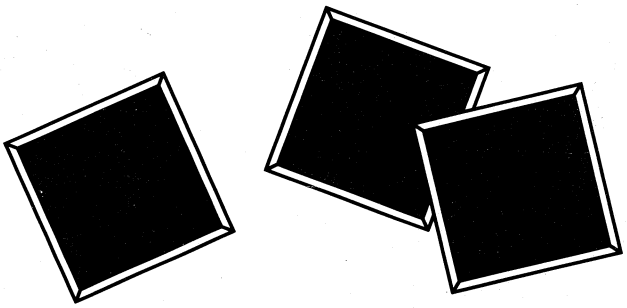
MVME188A VMEmodule™ RISC Microcomputer User's Manual



MOTOROLA INC.

MVME188A VMEmodule™

RISC Microcomputer

User's Manual

(MVME188A/D1)

NOTICE

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

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

RESTRICTED RIGHTS LEGEND

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

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

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

PREFACE

This manual provides general information, preparation for use and installation instructions, operating instructions, and functional description for the MVME188A RISC Microcomputer.

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

A basic knowledge of computers and digital logic is assumed.

To use this manual, you should be familiar with the publications listed in the *Related Documentation* section in Chapter 1 of this manual.

The computer programs stored in the Read Only Memories of this device contain material copyrighted by Motorola Inc., first published 1991, and may be used only under a license such as the License for Computer Programs (Article 14) contained in Motorola's Terms and Conditions of Sale, Rev. 1/79.

WARNING

THIS EQUIPMENT GENERATES, USES, AND CAN RADIATE RADIO FREQUENCY ENERGY AND IF NOT INSTALLED AND USED IN ACCORDANCE WITH THE INSTRUCTIONS MANUAL, MAY CAUSE INTERFERENCE TO RADIO COMMUNICATIONS. IT HAS BEEN TESTED AND FOUND TO COMPLY WITH THE LIMITS FOR A CLASS A COMPUTING DEVICE PURSUANT TO SUBPART J OF PART 15 OF FCC RULES, WHICH ARE DESIGNED TO PROVIDE REASONABLE PROTECTION AGAINST SUCH INTERFERENCE WHEN OPERATED IN A COMMERCIAL ENVIRONMENT. OPERATION OF THIS EQUIPMENT IN A RESIDENTIAL AREA IS LIKELY TO CAUSE INTERFERENCE IN WHICH CASE THE USER, AT HIS OWN EXPENSE, WILL BE REQUIRED TO TAKE WHATEVER MEASURES NECESSARY TO CORRECT THE INTERFERENCE.

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

Delta Series, HYPERmodule, M88000, SYSTEM V/88, VMEmodule, VMEsystem 3000, and 188Bug are trademarks of Motorola, Inc.

Rib-Cage is a trademark of DuPont.

© Copyright Motorola 1992
All Rights Reserved
Printed in the United States of America
February 1992

SAFETY SUMMARY
SAFETY DEPENDS ON YOU

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

GROUND THE INSTRUMENT.

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

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE.

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

KEEP AWAY FROM LIVE CIRCUITS.

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

DO NOT SERVICE OR ADJUST ALONE.

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

USE CAUTION WHEN EXPOSING OR HANDLING THE CRT.

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

DO NOT SUBSTITUTE PARTS OR MODIFY EQUIPMENT.

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

DANGEROUS PROCEDURE WARNINGS.

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

WARNING

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

CONTENTS

APPENDIX

INDEX

FIGURES

TABLES

CHAPTER 1
GENERAL INFORMATION

Introduction

This manual provides general information, preparation for use and installation instructions, operating instructions, and functional description for the MVME188A family of 32-bit Reduced-Instruction-Set-Computer (RISC) Microcomputers (referred to as the MVME188A throughout this manual, unless otherwise noted).

Model Designations

The MVME188A is available in several models, which are listed in Table 1-1. Note that the basic MVME188A includes only the System Controller (SYSCON) board and the 25 MHz Main Logic (CPU) board. It will not work without adding one, two, three, or four Memory Boards; and one HYPERmodule mezzanine module containing the RISC chips.

Enhancements to the MVME188A (over the MVME188) include:

- Single-cycle refresh to minimize VCC noise
- Increased onboard bypass capacitors
- Orthogonal data/address line layout to reduce crosstalk
- Onboard shielding of high-speed clock lines
- Change from CMOS to TTL drivers to reduce overshoot/undershoot.

The MVME188A also offers improved RESET functionality. It performs a full system RESET on MVME188A boards that are not the system controller and are memory mapped at a location other than $0. Both supervisory mode and user mode accesses are now permitted as well.

Table 1-1. MVME188A Model Designations

Product No.	Description
MVME188A	One System Controller (SYSCON) board and one 25 MHz Main Logic (CPU) board
MVME288-16	16MB DRAM Memory Board
MVME288-64	64MB DRAM Memory Board
MVME288EC-32	32MB ECC DRAM Memory Board
HM88K-1P32-2	25 MHz HYPERmodule mezzanine module with one MC88100 and two MC88200s
HM88K-1P64-2	25 MHz HYPERmodule mezzanine module with one MC88100 and four MC88200s
HM88K-1P128-2	25 MHz HYPERmodule mezzanine module with one MC88100 and eight MC88200s
HM88K-1P128-2	25 MHz HYPERmodule mezzanine module with one MC88100 and two MC88204s
HM88K-1P256-2	25 MHz HYPERmodule mezzanine module with one MC88100 and four MC88204s
HM88K-1P512-2	25 MHz HYPERmodule mezzanine module with one MC88100 and eight MC88204s
HM88K-2P64-2	25 MHz HYPERmodule mezzanine module with two MC88100s and four MC88200s
HM88K-2P128-2	25 MHz HYPERmodule mezzanine module with two MC88100s and eight MC88200s
HM88K-2P256-2	25 MHz HYPERmodule mezzanine module with two MC88100s and four MC88204s
HM88K-2P512-2	25 MHz HYPERmodule mezzanine module with two MC88100s and eight MC88204s
HM88K-4P128-2	25 MHz HYPERmodule mezzanine module with four MC88100s and eight MC88200s
HM88K-4P512-2	25 MHz HYPERmodule mezzanine module with four MC88100s and eight MC88204s

NOTE: The *WHOAMI Register* section in Chapter 4 has HYPERmodule descriptions.

Features

The MVME188A is an intelligent three-or-more-board set microcomputer module containing one or more MC88100 RISC microprocessor(s) and up to eight MC88200 or MC88204 RISC cache/memory management units (CMMUs). The boards are mechanically and electrically tightly connected to form a single unit, and the RISC chips are on a mezzanine module called the HYPERmodule.

The main features of the MVME188A are as follows:

- Three or more double-high/single-wide VMEboards including attached HYPERmodule mezzanine module
- Multiple RISC MPU/DRAM options on VMEmodule board set
- One to four MC88100 RISC microprocessors at 25 MHz featuring:
  Multiple execution units
  Floating point operations
  Single cycle integer, bit-field, branch, load, and store operations
- 32KB, 64KB, 128KB, 256KB, or 512KB of cache memory in two, four, or eight MC88200 or MC88204 Cache/Memory Management Units (CMMUs) featuring:
  16KB zero wait-state physical cache each (MC88200)
  or 64KB zero wait-state physical cache each (MC88204)
  Allows instructions to execute in one clock cycle
  MMU portion has two logical address ranges of 4 gigabytes each (user/supervisor)
  MMU has two Address Translation Caches: Page (PATC) and Block (BATC)
- 16MB parity DRAM (MVME288-16)
  or 64MB parity DRAM (MVME288-64)
  or 32MB Error Checking Correcting (ECC) RAM (MVME288EC-32)
  (The board set can be ordered with up to three additional memory boards (MVME288 series boards) to make a total of up to 64MB DRAM, or 256MB DRAM, or 128MB ECC memory maximum.)
- Full 32-bit master and slave VMEbus interfaces:
  A32/A24/A16 address
  D32/D16/D08 data
  Full system controller
  Four-level bus arbiter supports both priority (PRI) and round-robin-select (RRS)
  Bus requester supports release-on-request (ROR), release-when-done (RWD), and request-on-no-request (FAIR) modes
  7-level interrupt handler
  7-level interrupt requester
  Location monitor

- Orthogonal multi-processor interrupt controller
  Software direction of any of 25 interrupt sources to any of four processors
- Four JEDEC 32-pin ROM/PROM/EPROM/EEPROM sockets, wired as separate bytes of the 32-bit word, for 128K words (512KB) total (factory configuration is four 128K x 8 EPROMs containing the 188Bug debug/monitor firmware)
- Two EIA-232-D serial communication ports driven by a 68692 DUART
- Four programmable timers:
  Software configurable hardware watchdog
  Periodic tick
- Battery-backed-up clock/calendar
- 2KB non-volatile configuration storage
- Software programmable address decoders
  Local and VMEbus address maps are both dynamically configurable
- Full software support available:
  SYSTEM V/88 and Real-time Operating Systems
  Development tools
  Communications and applications

Specifications

General specifications for the MVME188A are provided in Table 1-2. Detailed specifications for the System Controller Board are provided in Table 1-3. Detailed specifications for the Memory Boards are provided in Table 1-4. Detailed specifications for the Main Logic Board are provided in Table 1-5.

The following sections detail cooling requirements and FCC compliance.

Cooling Requirements

The Motorola MVME188A VMEmodule board set is specified, designed, and tested to operate reliably with an incoming air temperature range from 0 degrees C to 55 degrees C (32 degrees to 131 degrees F) with forced air cooling at an inlet velocity of 300 LFM. Temperature qualification is performed in a standard Motorola VMEsystem 3000 chassis. The incoming air temperature and velocity is measured between the fan assembly and the card cage, where the incoming airstream first encounters the module under test. Test software is executed as the module is subjected to ambient temperature variations. Case temperatures of critical, high power density integrated circuits are monitored to ensure component vendors specifications are not exceeded.

While the exact amount of airflow required for cooling depends on the ambient air temperature and the type, number, and location of boards and other heat sources, adequate cooling can usually be achieved with 300 LFM flowing over the module. Less airflow is required to cool the module in environments having lower maximum ambients. Under more favorable thermal conditions, it may be possible to operate the module reliably at higher than 55 degrees C with increased airflow. It is important to note that there are other factors, such as the rated CFM of the air mover, which determine the actual volume and speed of air flowing over a module.

FCC Compliance

This VMEmodule board set (MVME188A) was tested in an FCC-compliant chassis, and meets the requirements for Class A equipment. FCC compliance was achieved under the following conditions:

a. Shielded cables on all external I/O ports.

b. Cable shields connected to earth ground via metal shell connectors bonded to a conductive module front panel.

c. Conductive chassis rails connected to earth ground. This provides the path for connecting shields to earth ground.

d. Front panel screws properly tightened.

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

Table 1-2. MVME188A Specifications

Characteristics	Specifications
Configuration	Full 32-Bit VMEbus Interface: A32/A24/A16 Address D32/D16/D08 Data Full System Controller Four-Level Bus Arbiter Modes: Priority (PRI) Round-Robin Select (RRS)

Table 1-2. MVME188A Specifications (cont'd)

Characteristics	Specifications
	Bus Requester Modes: Release On Request (ROR) Release When Done (RWD) Request On No Request (FAIR) 7-Level Interrupt Handler 7-Level Interrupt Requester Location Monitor
Form Factor	Three Single Wide, Double High Eurocards (minimum) (Expandable to Six Single Wide, Double High Eurocards)
Power Requirements (for three board set only)	+5 Vdc at 21.0 A (typical), 24.0 A (max) +12 Vdc at 16 mA (typical), 20 mA (max) -12 Vdc at 16 mA (typical), 20 mA (max)
Temperature	
Operating (refer to *Cooling Requirements* section)	0 degrees to 55 degrees C at point of entry of forced air (approximately 300 LFM)
Storage	-40 degrees C to +85 degrees C
Relative Humidity	5% to 90% (non-condensing)
Physical Dimensions	
Card Assembly Height Depth	 9.187 inches (233.35 mm) 6.299 inches (160.00 mm)
Front panel (3 cards) Height Width	 10.309 inches (261.85 mm) 2.40 inches (60.96 mm)

Table 1-3. System Controller Board Specifications

Characteristics	Specifications
Memory	ROM/PROM sockets for 512KB (contains the 188Bug debugger/diagnostic firmware) 128KB SRAM 2KB battery-backed SRAM
Registers	Eight Global Control and Status Registers (GCSR): GLOBAL0 - GLOBAL1, BRDID, GPCSR0 - GPCSR4 21 other registers

Table 1-4. Memory Board Specifications

Characteristics	Specifications
MVME288-16	16 MB DRAM with parity, arranged as four interleaved banks of 32-bit words to facilitate 16-byte burst transfers
MVME288-64	64 MB DRAM with parity, arranged as four interleaved banks of 32-bit words to facilitate 16-byte burst transfers
MVME288EC-32	32 MB DRAM with error checking/correction, arranged as two interleaved banks of 32-bit words with 7 check bits for error detection and correction
Registers	Control/Status Register (CSR), and diagnostics latch

Table 1-5. Main Logic Board (CPU Board) Specifications

Characteristics	Specifications
RISC Microprocessor(s)	One, two, or four MC88100s Speed: 25 MHz
RISC Cache/Memory Management Units	Two, four, or eight MC88200s or MC88204s
Memory	SRAM (used only for M bus address decoding and for VMEbus slave address decoding)
Registers	CPU Control/Status Register (CCSR) 10 other registers

General Description

The MVME188A board set consists of a system controller board (utility I/O RISC module), one, two, three, or four memory boards, and a main logic board (CPU processor) with a HYPERmodule (CPU cluster mezzanine module), all electrically and mechanically connected and with three or more single-wide front panels.

System Controller Board

The system controller board contains a VMEbus A16 slave interface, ROM, static RAM, UARTs, and timing elements plus local and global control/status registers.

Memory Board

On the memory board, the DRAM memory complement communicates with the CMMU devices on the HYPERmodule mezzanine over the high speed local bus or slave bus (S bus), a modified form of the M bus, the CMMU memory bus. (Because the S bus is a modified form of the M bus, descriptions of registers and other circuits sometimes describe it as the M bus.) Memory capacity can be increased by adding one to three MVME288 series memory boards.

Main Logic Board with HYPERmodule Mezzanine Module

Up to four clusters of MC88100 RISC Microprocessors closely coupled with MC88200 or MC88204 Cache/Memory Management Units are contained on a HYPERmodule mezzanine module plugged into the the main logic board. The main logic board contains a full 32-bit VMEbus interface, as well as address decode logic and S bus (MVME188A local Slave bus) interface logic.

Equipment Required

The following equipment is required to make a complete system using the MVME188A:

Terminal(s)
Disk drives and controllers
Chassis and power supply
Optional transition module MVME714M and connecting cables
Operating system

The MVME188Bug debug monitor firmware (188Bug) is provided in the four EPROM sockets on the MVME188A system controller board. It provides over 50 debug, up/downline load, and disk bootstrap load commands, as well as a full set of onboard diagnostics and a one-line assembler/disassembler. 188Bug includes a user interface which accepts commands from the system console terminal. 188Bug can also operate in a System Mode, which includes choices from a service menu. Refer to the *MVME188BUG Debugging Package User's Manual* for details.

The MVME714M transition module is a serial distribution module with the ability to connect two EIA-232-D devices to the MVME188A module. All of the circuitry necessary to convert from TTL serial I/O levels to EIA-232-D signal levels is on the MVME188A module. The MVME714M then provides one console port and one port for terminal/printer/modem, using DB-25 connectors. The MVME714M has a connector to have a UDS 2980, 3192, or 3382A modem plugged directly onto the PWB.

Note that the MVME188A contains no parallel ports. To use a parallel device, such as a printer, with the MVME188A, it is necessary to add a module such as the MVME335 Serial and Parallel I/O Module to the system.

Software support being made available for the MVME188A includes SYSTEM V/88 and real-time operating systems, programming languages, and other tools and applications. Contact your local Motorola sales office for more details.

Related Documentation

The following publications are applicable to the MVME188A and may provide additional helpful information. If not shipped with this product, they may be purchased from Motorola, Inc., Computer Group, Technical Literature Center, 1919 W. Fairmont, suite 8, Tempe, Arizona 85282; telephone 1-800-458-6443; FAX 602-438-0240.

Document Title	Motorola Publication Number
MVME188A VMEmodule RISC Microcomputer Support Information (Refer to the *Support Information* section in this chapter)	SIMVME188A
MVME188Bug Debugging Package User's Manual	MVME188BUG
MVME714, MVME714M, and MVME714M-1 2-Channel Serial I/O Distribution Modules User's Manual	MVME714
MC88100 RISC Microprocessor User's Manual	MC88100UM
MC88200 Cache/Memory Management Unit (CMMU) User's Manual	MC88200UM
MC88204 64K-Byte Cache/Memory Management Unit (CMMU) data sheet	MC88204/D
MC68681 Dual Asynchronous Receiver/Transmitter (DUART) Data Book	MC68681
MVME6000 VMEbus Interface User's Manual	MVME6000
HM88K HYPERmodule 32-Bit RISC Processor Mezzanine Module User's Manual	HM88KUM

NOTE: Although not shown in the above list, each Motorola Computer Group manual publication number is suffixed with characters which represent the revision level of the document, such as "/D2" (the second revision of a manual); a supplement bears the same number as a manual but has a suffix such as "/A1" (the first supplement to the manual).

The following publications are available from the sources indicated.

SCC68692 Dual Asynchronous Receiver/Transmitter (DUART) data sheet, Signetics Corporation, 811 E. Arques Avenue, P.O. Box 3409, Sunnyvale, California 94088-3409

MK48T02 2K x 8 Zeropower/Timekeeper RAM Data Sheet, Thompson Components Mostek, 1310 Electronics Drive, Carrollton, TX 75606

Z8536 Counter/Timer and Parallel I/O Unit (CIO) data sheet, Zilog, Inc., Corporate Communications, Building A, 1315 Dell Ave, Campbell, CA 95008

Am29C660 CMOS Cascadable 32-Bit Error Detection and Correction Circuit data sheet, publication# 10565 Rev. B, in Dynamic Memory Design 1990 Data Book/Handbook, Advanced Micro Devices, Inc., 901 Thompson Place, P.O. Box 3453, Sunnyvale, California, 94088-3453.

Versatile Backplane Bus: VMEbus, ANSI/IEEE Std 1014-1987, The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017

Support Information

The SIMVME188A manual contains the connector interconnect signal information, parts lists, and the schematics for the MVME188A.

This manual may be obtained from Motorola, Inc., Computer Group, Technical Literature Center, 1919 W. Fairmont, suite 8, Tempe, Arizona 85282; telephone 1-800-458-6443; FAX 602-438-0240.

Manual Terminology

Throughout this manual, a convention has been maintained whereby data and address parameters are preceded by a character which specifies the numeric format as follows:

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

Unless otherwise specified, all address references are in hexadecimal throughout this manual.

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

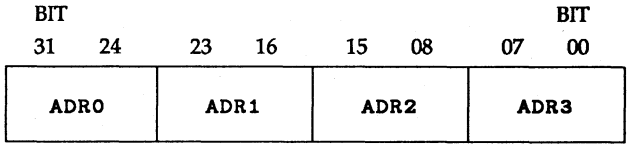
An asterisk (*) following the signal name for signals which are edge significant denotes that the actions initiated by that signal occur on high to low transition.

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

Data and address sizes are defined as follows:

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

Throughout this manual, it is assumed that the CPU(s) on the MVME188A always program the CMMUs with big-endian byte ordering, as shown below. Any attempt to use small-endian byte ordering will immediate render the MVME188Bug debugger unusable.



CHAPTER 2
HARDWARE PREPARATION AND INSTALLATION

Introduction

This chapter provides unpacking instructions, hardware preparation, and installation instructions for the MVME188A, including preparation and installation of extra memory boards.

Unpacking Instructions

NOTE

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

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

CAUTION

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

Hardware Preparation

To select the desired configuration and ensure proper operation of the MVME188A, certain modifications may be necessary before installation. These modifications are made through switch settings and header connections, as described in the following sections. Many other modifications are done by setting bits in control registers after the MVME188A has been installed in a system. (The MVME188A registers are described in Chapter 4.)

2

System Controller Board Switches and Header Connections

The location of the switches, jumpers, connectors, and LED indicators on the system controller board of the MVME188A is illustrated in Figure 2-1. The MVME188A has been factory tested and is shipped with factory switch settings that are described in the following sections. The MVME188A operates with its required and factory-installed Debug Monitor, MVME188Bug (188Bug), with these factory switch settings. Switches S1 through S4-8 are factory-configured as shown in Table 2-1.

Table 2-1. MVME188A Front Panel Switches Factory Configuration

Function	Configuration	Condition
ABORT switch (NOTE 1)	S1 momentary contact	Abort interrupt not started.
RESET switch (NOTE 1)	S2 momentary contact	MVME188A module not reset.
System controller (SCON*) enable	S3-1 closed	MVME188A module is system controller.
ENV0*-ENV2* functions (NOTE 2)	S3-2 closed S3-3 closed S3-4 closed	Z8536 CIO pin PB3 grounded. Z8536 CIO pin PB4 grounded. Z8536 CIO pin PB5 grounded.
Global Control and Status Register (GCSR) group base address within the VMEbus short I/O (A16) space	S3-5 open S3-6 open S3-7 closed S3-8 closed S4-1 open S4-2 closed S4-3 closed S4-4 closed	GCSR group base address is $**C8**00 in VMEbus short I/O (A16) space.
GCSR board base address select in VMEbus A16 space	S4-5 closed S4-6 closed S4-7 closed S4-8 closed	GCSR board base address is $C8**00** in VMEbus short I/O (A16) space.

NOTES: 1. The ABORT and RESET switches are described in Chapter 3.

2. The Z8536 is described in Chapter 4. ENV0* through ENV2* are reserved for use by the 188Bug firmware, if it is used.

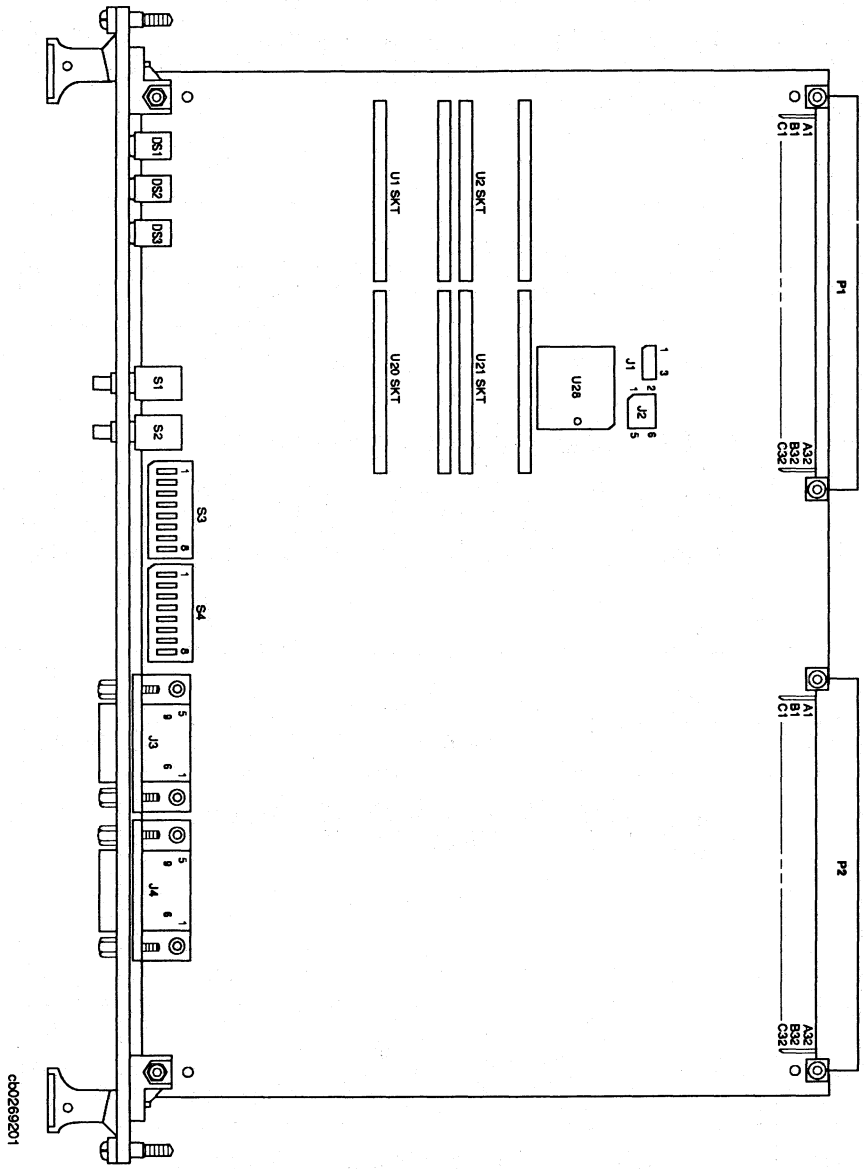
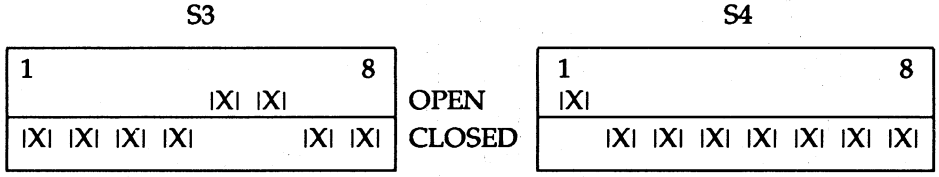


Figure 2-1. System Controller Board Switches, Headers, Connectors, and LED Indicators Location Diagram

2

Configuration Switches (S3 and S4), General Information

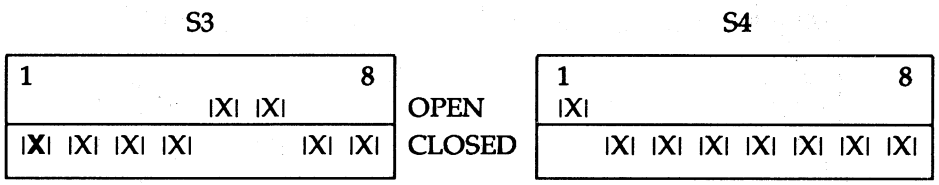
S3 and S4 are two banks of eight two-way switches accessible through the front panel. The factory configuration was given in Table 2-1. The following sections explain possible optional configurations.



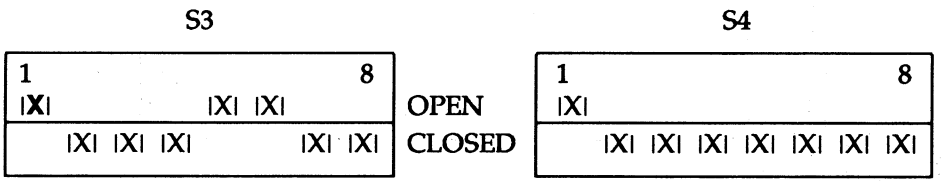
FACTORY CONFIGURATION

Configuration Switch S3-1, System Controller Enable Function

Segment 1 of S3 (S3-1) is the SCON switch. It determines whether the MVME188A module functions as VMEbus system controller. Factory configuration is with system controller enabled (SCON* = 0, S3-1 is on = closed).



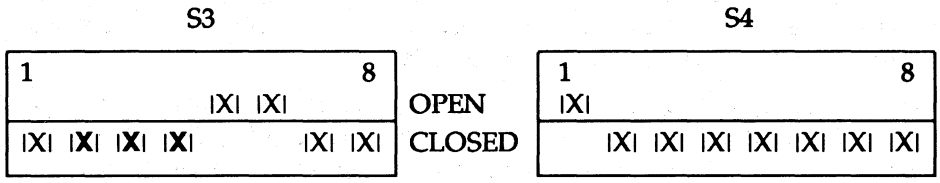
THIS MVME188A IS THE SYSTEM CONTROLLER.
(FACTORY CONFIGURATION)



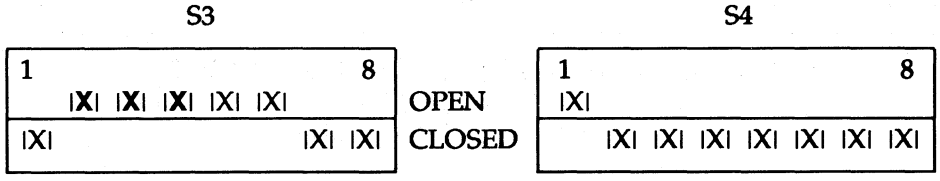
THIS MVME188A IS NOT THE SYSTEM CONTROLLER.

Configuration Switches S3-2 through S3-4, ENV0*-ENV2* Functions

Segments 2, 3, and 4 of S3 (S3-2, S3-3, and S3-4) are for the ENV0* through ENV2* functions. These switches can be read by the software from port B of the CIO. Their function is determined solely by the software. These switches are reserved by the 188Bug firmware if it is used.



THIS MVME188A SETS ENV0* THROUGH ENV2* TO 0.
S3-2 THROUGH S3-4 ARE ON = CLOSED.
Z8536 CIO PINS PB3 THROUGH PB5 ARE GROUNDED.
(FACTORY CONFIGURATION)



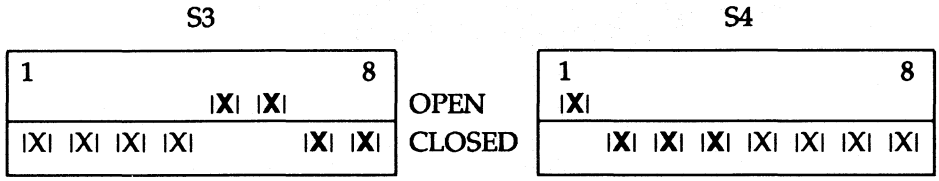
THIS MVME188A SETS ENV0* THROUGH ENV2* TO 1.
S3-2 THROUGH S3-4 ARE OFF = OPEN.
Z8536 CIO PINS PB3 THROUGH PB5 ARE NOT GROUNDED.

2

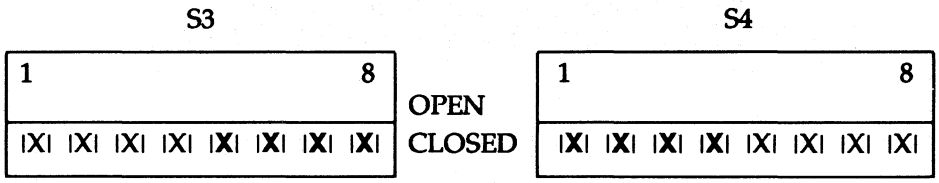
Configuration Switches S3-5 through S4-4, GCSR Group Address Functions

Segments 5 through 8 of S3 (S3-5, S3-6, S3-7, S3-8) and segments 1 through 4 of S4 (S4-1, S4-2, S4-3, S4-4) are the Group Address switches GRPAD7 through GRPAD0. These switches determine (along with the BDAD switches) the base address of the GCSR within VMEbus short I/O (A16) space. Note that the GCSR locations are mapped on two-byte boundaries: each register occupies only two bytes of local address space. (All other registers on the MVME188A are mapped on four-byte boundaries. Chapter 4 has details.)

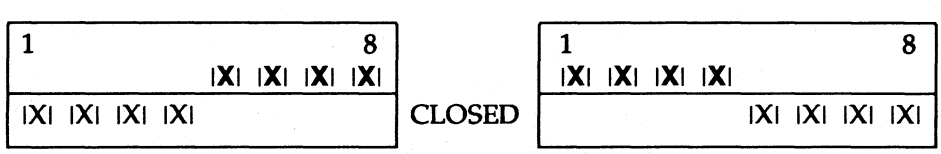
GRPAD[7:0] group address: This address is compared with VMEbus or slave bus (S bus) address bits 15:8 to determine where to place the GCSR in the map. It is also used to determine what addresses the location monitors respond to. It can be read by accessing the BASAD register. (Switch closed = on = 0, switch off = 1.)



THIS MVME188A HAS GCSR GROUP BASE ADDRESS
$C800 IN A16 SPACE.
(FACTORY CONFIGURATION)



THIS MVME188A HAS GCSR GROUP BASE ADDRESS
$0000 IN A16 SPACE.



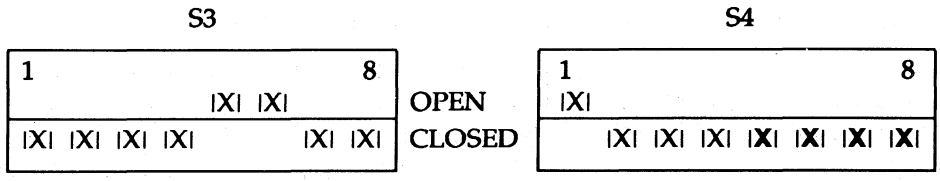
THIS MVME188A HAS GCSR GROUP BASE ADDRESS
$FF00 IN A16 SPACE.

Configuration Switches S4-5 through S4-8, GCSR Board Address Functions

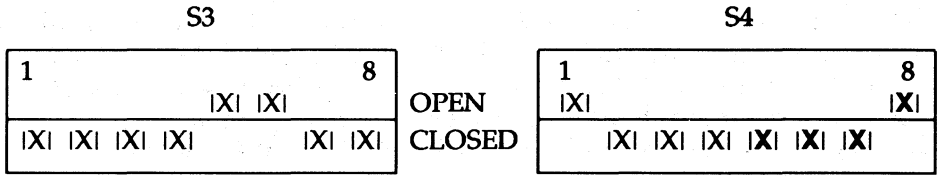
The Board Address switches (BDAD3 through BDAD0) consist of segments 5 through 8 of S4 (S4-5, S4-6, S4-7, S4-8). These switches determine (along with the GRPAD switches) the base address of the GCSR within VMEbus short I/O (A16) space. Note that the GCSR locations are mapped on two-byte boundaries: each register occupies only two bytes of local address space. (All other registers on the MVME188A are mapped on four-byte boundaries. Chapter 4 has details.)

BDAD[3:0] board address within group : This address is compared with VMEbus or S bus address bits 7:4 to determine where to place the GCSR in the map. It can be read by accessing the BASAD register. (Switch closed = on = 0, off = 1.)

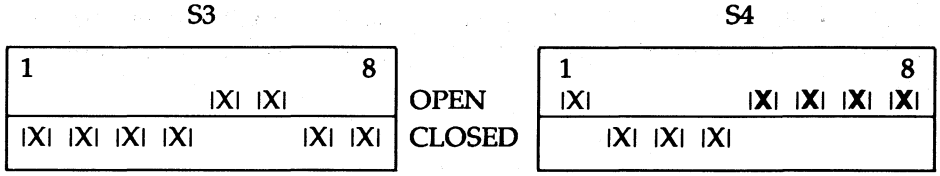
Note that BDAD[3:0] = %1111 ($F) is reserved for the location monitors; it is not meant to be used as a board address. If the BDAD switches are all off (BDAD[3:0] = %1111), GCSR accesses will still work, but location monitor interrupts will be generated to all boards within the same group address whenever a GCSR access occurs.



THIS MVME188A HAS GCSR BOARD BASE ADDRESS
$C800 IN A16 SPACE.
(FACTORY CONFIGURATION)



THIS MVME188A HAS GCSR BOARD BASE ADDRESS
$C810 IN A16 SPACE.

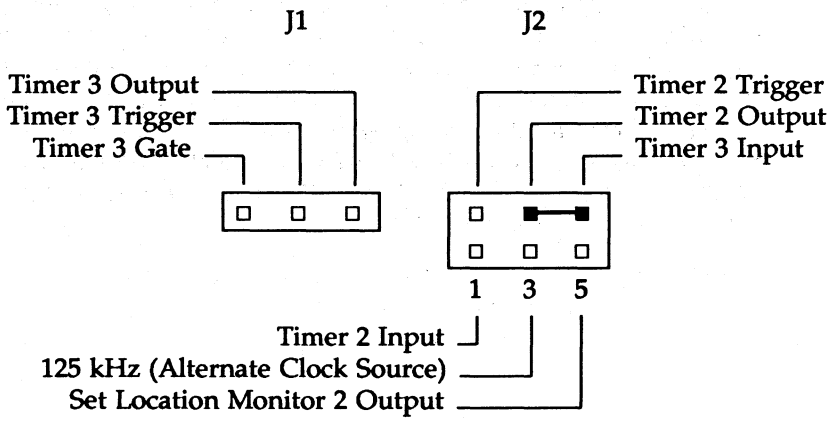


THIS MVME188A HAS GCSR BOARD BASE ADDRESS
$C8F0 IN A16 SPACE.
(NOT RECOMMENDED)

Z8536 CIO External Timer Select Jumper Headers J1 and J2

Spaces for headers J1 and J2 on the system controller board provide the means to access timers 2 and 3 of the Z8536 timer (U28). These jumper headers are designed to allow an alternate clock source and to allow cascading of the counters. The alternate clock is 125 kHz, derived from the 4 MHz clock that drives the clock input of the Z8536 (divide by 32). The factory setting for J1 and J2 has a zero-ohm shunt R37 installed that connects J2 pins 4 and 6. This connects timer 2 output to timer 3 input for diagnostic purposes. Also, the timer 3 output is used as the watchdog timer signal when the system software configures the Z8536 to perform the watchdog timer function. Location monitor 2 is also accessible as an input clock source to the Z8536. This could be set up to count events or perform synchronizing functions.

No actual jumper header pins are provided, just spaces for them to be connected. Some of the jumper header pin locations are not intended to be jumper configured. These provide external logic access to timer 2 trigger and to timer 3 output, trigger, and gate signals. These signals can be connected to other circuitry via wirewrap or jumper clip connections. Refer to the Z8536 data sheet (listed in the *Related Documentation* section in Chapter 1) for details regarding timer functionality.



Zero-ohm Resistor R61

Values listed for LRST in Chapter 3 are true only if the zero-ohm resistor R61 is removed from the SYSCON board in the MVME188A. If R61 is left in place, as shipped, the values for LRST are the same as those for SRST.

16MB or 64MB DRAM Memory Board Switch S1

The location of the four DRAM memory board switch segments is illustrated in Figure 2-2. The DRAM memory board has been factory tested and is shipped with factory switch settings as described in the following sections. However, if additional 16MB or 64MB DRAM boards (MVME288 series) are added to the MVME188A board set, they must have their switches changed. One memory board must always be set up as board 0. Switch S1 is factory-configured as shown in Table 2-2.

Table 2-2. 16MB or 64MB DRAM Memory Board Switches Factory Configuration

Function	Configuration	Condition
Memory board number	S1-1 closed S1-2 open S1-3 open S1-4 open	Memory board is board number 0.

NOTES: 1. Closed = on; open = off.

2. These memory boards do no address decoding, only board number decoding. Switches are used to set one of four possible memory board selects. Address ranges are set up by the software through the WMAD register. The MVME188Bug debugging package, for example, sets up the following default M bus address ranges for boards 0 through 3 on a 16MB board: $00000000 through $00FFFFFF, $01000000 through $01FFFFFF, $02000000 through $02FFFFFF, $03000000 through $03FFFFFF.

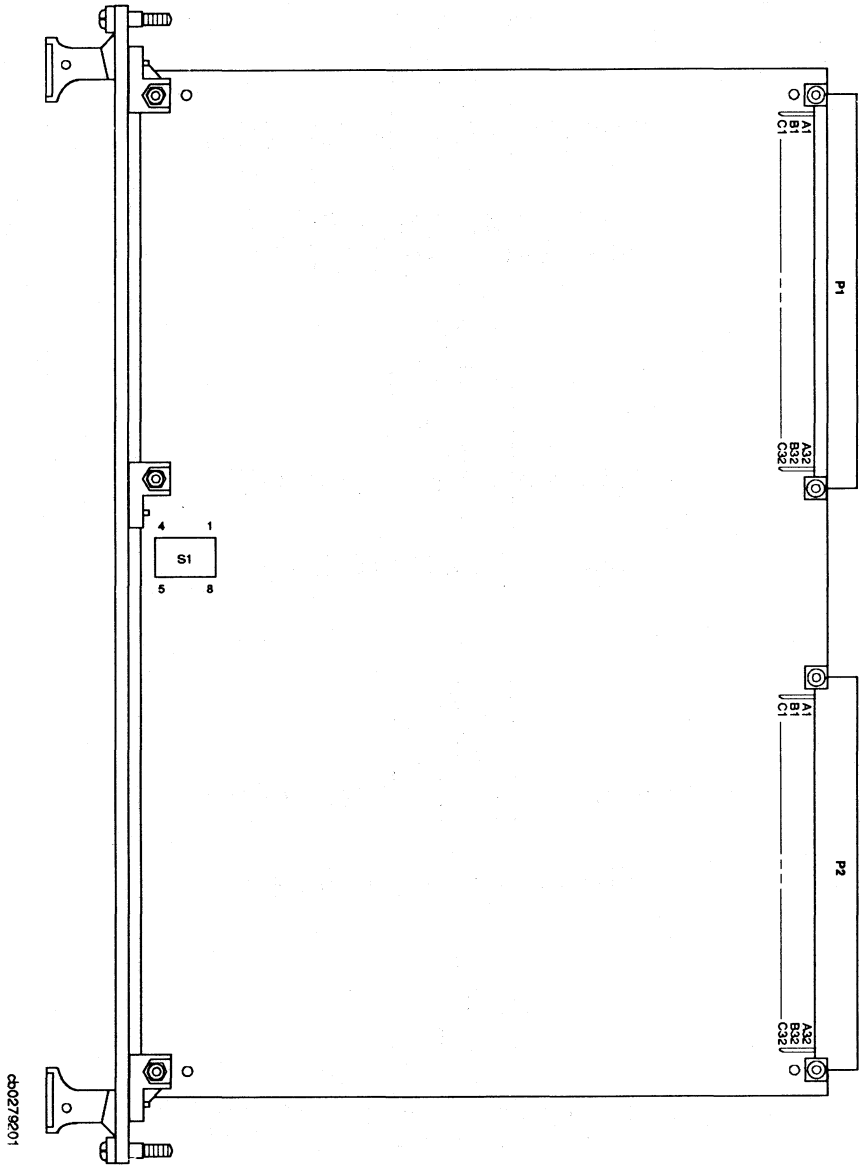
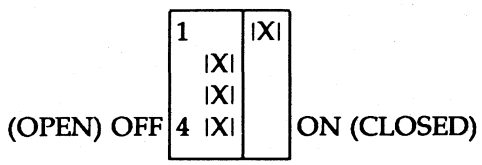


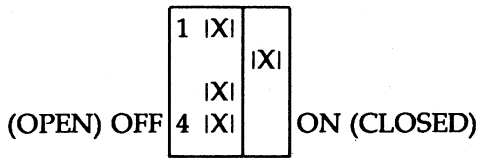
Figure 2-2. 16MB or 64MB DRAM Memory Board Switch Location Diagram

S1



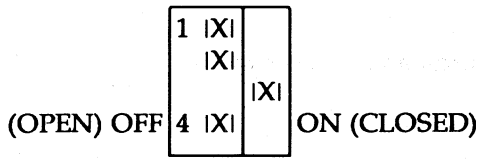
MEMORY BOARD 0 SELECTED.
(FACTORY CONFIGURATION)

S1



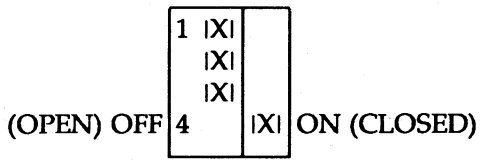
MEMORY BOARD 1 SELECTED.

S1



MEMORY BOARD 2 SELECTED.

S1



MEMORY BOARD 3 SELECTED.

32MB ECC DRAM Memory Board Switch SW1

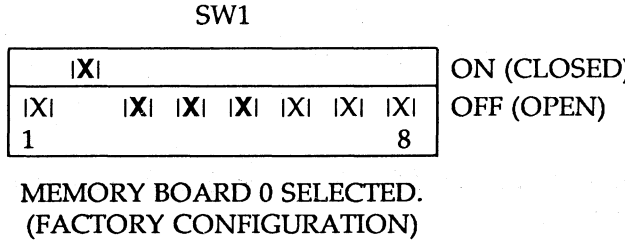
The location of the eight ECC DRAM memory board switch segments is illustrated in Figure 2-3. The ECC DRAM memory board has been factory tested and is shipped with factory switch settings as described in the following sections. However, if additional 32MB ECC DRAM boards (MVME288 series) are added to the MVME188A board set, they must have their switches changed. One memory board must always be set up as board 0. Switch SW1 is factory-configured as shown in Table 2-3.

Table 2-3. 32MB ECC DRAM Memory Board Switches Factory Configuration

Function	Configuration	Condition
Not used	SW1-1 open	None.
Memory board number	SW1-2 closed SW1-3 open SW1-4 open SW1-5 open	Memory board is board number 0.
Control and Status Register (CSR) decode	SW1-6 open SW1-7 open SW1-8 open	A31-A29 = 111 matches.

NOTES: 1. Closed = on; open = off.

2. The eight segment DIP switch SW1 enables two decode functions. Switch segment one is not used. Segments two through five select the memory board's position in the MVME188A slave bus memory map. Switch segment two selects memory board 0, while switch segment five selects memory board 3. Refer to Chapter 3 for the DRAM mapping addresses. Switch segments six through eight help determine the location of the two CSRs in the memory map.



MEMORY BOARD 0 SELECTED.
(FACTORY CONFIGURATION)

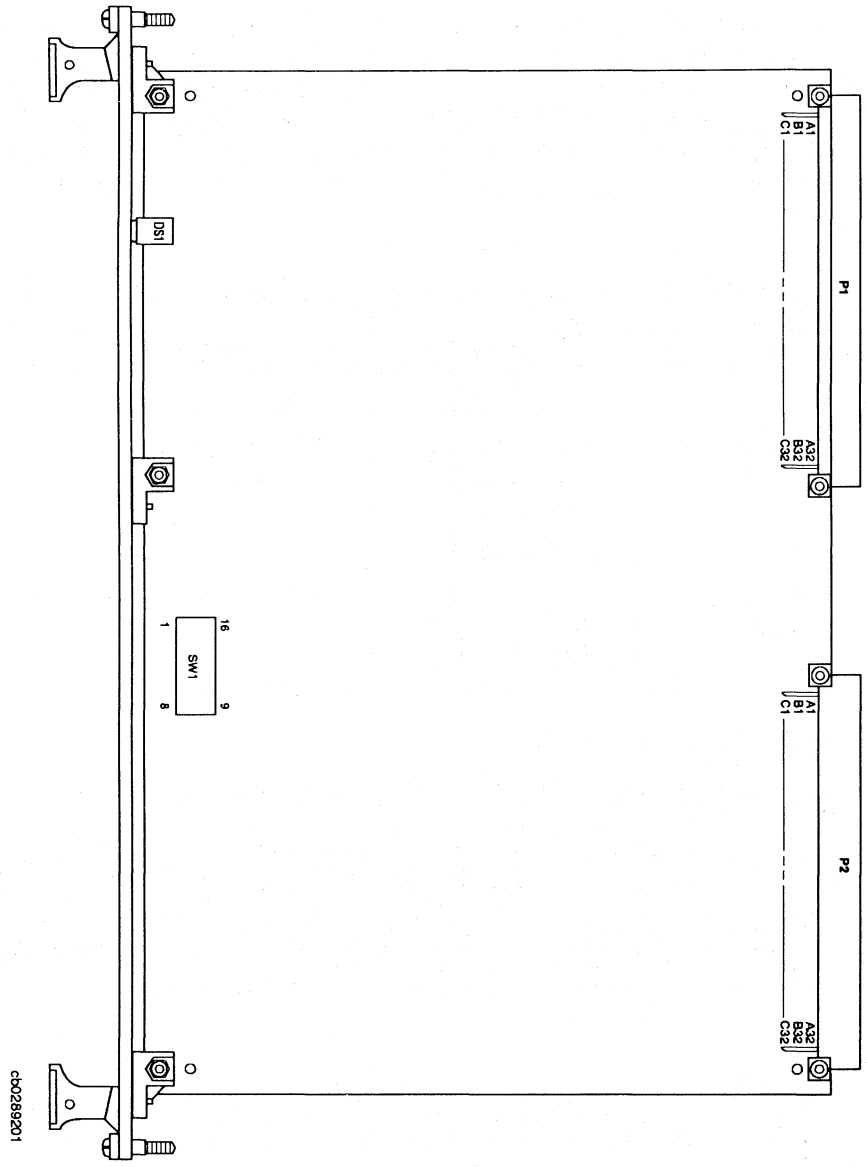
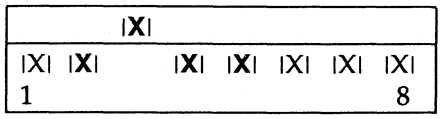


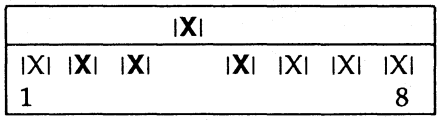
Figure 2-3. 32MB ECC DRAM Memory Board Switch and Indicator Location Diagram

SW1



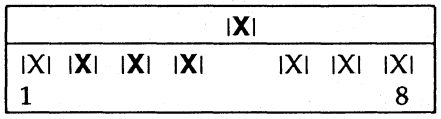
MEMORY BOARD 1 SELECTED.

SW1



MEMORY BOARD 2 SELECTED.

SW1



MEMORY BOARD 3 SELECTED.

Switch segments six through eight help determine the location of the two CSRs in the memory map. The actual decode of the CSR is determined by the settings for switch segments six through eight, the existence of a board select as determined by switch segments two through five, and address bit A2. Settings for switch segments six through eight are compared to bus address bits A31-A29. Leaving the switch segment setting open lets the lines pull high so that A31-A29 = 111 matches. This is the setting required by system software. Changing to a different switch setting is for specific users who have a conflict with that address. A2 = 0 enables the board level CSR. A2 = 1 and a write cycle cause a write to the diagnostic registers in the Am29C660 chips (U14 and U36). A read to this location starts check bit initialization. The diagnostic latches are write-only and are written concurrently. Refer to the Am29C660 data sheet listed in Chapter 1 for diagnostic latch definition. Note that the two locations of CSR for each ECC memory board require a 4MB window in the slave bus memory map.

The table below indicates the CSR addresses as initialized by the MVME188Bug firmware. Note that these decodes are soft-programmable and may change with future versions of firmware. Refer to Chapter 4 for details on programming the CSR.

Board Number	Switch Setting On (Closed)	CSR Address	Diagnostic Latch Register Address
0	SW1-2	$E0C00000	$E0C00004
1	SW1-3	$E0800000	$E0800004
2	SW1-4	$E0400000	$E0400004
3	SW1-5	$E0000000	$E0000004

Installation Instructions

The following sections discuss removal and installation of the HYPERmodule mezzanine module on the MVME188A module, installation of additional memory board(s) (MVME288 series) on the MVME188A, installation of the MVME188A into a VME chassis, connection of EIA-232-D terminal(s) and cable(s), and system considerations. Ensure that ROM/PROM/EPROM/EEPROM devices are installed. Factory configuration is with the EPROMs for the MVME188Bug debug monitor. For MVME188Bug version 4.4, these are B75 in socket for U21, B76 in socket for U1, B77 in socket for U20, and B78 in socket for U2. Ensure that all switch segments and header connections are configured as desired.

Removal and Installation of HYPERmodule Mezzanine Module

The HYPERmodule on the MVME188A is replaceable. It is possible to upgrade the MVME188A by replacing a single-processor HYPERmodule with a dual- or quad- processor HYPERmodule as follows.

a. Turn all equipment power OFF and disconnect power cable from ac power source.

CAUTION

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

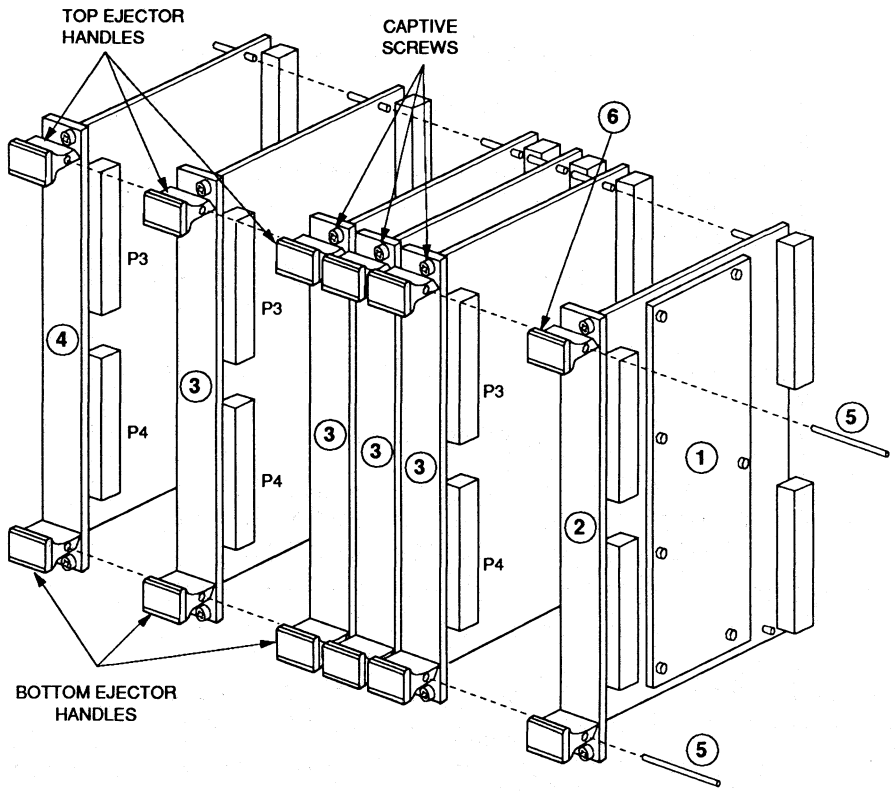
WARNING

DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS EQUIPMENT. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

b. Removal of the MVME188A board set from the system differs depending on the chassis your system is in. In general, proceed as follows:

   1. Open the appropriate cover(s) and/or access doors in the chassis to expose the front panels of the MVME188A.
   2. Locate the MVME188A board set in the leftmost three, four, five, or six slots (slots 1, 2, and 3; or 1, 2, 3, and 4; or 1, 2, 3, 4, and 5; or 1, 2, 3, 4, 5, and 6) of the card cage. See Figure 2-4. The top and bottom ejector handles of all boards in the set are ganged together by steel pins through the handles.
   3. Unscrew (loosen) all top and bottom captive screws on the front panels of the MVME188A board set. These screws mount the MVME188A in the chassis.
   4. Lift the top and bottom ejector handles and pull the MVME188A board set evenly forward out of its backplane slots.

c. Locate the screws that fasten the HYPERmodule to the MVME188A main logic board (CPU board) at the right side of the MVME188A board set. There are four screws at the front of the boards (near the front panel) and two or three at the back of the boards (near the backplane connectors).



1 HYPERMODULE
2 CPU BOARD
3 MEMORY BOARDS
4 SYSCON BOARD
5 STEEL PINS
6 CPU NAME PLATE

QUANTITY OF MEMORY BOARDS (ITEM3) CAN RANGE FROM 1 TO 4 (1 TO 2 ON 6 SLOT CHASSIS) AND IS DETERMINED BY SYSTEM CONFIGURATION

LENGTH OF PIN (ITEM 5) IS DETERMINED BY QUANTITY OF MEMORY BOARD ASSEMBLIES (ITEM 3)

CONFIGURE MEMORY BOARD ASSEMBLIES (ITEM 3) PER INSTRUCTIONS IN THIS CHAPTER.

cb0299201

Figure 2-4. MVME188A Board Set

CAUTION

Removing or installing screws incorrectly may warp or crack boards. Do not bend the HYPERmodule board, because this could break solder connections.

d. Carefully loosen the front and back screws, turning each screw equally one turn, then turning each one more turn, and so on until the screws and their washers are disconnected. Retain them for later use.

e. Remove the HYPERmodule from the main logic board, being careful not to damage the pins of the three 100-pin connectors that connect them electrically. Do not bend the HYPERmodule board, because this could break solder connections. Lift evenly along the front edge of the HYPERmodule to uniformly separate the connector pins from their socket holes. Do not attempt to do this by hand; instead, use a stiff non-conducting board, such as a 6-inch by 6-inch by 0.25-inch epoxy-glass board.

f. Install a new HYPERmodule, again being careful to mate the three 100-pin connectors exactly and evenly.

g. Carefully connect the front and back screws and their washers, tightening each screw equally one turn, then tightening each one turn more, and so on until the screws are tight.

NOTE

No adjustment or configuring of the MVME188A is required. Circuits on the HYPERmodule tell the CPU main logic board whether it is single, dual, or quad.

h. If additional MVME288 series memory boards are to be installed, proceed to the next section. If not, skip to the section on *MVME188A Module Installation in a System* and reinstall the MVME188A board set in the system.

Installation of Additional MVME288 Series Memory Board(s)

The MVME188A can be disassembled to add up to three additional 16MB or 64MB DRAM or 32MB ECC DRAM memory boards (MVME288 series). Proceed as follows.

a. Turn all equipment power OFF and disconnect power cable from ac power source.

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

> **WARNING**
>
> **DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS EQUIPMENT. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.**

b. If you have not already done so, remove the MVME188A board set from the system. Removal of the MVME188A board set from the system differs depending on the chassis your system is in. In general, proceed as follows:

   1. Open the appropriate cover(s) and/or access doors in the chassis to expose the front panels of the MVME188A.
   2. Locate the MVME188A board set in the leftmost three, four, five, or six slots (slots 1, 2, and 3; or 1, 2, 3, and 4; or 1, 2, 3, 4, and 5; or 1, 2, 3, 4, 5, and 6) of the card cage. See Figure 2-4. The top and bottom ejector handles of all boards in the set are ganged together by steel pins through the handles.
   3. Unscrew (loosen) all top and bottom captive screws on the front panels of the MVME188A board set. These screws mount the MVME188A in the chassis.

4. Lift the top and bottom ejector handles and pull the MVME188A board set evenly forward out of its backplane slots.

c. With the board set in a vertical position, push the CPU nameplate, which is on the top right ejector handle, out of the ejector handle. Push the Motorola Logo nameplate, which is on the bottom right ejector handle, out of the ejector handle. The two steel pins are now visible through the ejector handles. Retain both nameplates for later installation.

d. Using long-nose pliers, push (from the left) and pull (to the right) both steel pins until they are out of all the ejector handles. Retain these pins for possible further use. Note that if additional memory boards are installed, longer pins are required, and the original pins can not be used.

CAUTION

Separating or connecting plugs and spacers incorrectly may warp or crack boards.

e. The boards in the MVME188A board set are properly arranged as: leftmost, the SYSCON board; next the MEMORY board(s); and rightmost, the CPU board (with HYPERmodule). To separate the board set and/or remove memory board(s), gently pull the front panels of the boards apart and carefully separate the two 96-pin connectors P3 and P4 that are behind the front panels. Now pull the boards further apart until the compliant plugs and spacers at the two back corners of the boards pop apart.

f. Retain the SYSCON board, any memory board(s) you wish to reuse, and the CPU board (with HYPERmodule) for further use.

g. Obtain any new memory board(s) you wish to add.

h. A maximum of four memory boards may be used in any MVME188A board set. If you are using only 16MB memory boards, or only 64MB memory boards, or only 32MB ECC memory boards, they may be stacked in any order. If you are using a mixture of 32MB, 64MB, and/or 16MB memory boards, the 32MB ECC board(s) should be stacked to the left (nearest the SYSCON board), and the 64MB board(s) must be to the left of the 16MB board(s).

i. Lay the memory boards flat, from left to right, in the order you intend to stack them.

j. Configure any additional 16MB or 64MB DRAM or 32MB ECC DRAM memory board(s) you wish to install by setting S1 or SW1 on each board per the instructions earlier in this chapter. Configure them as board(s) 0, 1, 2, 3 in that order. (If your MVME188A already has two memory boards, you can add only two more.) The total number of memory boards that may be included in any one MVME188A board set is four.

k. Line up the boards for your modified MVME188A board set. Be sure that the SYSCON board is at the left, the MEMORY board(s) in correct order from left to right, and the CPU board at the right end.

l. Carefully press the boards together, mating the two 96-pin connectors P3 and P4 on each board that are just behind the front panels. The pins on the center row of the connectors (row B) are longer than the other pins and will serve to guide the connectors to a correct fit.

m. Next, press the back corners of the boards together until the compliant plugs and spacers at the two back corners of the boards pop together.

n. Line up the top and bottom ejector handles of all boards in the MVME188A board set. Select the right length steel pins. If you have added board(s), you need longer pins than those you removed. Insert the pins through the ejector handles from the right end. Gently tap the pins in place if they do not fit easily.

o. Reinstall the CPU nameplate and the Motorola Logo nameplate into the upper right and lower right ejector handles, respectively.

p. Proceed to the next section on *MVME188A Module Installation in a System* and reinstall the MVME188A board set in the system.

MVME188A Module Installation in a System

Now that the MVME188A module is ready for installation, proceed as follows:

a. Turn all equipment power OFF and disconnect power cable from ac power source.

CAUTION

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

CAUTION

After removal of system power, allow several seconds before restoring system power. Rapid power on-off cycling may result in system instability and power-up error messages. The actual off-time required varies from system to system, depending on the power supply hold-up time and on system payloads.

WARNING

DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS EQUIPMENT. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

b. Remove chassis cover(s) and/or access doors as instructed in the equipment user's manual.

c. Remove the filler panels from the appropriate card slots at the front and rear of the chassis (if the chassis has a rear card cage). The MVME188A is to be installed in the front of the chassis and the MVME714M (if required) may be installed in the front or the rear of the chassis. The MVME188A module requires power from both P1 and P2 for all of its boards (system controller, memory, and main logic with HYPERmodule mezzanine). It may be installed in any three or more adjacent double-height unused card slots, if it is not configured as system controller. (If the MVME188A has had extra memory boards added, it may take 4, 5, or 6 slots.) If the MVME188A is configured as system controller, it must be installed in the left-most three card slots (slots 1, 2, and 3) to correctly initiate the bus-grant daisy-chain and to have proper operation of the IACK-daisy-chain driver.

d. Remove IACK and BG jumpers from header on chassis backplane for the first card slot the MVME188A is installed in. (The memory board(s) and the main logic board (CPU board) use the succeeding card slots and have their IACK pins wired together and each pair of their BG pins wired together. Therefore, the header jumpers on these card slots can be present or absent. Refer to the *MVME188A VMEmodule RISC Microcomputer Support Information* document listed in the *Related Documentation* section in Chapter 1.

e. If there is a blank slot immediately to the right of the MVME188A board set you are ready to install, locate the corresponding slot on the backplane. Install five jumpers across the following pairs of lines for this slot. These connect the Bus Grant (BG) and Interrupt Acknowledge (IACK) daisy-chain signals across this slot.

Jumper These Pins	Connect These Signals
J1 pins A21 to A22	IACKIN* to IACKOUT*
J1 pins B4 to B5	BG0IN* to BG0OUT*
J1 pins B6 to B7	BG1IN* to BG1OUT*
J1 pins B8 to B9	BG2IN* to BG2OUT*
J1 pins B10 to B11	BG3IN* to BG3OUT*

f. If you installed jumpers at a slot in the previous step , place a one-slot wide filler panel in this slot and secure it by tightening its captive screws. Make good contact with the transverse mounting rails to minimize RFI emission.

NOTE

No adjustment or jumpering of the new MVME188A board set is required.

g. Carefully slide the MVME188A module into the card slots by aligning the PWBs in both the top and bottom card guides. Be sure all the boards are seated properly into the P1 and P2 connectors on the backplane. Do not damage or bend connector pins. Fasten modules in chassis with the front panel captive screws provided, making good contact with the transverse mounting rails to minimize RFI emissions.

h. Connect the specified cable(s) to the MVME188A at its J3 and/or J4 female 9-pin front-panel connector(s), or to P2 on the backplane at the system controller board slot (first or leftmost MVME188A slot), to mate with (optional) terminals or other peripherals at the EIA-232-D serial ports. Refer to the *Terminal Connection* section that follows. This cable is not provided with the MVME188A module, and therefore is made or provided by the user. (Motorola recommends using shielded cables for all connections to peripherals to minimize radiation, and can furnish cables for Motorola chassis.) Connect the peripherals to the cable.

i. Install any other required VMEmodules in the system.

j. Replace chassis cover(s) and/or access doors as instructed in the equipment user's manual.

k. Connect power cable to ac power source and turn equipment power ON.

> **CAUTION**
>
> **After removal of system power, allow several seconds before restoring system power. Rapid power on-off cycling may result in system instability and power-up error messages. The actual off-time required varies from system to system, depending on the power supply hold-up time and on system payloads.**

l. Test using EPROM-resident MVME188Diag diagnostics, including at least one pass of SST memory diagnostics. Get out of system mode by entering Menu # 3) Go to System Debugger. This takes the system to the system debugger, MVME188Bug (188Bug); as indicated by the diagnostic mode prompt, 188Diag>. Proceed as follows.

m. If needed, reconfigure the NVRAM (BBRAM) per the new HYPERmodule and/or new memory board configuration. The **ENV** command allows the user to interactively view/configure all 188Bug operational parameters kept in battery backed up RAM (BBRAM), also called non-volatile RAM (NVRAM). The operational parameters are saved in NVRAM and used whenever power is lost.

n. Any time the 188Bug uses a parameter from NVRAM, the NVRAM contents are first tested by checksum to insure the integrity of the NVRAM contents. In the instance of BBRAM checksum failure, certain default values are assumed.

o. The 188Bug operational parameters (which are kept in NVRAM) are not initialized automatically on power up/warm reset. It is up to the 188Bug user to invoke the **ENV** command. Once the **ENV** command is invoked and executed without error, 188Bug default and/or user parameters are loaded into NVRAM along with checksum data. If any of the operational parameters have been modified, these new parameters are not in effect until a reset/powerup condition. Refer to the *MVME188Bug Debugging Package User's Manual* for further information.

Terminal Connection

The two EIA-232-D ports at the front-panel 9-pin female connectors J3 and J4 are configured for DCE (to terminal) operation only. Two 9-pin EIA-232-D cables may be connected to J3 and J4, with the other ends connected to compatible terminals. These cables are not provided with the MVME188A module, and must be made or provided by the user. (Motorola recommends using shielded cables for all connections to peripherals to minimize RFI radiation.) Detailed information on the signals supported is found in Appendix A.

NOTE

The user may change J3 and/or J4 to a DTE (to modem) configuration by providing a "null-modem" cable that switches certain signals, by connecting to the MVME714M Transition Module, and by reprogramming U89, the 68692 DUART on the MVME188A system controller board.

Two suggested cabling arrangements are shown. Suggestion 1 is Figure 2-5. Two 9-pin flat-ribbon cable male connectors are used to connect to J3 and J4. These flat-ribbon cables are, then, crimped to two flat-cable female DB-25 (or DB-9) connectors for two EIA-232-D ports connected per Table 2-4.

Suggestion 2 is Figure 2-6. Here, a single flat-ribbon cable is, instead, connected to both EIA-232-D ports at the rear connector P2 on the MVME188A system controller board, and the other end of this cable has a 64-pin 3M female connector that mates with both the 10-pin male EIA-232-D ports on the MVME714M Transition Module. Refer to Table 2-4. Pin 1 on the cables must align with pin 1

on the connectors. The MVME714M must be installed in the VME chassis (according to the instructions in the *MVME714, MVME714M, and MVME714M-1 2-Channel Serial I/O Distribution Modules User's Manual*), and secured in place with screws at the top and bottom of its front panel. It is double height but approximately half the depth of a standard VMEmodule, and therefore does not connect into the backplane of the VME chassis. The peripherals are then connected to the EIA-232-D DB-25 ports on the MVME714M module. Each EIA-232-D port is defined to interface directly with a standard EIA-232-D connector.

If one terminal is to be used as the 188Bug system console, it must be connected either through the cable to pins 1 through 9 of J4 on the MVME188A, or to the default debug port marked "console" on the MVME714M front panel.

Note that the EIA-232-D ports are also made available at rows A and C of P2 of the MVME188A system controller board at the backplane. Refer to Table 2-4.

Set up the terminal as follows:

- eight bits per character
- one stop bit per character
- parity disabled (no parity)
- baud rate 9600 baud

After power-up, the baud rate of the debug port can be reconfigured by using the Port Format (**PF**) command of the 188Bug debugger.

Table 2-4. Mating Cable Connections for EIA-232-D Ports

MVME188A P2 Backplane Pin No. (SYSCON Bd)	MVME714M 10-Pin Connector Pin No.	MVME188A J3 and J4 DB-9 Pin No. (SYSCON Bd)	DB-25 Pin No.	EIA-232-D Signal Mnemonic and Name
C23 and C28	1	J4-1	8	DCDA Data Carrier Detect
A23 and A28	2	J4-6	6	DSRA Data Set Ready
C24 and C29	3	J4-2	3	RXDA* Receive Data
A24 and A29	4	J4-7	4	RTSA Ready To Send
C25 and C30	5	J4-3	2	TXDA* Transmit Data
A25 and A30	6	J4-8	5	CTSA Clear To Send
C26 and C31	7	J4-4	20	DTRA Data Terminal Ready
none	8	J4-9	1	Not used (tied to DB-9 connector shell)
C27 and C32	9	J4-5	7	GND Signal Return GND
	10			Not used
			9 through 14	Not used
			15	RTXC
			16	Not used
			17	RXC
			18 through 19	Not used
			21 through 25	Not used
C1 and C6	1	J3-1	8	DCDB Data Carrier Detect
A1 and A6	2	J3-6	6	DSRB Data Set Ready
C2 and C7	3	J3-2	3	RXDB* Receive Data
A2 and A7	4	J3-7	4	RTSB Ready To Send
C3 and C8	5	J3-3	2	TXDB* Transmit Data
A3 and A8	6	J3-8	5	CTSB Clear To Send
C4 and C9	7	J3-4	20	DTRB Data Terminal Ready
none	8	J3-9	1	Not used (tied to DB-9 connector shell)
C5 and C10	9	J3-5	7	GND Signal Return GND
	10			Not used
			9 through 14	Not used
			15	RTXC
			16	Not used
			17	RXC
			18 through 19	Not used
			21 through 25	Not used

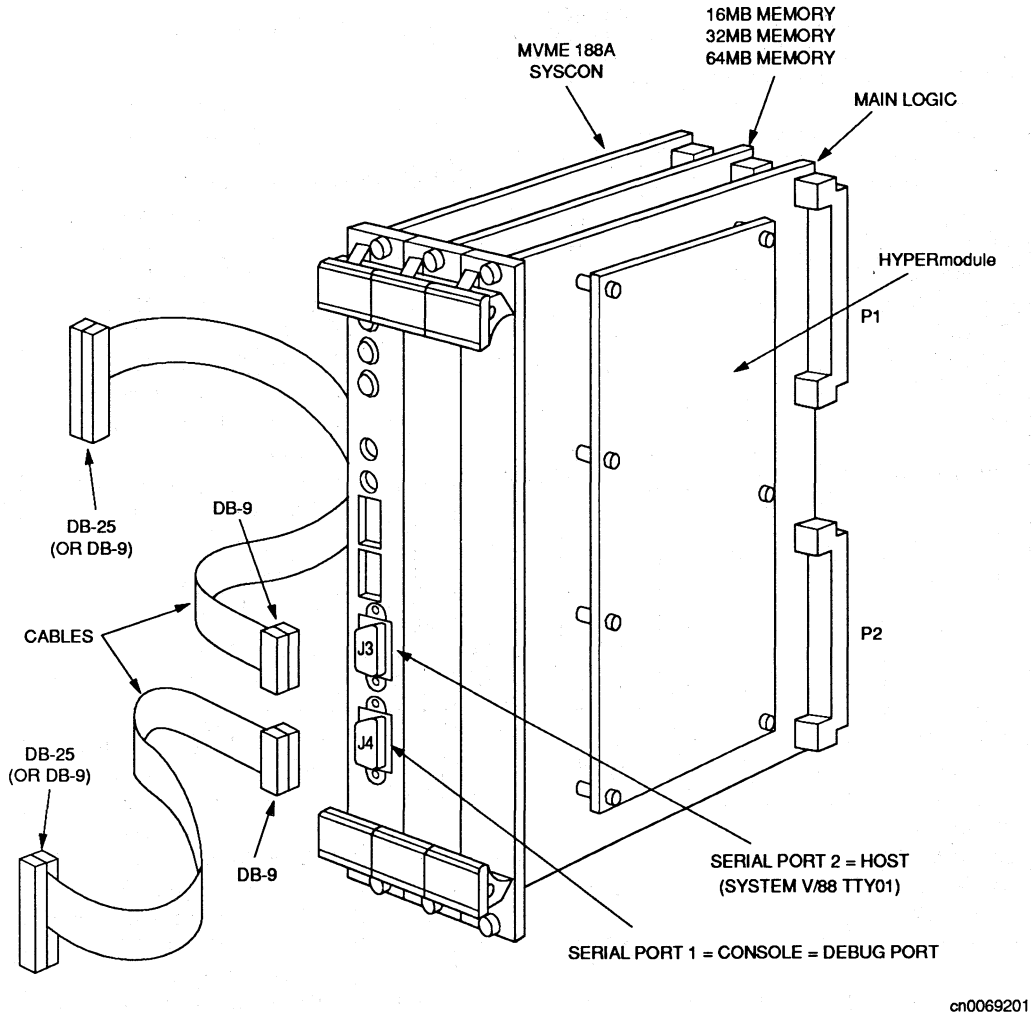


Figure 2-5. Cabling Connections to J3 and J4 EIA-232-D Ports with No Transition Module

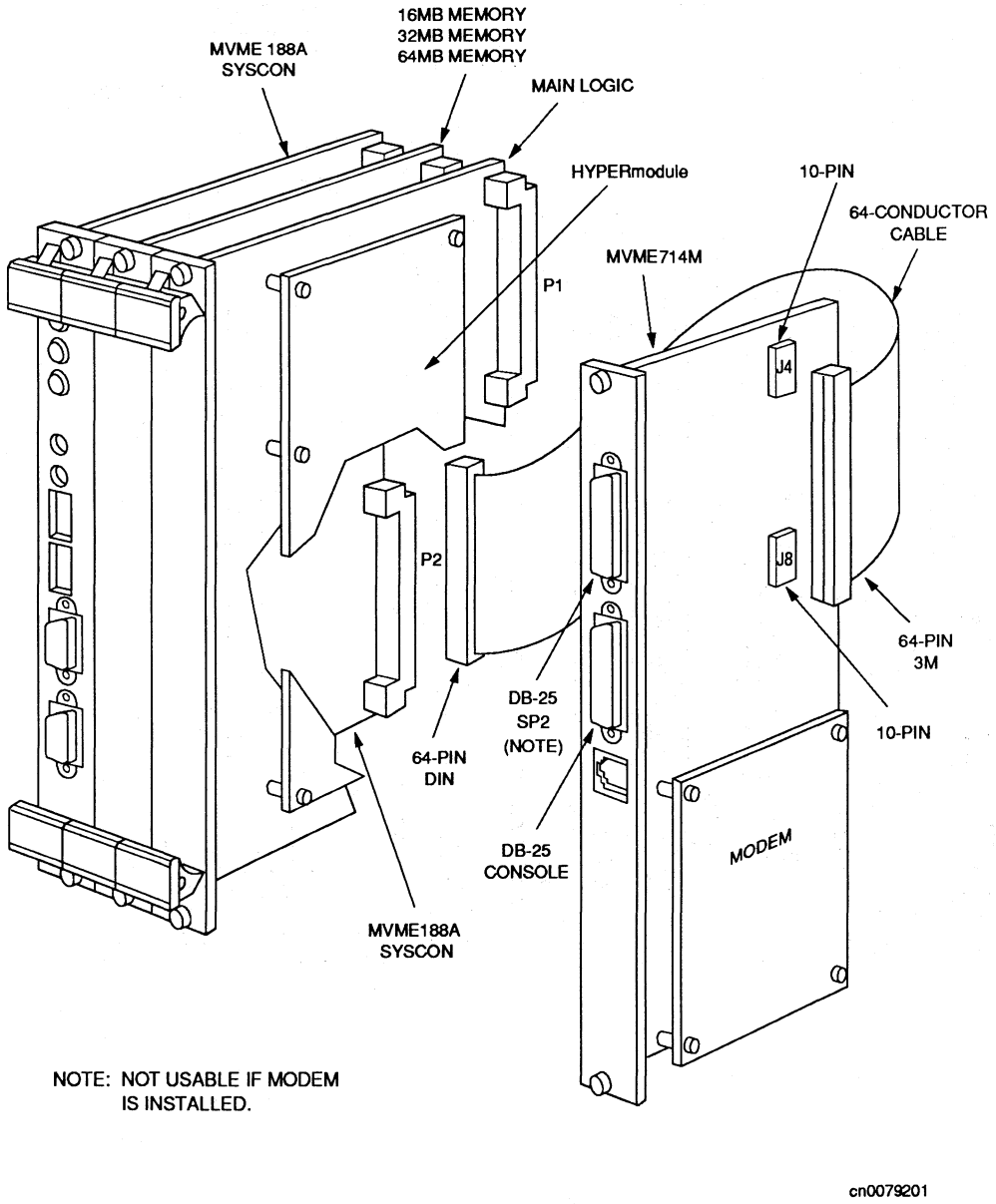


Figure 2-6. Cabling Connections to EIA-232-D Ports at P2 with MVME714M Transition Module

System Considerations

The MVME188A needs to draw power from both P1 and P2 of the VMEbus backplane. P2 is also used for the upper 16 bits of data for 32-bit transfers, and for the upper 8 address lines for extended addressing mode. The MVME188A may not operate properly without all its boards (system controller, memory, and main logic) connected to P1 and P2 of the VMEbus backplane.

Whether the MVME188A operates as a VMEbus master or as a VMEbus slave, it is configured for 32 bits of address and for 32 bits of data (A32/D32). However, it handles A16 or A24 devices in the address ranges indicated in Chapter 3. D8 and/or D16 devices in the system must be handled by the MC88100 software. Refer to the memory maps in Chapter 3.

The MVME188A contains 16MB (or more) of shared onboard DRAM whose base address is software-selectable. Both the onboard processor and offboard VMEbus devices see this local DRAM at physical address $00000000 through $00FFFFFF (for one 16MB parity DRAM board), or $00000000 through $01FFFFFF (for one 32MB ECC board), or $00000000 through $03FFFFFF (for one 64MB parity DRAM board) as programmed by the MVME188Bug firmware. This may be changed, by software, to any other base address. Refer to the *WMAD Register* description in Chapter 4 for details.

Note that the MVME188A contains no parallel ports. To use a parallel device, such as a parallel printer, with the MVME188A, it is necessary to add a module such as the MVME335 Serial and Parallel I/O Module to the system.

If the MVME188A tries to access offboard resources in a non-existent location, and is not system controller, and if the system does not have a global bus time-out, the MVME188A waits forever for the VMEbus cycle to complete. This would cause the system to hang up. There is only one situation in which the system might lack this global bus time-out: the MVME188A is not the system controller, and there is no global bus time-out elsewhere in the system.

Multiple MVME188A modules may be configured into a single VME card cage. Thus, multiple processors may be on one MVME188A, or on several MVME188As, or both. In general, hardware multiprocessor features are supported. Thus, VMEbus accesses to local memory are retried to insure cache coherency. Most MVME188A resources are accessible from the VMEbus.

Other CPUs on the VMEbus can interrupt, disable, communicate with and determine the operational status of the RISC processor(s). One register of the GCSR set includes four bits which function as location monitors to allow one MVME188A processor to broadcast a signal to other MVME188A processors, if any. All eight registers are accessible from any local processor as well as from the VMEbus.

CHAPTER 3
OPERATING INSTRUCTIONS

Introduction

This chapter provides necessary information to use the MVME188A module in a system configuration. This includes controls and indicators, memory map details, and software initialization of the module.

Controls and Indicators

The MVME188A module has ABORT and RESET switches, and FAIL, HALT, and RUN indicators, all located on the front panel of the system controller board of the module (the left-most of the front panels). (See Figure 2-1.) (Refer to Chapter 2 for front-panel configuration switches S3 and S4.) There is also a MEM ERROR indicator located on the front panel of the 32MB ECC memory board, if it is used in your MVME188A. (See Figure 2-3.)

ABORT Switch S1

ABORT is a momentary-action switch used to generate a local interrupt to the processor when it is pressed (not pressed = 1, pressed = 0). The button is debounced and fed to an edge-detector. The edge-detector output (the assertion of the ABORT bit in the ISTATE register, a 0-to-1 transition) is used to generate the ABRT interrupt request. The ABRT interrupt is used to gain access to the 188Bug debugger firmware located in the MVME188A EPROMs. Refer to interrupt details in Chapter 4.

RESET Switch S2

RESET is a momentary-action switch which, when pressed, if MVME188A is VMEbus system controller, generates a local and VMEbus system reset (SRST) that is asserted for 200 ms from the time that the switch is released. If the MVME188A is not the system controller, the RESET switch generates only a local reset (LRST); it is also asserted for 200 ms from the time that the switch is released. (Not pressed = 1, pressed = 0.) The switch is fully debounced in hardware. The *Software Initialization* section later in this chapter has reset details.

FAIL Indicator DS1

The yellow FAIL LED is illuminated whenever the MVME188A is driving the VMEbus SYSFAIL* line, or would drive it if the ISF bit was not set in the Global Control and Status Register (GCSR). This is caused by an asserted DRVSF* bit in the UCSR. (An expired watchdog timer, if it is enabled, causes DRVSF* to be asserted if the WDA[1:0] and EWDTO bits of the UCSR allow a reset to be generated by the watchdog timer.)

HALT Indicator DS2

Either of two reset sources cause illumination of the yellow HALT LED: a local reset (POR* or LRST* is asserted on the MVME188A) or a system reset (SYSRESET* is asserted on the VMEbus). If the R&H bit of the GCSR is set, local reset will keep the HALT LED lit. Because the MC88100 does not have a "halt" state, reset is the only case in which this LED is illuminated.

RUN Indicator DS3

The green RUN LED illuminates on any access of local memory (DRAM) or system controller board resources (ROM, SRAM, etc), that is, whenever any device is Being accessed on the MVME188A local bus (slave bus) (MBSEL[3:0]*, USEL*, or UCCHB* asserted). It is also lit whenever the MVME188A accesses the VMEbus.

MEM ERROR Indicator DS1

The yellow MEM ERROR LED is on the front panel of the 32MB ECC DRAM memory board used in some MVME188As. Uncorrectable multiple-bit errors which are detected by the Am29C660 error detection and correction circuits (EDCs) on the board generate a bus error on the slave bus. Status is set indicating the source of the error and the yellow MEM ERROR LED is illuminated. Clearing the ECC CSR bit 31 or using system reset can switch the LED to the off state.

MVME188A Memory Maps

The address maps are defined in the following sections. The P bus map shows how the CMMUs respond to CPU addresses. The M bus maps show how the MVME188A hardware responds to addresses generated by the CMMUs. The VMEbus slave address map shows how the MVME188A responds to VMEbus addresses generated by another master. Note that there is one case where the MVME188A can "address itself": when an on-board GCSR access is performed via VMEbus A16 space. (Refer to Chapter 4 for GCSR details.)

P Bus Address Maps

The P bus address map depends on the type of HYPERmodule (mezzanine module) installed and the value written into the PCNFA and PCNFB registers on the main logic board (CPU processor). There are six possible configurations of HYPERmodules: 0, 1, 2, 5, 6, and A. (Each configuration can use MC88200 or MC88204 CMMUs, making a total of 12 possible HYPERmodules.) These configurations are defined in the *WHOAMI Register* description in Chapter 4. Refer to that section for information on the different HYPERmodule configurations.

There are no P bus address decoders on the following HYPERmodule configurations:

HM88K-4P128-2 or HM88K-4P512-2	configuration 0	4 CPUs, 8 CMMUs;
HM88K-2P64-2 or HM88K-2P256-2	configuration 5	2 CPUs, 4 CMMUs;
HM88K-1P32-2 or HM88K-1P128-2	configuration A	1 CPU, 2 CMMUs;

because all CMMUs are selected at all times. (This is true any time there is only one CMMU on each P bus.)

The following tables define the P bus map for each of the remaining HYPERmodule configurations (1, 2, and 6) for each value of PCNFA and PCNFB.

Table 3-1. HM88K-2P128-2 configuration 1 2 CPUs, 8 CMMUs
Table 3-2. HM88K-2P512-2 configuration 1 2 CPUs, 8 CMMUs
Table 3-3. HM88K-1P128-2 configuration 2 1 CPU, 8 CMMUs
Table 3-4. HM88K-1P512-2 configuration 2 1 CPU, 8 CMMUs
Table 3-5. HM88K-1P64-2 configuration 6 1 CPU, 4 CMMUs
Table 3-6. HM88K-1P256-2 configuration 6 1 CPU, 4 CMMUs

For example, in an HM88K-1P128-2 configuration 2 HYPERmodule (1 CPU, 8 CMMUs), there are four CMMUs on each P bus (i.e., four on the code side and four on the data side). If PCNFx registers on the main logic board are set to 0, P bus address bits 14 and 12 are decoded to select each CMMU. If the PCNFx are set to 5, however, P bus address bit 14 and the P bus supervisor bit are used to select each CMMU.

Table 3-1. P Bus Address Map, HM88K-2P128-2, Config. 1 (2 CPU, 8 CMMU)

PCNFA (PCNFB) Mode (NOTE)	CPU0 (CPU1) Code CMMU#	CPU0 (CPU1) Data CMMU#	CPU0 (CPU1) Code P Bus Address S 31 30 12	CPU0 (CPU1) Data P Bus Address S 31 30 12	Description
0	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
1	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
2	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
3	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
4	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
5	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
6	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
7	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
8	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU

Table 3-1. P Bus Address Map, HM88K-2P128-2, Config. 1 (2 CPU, 8 CMMU) (cont'd)

PCNFA (PCNFB) Mode (NOTE)	CPU0 (CPU1) Code CMMU#	CPU0 (CPU1) Data CMMU#	CPU0 (CPU1) Code P Bus Address S 31 30 12	CPU0 (CPU1) Data P Bus Address S 31 30 12	Description
9	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
A	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
B	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
C	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
D	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
E	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
F	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM

NOTE: PCNFA maps CPU0 CODE & DATA CMMUs; PCNFB maps CPU1 CODE & DATA CMMUs.

Table 3-2. P Bus Address Map, HM88K-2P512-2, Config. 1 (2 CPU, 8 CMMU)

PCNFA (PCNFB) Mode (NOTE)	CPU0 (CPU1) Code CMMU#	CPU0 (CPU1) Data CMMU#	CPU0 (CPU1) Code P Bus Address S 31 30 14	CPU0 (CPU1) Data P Bus Address S 31 30 14	Description
0	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
1	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
2	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
3	0 (2)		X X X 1	X X X X	shared CMMU
	1 (3)		X X X 0	X X X X	shared CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
4	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
5	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
6	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
7	0 (2)		1 X X X	X X X X	supervisor CMMU
	1 (3)		0 X X X	X X X X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
8	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU

Table 3-2. P Bus Address Map, HM88K-2P512-2, Config. 1 (2 CPU, 8 CMMU) (cont'd)

PCNFA (PCNFB) Mode (NOTE)	CPU0 (CPU1) Code CMMU#	CPU0 (CPU1) Data CMMU#	CPU0 (CPU1) Code P Bus Address S 31 30 14	CPU0 (CPU1) Data P Bus Address S 31 30 14	Description
9	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
A	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
B	0 (2)		0 1 0 X	X X X X	user CMMU
	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		1 1 0 X	X X X X	supervisor SRAM
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM
C	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	X X X 1	shared CMMU
		1 (3)	X X X X	X X X 0	shared CMMU
D	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	1 X X X	supervisor CMMU
		1 (3)	X X X X	0 X X X	user CMMU
E	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	0 1 0 X	user CMMU
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	1 1 0 X	supervisor SRAM
F	0 (2)		X 0 X X	X X X X	shared CMMU
	0 (2)		X 1 1 X	X X X X	shared CMMU
	1 (3)		X 1 0 X	X X X X	shared SRAM
		0 (2)	X X X X	X 0 X X	shared CMMU
		0 (2)	X X X X	X 1 1 X	shared CMMU
		1 (3)	X X X X	X 1 0 X	shared SRAM

NOTE: PCNFA maps CPU0 CODE & DATA CMMUs; PCNFB maps CPU1 CODE & DATA CMMUs.

Table 3-3. P Bus Address Map, HM88K-1P128-2, Config. 2 (1 CPU, 8 CMMU)

PCNFA (PCNFB) Mode (NOTE)	Code (Data) CMMU#	P Bus Address S 31 30 14 12	Description
0	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
1	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
2	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
3	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
4	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
5	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
6	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
7	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
8	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM

Table 3-3. P Bus Address Map, HM88K-1P128-2, Config. 2 (1 CPU, 8 CMMU) (cont'd)

PCNFA (PCNFB) Mode (NOTE)	Code (Data) CMMU#	P Bus Address S 31 30 14 12	Description
9	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
A	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
B	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
C	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
D	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
E	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
F	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM

NOTE: PCNFA maps CODE CMMUs; PCNFB maps DATA CMMUs.

Table 3-4. P Bus Address Map, HM88K-1P512-2, Config. 2 (1 CPU, 8 CMMU)

PCNFA (PCNFB) Mode (NOTE)	Code (Data) CMMU#	P Bus Address S 31 30 16 14	Description
0	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
1	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
2	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
3	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	X X X 0 1	shared CMMU
	3	X X X 0 0	shared CMMU
4	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
5	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
6	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
7	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	1 X X 0 X	supervisor CMMU
	3	0 X X 0 X	user CMMU
8	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM

Table 3-4. P Bus Address Map, HM88K-1P512-2, Config. 2 (1 CPU, 8 CMMU) (cont'd)

PCNFA (PCNFB) Mode (NOTE)	Code (Data) CMMU#	P Bus Address S 31 30 16 14	Description
9	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
A	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
B	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	0 1 0 0 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	1 1 0 1 X	supervisor SRAM
C	0	X X X 1 1	shared CMMU
	1	X X X 1 0	shared CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
D	0	1 X X 1 X	supervisor CMMU
	1	0 X X 1 X	user CMMU
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
E	0	0 1 0 1 X	user CMMU
	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	1 1 0 1 X	supervisor SRAM
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM
F	0	X 0 X 1 X	shared CMMU
	0	X 1 1 1 X	shared CMMU
	1	X 1 0 1 X	shared SRAM
	2	X 0 X 0 X	shared CMMU
	2	X 1 1 0 X	shared CMMU
	3	X 1 0 0 X	shared SRAM

NOTE: PCNFA maps CODE CMMUs; PCNFB maps DATA CMMUs.

Table 3-5. P Bus Address Map, HM88K-1P64-2, Config. 6 (1 CPU, 4 CMMU)

PCNFA Mode (NOTE)	Code CMMU#	Data CMMU#	Code P Bus Address S 31 30 12	Data P Bus Address S 31 30 12	Description
0	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
1	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
2	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
3	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
4	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
5	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
6	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
7	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
8	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU

Table 3-5. P Bus Address Map, HM88K-1P64-2, Config. 6 (1 CPU, 4 CMMU) (cont'd)

PCNFA Mode (NOTE)	Code CMMU#	Data CMMU#	Code P Bus Address S 31 30 12	Data P Bus Address S 31 30 12	Description
9	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
A	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
B	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
C	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
D	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
E	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
F	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM

NOTE: PCNFA maps CODE & DATA CMMUs; PCNFB is not used.

Table 3-6. P Bus Address Map, HM88K-1P256-2, Config. 6 (1 CPU, 4 CMMU)

PCNFA Mode (NOTE)	Code CMMU#	Data CMMU#	Code P Bus Address S 31 30 14	Data P Bus Address S 31 30 14	Description
0	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
1	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
2	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
3	0		X X X 1	X X X X	shared CMMU
	1		X X X 0	X X X X	shared CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
4	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
5	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
6	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
7	0		1 X X X	X X X X	supervisor CMMU
	1		0 X X X	X X X X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
8	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU

Table 3-6. P Bus Address Map, HM88K-1P256-2, Config. 6 (1 CPU, 4 CMMU) (cont'd)

PCNFA Mode (NOTE)	Code CMMU#	Data CMMU#	Code P Bus Address S 31 30 14	Data P Bus Address S 31 30 14	Description
9	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
A	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
B	0		0 1 0 X	X X X X	user CMMU
	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		1 1 0 X	X X X X	supervisor SRAM
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM
C	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	X X X 1	shared CMMU
		1	X X X X	X X X 0	shared CMMU
D	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	1 X X X	supervisor CMMU
		1	X X X X	0 X X X	user CMMU
E	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	0 1 0 X	user CMMU
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	1 1 0 X	supervisor SRAM
F	0		X 0 X X	X X X X	shared CMMU
	0		X 1 1 X	X X X X	shared CMMU
	1		X 1 0 X	X X X X	shared SRAM
		0	X X X X	X 0 X X	shared CMMU
		0	X X X X	X 1 1 X	shared CMMU
		1	X X X X	X 1 0 X	shared SRAM

NOTE: PCNFA maps CODE & DATA CMMUs; PCNFB is not used.

M Bus Address Map

The M bus address map is defined in the following sections. Three usable address spaces are available to M bus masters: local DRAM space, VMEbus space, and utility space. In addition, a fourth address space is available which always returns an error. At power-up time (or any time the MADV bit of the CPU Control/Status Register (CCSR) is = 0), the only access available to M bus masters is utility space. When MADV = 0, utility space is a 4MB block that is repeated throughout the entire 4Gb M bus address space (1024 times). All of the MVME188A registers and the MC88200 and/or MC88204 registers (as well as the VMEbus short I/O space) are mapped to addresses within utility space.

Refer to Chapter 4 for information on the *M Bus Address Decoder* and the registers.

DRAM Address Map

The memory board supplied as part of the MVME188A is a 16MB DRAM, 64MB DRAM, or 32MB ECC DRAM board. Note that there are restrictions as to how the DRAM addresses are mapped; some of the M bus address bits are used to select both the 4MB page number and the DRAM address at the same time.

Refer to Chapter 4 for information on the *M Bus Address Decoder* for an explanation of how DRAM address mapping is handled. The decoding is summarized in Table 3-7.

Table 3-7. DRAM Space Address Map

Device Name	M Bus Address Bits							
	3322 1098	2222 7654	2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	0000 3210
16MB DRAM	PPPP	PPPP	SSAA	AAAA	AAAA	AAAA	AAAA	AAAA
64MB DRAM	PPPP	PPSS	SSAA	AAAA	AAAA	AAAA	AAAA	AAAA
32MB ECC DRAM	PPPP	PPPS	SSAA	AAAA	AAAA	AAAA	AAAA	AAAA

NOTES: P = page 4MB page number (upper bits)
S = section 4MB page number (lower bits) & address within DRAM board
A = address DRAM address within 4MB page
32MB DRAM boards also have CSR and Diagnostic Registers at addresses $E0C00000 & $E0C00004 (board 0), $E0800000 & $E0800004 (board1), $E0400000 & $E0400004 (board 2), and $E0000000 & $E0000004 (board 3) as initialized. Refer to Chapter 2 for switch settings.

VMEbus Master Address Map

The VMEbus A16, A24, and A32 spaces can be accessed by the MVME188A from the M bus. The VMEbus A16 space is accessed via the utility address space (described in a following section). The A24 and A32 spaces are separately mappable in 4MB pages via the M bus address decoder. Note that there are restrictions as to how the VMEbus addresses are mapped; some of the M bus address bits are used to select both the 4MB page number and the VMEbus address at the same time.

Refer to Chapter 4 for information on the *M Bus Address Decoder* for an explanation of how VMEbus master address mapping is handled. The decoding is summarized in Table 3-8.

Table 3-8. VMEbus Master Space Address Map

Device Name	M Bus Address Bits							
	3322 1098	2222 7654	2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	0000 3210
VMEbus A24 SPACE	PPPP	PPPP	SSAA	AAAA	AAAA	AAAA	AAAA	AAAA
VMEbus A32 SPACE	SSSS	SSSS	SSAA	AAAA	AAAA	AAAA	AAAA	AAAA

NOTES: P = page — 4MB page number (upper bits)
S = section — 4MB page number (lower bits) & address within VMEbus space
A = address — VMEbus address within 4MB page
AM[5:0] are derived from EXTAM register and M bus address decoder contents for the page addressed. Chapter 4 has register description.

Utility Address Map

The utility address space is a 4MB page that contains all of the MVME188A registers and MC88200 and/or MC88204 registers (as well as the VMEbus master short I/O space).

Refer to Chapter 4 for information on the *M Bus Address Decoder* for an explanation of how utility space address mapping is handled and for register descriptions. The utility space address map is shown in Table 3-9. Utility space address assignments in hexadecimal are shown in Table 3-10.

Table 3-9. Utility Space Address Map

Device Name	M Bus Address Bits 3322 1098	2222 7654	2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	0000 3210
EPROM	PPPP	PPPP	PP0X	XAAA	AAAA	AAAA	AAAA	AA..
SRAM	PPPP	PPPP	PP10	XXXA	AAAA	AAAA	AAAA	AA..
MC88200/204 N, REGISTER R	1111	1111	1111	0NNN	NNNN	RRRR	RRRR	RR..
UTILITY REGISTERS	PPPP	PPPP	PP11	1000	0XXX	XXXX	XXXX	XX..
BBSRAM/RTC	PPPP	PPPP	PP11	1000	000A	AAAA	AAAA	AA..
DUART REGISTERS	PPPP	PPPP	PP11	1000	0010	XXXX	XXRR	RR..
CIO REGISTERS	PPPP	PPPP	PP11	1000	0011	XXXX	XXXX	RR..
IEN0	PPPP	PPPP	PP11	1000	0100	XXXX	0XXX	X1..
IEN1	PPPP	PPPP	PP11	1000	0100	XXXX	0XXX	1X..
IEN2	PPPP	PPPP	PP11	1000	0100	XXXX	0XX1	XX..
IEN3	PPPP	PPPP	PP11	1000	0100	XXXX	0X1X	XX..
IST	PPPP	PPPP	PP11	1000	0100	XXXX	01XX	XX..
SETSWI	PPPP	PPPP	PP11	1000	0100	XXXX	1XXX	00..
CLRSWI	PPPP	PPPP	PP11	1000	0100	XXXX	1XXX	01..
ISTATE	PPPP	PPPP	PP11	1000	0100	XXXX	1XXX	10..
CLRINT	PPPP	PPPP	PP11	1000	0100	XXXX	1XXX	11..
VIRQLV	PPPP	PPPP	PP11	1000	0101	XXXX	XX00	00..
VIACK1V	PPPP	PPPP	PP11	1000	0101	XXXX	XX00	01..
VIACK2V	PPPP	PPPP	PP11	1000	0101	XXXX	XX00	10..
VIACK3V	PPPP	PPPP	PP11	1000	0101	XXXX	XX00	11..
VIACK4V	PPPP	PPPP	PP11	1000	0101	XXXX	XX01	00..
VIACK5V	PPPP	PPPP	PP11	1000	0101	XXXX	XX01	01..
VIACK6V	PPPP	PPPP	PP11	1000	0101	XXXX	XX01	10..
VIACK7V	PPPP	PPPP	PP11	1000	0101	XXXX	XX01	11..
VIRQV	PPPP	PPPP	PP11	1000	0101	XXXX	XX1X	XX..
GCSR BLOCK	PPPP	PPPP	PP11	1000	0110	XXXX	XXXX	RRR.
UCSR	PPPP	PPPP	PP11	1000	0111	XXXX	XXXX	00..
BASAD	PPPP	PPPP	PP11	1000	0111	XXXX	XXXX	01..
GLBRES	PPPP	PPPP	PP11	1000	0111	XXXX	XXXX	11..
CPU REGISTERS	PPPP	PPPP	PP11	1000	1XXX	XXXX	XXXX	XX..
CCSR	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX00	00..
ERROR	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX00	01..
PCNFA	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX00	10..
PCNFB	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX00	11..
EXTAD	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX01	00..
EXTAM	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX01	01..
WHOAMI	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX01	10..
WMAD	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX10	00..
RMAD	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX10	01..
WVAD	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX10	10..
RVAD	PPPP	PPPP	PP11	1000	1XXX	XXXX	XX10	11..

Table 3-9. Utility Space Address Map (cont'd)

Device Name	M Bus Address Bits 3322 2222 2222 1111 1111 1100 0000 0000 1098 7654 3210 9876 5432 1098 7654 3210
causes error	PPPP PPPP PP11 1001 XXXX XXXX XXXX XX..
causes error	PPPP PPPP PP11 101X XXXX XXXX XXXX XX..
causes error	PPPP PPPP PP11 110X XXXX XXXX XXXX XX..
causes error	PPPP PPPP PP11 1110 XXXX XXXX XXXX XX..
VMEbus A16 SPACE	PPPP PPPP PP11 1111 AAAA AAAA AAAA AAAA

NOTES: P = page number (MPN[9:0]) with contents MDS[2:0] = 111 in M bus address decoder. Whenever MADV = 0, MDS[2:0] is all ones (utility space) for every M bus access, regardless of address decoder contents.
Page $3FF (base address $FFC00000) MUST always be mapped as utility space.
A = address
N = encoded CMMU number. (Refer to Table 3-11.)
R = register number. Chapter 4 describes system controller board and main logic board registers.
X = not decoded
. = not used

Table 3-10. Utility Space Address Assignments (Hex)

Device Name	M Bus Address
EPROM (512KB)	$FFC00000 through $FFC7FFFF
SRAM (128KB)	$FFE00000 through $FFE1FFFF
MC88200/204 N, REGISTER R	$FFF00000 through $FFF7FFFF
BBSRAM/RTC	$FFF80000 through $FFF81FFF
DUART REGISTERS	$FFF82000 through $FFF8203F
CIO REGISTERS	$FFF83000 through $FFF8300F
IEN0	$FFF84004
IEN1	$FFF84008
IEN2	$FFF84010
IEN3	$FFF84020
IEN (all)	$FFF8403C
IST	$FFF84040
SETSWI	$FFF84080
CLRSWI	$FFF84084
ISTATE	$FFF84088
CLRINT	$FFF8408C

Table 3-10. Utility Space Address Assignments (Hex) (cont'd)

Device Name	M Bus Address
VIRQLV	$FFF85000
VIACK1V	$FFF85004
VIACK2V	$FFF85008
VIACK3V	$FFF8500C
VIACK4V	$FFF85010
VIACK5V	$FFF85014
VIACK6V	$FFF85018
VIACK7V	$FFF8501C
VIRQV	$FFF85020
GCSR BLOCK	$FFF86000 through $FFF8600F
UCSR	$FFF87000
BASAD	$FFF87004
GLBRES	$FFF8700C
CCSR	$FFF88000
ERROR	$FFF88004
PCNFA	$FFF88008
PCNFB	$FFF8800C
EXTAD	$FFF88010
EXTAM	$FFF88014
WHOAMI	$FFF88018
WMAD	$FFF88020
RMAD	$FFF88024
WVAD	$FFF88028
RVAD	$FFF8802C
causes error	$FFF90000 through $FFFEFFFF
VMEbus A16 SPACE	$FFFF0000 through $FFFFFFFF

NOTES:

Utility address space = $FFC00000 through $FFFFFFFF when base address is $FFC00000. This puts VMEbus A16 space at the desired base address of $FFFF0000.

$FFC00000 through $FFFFFFFF MUST always be mapped as utility space.

MC88200 and/or MC88204 registers only appear in the page whose base address is $FFC00000. (Refer to Table 3-11.)

MC88200 and/or MC88204 CMMU Registers

The HYPERmodule uses either MC88200 CMMUs (16KB) or MC88204 CMMUs (64KB). The addresses of the MC88200 and/or MC88204 CMMUs internal registers are not mappable by the M bus address decoder. The upper address bits (i.e., register base address) are decoded internally by each CMMU. In order to access these registers, page $3FF (base address $FFC00000) must be mapped as utility space. Utility space can also be mapped elsewhere, but the CMMU registers will not be accessible (reads will return $FFFFFFFF and writes will be ignored).

The default power-up CMMU register base address assignments are shown in Table 3-11. Note that all CMMUs may or may not be present, depending on the HYPERmodule configuration installed.

Note that this table represents the default mapping; the address bits shown (19 through 12) can be remapped by writing to the CMMU ID register (IDR). The remaining upper address bits (31 through 20) are hard-decoded inside the CMMUs and cannot be remapped. Note that bit 31 of the IDR should **NEVER** be set to 1; this would map the CMMU registers on top of the other utility space registers.

> **CAUTION**
>
> **Use caution when changing the CMMU map; if two CMMUs respond to the same address, permanent damage to the CMMUs may result. If a CMMU is mapped to any address with bit 19 set, permanent damage to the CMMUs may also result.**

Table 3-11. MC88200 and/or MC88204 CMMU Register Address Assignments

CMMU Number	M Bus Address		
	1111 9876	1111 5432	Hex Address
code CMMU 0	0111	1110	$FFF7ERRR
code CMMU 1	0111	1101	$FFF7DRRR
code CMMU 2	0111	1011	$FFF7BRRR
code CMMU 3	0111	0111	$FFF77RRR
data CMMU 0	0110	1111	$FFF6FRRR
data CMMU 1	0101	1111	$FFF5FRRR
data CMMU 2	0011	1111	$FFF3FRRR
data CMMU 3	0111	1111	$FFF7FRRR

NOTE: RRR = MC88200/204 register address (offset from base address).

Real-Time Clock Registers Address Map

The real-time clock registers are byte-wide registers mapped to 32-bit addresses. The data is mapped to the least-significant byte (bits 7 through 0). Byte or half-word accesses are permitted, but (for consistency) not recommended. These registers occupy the last eight locations of the 2048 reserved for the Battery-Backed-up RAM/Real-Time Clock (BBRAM/RTC). Refer to the MK48T02 data sheet listed in the *Related Documentation* section in Chapter 1 for explanation of the bit assignments of the register functions listed in Table 3-12.

Table 3-12. Real-Time Clock Register Addresses (Hex)

Register	M Bus Word Address
control	$FFF81FE0
seconds	$FFF81FE4
minutes	$FFF81FE8
hour	$FFF81FEC
day	$FFF81FF0
date	$FFF81FF4
month	$FFF81FF8
year	$FFF81FFC

NOTE: These addresses assume a utility base address of $FFC00000.

DUART Registers Address Map

The 68692 DUART registers are byte-wide registers mapped to 32-bit addresses. The data is mapped to the least-significant byte (bits 7 through 0). Byte or half-word accesses are permitted, but (for consistency) not recommended. DUART registers assigned addresses are given in Table 3-13. Refer to register function descriptions and bit assignments in the 68692 data sheet in the *Related Documentation* section in Chapter 1. The 68692 is a CMOS version of the MC68681.

Table 3-13. DUART Register Addresses (Hex)

Read Register	Write Register	M Bus Word Address
mode A (MR1A, MR2A)	mode A (MR1A, MR2A)	$FFF82000
status A (SRA)	clock select A (CSRA)	$FFF82004
do not access	command A (CRA)	$FFF82008
receive buffer A (RBA)	transmit buffer A (TBA)	$FFF8200C
input port change (IPCR)	auxiliary control (ACR)	$FFF82010
interrupt status (ISR)	interrupt mask (IMR)	$FFF82014
count upper (CUR)	counter/timer upper (CTUR)	$FFF82018
count lower (CLR)	counter/timer lower (CTLR)	$FFF8201C
mode B (MR1B, MR2B)	mode B (MR1B, MR2B)	$FFF82020
status B (SRB)	clock select B (CSRB)	$FFF82024
do not access	command B (CRB)	$FFF82028
receive buffer B (RBB)	transmit buffer B (TBB)	$FFF8202C
interrupt vector (IVR)	interrupt vector (IVR)	$FFF82030
input port (unlatched)	output port configuration (OPCR)	$FFF82034
start counter command	output port bit set (OPR)	$FFF82038
stop counter command	output port bit reset (OPR)	$FFF8203C

NOTE: These addresses assume a utility base address of $FFC00000.

CIO (Counter-Timer) Registers Address Map

The Z8536 Counter/Timer and Parallel I/O Unit (CIO) registers are byte-wide registers mapped to 32-bit addresses. The data is mapped to the least-significant byte (bits 7 through 0). Byte or half-word accesses are permitted, but (for consistency) not recommended. The assigned addresses of the CIO registers are shown in Table 3-14. The Z8536 is a non-multiplexed bus version of the Z8036. The register bit assignments are the same, but accessing the registers requires a two-step procedure rather than the simple read/write of the Z8036. For a description of the bit assignments and register functions, refer to the Z8536 data sheet listed in the *Related Documentation* section in Chapter 1.

Because Z8536 register access requires two bus cycles, care must be exercised in the MVME188A multi-CPU environment. If two or more CPUs attempt simultaneous CIO access, the register reads and/or writes will be interleaved. This will most likely produce an invalid access sequence resulting in erroneous data being written to or read from the CIO.

Table 3-14. CIO (Counter-Timer) Register Addresses (Hex)

Register	M Bus Word Address
port C data register	$FFF83000
port B data register	$FFF83004
port A data register	$FFF83008
control registers	$FFF8300C

NOTE: These addresses assume a utility base address of $FFC00000.

VMEbus Slave Address Maps

The VMEbus slave map consists of two parts: the A16 (short I/O) map and the A24/A32 map. The GCSR and location monitors are the only resources that reside in the A16 map. There is a basic difference in mapping implementation between the A16 and A24/A32 spaces. VMEbus A24 and A32 addresses are mapped to M bus addresses by the software-programmable VMEbus address decoder; VMEbus A16 addresses are mapped directly to the GCSR and location monitor addresses by switches. (Refer to the *Hardware Configuration* section in Chapter 2 for details.)

Note that because the A16 map is configured by the hardware, the GCSR is immediately accessible from the VMEbus after reset. The A24 and A32 address decoders must be configured by the local CPU before any VMEbus A24 or A32 slave accesses can occur.

VMEbus A16 Slave Address Map

The GCSR base address is determined by switches S3-5 through S3-8, S4-1 through S4-4 (GRPAD7 through GRPAD0, respectively), and S4-5 through S4-8 (BDAD3 through BDAD0, respectively). The location monitor base address is determined by GRPAD7 through GRPAD0 (the same switches used to set the GCSR base address).

Each switch is turned on to set its corresponding address bit to a 0. Off represents a 1 address bit. Moving the switch lever to the left (closer to the PWB) turns the switch on; moving it to the right (away from the PWB) turns the switch off. (Refer to the *Hardware Configuration* section in Chapter 2 for details.)

GCSR Base Address Selection

The GCSR base address is set by the GRPAD (group address) and BDAD (board address within that group) switches. Note that setting BDAD = %1111 is not recommended. Refer to Table 3-15.

Table 3-15. VMEbus A16 Slave Space Address Map

Device Name	VMEbus A16 Address 15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0	Hex Address
GCSR	G G G G	G G G G	B B B B	R R R 1	$GGBR
Location monitor 0	G G G G	G G G G	1 1 1 1	X 0 0 X	$GGF0
Location monitor 1	G G G G	G G G G	1 1 1 1	X 0 1 X	$GGF2
Location monitor 2	G G G G	G G G G	1 1 1 1	X 1 0 X	$GGF4
Location monitor 3	G G G G	G G G G	1 1 1 1	X 1 1 X	$GGF6

NOTES: G = GRPAD, B = BDAD, R = GCSR offset.
Setting BDAD = %1111 is not recommended.

Location Monitor Address Mapping

There are four location monitors on the MVME188A. Each of them can cause an interrupt to any local CPU. The base address of the location monitors is determined by the setting of the GRPAD switches. Any access to a location monitor location causes the appropriate bit (LM0* through LM3*) to be asserted in the GCSR. The assertion of any of the location monitor bits in the GCSR causes an LMI interrupt to all processors that have LMI unmasked. Note that BDAD = %1111 is reserved for location monitor addresses; no MVME188A boards should be set up with this BDAD. (It is impossible to access any GCSR registers on an MVME188A with BDAD set to %1111 without setting a location monitor bit.) All MVME188A boards in the system that have the same GRPAD respond to the same location monitor addresses.

The MVME188A does not generate a DTACK* to the VMEbus on location monitor accesses. Typically, a BERR* (from the VMEbus time-out timer) is used to end the cycle, unless the bus master generates a DTACK* for location monitor accesses.

The MVME188A location monitors respond to D08 (EO) or D16 accesses (i.e., LWORD* must be negated). The GCSR registers respond to D08 (O) or D16 accesses.

VMEbus A24/A32 Slave Address Map

The VMEbus slave address space is mapped in 4MB pages in a manner similar to M bus address space. Only A32 and A24 spaces are mappable via software; the A16 base address is fixed by DIP switch settings. (Refer to the previous section, and to the *Hardware Preparation* section in Chapter 2.) The A32 and A24 spaces are individually mappable (in 4MB pages) by the VMEbus SRAM address decoder. The WVAD register is used to load the decoder, and the RVAD register is used to read back the decoder contents. (Refer to Chapter 4 for the description of the *VMEbus Slave Address Decoder* and all the main logic board (CPU board) registers.)

For VMEbus A32 slave accesses, VMEbus address bits A31 through A22 select a page, and AM5 through AM4 determine which one of the decoder maps is used (A32 or A24). If the selected page in the selected map is configured to be in VMEbus A32 space, and AM3 = 1, all 32 bits of the VMEbus address are used to address an M bus device.

VMEbus A24 slave accesses are handled in a similar manner. VMEbus address bits A31 through A24 supply the top eight bits of the page address, and VMEbus address bits A23 through A22 supply the remaining two. Note that because VMEbus address bits A31 through A24 may or may not be driven during an A24 access, all 256 locations in the SRAM that have the same lower two address bits must be written with the same value. Effectively, the SRAM decoder has only four locations (one for each 4MB page contained within VMEbus A24 space). The decoder map (A24 or A32) is selected in the same manner as above (decoded from AM5 through AM4). If the selected page is configured to be in VMEbus A24 space, and AM3 = 1, the EXTAD register (top eight bits) and VMEbus address (remaining address bits) form an address used to access an M bus device.

As long as the VADV bit in the CCSR is set, the VMEbus is constantly monitored for cycles in which the address and address modifiers (A24 or A32 only) select a matching mapped page. When this occurs, a request is generated to the M bus arbiter for a VMEbus cycle. Note that VMEbus master space is not accessible from VMEbus slave space; it is possible (but not recommended) to set up the VMEbus and M bus address decoders in this configuration. Any attempt to do this will result in a bus error being returned on the VMEbus cycle.

Note that the MC88200 and/or MC88204 registers are NOT accessible from the VMEbus. Any reads attempted from these CMMU registers return the value $FFFFFFFF. Any writes to the registers write the value $FFFFFFFF.

The way the M bus address is generated on VMEbus A32 and A24 slave accesses to the MVME188A is shown in Table 3-16.

CAUTION

MC88200 and/or MC88204 registers should not be accessed from the VMEbus.

Table 3-16. VMEbus A32/A24 Slave Access M Bus Address Generation

Device Name	M Bus Address Bits							
	3322 1098	2222 7654	2222 3210	1111 9876	1111 5432	1100 1098	0000 7654	0000 3210
VMEbus A24 SPACE	EEEE	EEEE	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA
VMEbus A32 SPACE	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAAA

NOTES: E = extended address: contents of EXTAD register. Refer to Chapter 4.
A = address: VMEbus address bits. The top ten bits (A31 through A22) select the page number for address decoding.
AM5 through AM4 select the VMEbus address decode map from SRAM (there are 1024 pages in the decoder for A32 space, and 1024 pages for A24 space).

Software Initialization

Most functions that have been done with switches or jumpers on other modules are done by setting control registers on the MVME188A. At powerup or reset, the EPROMs that contain the 188Bug debugging package set up the default values of many of these registers. Address decoding is done by SRAMs that are also initially set up by the EPROMs.

Specific programming details may be determined by study of the *MC88100 RISC Microprocessor User's Manual*, the *MC88200 Cache/Memory Management Unit User's Manual*, and the *MC88204 64K-Byte Cache/Memory Management Unit (CMMU)* data sheet. Then check the details of all the MVME188A on-board registers as given in Chapter 4 of this manual.

Multi-CPU Programming Considerations

The multiple-CPU architecture of the HYPERmodule requires some special care in the use of some of the MVME188A registers. Good programming practice dictates that only one CPU at a time have control of the control registers.

Of particular note are:

> registers that modify the address map (WMAD, WVAD, CCSR bits MADV/VADV); registers that require two cycles to access (CIO registers); and VMEbus interrupt request registers (VIRQLV, VIRQV).

The GLBRES register should also be used with caution, because its action can be aborted by a local reset from the watchdog timer (perhaps controlled by a different CPU).

The CMMU page tables should NEVER be located out on the VMEbus. The CMMU does not support retry protocol on the last cycle of a table-walk operation. If the CMMU page tables map to a VMEbus address and another master tries to access an MVME188A resource in between the last two cycles of a table-walk, a fatal CMMU error will occur.

Local Reset Operation

Local reset (LRST) is a subset of system reset (SRST). Local reset can be generated three ways: expiration of the watchdog timer, pressing the front panel RESET switch (if the SCON switch is off), or by asserting the R&H bit in the GCSR.

A local reset removes any pending GLBRES. A GLBRES is made pending by any write to the GLBRES register. It may take up to 32 usec for the resulting system reset to occur. During this time, the assertion of local reset from any of its three sources clears the pending GLBRES (and thus prevents a system reset from being generated). This could be a concern in a multi-CPU or multi-MVME188A system if the watchdog and/or accesses to the GCSR are not carefully controlled. Note that the front-panel RESET pushbutton switch could also prevent a system reset from being generated (if the SCON switch is off) if it was pressed at just the right time.

Any VMEbus access to the MVME188A while it is in the reset state will return a bus error.

Summary of Register Values After Reset

A summary of how each MVME188A register is affected by local reset (LRST), system reset (SRST), and power-on reset (POR) is given in Table 3-17. All of this information is contained in Chapter 4 in this manual, but is summarized here.

Table 3-17. MVME188A Register Values After Reset

Device Name	Value On LRST (NOTE 1)	Value On SRST	Value On POR
RTC	X	X	X
DUART REGISTERS	R	R	R
CIO REGISTERS	X	R	R
IEN0	$0	$0	$0
IEN1	$0	$0	$0
IEN2	$0	$0	$0
IEN3	$0	$0	$0
IST	NOTE 2	$0	$0
SETSWI	X	X	X
CLRSWI	X	X	X
ISTATE	X	X	X
CLRINT	X	X	X
VIRQLV	$0	$0	$0
VIACK1V	X	X	X
VIACK2V	X	X	X
VIACK3V	X	X	X
VIACK4V	X	X	X
VIACK5V	X	X	X
VIACK6V	X	X	X
VIACK7V	X	X	X
VIRQV	$0	$0	$0
GCSR GLOBAL0	X	$FF	$FF
GCSR GLOBAL1	NOTE 3	$10 IF NOT SCON $50 IF SCON	$10 IF NOT SCON $50 IF SCON
GCSR BRDID	X	$0	$0
GCSR GPCSR0	X	$FF	$FF
GCSR GPCSR1	X	$0	$0
GCSR GPCSR2	X	$0	$0

Table 3-17. MVME188A Register Values After Reset (cont'd)

Device Name	Value On LRST	Value On SRST	Value On POR
GCSR GPCSR3	X	$0	$0
GCSR GPCSR4	X	$0	$0
UCSR	NOTE 4	$461B (NOTE 5)	$061B
BASAD	X	X	X
GLBRES	X	X	X
CCSR	X	$0	$0
ERROR	X	$0	$0
PCNFA	X	$0	$0
PCNFB	X	$0	$0
EXTAD	X	X	X
EXTAM	X	X	X
WHOAMI	X	X	X
WMAD	X	X	X
RMAD	X	X	X
WVAD	X	X	X
RVAD	X	X	X

NOTES: (1) Values listed for LRST are true only if the zero-ohm resistor R61 is removed from the SYSCON board in the MVME188A. If R61 is left in place, as shipped, the values for LRST are the same as those for SRST.

(2) Interrupt sources cleared by LRST: ABRT, ACF, DTI, SF, SWI0 through SWI7, DI.

(3) In GCSR GLOBAL1, BRDFAIL (bit 4) is the only bit affected; it is set to 1 by LRST.

(4) In UCSR, only DRVSF* (bit 13), BRIRQO (bit 12), EARBTO (bit 5), and EWDTO (bit 2) are cleared by LRST; all others are unaffected.

(5) PWRUP* (bit 14) is unaffected by SRST. This value assumes that PWRUP* has been set by software prior to SRST.

X = unaffected

R = reset is input to LSI chip; consult appropriate data sheet for reset effects. Refer to *Related Documentation* in Chapter 1.

CHAPTER 4
FUNCTIONAL DESCRIPTION

Introduction

This chapter provides the overall block diagram level description for the MVME188A module. The general description provides an overview of the module, followed by a detailed description of each board of the module. The simplified block diagram of the MVME188A is shown in Figure 4-1.

MVME188A General Functional Description

The MVME188A family of RISC microcomputers offers a flexible new means of achieving next generation increases in computing power for technical and commercial computer system OEM applications. Specific applications encompass large multi-user compute servers, network/communications controllers, large file and database servers, as well as image processing, visualization, real-time simulation, and artificial intelligence. Up to 132 MIPS are provided by the MVME188A concept which combines in a single unit three standard single-wide, double-high VMEmodules: the system controller board, one or more memory boards, and the main logic board. The three boards are interconnected through a very high-speed local bus (also called the slave bus, or S bus). Because the S bus is a modified form of the M bus, descriptions of registers and other circuits sometimes describe it as the M bus.

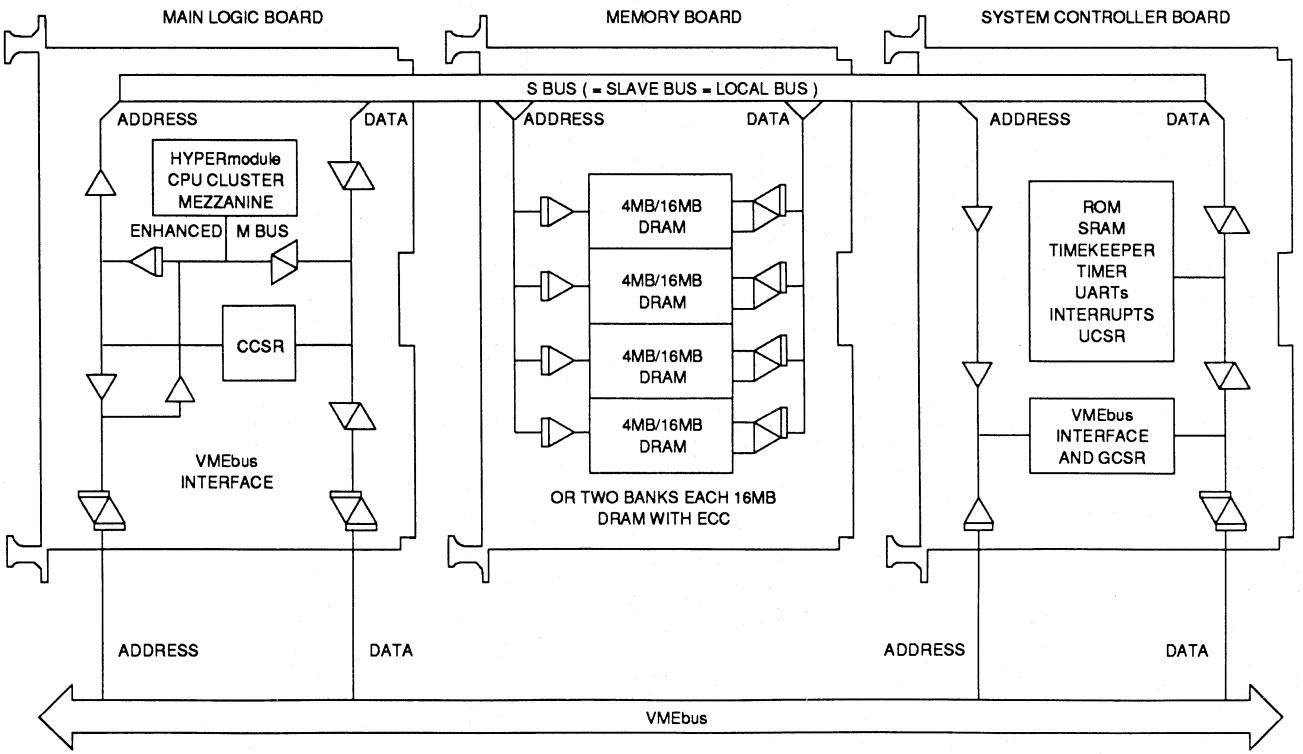


Figure 4-1. MVME188A Block Diagram

bd0529201

All-over MVME188A timing is as follows:

Type Of Access	Read	Write	Notes
Onboard CPU to CMMU registers	3+N cycles	3+N cycles	1
CMMU to onboard ROM/PROM/EPROM/EEPROM	3+X cycles	3+X cycles	2
CMMU to onboard DRAM	6 cycles	3 cycles	3
Onboard CMMU to VMEbus	6+Z cycles	6+Z cycles	4
VMEbus to onboard DRAM	6+Y cycles	6+Y cycles	5

NOTES: (1) N = wait states generated by CMMU.
(2) X = wait states generated by selected slave.
(3) Includes wait states listed in Memory Board description.
(4) Z = bus access + slave response. Bus access = (BRX* asserted to BBSY* asserted)/(CPU clock cycle time). Slave response = (DS0*/DS1* to DTACK* time)/(CPU clock cycle time). Z should be rounded up to the nearest integer.
(5) Y = M bus access time.

The system controller board contains a VMEbus A16 slave interface, ROM, static RAM, UARTs, timing elements, and local and global control/status registers.

On the memory board, a 16MB DRAM (or 64MB DRAM or 32MB ECC DRAM) complement communicates with the CMMU devices on the mezzanine over the high speed S bus. Memory capacity can be increased to 64MB (or 256MB or 128MB) by adding one to three MVME288 series memory boards with 16MB DRAM (or 64MB DRAM or 32MB ECC DRAM) each.

Up to four clusters of MC88100 RISC microprocessors closely coupled with MC88200 or MC88204 Cache/Memory Management Units (CMMUs) are contained on a HYPERmodule mezzanine module plugged into the main logic board (CPU board). The main logic board contains a full 32-bit VMEbus interface, as well as address decode logic and M bus (CMMU memory bus) interface logic.

System Controller Board Functional Description

Located on the left of the MVME188A board set, the system controller board contains the VMEbus A16 slave interface, system controller functions, and other facilities including some commonly required by a microcomputer operating in a VMEbus environment. The system controller board complement includes: two serial ports, four programmable timers, four 32-pin JEDEC ROM sockets, SRAM and battery-backed SRAM, switches, and indicator LEDs. Also provided are 21 registers, not including the 8-register Global Control and Status Register (GCSR) set, or the registers in the RTC, DUART, and CIO.

For a block diagram of the system controller board, see Figure 4-2.

System Controller Board Registers

The following sections describe in detail the system controller board registers and their functions, as well as the state of all bits at power-up time (SRST) or after a local reset (LRST).

All registers (with the exception of the GCSR) are located on 4-byte boundaries. Accesses to all of these registers should be 32-bit operations. All accesses to the GCSR should be 8-bit or 16-bit operations. Unless otherwise noted in the register bit descriptions, all undefined bits will be read as "1". Undefined bits are ignored on writes.

All of the following registers reside on the system controller board; some are contained within LSI parts (such as the 68692 DUART). Registers that are described in an LSI data sheet are not further described here, unless their functions are defined by external hardware (such as which DUART pins connect to RTS, CTS, etc). Please refer to the data sheets of the 68692 DUART, Z8536 CIO, and MK48T02 clock for specific operation of these parts. These data sheets are listed in the *Related Documentation* section in Chapter 1.

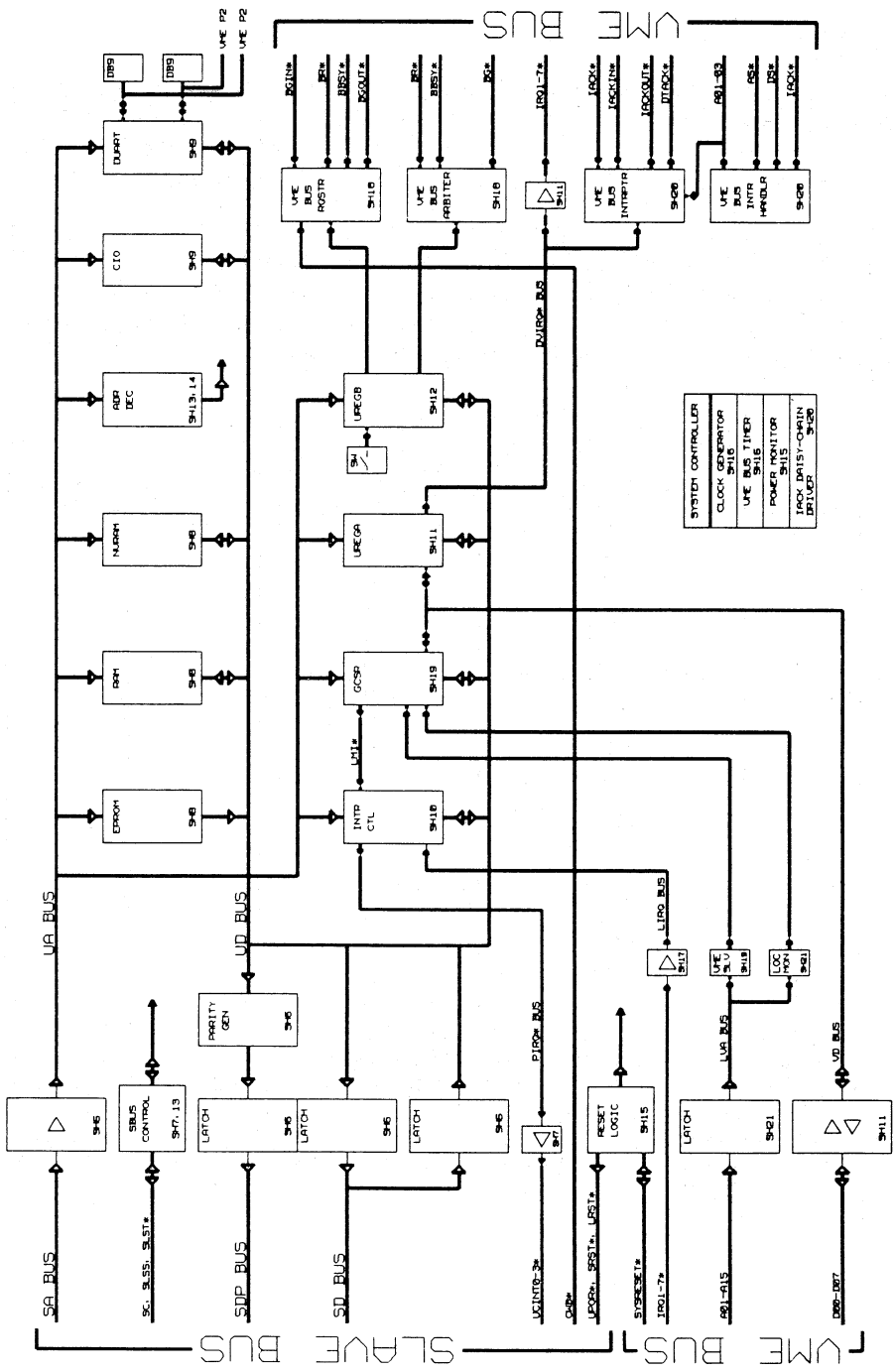


Figure 4-2. System Controller Board Block Diagram

68692 DUART Registers

The DUART register bit assignments are described in the 68692 data sheet. How the hardware connection is made to each of the EIA-232-D serial ports on the MVME188A front panel is shown in Table 4-1. This table provides all of the required information to properly initialize and use the two serial ports.

Given the clock rate of the DUART (3.6864 MHz), a square wave can be produced with anywhere from 540 nsec to 563 msec period. In 1x clock mode, the range is 542 nsec to 35.5 msec in 542 nsec increments. In 16x clock mode, the range is 8.6 μsec to 568 msec in 8.6 μsec increments. All these times are approximate; actual values can be calculated from the formula in the DUART data sheet.

Table 4-1. DUART Hardware I/O Connections

DUART Pin	I/O	Connects To		
		EIA-232-D Signal	DUART Port	MVME188A Port
IP0	I	CTS	A	1
IP1	I	CTS	B	2
IP2	I	DCD	A	1
IP3	I	DCD	B	2
IP4	I	DCD	A	1
IP5	I	DCD	B	2
OP0	O	RTS	A	1
OP1	O	RTS	B	2
OP2	O	DTR	A	1
OP3	O	*	*	*
OP4	O	-	-	-
OP5	O	DTR	B	2
OP6	O	-	-	-
OP7	O	-	-	-
RxDA	I	RXD	A	1
RxDB	I	RXD	B	2
TxDA	O	TXD	A	1
TxDB	O	TXD	B	2

NOTES: I = input to DUART; O = output from DUART.
MVME188A port 1 (bottom connector J4) is used for the console. Port 2 is the top connector J3. The DB-9 board-edge connectors have the same pin assignments as defined in Chapter 2.
* This output is connected to the DUART timer interrupt (DTI) of the on-board interrupt controller; it should be programmed as the counter-ready (CTRDY*) output.

Z8536 CIO Registers

The CIO register bit assignments are described in the Z8536 data sheet. What hardware connection is made to each of the CIO pins that are used is shown in Table 4-2. This table provides required data to properly initialize and use the three CIO timers.

The clock rate of the CIO is 4 MHz; a square wave can be produced with anywhere from 1 μs to 4295 s (71 min.) period. Using only one timer, the range is 1 μs to 65.5 ms in 1 μs increments. By using two cascaded timers, the range can be increased a factor of 65536. This yields a minimum period of 1 μs, and a maximum period of 4295 s (in 1 μs increments). All these times are approximate; actual values can be calculated from the count values loaded into CIO registers based on a PCLK of 4 MHz (250 ns period).

Several timer I/O pins are brought out to two optional header areas (timer control headers). The pinouts for these headers are described in the *Hardware Preparation* section in Chapter 2. Normally these two headers do not have pins in production versions; instead, a zero-ohm shunt (R37) is installed to connect the output of timer #2 to the input of timer #3 so the timers can be cascaded.

Note that the CIO is given a hardware reset whenever SRST or LRST is asserted.

Table 4-2. CIO Hardware I/O Connections

CIO Pin	I/O	CIO Function	Connection Options
PA0	X	bit I/O	-
PA1	X	bit I/O	-
PA2	X	bit I/O	-
PA3	X	bit I/O	-
PA4	X	bit I/O	-
PA5	X	bit I/O	-
PA6	X	bit I/O	-
PA7	X	bit I/O	-
PB0	O	timer 2 output	timer 3 input no connection
PB1	I	timer 2 input	alternate clock (125 kHz) location monitor 2 no connection
PB2	I	timer 2 trigger	timer control header
PB3	I	bit input	ENV0* (switch S3-2)
PB4	I	bit input	ENV1* (switch S3-3)
PB5	I	bit input	ENV2* (switch S3-4)
PB6	X	bit I/O	-
PB7	I	bit input	BRIRQI* (VMEbus IRQ1*)
PC0	O	timer 3 output	watchdog timer (enabled by EWDTO bit of UCSR) timer interface header
PC1	I	timer 3 input	alternate clock timer 2 output location monitor 2 no connection
PC2	I	timer 3 trigger	timer interface header
PC3	I	timer 3 gate	timer interface header

NOTES: I = input to CIO; O = output from CIO.
ALL pins (both input and output) are pulled up, so unused pins can be programmed as either inputs or outputs.

Interrupt Enable and Status Registers

The Interrupt Status (IST) and Interrupt Enable (IEN0 through IEN3) registers all have the same bit assignments. These are shown in Table 4-3. The IST register shows the current state of all interrupt requests, independent of any enables. The IEN0 through IEN3 registers (one for each processor) allow each individual interrupt to be enabled. For each active-high IEN register bit, a logic 1 enables the corresponding interrupt. All IEN bits are cleared to 0 on SRST or LRST. Because the IST register reflects the state of the interrupt requests, IST bits are not directly affected by reset. Interrupt requests may or may not go away with reset; some may have to be cleared by software before use. All reserved interrupt request bits are read as "0" (negated).

Table 4-3. Interrupt Status Register (IEN0-IEN3, IST) Bit Definitions

IEN0 - IEN3, IST (upper half)

BIT 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	BIT 16
ABRT	ACF	ARBTO	DTI	SWI7	SWI6	SWI5	SWI4	IRQ7	rsrvd	CIOI	SF	IRQ6	rsrvd	DI	SIGHPI

IEN0 - IEN3, IST (lower half)

BIT 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
rsrvd	IRQ5	rsrvd	IRQ4	rsrvd	IRQ3	rsrvd	LMI	SIGLPI	IRQ2	rsrvd	IRQ1	SWI3	SWI2	SWI1	SWI0

IEN0[31:0]	WR/rd	Interrupt enable register, CPU 0
IEN1[31:0]	WR/rd	Interrupt enable register, CPU 1
IEN2[31:0]	WR/rd	Interrupt enable register, CPU 2
IEN3[31:0]	WR/rd	Interrupt enable register, CPU 3
IEN(all)[31:0]	WR only	Simultaneous write, IEN0 - IEN3
IST[31:0]	RD only	Interrupt status register

Bit 31 ABRT ABORT switch, active edge

1 : ABORT switch was pressed since last write of CLRABRTI (CLRINT register).

0 : ABORT switch has not been pressed since last write of CLRABRTI.

Table 4-3. Interrupt Status Register (IEN0-IEN3, IST) Bit Definitions (cont'd)

Bit 30 ACF ACFAIL*, active edge
1 : ACFAIL* was asserted on VMEbus since last write of CLRACFI (CLRINT register).
0 : ACFAIL* has not been asserted since last write of CLRACFI.

Bit 29 ARBTO VMEbus arbiter time out
1 : VMEbus arbiter BG* has timed out since EARBTO was set to 1 Utility Control and Status Register (UCSR).
0 : VMEbus arbiter BG* has not timed out since EARBTO was set to 1.

Bit 28 DTI DUART timer interrupt
1 : DUART timer output is = 0 (interrupt request).
0 : DUART timer output is = 1 (no interrupt request).
NOTE: This is the timer output ONLY - not the DUART's interrupt request.

Bit 27 SWI7 Software generated interrupt 7
Bit 26 SWI6 Software generated interrupt 6
Bit 25 SWI5 Software generated interrupt 5
Bit 24 SWI4 Software generated interrupt 4
1 : Corresponding SWI bit is set (interrupt requested).
0 : Corresponding SWI bit is clear (no interrupt requested).
NOTE: SWI bits are all cleared by SRST or LRST.

Bit 23 IRQ7 VMEbus level 7 interrupt
1 : VMEbus IRQ7* is asserted (i.e., it is low).
0 : VMEbus IRQ7* is not asserted (i.e., it is high).

Bit 22 Reserved, always "0"

Bit 21 CIOI CIO (Z8536) interrupt
1 : The Z8536 CIO is requesting an interrupt.
0 : The Z8536 CIO is not requesting an interrupt.

Bit 20 SF SYSFAIL*, active edge
1 : SYSFAIL* was asserted on VMEbus since last write of CLRSFI (CLRINT register).
0 : SYSFAIL* has not been asserted since last write of CLRSFI.

Bit 19 IRQ6 VMEbus level 6 interrupt
1 : VMEbus IRQ6* is asserted (i.e., it is low).
0 : VMEbus IRQ6* is not asserted (i.e., it is high).

Bit 18 Reserved, always "0"

Table 4-3. Interrupt Status Register (IEN0-IEN3, IST) Bit Definitions (cont'd)

Bit 17 DI DUART interrupt
1 : The 68692 DUART is requesting an interrupt.
0 : The 68692 DUART is not requesting an interrupt.

Bit 16 SIGHPI GCSR SIGHP assertion
1 : GCSR SIGHP bit is asserted (i.e., SIGHP = 1).
0 : GCSR SIGHP bit is not asserted (i.e., SIGHP = 0).

Bit 15 Reserved, always "0"

Bit 14 IRQ5 VMEbus level 5 interrupt
1 : VMEbus IRQ5* is asserted (i.e., it is low).
0 : VMEbus IRQ5* is not asserted (i.e., it is high).

Bit 13 Reserved, always "0"

Bit 12 IRQ4 VMEbus level 4 interrupt
1 : VMEbus IRQ4* is asserted (i.e., it is low).
0 : VMEbus IRQ4* is not asserted (i.e., it is high).

Bit 11 Reserved, always "0"

Bit 10 IRQ3 VMEbus level 3 interrupt
1 : VMEbus IRQ3* is asserted (i.e., it is low).
0 : VMEbus IRQ3* is not asserted (i.e., it is high).

Bit 9 Reserved, always "0"

Bit 8 LMI Location monitor interrupt
1 : Any of the GCSR bits LM0* - LM3* are asserted (i.e., one or more are "0").
0 : All of the GCSR bits LM0* - LM3* are negated (i.e., they are all "1").

Bit 7 SIGLPI GCSR SIGLP assertion
1 : GCSR SIGLP bit is asserted (i.e., SIGLP = 1).
0 : GCSR SIGLP bit is not asserted (i.e., SIGLP = 0).

Bit 6 IRQ2 VMEbus level 2 interrupt
1 : VMEbus IRQ2* is asserted (i.e., it is low).
0 : VMEbus IRQ2* is not asserted (i.e., it is high).

Bit 5 Reserved, always "0"

Bit 4 IRQ1 VMEbus level 1 interrupt
1 : VMEbus IRQ1* is asserted (i.e., it is low).
0 : VMEbus IRQ1* is not asserted (i.e., it is high).

Table 4-3. Interrupt Status Register (IEN0-IEN3, IST) Bit Definitions (cont'd)

Bit 3 SWI3 Software generated interrupt 3
Bit 2 SWI2 Software generated interrupt 2
Bit 1 SWI1 Software generated interrupt 1
Bit 0 SWI0 Software generated interrupt 0
1 : Corresponding SWI bit is set (interrupt requested).
0 : Corresponding SWI bit is clear (no interrupt requested).
NOTE: SWI bits are all cleared by SRST or LRST.

SETSWI Register

The SETSWI register is used by the software to generate an interrupt. It is not really a register itself, but it is used to control the setting of the SWI bits. The state of the SWI bits can be read from the interrupt status (IST) register. For every SETSWI bit that is written as a "1", the corresponding SWI bit is set (requesting an interrupt). Every bit that is written as a "0" does nothing. This makes each SWI bit individually settable by writing a "1" to the corresponding bit in the SETSWI register. Bit assignments are shown in Table 4-4.

Because this location is write only, it is unaffected by reset. (However, all SWI bits are cleared by SRST or LRST.)

Table 4-4. Set Software Interrupt Request Register (SETSWI) Bit Definitions

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SETSWI7	SETSWI6	SETSWI5	SETSWI4	SETSWI3	SETSWI2	SETSWI1	SETSWI0

Bits 7-0 SETSWI[7:0] WR only Set software interrupt request
1 : Set software interrupt request corresponding to this bit.
0 : Do nothing to the software interrupt request corresponding to this bit.
NOTE: The state of each SWI bit is reflected in the IST register.

CLRSWI Register

The CLRSWI register is used by the software to clear an SWI interrupt. It is not really a register itself, but it is used to control the clearing of the SWI bits. The state of the SWI bits can be read from the IST register. For every CLRSWI bit that is written as a "1", the corresponding SWI bit is cleared (removing an interrupt request). Every bit written as a "0" does nothing. This makes each SWI bit individually resettable by writing a "1" to the corresponding bit in the CLRSWI register. Bit assignments are shown in Table 4-5.

Because this location is write only, it is unaffected by reset. (However, all SWI bits are cleared by SRST or LRST.)

Table 4-5. Clear Software Interrupt Request Register (CLRSWI) Bit Definitions

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CLRSWI7	CLRSWI6	CLRSWI5	CLRSWI4	CLRSWI3	CLRSWI2	CLRSWI1	CLRSWI0

Bits 7-0 CLRSWI[7:0] WR only Clear software interrupt request
1 : Clear software interrupt request corresponding to this bit.
0 : Do nothing to the software interrupt request corresponding to this bit.
NOTE: The state of each SWI bit is reflected in the IST register.

ISTATE Register

The ISTATE register is used to determine the current state of the three interrupt sources that are generated by transitions: ABRT, ACF, and SF. The value returned on read indicates the current state of each signal. Bit assignments are shown in Table 4-6.

Because this register reflects the current state of external requests (not latched), it is not affected by reset.

Table 4-6. Hardware Interrupt State Register (ISTATE) Bit Definitions

BIT 2	BIT 1	BIT 0
ABORT	ACFAIL	SYSFAIL

Bit 2 ABORT RD only ABORT switch status
1 : The ABORT switch is depressed.
0 : The ABORT switch is not depressed.

Bit 1 ACFAIL RD only ACFAIL* status
1 : ACFAIL* is asserted (i.e., low) on the VMEbus.
0 : ACFAIL* is not asserted (i.e., high) on the VMEbus.

Bit 0 SYSFAIL RD only SYSFAIL* status
1 : SYSFAIL* is asserted (i.e., low) on the VMEbus.
0 : SYSFAIL* is not asserted (i.e., high) on the VMEbus.

CLRINT Register

The CLRINT register is used to clear the hardware edge-detectors that generate ABRT, ACF, and SF interrupts. The events that caused the interrupt are only transitions from inactive to active. Writing a "1" to a particular bit clears that bit. Another transition from inactive to active must then occur on that edge-detector input for that interrupt request to be re-asserted. For example, if SYSFAIL* is asserted, a "1" written to the CLRSFI bit sets SF = 0 in the IST register. However, it does not negate SYSFAIL*. In order to get another SF interrupt, SYSFAIL* would have to be negated and then re-asserted. Bit assignments are shown in Table 4-7.

Because the CLRINT register is write-only, it is not affected by reset. (However, the interrupt request bits (ABRT, ACF, and SF) are cleared by system reset or local reset.)

Table 4-7. Clear Hardware Interrupt Request Register (CLRINT) Bit Definitions

BIT 2	BIT 1	BIT 0
CLRABRTI	CLRACFI	CLRSFI

Bit 2 CLRABRTI WR only Clear ABORT switch interrupt request
1 : Clear ABRT interrupt request.
0 : Do nothing to the ABRT interrupt request.

Bit 1 CLRACFI WR only Clear ACFAIL interrupt request
1 : Clear ACF interrupt request.
0 : Do nothing to the ACF interrupt request.

Bit 0 CLRSFI WR only Clear SYSFAIL interrupt request
1 : Clear SF interrupt request.
0 : Do nothing to the SF interrupt request.

NOTE: The status of ABRT, ACF, and SF is indicated in the IST register. These three interrupt requests are generated on the detection of the interrupting condition by a hardware edge-detector. Clearing the interrupt requests by writing to the CLRINT register does not clear the interrupt condition, it merely clears the interrupt request. The ISTATE register (described previously) indicates the state of these three interrupting conditions.

VIRQLV Register

The VIRQLV register is used to request a VMEbus interrupt. A VMEbus interrupt request is asserted corresponding to the value in VIRQLV. A 0 value means that no interrupt is being requested by the MVME188A. When the interrupt is acknowledged by the VMEbus, this register is reset to 0. As long as this register is non-zero, all writes to the VMEbus interrupt vector register (VIRQV) are inhibited. This prevents the vector from changing while it is being read. It is legal on the VMEbus to remove the request (i.e., write 0 to the VIRQLV register when it is currently non-zero), but it may cause a spurious interrupt on the VMEbus. This should be done only as a last resort (in the case of no response to an interrupt for an extended time, for instance).

Note that the MVME188A can only assert one VMEbus interrupt request at a time. The software should verify that bits 2 through 0 of VIRQLV are 0 before requesting any VMEbus interrupt. Bit assignments are shown in Table 4-8.

This register is cleared by both LRST and SRST.

Table 4-8. VMEbus Interrupt Request Level Register (VIRQLV) Bit Definitions

BIT 2	BIT 1	BIT 0
VIRQLV2	VIRQLV1	VIRQLV0

Bits 2-0 VIRQLV[2:0] WR/RD VMEbus interrupt request level

Bit 2	Bit 1	Bit 0	Condition
0	0	0	VMEbus interrupter is not busy, writes to VIRQV register OK.
0	0	1	VMEbus interrupt requested on level 1.
0	1	0	VMEbus interrupt requested on level 2.
0	1	1	VMEbus interrupt requested on level 3.
1	0	0	VMEbus interrupt requested on level 4.
1	0	1	VMEbus interrupt requested on level 5.
1	1	0	VMEbus interrupt requested on level 6.
1	1	1	VMEbus interrupt requested on level 7.

NOTE: All writes to the VIRQV register are ignored unless this register = 0. This register is reset to 0 upon completion of a VMEbus IACK cycle where VMEbus A[03:01] match VIRQLV[2:0]. It is also set to 0 by LRST or SRST.

VIACK(n)V Registers

These registers are used to acknowledge VMEbus interrupt requests. There is one register for each VMEbus interrupt request level. When one of these registers is read, an interrupt acknowledge cycle is generated on the VMEbus. The vector obtained from the responding interrupter is returned as data in bits 0 through 7, and bit 8 is clear. If no interrupter responds to the IACK cycle (i.e., a BUSERR* terminated the cycle), a value of $FFFFFF00 is returned (bit 8 is set, bits 0 through 7 are cleared, and bits 9 through 31 are unused and therefore read as set). Bit assignments are shown in Table 4-9.

Note that the MVME188A can "interrupt itself" by requesting on a particular level and then acknowledging its own interrupt request. In this case (when the MVME188A is both interrupter and interrupt handler), no vector is supplied to the VMEbus. The interrupt vector is supplied by the VIRQV register directly to the addressed VIACK(n)V register without being driven onto the VMEbus.

Because this register reads a vector from the bus each time it is accessed, it is unaffected by reset.

Table 4-9. VMEbus IACK Vector Read Register (VIACK(n)V) Bit Definitions

VIACK1V - VIACK7V

BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
IBERR	VIACKV7	VIACKV6	VIACKV5	VIACKV4	VIACKV3	VIACKV2	VIACKV1	VIACKV0

VIACK1V[8:0]	RD only	VMEbus IACK level 1 generation and vector read
VIACK2V[8:0]	RD only	VMEbus IACK level 2 generation and vector read
VIACK3V[8:0]	RD only	VMEbus IACK level 3 generation and vector read
VIACK4V[8:0]	RD only	VMEbus IACK level 4 generation and vector read
VIACK5V[8:0]	RD only	VMEbus IACK level 5 generation and vector read
VIACK6V[8:0]	RD only	VMEbus IACK level 6 generation and vector read
VIACK7V[8:0]	RD only	VMEbus IACK level 7 generation and vector read

Table 4-9. VMEbus IACK Vector Read Register (VIACK(n)V) Bit Definitions (cont'd)

Bit 8 IBERR RD only IACK bus error
1 : VMEbus IACK cycle generated by this register read was terminated by a BERR*; bits 7:0 are all = "0".
0 : VMEbus IACK cycle generated by this register read was terminated by a DTACK*; bits 7:0 are valid vector.

Bits 7-0 VIACKV[7:0] RD only VMEbus interrupt vector from D[07:00]
A read of any of these registers generates a VMEbus IACK cycle on the level corresponding to the register accessed. The 8-bit vector from the VMEbus is returned as the read data. These bits are only valid when IBERR = 0; otherwise they read as all "0".

VIRQV Register

The VIRQV register supplies the vector to the VMEbus during an IACK cycle if the level being acknowledged is the same one that the MVME188A is requesting on. Writes are inhibited to this register if the VIRQLV register is non-zero (i.e., if an MVME188A interrupt request is pending). Bit assignments are shown in Table 4-10.

This register is cleared by SRST or LRST. It should be initialized before any VMEbus interrupts are generated.

Table 4-10. VMEbus Interrupter Interrupt Vector Register (VIRQV) Bit Definitions

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
VIRQV7	VIRQV6	VIRQV5	VIRQV4	VIRQV3	VIRQV2	VIRQV1	VIRQV0

Bits 7-0 VIRQV[7:0] WR/rd VMEbus interrupter vector register
VIRQV[7:0] drives VMEbus D[07:00] during VMEbus interrupt acknowledge cycles if the level on VMEbus A[03:01] matches VIRQLV.

Global Control and Status Registers (GCSRs)

The GCSRs are a block of eight 1-byte registers located non-coterminously on consecutive *2-byte* boundaries which appear both in the M bus and the VMEbus A16 (short I/O) address spaces. All eight are intended to function identically to the corresponding GCSR registers in the Motorola MVME6000 VMEbus Interface chip (also called the VMEchip).

The purpose of the MVME188A GCSR is to facilitate operation in a multi-CPU environment by allowing other CPUs on the VMEbus to interrupt, disable, communicate with, and determine the operational status of the RISC processor(s) local to the GCSR. One register of the set includes four bits which function as location monitors to allow one MVME188A processor to broadcast a signal to other MVME188A processors, if any. All eight registers are accessible from any local processor as well as from the VMEbus. Some of the bits can be individually set or cleared. Some of the bits have restricted access on either M bus or VMEbus A16 space. Bit assignments are shown in Table 4-11. Other than the location monitors, these registers are the only VMEbus A16 resource on the MVME188A.

The GCSR appears as eight one-byte registers addressed on 2-byte boundaries in both the M bus and VMEbus A16 space. Note that this is different from all of the other MVME188A registers, which appear on 4-byte boundaries. If a 32-bit access is attempted from the VMEbus, the MVME188A does not respond; normally this cycle is terminated by a bus error from the VMEbus time-out timer. If a 4-byte access is attempted from M bus, no error is generated. However, the data returned will be two copies of the register whose 2-byte address appears on the M bus (i.e., the lower of the two addressed registers).

To guarantee proper operation of the location monitors, LM0* through LM3* should be set high by writing to Global Register 0 in VMEbus A16 space, rather than in M bus space. This will guarantee that consecutive interrupts from the same location monitor will be properly recognized, even in the absence of any VMEbus activity.

CAUTION

The hardware does not support indivisible M bus accesses to the GCSR. Do not use the XMEM instruction directly on any GCSR except via VMEbus A16 (short I/O) space. This will insure that no other VMEbus masters access the GCSR in between the XMEM load and store operations.

Table 4-11. Global Control and Status Registers (GCSR) Bit Definitions

		GCSR							
NAME	**OFFSET**	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
GLOBAL0	$1	LM3*	LM2*	LM1*	LM0*	1	1	1	1
GLOBAL1	$3	R&H	SCON	ISF	BRDFAIL	rsrvd	rsrvd	SIGHP	SIGLP
BRDID	$5	BRDID7	BRDID6	BRDID5	BRDID4	BRDID3	BRDID2	BRDID1	BRDID0
GPCSR0 through GPCSR4	$7,$9, $B,$D, $F	General purpose user defined registers. These bits have user defined functions.							

NOTE: These registers function similarly to those in the Motorola MVME6000 VMEbus Interface chip, also called the VMEchip, (which is used in the MVME147 MPU VMEmodule). Refer to the *MVME6000 VMEbus Interface User's Manual* listed in the *Related Documentation* section in Chapter 1 herein.

GLOBAL0 Bit Functions

Bits 7-4 LM[3:0]* RD/WR1 VMEbus/M bus Location monitors
1 : No location monitor access was detected since this bit was set by software or SRST.
0 : A location monitor access was detected since this bit was set by software or SRST.
Set to 1 by SRST or LRST.

Bits 3-0 CHID[3:0] RD only VMEbus/M bus Chip identification number
These bits are hardwired as %1111 to indicate that the GCSR is the MVME188A implementation of the VMEchip function.
Unaffected by SRST or LRST.

GLOBAL1 Bit Functions

Bit 7 R&H RD/WR VMEbus/M bus Local reset-and-hold
1 : LRST is held asserted for this MVME188A.
0 : LRST is not held asserted for this MVME188A.
Cleared by SRST or LRST.

Table 4-11. Global Control and Status Registers (GCSR) Bit Definitions (cont'd)

Bit 6 SCON RD only VMEbus/M bus System controller on
1 : SCON switch is on, making this MVME188A the system controller.
0 : SCON switch is off; this MVME188A is not the system controller.
Unaffected by SRST or LRST.

Bit 5 ISF RD/WR VMEbus/M bus Inhibit SYSFAIL*
1 : This MVME188A will not assert SYSFAIL* on the VMEbus.
0 : Assert SYSFAIL* if DRVSF* is asserted.
Cleared by SRST or LRST.

Bit 4 BRDFAIL RD only VMEbus/M bus Board fail status
1 : DRVSF* = 0 (asserted).
0 : DRVSF* = 1 (negated).
Not directly affected by reset; always reflects the state of DRVSF* and the watchdog timer output.

Bits 3-2 Reserved, always = "0".

Bit 1 SIGHP RD/WR1 VMEbus, RD/WR0 M bus Signal high priority
1 = SIGHP interrupt request pending.
0 = SIGHP interrupt request not pending.
Cleared by SRST or LRST.

Bit 0 SIGLP RD/WR1 VMEbus, RD/WR0 M bus Signal low priority
1 = SIGLP interrupt request pending.
0 = SIGLP interrupt request not pending.
Cleared by SRST or LRST.

BRDID Bit Functions

Bits 7-0 BRDID[7:0] RD only VMEbus, RD/WR M bus Board identification
These bits are written by a local CPU and are read-only from the VMEbus.
Cleared by SRST or LRST.

Table 4-11. Global Control and Status Registers (GCSR) Bit Definitions (cont'd)

GPCSR0 Bit Functions
GPCSR1 Bit Functions
GPCSR2 Bit Functions
GPCSR3 Bit Functions
GPCSR4 Bit Functions

Bits 7-0 RD/WR VMEbus/M bus General purpose user defined registers
These bits have user defined functions.
GPCSR0 is set to $ FF by SRST.
GPCSR1 through GPCSR4 are cleared by SRST or LRST.

Utility Control and Status Register (UCSR)

The UCSR provides status (on read) and control (on write) of many hardware functions on the system controller (utility) board. The bits are all read/write with the exception of PWRUP*, which is writable as a 1 only. (It can only be cleared by the hardware.) Bit assignments are shown in Table 4-12.

Table 4-12. Utility Control and Status Register (UCSR) Bit Definitions

BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
PWRUP*	DRVSF*	BRIRQO	ROBIN	BRLV1	BRLVO	RNEVER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
RONR	RWD	EARBTO	VTOSEL1	VTOSEL0	EWDTO	WDA1	WDA0

Bit 14 PWRUP* RD/WR1 Power up initialization indicator
1 : No ACFAIL assertion has occurred since the CPU negated PWRUP*.
0 : ACFAIL was asserted since the last time the CPU negated PWRUP*.

NOTE: This bit is set to 0 when ACFAIL is asserted, and is guaranteed to be = 0 when power comes up. It can only be negated (set = 1) by the CPU. It is unaffected by SRST or LRST.

Table 4-12. Utility Control and Status Register (UCSR) Bit Definitions (cont'd)

Bit 13 DRVSF* WR/rd Board/system fail
1 : Do not assert SYSFAIL* on the VMEbus.
0 : If ISF = 0 in GCSR, then assert SYSFAIL* on the VMEbus; if ISF = 1 in GCSR, then do not assert SYSFAIL* on the VMEbus.

NOTE: This bit will light the FAIL LED if asserted (= 0), even if SYSFAIL* is not driven onto the VMEbus. Cleared by SRST or LRST.

Bit 12 BRIRQO WR/rd Broadcast interrupt (drive VMEbus IRQ1* low)
1 : Assert VMEbus IRQ1*.
0 : Do not assert VMEbus IRQ1*.

NOTE: IRQ1* is an open-collector line; it may still be asserted on the VMEbus (by another interrupter) even though BRIRQO = 0. Cleared by SRST or LRST.

Bit 11 ROBIN WR/rd Select round-robin arbiter mode
1 : VMEbus arbiter operates in the Round-Robin Select (RRS) mode.
0 : VMEbus arbiter operates in the Priority (PRI) mode.
Cleared by SRST or LRST.

Bits 10-9 BRLV[1:0] WR/rd VMEbus request/bus grant level

BRLV1	BRLV0	Level
0	0	level 0
0	1	level 1
1	0	level 2
1	1	level 3

NOTE: Do not change these bits while any MVME188A VMEbus request is pending. These bits are set to "11" by SRST or LRST.

Bit 8 RNEVER WR/rd Release never
1 : Once the bus has been acquired, do not release it.
0 : Operate in the release mode set by RWD.
Cleared by SRST or LRST.

Bit 7 RONR WR/rd Request on no request
1 : Set VMEbus requester to operate in fairness (FAIR) mode. (Do not request bus if anyone else is requesting on currently assigned level).
0 : Request the bus whenever required.
Cleared by SRST or LRST.

Table 4-12. Utility Control and Status Register (UCSR) Bit Definitions (cont'd)

Bit 6 RWD WR/rd VMEbus requester type
1 : VMEbus requester is a Release-When-Done (RWD) type.
0 : VMEbus requester is a Release-On-Request (ROR) type.
Cleared by SRST or LRST.

Bit 5 EARBTO WR/rd Enable VMEbus arbiter timeout
1 : MVME188A removes bus grant if BBSY* is not driven within 1 second of bus grant assertion.
0 : MVME188A will hold bus grant until BBSY* is asserted.

NOTE: An interrupt request (ARBTO) is generated when bus grant is removed because of a timeout. This bit is cleared by SRST or LRST.

Bits 4-3 VTOSEL[1:0] WR/rd VMEbus data transfer timeout select
These bits determine the length of time that DSx* can be asserted on the VMEbus before the MVME188A asserts a BERR* if SCON = 1. If SCON = 0, this timer is always disabled, regardless of the state of these bits.
These bits are set to "11" by SRST or LRST.

VTOSEL1	VTOSEL0	Interval
0	0	32 µs
0	1	64 µs
1	0	128 µs
1	1	disabled

Bit 2 EWDTO WR/rd Enable watchdog timeout
1 : When timer #3 output = 0, action selected by WDA[1:0] is performed.
0 : Timer #3 output causes no watchdog action.
NOTE: Timer #3 can be used for general purposes if this bit is = 0. This bit is cleared by SRST or LRST.

Bits 1-0 WDA[1:0] WR/rd Watchdog action on timeout
These bits determine what happens when the watchdog timer expires. If EWDTO = 0, then no action will occur regardless of the state of these bits.
These bits are set to "11" by SRST or LRST.

Table 4-12. Utility Control and Status Register (UCSR) Bit Definitions (cont'd)

WDA1	WDA0	Action
0	0	momentary system reset (200 ms)
0	1	momentary local reset (200 ms)
1	0	local reset and hold
1	1	watchdog timer disabled

BASAD Register

The BASAD register is used to read the state of the VMEbus A16 base address DIP switches. It is read only, and unaffected by reset. Bit assignments are shown in Table 4-13.

Table 4-13. Base Address Register (BASAD) Bit Definitions

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
GRPAD7	GRPAD6	GRPAD5	GRPAD4	GRPAD3	GRPAD2	GRPAD1	GRPAD0

BIT 7	BIT 6	BIT 5	BIT 4	BITS 3 THROUGH 0
BDAD3	BDAD2	BDAD1	BDAD0	1

Bits 15-8 GRPAD[7:0] RD only Group address
This address is compared with VMEbus or slave address bits 15 through 8 to determine where to place the GCSR in the map. It is also used to determine what addresses the location monitors respond to. It is a read-only reflection of the GRPAD switch setting. (Refer to the *Hardware Preparation* section in Chapter 2.)

Bits 7-4 BDAD[3:0] RD only Board address within group
This address is compared with VMEbus or slave address bits 7 through 4 to determine where to place the GCSR in the map. It is a read-only reflection of the BDAD switch setting. (Refer to the *Hardware Preparation* section in Chapter 2.)

Bits 3-0 Not used, always 1.

GLBRES Register

The GLBRES register is not actually a register; when this location is written to, a SRST is generated. This resets the entire system, including the VMEbus. Note that the VMEbus is reset even if the MVME188A is not system controller. Bit assignments are shown in Table 4-14.

CAUTION

It may take up to 32 μs before SRST is actually asserted, even though the M bus write cycle has been successfully completed. Any LRST that occurs during this window will reset the MVME188A and prevent the SRST from occurring. Refer to the section on *Local Reset Operation* in Chapter 3.

Table 4-14. Global Reset Register (GLBRES) Bit Definitions

BITS 31 THROUGH 0
X

Bits 31-0 GLBRES[31:0] WR only Global reset for 200 ms
X : Any write access causes a global system reset.
NOTE: Any read access returns all "1s" ($FFFFFFFF) without performing a reset.

VMEbus A16 Slave Address Decoder

Because the GCSR can be used in configuring and bootstrapping the MVME188A, 12 switches are provided which allow the base address of the GCSR to be located anywhere within the VMEbus A16 address space, as desired. Also, to provide system development flexibility, the switches are accessible through the front panel. The GCSR is the only VMEbus A16 resource on the MVME188A RISC microcomputer.

SRAM

The MVME188A has 128KB of SRAM, configured as 32K by 32 bits. This 32-bit wide SRAM is not battery backed up.

Timekeeper RAM

The battery backed up SRAM and clock calendar functions provided by the MVME188A are implemented with the MK48T02 Timekeeper RAM. This device offers battery-backed-up operation (up to 11 years if the RTC is off), a clock/calendar function, 2KB of SRAM and circuitry for automatically and safely switching to battery power when system power is removed.

VMEbus System Controller

The MVME188A RISC microcomputer has a VMEbus system controller which may be enabled by means of a switch accessible through the front panel. Several features are offered by the system controller. A system clock driver within the system controller circuitry drives the VMEbus SYSCLK line with a 16 Mhz clock signal. This signal is not used by the MVME188A.

When the MVME188A is configured to be system controller, its power monitor is enabled to drive the VMEbus SYSRESET* line. To determine when to assert SYSRESET*, the power monitor reads the VMEbus ACFAIL* line. In addition, the power monitor guarantees that SYSRESET* will be asserted for at least 200 ms after the +5 Vdc line reaches a voltage of +4.75 Vdc.

As determined by the state of the RONR bit in the UCSR, the arbiter functions either as a PRI or as an RRS arbiter. In the PRI mode, Bus Clear operation is supported; i.e., BCLR* is asserted when a request higher in priority than the current active bus grant level occurs.

The arbiter includes a function for generation of a local interrupt following a bus grant timeout. If the arbiter grants the VMEbus to a requester which does not drive BBSY* low within one second, the arbiter withdraws the bus grant and interrupts the local processor. The interrupt acknowledge out (IACKOUT*) line to the local interrupter is driven by the IACK daisy chain driver whenever IACKIN* is low followed by the assertion of either data strobe (DS0* or DS1*).

A choice of 32 μs, 64 μs, 128 μs, or infinity may be selected as the time-out period before a bus error is generated by the VMEbus data transfer time-out timer. The time-out is measured from the assertion of either VMEbus data strobe.

VMEbus Requester

To obtain the VMEbus for both interrupt handling and data transfers, a bus requester is used which is configurable for any level. It supports the RWD, ROR, and FAIR modes for data transfer.

7-Level VMEbus Interrupt Requester

An interrupt on any of the seven VMEbus levels can be generated by writing the interrupt vector into the 8-bit VME IRQ vector register (VIRQV) and then writing the interrupt level into the appropriate system control register (VIRQLV). The specified interrupt line is kept asserted until a VMEbus interrupt acknowledge cycle occurs on the requested level; this causes a read of the VIRQV register and a release of the interrupt request. Only one interrupt request may be active at a time.

7-Level VMEbus Interrupt Handler

The interrupt enable lines can be individually set by software to allow one or more of the seven VMEbus interrupt requests to generate an MVME188A interrupt. Interrupt handler software reads one of the VMEbus IACK cycle registers (VIACK1V through VIACK7V) to initiate a VMEbus IACK cycle. Each VIACK(n)V register is used to acknowledge interrupts on a different VMEbus interrupt level. The IACK cycle is a read of the 8-bit interrupt vector from the interrupt requester by the active processor.

25-Line Orthogonal Multiprocessor Interrupt Controller

All local interrupts for the HYPERmodule CPUs are controlled by logic on the system controller board.

There are no CPU interrupt restrictions. Every processor can receive any interrupt. Any processor can disable any interrupt. Software must use the individual interrupt enable bits to coordinate interrupts.

Although the interrupt controller is given no hardware knowledge of CPU level, a group/level organization can be programmed into the interrupt handler.

Additional interrupts (SWI7:0) enable one processor to interrupt the other (2-processor model), or one processor to interrupt any of the other three processors (4-processor model).

Interrupt lines supported include:

ABRT	ABORT switch on system controller board
ACF	VMEbus ACFAIL* interrupt
ARBTO	VMEbus arbiter BG* timeout
DTI	DUART timer interrupt
SWI7:0	Software generated interrupts
IRQ7:1	VMEbus interrupt request lines
CIOI	Counter-timer interrupt
SF	VMEbus SYSFAIL* interrupt
DI	DUART serial port interrupt
SIGHPI	VMEbus GCSR high priority interrupt
LMI	VMEbus location monitor interrupt
SIGLPI	VMEbus GCSR low priority interrupt

Each processor has one 32-bit register (IEN) of interrupt enables that any of the processors can read or write. A single IST register provides the states of all the individual interrupt request lines. Interrupt handling software is responsible for disabling an interrupt request by changing the state of the individual enabling bit in the appropriate IEN register.

Any reset (power-on, system, or local) disables interrupts by clearing all bits of all four IEN registers. Power-on, SRST, and LRST also clears all interrupt requests generated by the MVME188A.

Status Indicator LEDs

The MVME188A system controller board front panel contains three LEDs: FAIL, HALT, and RUN. Refer to Chapter 3 for detailed information on each.

Control Switches

Two momentary-action switches are provided on the MVME188A system controller board front panel: ABORT and RESET. Refer to Chapter 3 for detailed information on them.

Onboard Switches

The system controller board contains 16 onboard switches: 12 for address mapping, one for SCON select, and three for general purpose use. The twelve address select switches map the 16-byte block containing the GCSR registers to one of 4096 possible locations within the 64KB VMEbus A16 space. The SCON switch enables the MVME188A system controller logic. The three remaining switches have user-defined functions; they can be read via an input port of the counter-timer. Holes are provided on the MVME188A system controller board for nine wire-wrap posts. These posts provide external connections for three of the four on-board timers. They also allow cascading of the timers and connection of alternate clock sources. Further information on these switches and timer connections is found in Chapter 2 in the section on *Hardware Preparation.*

16MB or 64MB DRAM Memory Board Functional Description

Located in the middle of the MVME188A board set, the DRAM board has 144 locations for one megabit (or four megabit) memory chips providing storage of 16MB (or 64MB), with parity. The board set can be ordered with up to three additional memory boards (MVME288 series) to make a total of up to 64MB DRAM (or 256MB DRAM) maximum. Although the board packaging scheme limits MVME188A local DRAM to 64MB (or 256MB) using currently available chips, the DRAM boards decode and buffer the entire 4GB address range.

The memory board is optimized for 16 byte cache line fills and spills. To allow byte writes without the delay of read-modify-write cycles, separate byte write enabling and byte storage decoding is used. Four interleaved banks of 32-bit words facilitate much faster 16 byte burst transfers than would single word random accesses to each of the sixteen bytes. When the DRAM board receives the memory address and a board select signal, bank decoders sequence through the interleaved banks. The memory board contains a wait-state generator to properly control accesses from the S bus. Each DRAM read or write operation consists of one address cycle, one data cycle for each word (32-bit) transfer, plus any inserted wait states, as shown below.

Generated Wait States	Read/Write Option
4	1 to 16 byte Read
1	1 to 16 byte Write

This yields the following maximum data handling rates for the CMMU accessing the onboard DRAM via the S bus.

Read/Write Operation	Peak Bandwidth (MB/sec)
Read DRAM	44.4
Write DRAM	66.7

Memory is refreshed every 13.1 microseconds. The operation uses 5 clock cycles. To minimize the heavy loading on the backplane power lines during refresh, board design insures that only one refresh cycle occurs per module in the system.

For a block diagram of the 16MB or 64MB DRAM memory board, see Figure 4-3.

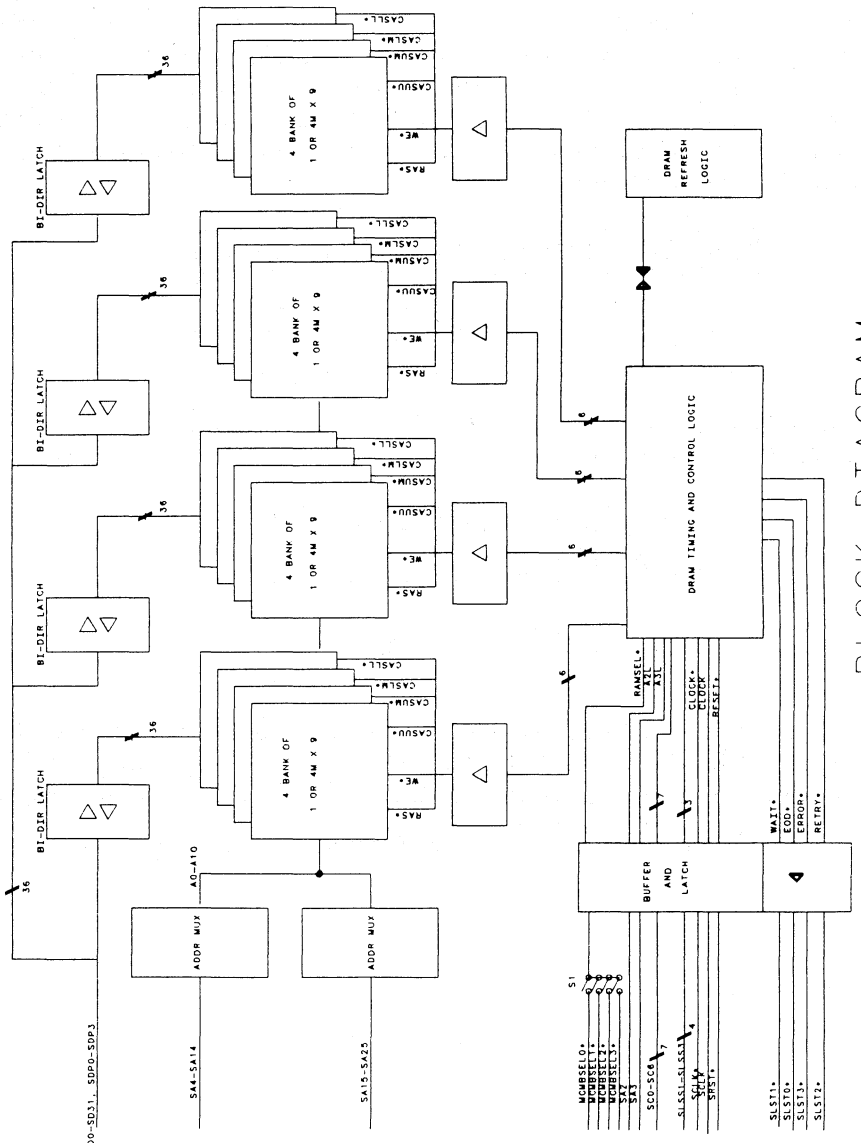


Figure 4-3. 16MB or 64MB DRAM Memory Board Block Diagram

32MB ECC DRAM Memory Board Functional Description

The MVME288EC-32 module is a 32MB memory module designed for the MVME188A system. It implements single-bit error detection and correction, and several diagnostic features useful for testing and diagnosing memory module status.

For a block diagram of the 32MB ECC DRAM memory board, see Figure 4-4.

The MVME288EC-32 32MB storage module resides on the MVME188A local bus (slave bus). The board-level architecture is implemented as two banks of DRAM. Each bank consists of a data section, 4 megabits X 32, and a check bit section, 4 megabits X 7. Each bank uses an Am29C660 error detection and correction integrated circuit.

There is a control/status (CSR) register and a diagnostic latch. The CSR is used to set the mode of the memory controller. The diagnostic latch is integrated into the Am29C660 device. It is used to initialize the memory with single- and double-bit errors.

Summary of Major Features

- 32MB storage.
- Single-bit error detection and correction.
- Double-bit error detection.
- Refresh cycles detect and correct single-bit errors.
- DRAM implementation uses 4 megabit X 1 devices which allow operation with one dead memory device per bank.
- Single-bit error counting.
- Hardware diagnostic features allow planting and detection of bit errors.
- Hardware diagnostic features allow syndrome field examination to pinpoint device failure.
- Slave bus parity is monitored and error status is maintained.
- Bus performance equal to parity-protected memory for the most frequently used cycles.

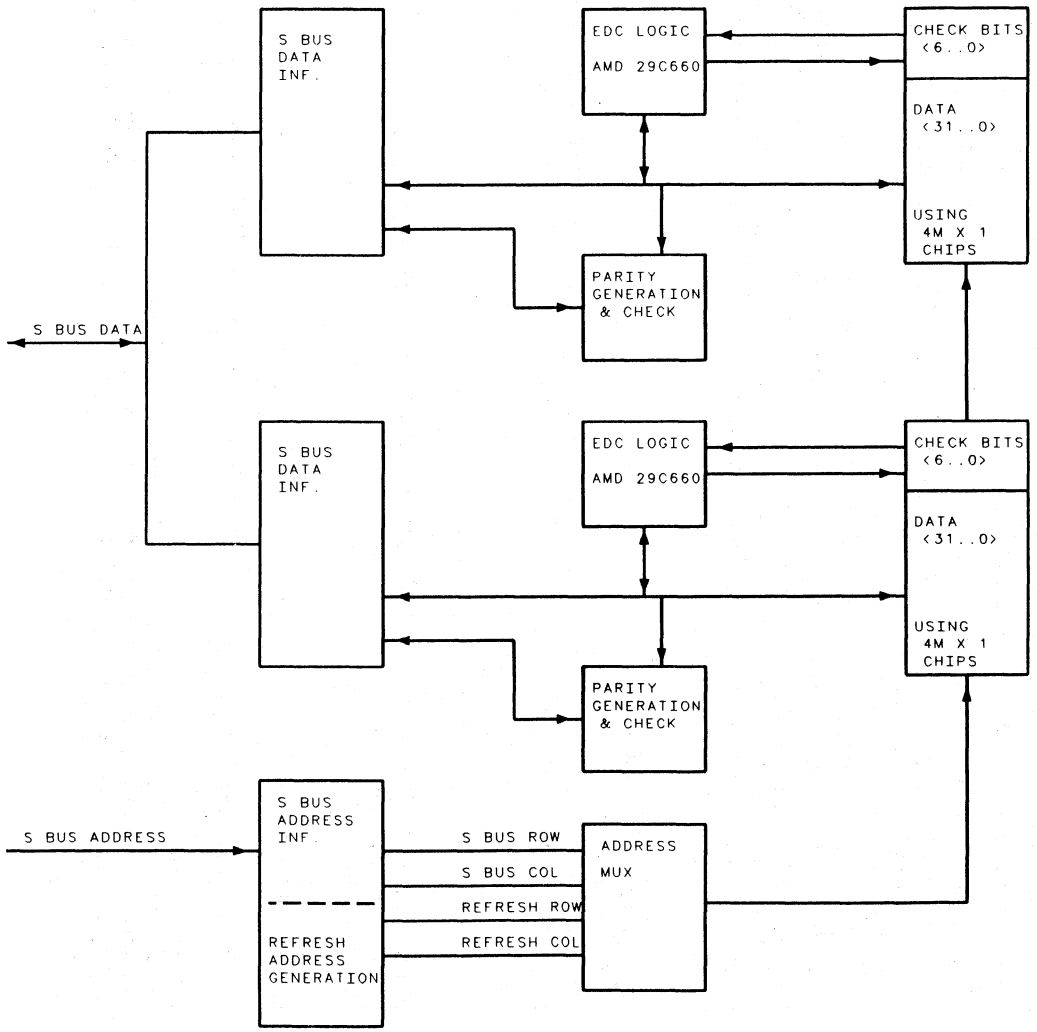


Figure 4-4. 32MB ECC DRAM Memory Board Block Diagram

Error Detection and Correction

As noted in the introduction, there are two memory banks, each with an Am29C660 error detection and correction circuit (EDC). The Am29C660 contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the Am29C660 detects and corrects all single-bit errors and detects all double- and some triple-bit errors. For 32-bit words, seven check bits are used.

Data items which have single-bit errors are corrected by the EDC and passed on to the MVME188A local bus (slave bus). The corrected data is also written back to the DRAM by hardware control.

Multiple-bit errors which are detected by the EDC generate a bus error on the slave bus. Status is set indicating the source of the error and the yellow MEM ERROR LED is turned on.

Data parity is generated when the data is passed to the slave bus. Parity is also checked when data is written to the memory.

Refresh Scrubbing

The 32MB of memory is examined by the refresh logic every 63 seconds. Single-bit errors are corrected and written back to DRAM. Multiple-bit errors are ignored. Single-bit errors encountered during refresh are counted by the error counter.

Hardware Diagnostic Features and Operating Modes

This section describes operating modes and other hardware diagnostic features.

Mode Selection

Modes are selected via the CSR.

Normal Mode

The normal mode is set by writing a 00 to CSR bits 2 and 1. Single-bit error detection/correction, double error detection, refresh scrubbing, parity error detection, and error counting are enabled in this mode.

Diagnostic Generate Mode

Diagnostic generate mode is set by writing a 01 to CSRbits 2 and 1. While in diagnostic generate mode, software can write data/check bit error sets to DRAM locations. These locations can be accessed while in the normal mode to test the error detection/correction features of the memory module. The actual check bit pattern delivered to the DRAM array while in the diagnostic generate mode is taken from the diagnostic latch in the Am29C660. Refer to the AMD data sheet (listed in Chapter 1) for the diagnostic latch definition.

Note that refresh can have two unexpected results while using the diagnostic generate mode. Both are a result of the refresh scrubbing action. If refresh encounters a single-bit error while in the diagnostic generate mode, it will try to correct the error. Since the correct mode is not enabled, it will write the uncorrected data back to memory but generate new check bits. Therefore it corrects the check bits. For this case, test software will access the location but not get corrected data. The second result which must be considered is that once the error has been planted and the board has been switched back to the normal mode, refresh may correct the data before test software accesses the location. In this case, the correction is transparent. The only way to determine if refresh or software access fixed the error location is to examine the error counter before the software access. The effects that refresh can have on this test can be minimized. The test code should minimize the windows in which these incidents can occur. It takes refresh 63 seconds to cycle through the DRAM array. Therfore, testing at two points in the memory map which are 16MB apart will guarantee that only one of the regions will be affected.

Diagnostic Detect Mode

The diagnostic detect mode is set by writing a 10 to CSR bits 2 and 1. This mode isolates the Am29C660 from its surrounding logic for testing purposes. It allows better fault resolution for diagnostic software.

Pass Through Mode

The pass through mode is set by writing a 11 to CSR bits 2 and 1. While in pass through mode, the DRAM array is unprotected. Software can perform tests without the error correcting circuits masking the effects of a defective memory device. Valid check bits are not generated in the pass thru mode. Reset will set the state of the board to pass through mode.

Initialize Command

The initialize command is set by reading the diagnostic latch. It is exited by reading the CSR. The initialize mode should not be entered unless the memory controller is in the pass through mode with fast refresh enabled. This command will modify the checkbits of each DRAM location so that they will not represent an error condition when accessed in the normal mode. The initialize command takes 131,072,000 clock cycles to complete. This is 5.24 seconds at 25 MHz.

Note that the diagnostic latch is a write-only latch. Therefore, reading the latch to set the initialize commnd does not restrict programming flexibility.

This command should be executed before the normal mode is entered.

Refresh Period

CSR bit 0 controls the refresh period. The normal refresh period is 15.4 µsec. Setting CSR bit 0 to a one will decrease the refresh period to 1.4 µsec. This will reduce the test time of the refresh scrubbing function.

Error Counter

There is an eight bit counter which is accessable at the CSR. It will increment when single-bit errors are encountered. The single-bit errors can be a result of refresh or bus access. It does not increment for double-bit errors. It is cleard under CSR control or by reset.

Syndrome Bits

The CSR has two seven-bit fields which captures the syndrome bits of the last memory read access. The contents are preserved when a single-bit error is detected by the EDC. The syndrome bits will point to the memory device which caused the error. Reading the CSR will re-enable the captures. Refer to the table for syndrome bit encoding in the Am29C660 data sheet.

Parity Detection

All write activity to the module will be checked for correct parity. Parity status is maintained in the CSR. The parity status can be cleared by a power-on reset or by writing a bit to the CSR.

Performance

The number of wait states required for specific cycle types are indicated in the following table. In the case of refresh, the number indicates clock ticks required to complete the refresh cycle.

Cycle Type	Wait States	Notes
refresh	7	no error
refresh	12	with correctable error
read cycles	4	no error
read cycle	9	with correctable error
write word	2	don't care error condition
write burst	2	don't care error condition
write byte	4	no error
write byte	8	with correctable error

Note that the two wait states for writes do not affect system performance since the caches requires two wait states for snooping. The performance parameters are for isolated cycles. Since the memory cycles are write posted, a second consecutive write cycle would require an additional three ticks. The seven ticks for refresh includes the RAS pre-charge time required for the pending cycle.

Status Indicator LED

The 32MB ECC DRAM memory board front panel contains an LED that indicates MEM ERROR. Refer to Chapter 3 for detailed information.

Onboard Switch

The 32MB ECC DRAM memory board contains an eight-segment switch SW1. Its segments are used to select the memory board's position in the MVME188A slave bus memory map, and to decode and set addresses for the Control and Status Registers (CSRs) and the diagnostic latch registers. Chapter 2 has further information on these switch segments. The registers are described in detail in the following sections of the present chapter (Chapter 4).

32MB ECC DRAM Memory Board Registers

The diagnostic latch register is described in the Am29C660 data sheet listed in Chapter 1. The CSR is described as follows.

Programming the Control and Status Register

The control and status register (CSR) is used to program the operation of the MVME288EC-32. This register is read/write. Various operating modes can be programmed. They include error checking and correcting on or off, diagnostic modes, and fast refresh cycling.

The CSR register definition in Table 4-15 has four lines. Line 1 shows the bits defined by this table. Line 2 defines the name of the bits in the register. Line 3 defines the operations possible on the register bits as follows.

1 R - This bit is a read-only status bit.

2 R/W - This bit is readable and writable.

3 R/W1 - This bit can be read and it is cleared by writing a 1 to it.

Line 4 defines the state of the bit following a reset as defined below.

1 P - The bit is affected by power-up reset.

2 S - The bit is affected by SYSRESET.

3 L - The bit is affected by local reset.

4 X - The bit is not affected by reset.

Table 4-15. ECC Control and Status Register

BIT	31	30	29	28	27	26	25	24
NAME	DBE	SOD6	SOD5	SOD4	SOD3	SOD2	SOD1	SOD0
OPER	R/W1	R	R	R	R	R	R	R
RESET	0 PSL	x	x	x	x	x	x	x

The DBE bit indicates that a double-bit error was detected. A bus error (M bus error) is generated for this cycle. Reset to zero. Bit 31 is cleared by writing a 1 to that location. DBE will also drive the MEM ERROR LED.

The DRAM array is tested for errors across two words. It is possible to access a word which does not have an error but still generate an error. This can be prevented by initializing all of memory before enabling error checking.

SOD[6..0] are the syndrome bits for the odd bank of memory. When the field is non-zero, an error has been detected. Refer to a table in the Am29C660 data sheet for the error definition. The first error encountered will freeze the state of this field. Examining the CSR will allow the syndrome bits of the next error to be captured.

The control and status bits are defined below.

Bit 31 Double bit error has occurred

Bit 30 Syndrome bit 6 (Odd bank)

Bit 29 Syndrome bit 5 (Odd bank)

Bit 28 Syndrome bit 4 (Odd bank)

Bit 27 Syndrome bit 3 (Odd bank)

Bit 26 Syndrome bit 2 (Odd bank)

Bit 25 Syndrome bit 1 (Odd bank)

Bit 24 Syndrome bit 0 (Odd bank)

Table 4-15. ECC Control and Status Register (cont'd)

BIT	23	22	21	20	19	18	17	16
NAME	ECNT7	ECNT6	ECNT5	ECNT4	ECNT3	ECNT2	ECNT1	ECNT0
OPER	R	R	R	R	R	R	R	R
RESET	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL	0 PSL

ECNT[7..0] represent the contents of the error counter. ECNT counts all single-bit errors that are encountered. The errors can be a result of refresh or bus access. Multiple-bit errors are not counted. The ECNT bits are reset to 0 and cleared by control and status register bit 3.

The control and status bits are defined below.

Bit 23 Error count bit 7

Bit 22 Error count bit 6

Bit 21 Error count bit 5

Bit 20 Error count bit 4

Bit 19 Error count bit 3

Bit 18 Error count bit 2

Bit 17 Error count bit 1

Bit 16 Error count bit 0

Table 4-15. ECC Control and Status Register (cont'd)

BIT	15	14	13	12	11	10	9	8
NAME	SEV6	SEV5	SEV4	SEV3	SEV2	SEV1	SEV0	OVF
OPER	R	R	R	R	R	R	R	R
RESET	x	x	x	x	x	x	x	0 PSL

SEV[6..0] access the syndrome storage latch of the even bank of memory. The control of this field is the same as the syndrome bits for the odd field.

OVF, the overflow bit, indicates the error counter exceeded the count of 255. It is reset to 0 and is read only, cleared by bit 3. Note that the OVF indicator is passed through the syndrome latch. Therefore, the CSR must be polled twice to get accurate OVF status (because the latch may have saved the single-bit error syndrome information before the error counter reached a terminal count). Also, a memory reference to the board must occur between the two reads of the CSR so that the syndrome latch is updated.

The control and status bits are defined below.

Bit 15 Syndrome bit 6 (Even bank)

Bit 14 Syndrome bit 5 (Even bank)

Bit 13 Syndrome bit 4 (Even bank)

Bit 12 Syndrome bit 3 (Even bank)

Bit 11 Syndrome bit 2 (Even bank)

Bit 10 Syndrome bit 1 (Even bank)

Bit 9 Syndrome bit 0 (Even bank)

Bit 8 Overflow bit

Table 4-15. ECC Control and Status Register (cont'd)

BIT	7	6	5	4	3	2	1	0
NAME	SPE3	SPE2	SPE1	SPE0	CCLR	MOD1	MOD0	FREF
OPER	R	R	R/W	R	R/W	R/W	R/W	W
RESET	0 P	0 P	0 P	0 P	1 PSL	1 PSL	1 PSL	0 PSL

SPE[3..0] set to a 1 indicates that a bus parity error occured. The parity error was ether generated by the processor board or data/noise was clocked into the bus register on the ECC board. The system is not notified if this condition is detected. These bits must be polled or examined after a malfunction has been detected. They are cleared to zero by a power-on reset. Bits 4 thru 7 are also cleared by writing a 1 to bit 5. Bit 4 is for the least significant byte (highest address) and bit 7 is for the most significant byte.

CCLR: writing CCLR to a 1 will clear the ECNT field and OVF bit. Reading CCLR indicates the status of the board-level state machine. This state machine is implemented with three separate sequencer circuits. If a malfunction occurs and the seperate circuits are not synchronized, CCLR will be set to a 0. A 0 is the error condition.

MOD[1..0]: the mode bits control the mode inputs to each of the error checking and correcting circuits (Am29C660). Reset to 11 (pass through mode). MOD[1..0] are read/write bits.

FREF: setting the FREF to a one will decrease the refresh period to 1.44 μsec. This feature is used to decrease the test time of refresh scrubbing. Writing FREF to a zero will reduce the refresh rate to a 15.4 μsec period.

The control and status bits are defined below.

Bit 7 S bus parity error bit3

Bit 6 S bus parity error bit2

Bit 5 S bus parity error bit1

Bit 4 S bus parity error bit0

Bit 3 Count Clear bit

Bit 2 Mode bit 1

Bit 1 Mode bit 0

Bit 0 Fast refresh bit

Main Logic Board (CPU Board) with HYPERmodule Mezzanine Module Functional Description

The main logic board on the right of the MVME188A board set contains three female, 100-pin miniature Rib-Cage connectors which mate with male connectors on the mezzanine module. The connectors serve both as a mechanical mounting and to provide paths for the CMMU memory bus (M bus). Also contained on the main logic board are the buffers, latches, decoders, and control logic required for the S bus, and a VMEbus interface.

For a block diagram of the main logic board, see Figure 4-5.

CPU/CMMU Cluster HYPERmodule

The MVME188A family realizes the computing power of the MC88100 RISC microprocessor and the MC88200 or MC88204 Cache/Memory Management Unit (CMMU) using a mezzanine board called a HYPERmodule which has one or more extremely fast CPU clusters. Each cluster has a single CPU and from 32KB to 512KB of cache memory. All modules run at 25 MHz.

Number Of CPUs	Number Of CMMUs	Description	Board Type
1	8	Single Processor Module with 128KB or 512KB cache memory (64KB or 256KB data cache, 64KB or 256KB instruction cache).	HM88K-1P128 or HM88K-1P512
1	4	Single Processor Module with 64KB or 256KB cache memory (32KB or 128KB data cache, 32KB or 128KB instruction cache).	HM88K-1P64 or HM88K-1P256
1	2	Single Processor Module with 32KB or 128KB cache memory (16KB 0r 64KB data cache, 16KB or 64KB instruction cache).	HM88K-1P32 or HM88K-1P128
2	8	Dual Processor Module with 128KB or 512KB cache memory. Each processor has 32KB or 128KB data cache, 32KB or 128KB instruction cache.	HM88K-2P128 or HM88K-2P512
2	4	Dual Processor Module with 64KB or 256KB cache memory. Each processor has 16KB or 64KB data cache, 16KB or 64KB instruction cache.	HM88K-2P64 or HM88K-2P256
4	8	Quad Processor Module with 128KB or 512KB cache memory. Each processor has 16KB or 64KB data cache, 16KB or 64KB instruction cache.	HM88K-4P128 or HM88K-4P512

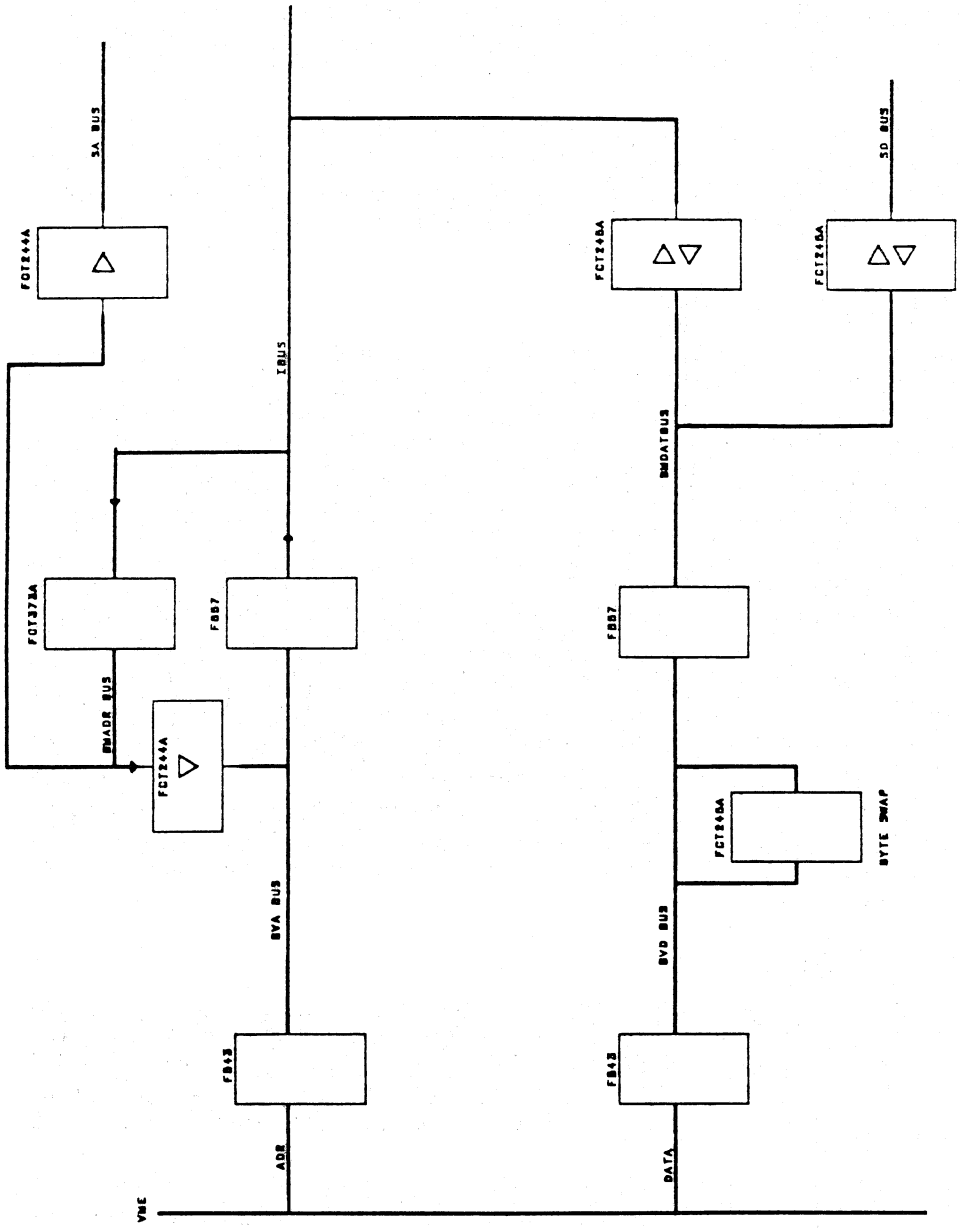


Figure 4-5. Main Logic Board Block Diagram

HYPERmodules measure only 3.5 inches by 8.5 inches (89 mm x 216 mm). They are made using the latest techniques in miniature packaging and printed circuit board manufacturing, adding high quality and reliability to match their performance.

A HYPERmodule connects to the main logic board via a multiplexed M bus interface which has been enhanced with additional signals to support multiple CPU clusters, multiple CMMUs, fault checking modes, and HYPERmodule factory testing. (This is sometimes referred to as the Enhanced M Bus.)

M Bus Address Decoder

The complex states of the multiplexed M bus are closely followed by an SRAM address decoder that determines which resource is the current M bus slave. Rather than configured through the use of switches or jumpers, the volatile SRAM is initialized by a local CPU running out of the ROM on the system controller board. This readily supports the various CPU cluster configurations on the mezzanine module and the address space assignments required by an application.

The SRAM M bus address decoder can map any 4MB segment of M bus address space to utility space (local SRAM, ROM, control registers, etc), DRAM space (local memory boards), VMEbus A32 address space, or VMEbus A24 address space. Within the 4MB utility address space, hardware reserves a 2MB space for ROM, a 1MB space for SRAM, and 1MB of CPU address space divided into three parts: CMMU register space, a block for the local control/status registers, and the VMEbus A16 short I/O address space. Because the CMMU chips require a 1MB space at the top of memory, the top 4MB of M bus space must be mapped to utility space. A power-on reset causes the entire M bus address space to be mapped to the utility address space, which enables the initial bootstrap sequences to be fetched from ROM.

Cache Coherency

The CMMU chips support a cache coherency protocol in the respect that all eight caches contain a coherent image of their shared RAM on the M bus. For the fastest operation, it is suggested that the copy back mode be used to maintain the cache so that most writes are zero wait-state P bus accesses rather than multiple cycle M bus accesses.

Random access memory on the DRAM module is coherent with the VMEbus. The SRAM VMEbus slave address decoder can be configured to request snooping on every VMEbus access of any MVME188A resource, and for the mezzanine module CMMUs to intervene when the local memory and local caches are not coherent.

Snooping is programmable on any 4MB boundary in the VMEbus slave address decoder. It should be noted that the RISC complex is not coherent with the VMEbus; this requires, in systems with multiple masters on the VMEbus, that memory pages which must be accessed across the VMEbus be marked cache inhibit. So that an operating system kernel obtains strictly current device status and control information, I/O device registers must always be marked cache inhibit. While the VMEbus itself is not snooped, any VMEbus access to local memory may be snooped if that page is snoop-enabled in the VMEbus address decoder via the WVAD and RVAD registers.

M Bus Arbiter

So that processors on the global VMEbus are not held up by local accesses of DRAM, the VMEbus slave interface on the main logic board operates in the unfair mode when accessing the M bus. The M bus arbiter accommodates this by assigning highest priority to requests from the VMEbus, followed by code CMMUs 0 through 3 and data CMMUs 0 through 3 in descending order. To equally share M bus bandwidth, CMMU chips are programmed to operate in fairness mode.

As an aid to developers of systems in which dual- and quad-processor versions of the MVME188A are used, a status register reports the CPU/CMMU configuration using the states of the M bus acknowledge lines. This permits the startup master to be selected by code in ROM.

M Bus Master/Slave Interface To VMEbus

For all accesses to VMEbus peripherals such as hard and floppy disk drives, 9-track and streaming tape drives, accesses to graphics systems, and accesses to networks, the SRAM M bus address decoder determines whether the access is to VMEbus A32, A24, or A16 address space. A second SRAM address decoder allows the DRAM board and other MVME188A resources to be mapped anywhere in VMEbus address space. This permits any VMEbus master, such as an I/O controller, to access any M bus address space.

VMEbus Slave Address Decoder

Rather than jumpers, switches, and comparators, volatile SRAM is used to configure VMEbus A32 and A24 address space. Because it is volatile, boot firmware must initialize the SRAM on startup. The SRAM decoder allows the base address of the MVME188A RISC microcomputer to be located on any 4MB boundary of the A32 and A24 VMEbus address ranges.

When the MVME188A RISC microcomputer is accessed from the VMEbus, the access address is buffered to the 32-bit M bus with an 8-bit address extension register supplying the additional bits on an A24 access. The SRAM M bus decoder then performs the further decoding required to direct the access to the M bus space assigned to the utility space or the DRAM board, as required.

In addition to decoding logic, the SRAM address decoder contains bits which can be set to cause the CMMUs to snoop 4MB segments of M bus space for accesses from the VMEbus.

VMEbus Data Transfer Master/Slave Interface

The 32-bit master/slave VMEbus data transfer interface on the main logic board supports 32-bit address and data paths and supports arbitration pipelining and address pipelining. The interface includes the following features which comply with version C.1 of the VMEbus standard:

Data Transfer Master

- D32, D16, D08(EO)
- No UAT (Unaligned Address Transfer)
- A32, A24, A16
- RMW (Read-Modify-Write)
- No BLT (Block Transfer)

Data Transfer Slave

- D32, D16, D08(EO)
- UAT (Unaligned Address Transfer)
- A32, A24, A16
- ADO (Address Only Cycle)
- RMW
- No BLT

The interface does not require data transfer master unaligned address transfers, and does not support block transfer or full cache coherency with the VMEbus.

P Bus Address Decoder

At any instant, each RISC processor can generate an access to two of four 4Gb logical address spaces: user code, user data, supervisor code, or supervisor data. Code and data accesses can be simultaneous, because they occur on separate buses. For any access, the selected CMMU immediately resolves the access and,

if an M bus access is needed, translates the logical processor address to a physical memory bus access.

To obtain a choice of interleaving, an operating system kernel can use the P bus configuration registers on the main logic board. If desired, real time systems can use separate CMMUs for kernel and user tasks so that interrupt response times and critical routine execution times are independent of the cache behavior. Real time systems can also program up to one half of the CMMUs to operate in the SRAM mode and thus guarantee that critical interrupt routines encounter no wait states. The SRAM mode CMMUs can be configured to appear in the kernel logical address space only or in both the kernel and user logical address spaces.

Main Logic Board Registers

The main logic (CPU) board includes eleven registers to facilitate coherent communications with the memory and system controller boards in a VMEbus multiprocessor environment. The following sections describe in detail the main logic board registers and their functions, as well as the state of all bits at power-up time (SRST) or after an LRST.

All registers (with the exception of the GCSR) are located on 4-byte boundaries. Accesses to all of these registers should be 32-bit operations. All accesses to the GCSR should be 8-bit or 16-bit operations. Unless otherwise noted in the register bit descriptions, all undefined bits will be read as "1". Undefined bits are ignored on writes.

All of the following registers reside on the main logic (CPU) board.

CPU Control and Status Register (CCSR)

The CCSR,provides the status (on read) and control (on write) of all the hardware functions on the main logic (CPU) board. The bits are all read/write. Bit assignments are shown in Table 4-16.

Note that extreme caution should be exercised when changing the state of the MADV or VADV bits. The VADV bit is especially dangerous to modify, because the other VMEbus masters are not necessarily under direct control of the MVME188A. Any VMEbus A24 or A32 slave accesses (by these other masters) to MVME188A resources are ignored when VADV is clear. This may cause system problems if other VMEbus masters depend on MVME188A resources to function properly.

This register is cleared by SRST or LRST.

Table 4-16. CPU Control and Status Register (CCSR) Bit Definitions

BIT 3	BIT 2	BIT 1	BIT 0
WWP	PAREN	MADV	VADV

Bit 3 WWP WR/rd Write wrong parity
1 : Local CPU writes ODD parity to slave (S) bus (inverted).
0 : Local CPU writes EVEN parity to slave (S) bus (normal).
NOTE: M bus transactions are EVEN parity protected. All read cycles check for EVEN parity (if parity is enabled in CMMUs) regardless of WWP state. DRAM is the only resource protected by parity. All VMEbus writes to DRAM write EVEN parity, regardless of WWP state.
Cleared by SRST or LRST.

Bit 2 PAREN WR/rd Enable VMEbus BERR* on slave access parity error
1 : DRAM parity errors cause BERR* on VMEbus slave reads.
0 : DRAM parity errors are ignored on VMEbus slave reads.
Cleared by SRST or LRST.

Bit 1 MADV RD/WR M bus address decoder SRAM valid
1 : M bus decoder RAM has been loaded by firmware.
0 : M bus decoder RAM contents not guaranteed valid; all M bus accesses mapped to utility space.
Cleared by SRST or LRST.

CAUTION

Clearing this bit modifies the M bus address map. Use caution when clearing it, as all M bus resources except utility space will be removed from the map.

Bit 0 VADV RD/WR VMEbus address decoder SRAM valid
1 : VMEbus decoder RAM has been loaded by firmware.
0 : VMEbus decoder RAM contents not guaranteed valid; no VMEbus A24/A32 slave response.
Cleared by SRST or LRST.

Table 4-16. CPU Control and Status Register (CCSR) Bit Definitions (cont'd)

CAUTION

Clearing this bit modifies the VMEbus address map. Use caution when clearing it, as all MVME188A resources will be removed from the VMEbus A24 and A32 maps.

Never clear this bit if there is any possibility that another VMEbus master could be accessing the MVME188A via VMEbus A24 or A32 address space. If it must be cleared, the safest procedure is to write to the global reset register and let SRST clear it. Unpredictable (and perhaps fatal) behavior will result if a VMEbus A24 or A32 slave access is attempted during a CCSR access that clears VADV.

ERROR Register

The error register provides information on the cause of M bus faults. The bits are all read only, and the register is automatically cleared after it is read. This register is also cleared by SRST or LRST. Bit assignments are shown in Table 4-17.

Table 4-17. CPU/CMMU Error Register (ERROR) Bit Definitions

BIT 31-15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	BIT 0
1	MELA	PE	VMEBE	PBE3	PBE2	PBE1	PBE0	DCE3	DCE2	DCE1	DCE0	CCE3	CCE2	CCE1	CCE0

Bits 31-15 Unused, always "1".

Table 4-17. CPU/CMMU Error Register (ERROR) Bit Definitions (cont'd)

Bit 14 NELA RD only Non-existent local address
1 : An access was attempted by the local CPU to a non-mapped address on M bus since last read of error register.
0 : No access was attempted by the local CPU to any non-mapped address on M bus since last read of error register.
NOTE: An M bus fault is generated if this bit is ever set.
Cleared by SRST or LRST.

Bit 13 PE RD only M bus parity error
1 : Parity error has occurred on a VMEbus slave read of M bus since last read of error register and PAREN = 1 in the CCSR.
0 : No parity error has occurred on any VMEbus slave read of M bus since last read of error register or PAREN = 0 in the CCSR.
NOTE: A VMEbus Bus Error (BERR*) is also generated if this bit is ever set by a parity error on a VMEbus slave read of M bus while PAREN = 1.
Cleared by SRST or LRST.

Bit 12 VMEBE RD only VMEbus error on MVME188A master cycle
1 : VMEbus BERR* was asserted in response to VMEbus master access from MVME188A since last read of error register.
0 : No VMEbus BERR* was asserted in response to any VMEbus master access from MVME188A since last read of error register.
NOTE: An M bus fault is generated on the cycle that sets this bit.
Cleared by SRST or LRST.

Bits 11-8 PBE[3:0] RD only P bus error (listed below)
1 : One or more P bus errors have occurred since last read of error register.
0 : No P bus error has occurred since last read of error register.
Cleared by SRST or LRST.

3	2	1	0	Quad CPU	Dual CPU	Single CPU
X	X	X	1	CPU 0	CPU 0	CPU 0
X	X	1	X	CPU 1	CPU 1	0
X	1	X	X	CPU 2	0	0
1	X	X	X	CPU 3	0	0

NOTE: The "0" entries above mean that the error condition will never be indicated (the bit will always be "0").

Table 4-17. CPU/CMMU Error Register (ERROR) Bit Definitions (cont'd)

Bits 7-4 DCE[3:0] RD only Data CMMU error

1 : One or more data CMMU errors have occurred since last read of error register.

0 : No data CMMU error has occurred since last read of error register. Cleared by SRST or LRST.

3	2	1	0	Device Indicating Error
X	X	X	1	data CMMU 0
X	X	1	X	data CMMU 1
X	1	X	X	data CMMU 2
1	X	X	X	data CMMU 3

Bits 3-0 CCE[3:0] RD only Code CMMU error

1 : One or more code CMMU errors have occurred since last read of error register.

0 : No code CMMU error has occurred since last read of error register. Cleared by SRST or LRST.

3	2	1	0	Device Indicating Error
X	X	X	1	code CMMU 0
X	X	1	X	code CMMU 1
X	1	X	X	code CMMU 2
1	X	X	X	code CMMU 3

PCNFA and PCNFB Registers

The PCNFA register defines the operation of the P bus "A" decoder. The decoder function (and hence the PCNFA register bit definition) changes with the HYPERmodule configuration. The PCNFB register defines the operation of the P bus "B" decoder. The decoder function (and hence the PCNFB register bit definition) changes with the HYPERmodule configuration.

For a given HYPERmodule configuration, both P bus decoders are identical. Bit assignments and functions for PCNFA and PCNFB are contained in the following text and tables. Bit assignments for the PCNFA register are shown in Table 4-18; for the PCNFB register in Table 4-25. PCNFA and PCNFB are cleared by SRST or LRST.

Using HYPERmodule configuration 1 (HM88K-2P128 or HM88K-2P512) (two CPUs, eight CMMUs), PCNFA controls the P bus mapping of both code and data CMMUs 0 and 1 on CPU 0. Code and data CMMUs 2 and 3 (on CPU 1) are controlled by PCNFB.

Using HYPERmodule configuration 2 (HM88K-1P128 or HM88K-1P512) (one CPU, eight CMMUs), PCNFA controls the P bus mapping of all four code CMMUs. The data CMMU mapping is controlled by PCNFB.

Using HYPERmodule configuration 6 (HM88K-1P64 or HM88K-1P256) (one CPU, four CMMUs), PCNFA also controls the P bus mapping of both code and data CMMUs; PCNFB is not used with this HYPERmodule configuration.

HYPERmodule configurations 0, 5, and A (HM88K-4P128 or -4P512, -2P64 or 2P256, and -1P32 or 1P128, respectively) do not implement either P bus decoder "A" or "B", and consequently all writes to the PCNFA and PCNFB registers will have no effect on the P bus mapping. The PCNFA and PCNFB registers can still be accessed in these configurations; the contents are simply ignored.

In all decoder configurations, each pair of CMMUs are controlled by two register bits. The two register bits determine the method used to select one of the two CMMUs, as follows.

Bits	CMMU Selection Method
00	Low order interleave using P bus A12 bit (A14 bit using MC88204s)
01	High order interleave using P bus S (user/supervisor) bit
10	One CMMU in SRAM space for supervisor only, other CMMU covers rest of map
11	One CMMU in SRAM space for both user and supervisor, other CMMU covers rest of map

Refer to the *WHOAMI Register* description for a definition of the valid HYPERmodule configurations.

Table 4-18. P bus (A) Configuration Register (PCNFA) Bit Definitions

BIT 3	BIT 2	BIT 1	BIT 0
PCNFA3	PCNFA2	PCNFA1	PCNFA0

Bits 3-0 PCNFA[3:0] WR/rd P bus "A" address decoder configuration
These bits define the configuration of the P bus "A" decoder. Their action depends upon the HYPERmodule configuration installed. Refer to Tables 4-19, 4-20, 4-21, 4-22, 4-23, and 4-24. These bits are ignored in HYPERmodule configurations 0, 5, and A. Refer to the *WHOAMI Register* description for information on how to determine the HYPERmodule configuration.
Cleared by SRST or LRST.

Table 4-19. PCNFA Register Bit Definitions for HYPERmodule HM88K-2P128, Configuration 1 (2 CPU/8 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	CPU 0 Code CMMU (0,1) Mode	CPU 0 Data CMMU (0,1) Mode
0	0	0	0	0	A12 split	A12 split
1	0	0	0	1	A12 split	spv/usr split
2	0	0	1	0	A12 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A12 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A12 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A12 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A12 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CPU 0 CODE and DATA CMMUs; PCNFB maps CPU 1 CODE and DATA CMMUs.

Table 4-20. PCNFA Register Bit Definitions for HYPERmodule HM88K-2P512, Configuration 1 (2 CPU/8 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	CPU 0 Code CMMU (0,1) Mode	CPU 0 Data CMMU (0,1) Mode
0	0	0	0	0	A14 split	A14 split
1	0	0	0	1	A14 split	spv/usr split
2	0	0	1	0	A14 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A14 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A14 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A14 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A14 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CPU 0 CODE and DATA CMMUs; PCNFB maps CPU 1 CODE and DATA CMMUs.

Table 4-21. PCNFA Register Bit Definitions for HYPERmodule HM88K-1P128, Configuration 2 (1 CPU/8 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	Code CMMU (0,1) Mode	Code CMMU (2,3) Mode
0	0	0	0	0	A12, A14 split	A12, A14 split
1	0	0	0	1	spv,A14 split	A12, A14 split
2	0	0	1	0	spv only SRAM /shared CMMU split	A12, A14 split
3	0	0	1	1	shared SRAM /shared CMMU split	A12, A14 split
4	0	1	0	0	A12, A14 split	spv, A14 split
5	0	1	0	1	spv,A14 split	spv, A14 split
6	0	1	1	0	spv only SRAM /shared CMMU split	spv, A14 split
7	0	1	1	1	shared SRAM /shared CMMU split	spv, A14 split
8	1	0	0	0	A12, A14 split	spv only SRAM /shared CMMU split
9	1	0	0	1	spv,A14 split	spv only SRAM /shared CMMU split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
C	1	1	0	0	A12, A14 split	shared SRAM /shared CMMU split
D	1	1	0	1	spv,A14 split	shared SRAM /shared CMMU split
E	1	1	1	0	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CODE CMMUs; PCNFB maps DATA CMMUs.

Table 4-22. PCNFA Register Bit Definitions for HYPERmodule HM88K-1P512, Configuration 2 (1 CPU/8 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	Code CMMU (0,1) Mode	Code CMMU (2,3) Mode
0	0	0	0	0	A14, A16 split	A14, A16 split
1	0	0	0	1	spv,A16 split	A14, A16 split
2	0	0	1	0	spv only SRAM /shared CMMU split	A14, A16 split
3	0	0	1	1	shared SRAM /shared CMMU split	A14, A16 split
4	0	1	0	0	A14, A16 split	spv, A16 split
5	0	1	0	1	spv,A16 split	spv, A16 split
6	0	1	1	0	spv only SRAM /shared CMMU split	spv, A16 split
7	0	1	1	1	shared SRAM /shared CMMU split	spv, A16 split
8	1	0	0	0	A14, A16 split	spv only SRAM /shared CMMU split
9	1	0	0	1	spv,A16 split	spv only SRAM /shared CMMU split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
C	1	1	0	0	A14, A16 split	shared SRAM /shared CMMU split
D	1	1	0	1	spv,A16 split	shared SRAM /shared CMMU split
E	1	1	1	0	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CODE CMMUs; PCNFB maps DATA CMMUs.

Table 4-23. PCNFA Register Bit Definitions for HYPERmodule HM88K-1P64, Configuration 6 (1 CPU/4 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	Code CMMU (0,1) Mode	Data CMMU (0,1) Mode
0	0	0	0	0	A12 split	A12 split
1	0	0	0	1	A12 split	spv/usr split
2	0	0	1	0	A12 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A12 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A12 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A12 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A12 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CODE and DATA CMMUs; PCNFB is not used.

Table 4-24. PCNFA Register Bit Definitions for HYPERmodule HM88K-1P256, Configuration 6 (1 CPU/4 CMMU)

PCNFA Mode	PCNFA3	PCNFA2	PCNFA1	PCNFA0	Code CMMU (0,1) Mode	Data CMMU (0,1) Mode
0	0	0	0	0	A14 split	A14 split
1	0	0	0	1	A14 split	spv/usr split
2	0	0	1	0	A14 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A14 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A14 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A14 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A14 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFA maps CODE and DATA CMMUs; PCNFB is not used.

Table 4-25. P bus (B) Configuration Register (PCNFB) Bit Definitions

BIT 3	BIT 2	BIT 1	BIT 0
PCNFB3	PCNFB2	PCNFB1	PCNFB0

Bits 3:0 PCNFB[3:0] WR/rd P bus "B" address decoder configuration
These bits define the configuration of the P bus "B" decoder. Their action depends upon the HYPERmodule configuration installed. Refer to Tables 4-26, 4-27, 4-28, and 4-29. These bits are ignored in HYPERmodule configurations 0, 5, 6, and A. Refer to the *WHOAMI Register* description for information on how to determine the HYPERmodule configuration.
Cleared by SRST or LRST.

Table 4-26. PCNFB Register Bit Definitions for HYPERmodule HM88K-2P128, Configuration 1 (2 CPU/8 CMMU)

PCNFB Mode	PCNFB3	PCNFB2	PCNFB1	PCNFB0	CPU 1 Code CMMU (2,3) Mode	CPU 1 Data CMMU (2,3) Mode
0	0	0	0	0	A12 split	A12 split
1	0	0	0	1	A12 split	spv/usr split
2	0	0	1	0	A12 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A12 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A12 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A12 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A12 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFB maps CPU 1 CODE and DATA CMMUs; PCNFA maps CPU 0 CODE and DATA CMMUs.

Table 4-27. PCNFB Register Bit Definitions for HYPERmodule HM88K-2P512, Configuration 1 (2 CPU/8 CMMU)

PCNFB Mode	PCNFB3	PCNFB2	PCNFB1	PCNFB0	CPU 1 Code CMMU (2,3) Mode	CPU 1 Data CMMU (2,3) Mode
0	0	0	0	0	A14 split	A14 split
1	0	0	0	1	A14 split	spv/usr split
2	0	0	1	0	A14 split	spv only SRAM /shared CMMU split
3	0	0	1	1	A14 split	shared SRAM /shared CMMU split
4	0	1	0	0	spv/usr split	A14 split
5	0	1	0	1	spv/usr split	spv/usr split
6	0	1	1	0	spv/usr split	spv only SRAM /shared CMMU split
7	0	1	1	1	spv/usr split	shared SRAM /shared CMMU split
8	1	0	0	0	spv only SRAM /shared CMMU split	A14 split
9	1	0	0	1	spv only SRAM /shared CMMU split	spv/usr split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
C	1	1	0	0	shared SRAM /shared CMMU split	A14 split
D	1	1	0	1	shared SRAM /shared CMMU split	spv/usr split
E	1	1	1	0	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFB maps CPU 1 CODE and DATA CMMUs; PCNFA maps CPU 0 CODE and DATA CMMUs.

Table 4-28. PCNFB Register Bit Definitions for HYPERmodule HM88K-1P128, Configuration 2 (1 CPU/8 CMMU)

PCNFB Mode	PCNFB3	PCNFB2	PCNFB1	PCNFB0	Data CMMU (0,1) Mode	Data CMMU (2,3) Mode
0	0	0	0	0	A12, A14 split	A12, A14 split
1	0	0	0	1	spv,A14 split	A12, A14 split
2	0	0	1	0	spv only SRAM /shared CMMU split	A12, A14 split
3	0	0	1	1	shared SRAM /shared CMMU split	A12, A14 split
4	0	1	0	0	A12, A14 split	spv, A14 split
5	0	1	0	1	spv,A14 split	spv, A14 split
6	0	1	1	0	spv only SRAM /shared CMMU split	spv, A14 split
7	0	1	1	1	shared SRAM /shared CMMU split	spv, A14 split
8	1	0	0	0	A12, A14 split	spv only SRAM /shared CMMU split
9	1	0	0	1	spv,A14 split	spv only SRAM /shared CMMU split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
C	1	1	0	0	A12, A14 split	shared SRAM /shared CMMU split
D	1	1	0	1	spv,A14 split	shared SRAM /shared CMMU split
E	1	1	1	0	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFB maps DATA CMMUs; PCNFA maps CODE CMMUs.

Table 4-29. PCNFB Register Bit Definitions for HYPERmodule HM88K-1P512, Configuration 2 (1 CPU/8 CMMU)

PCNFB Mode	PCNFB3	PCNFB2	PCNFB1	PCNFB0	Data CMMU (0,1) Mode	Data CMMU (2,3) Mode
0	0	0	0	0	A14, A16 split	A14, A16 split
1	0	0	0	1	spv,A16 split	A14, A16 split
2	0	0	1	0	spv only SRAM /shared CMMU split	A14, A16 split
3	0	0	1	1	shared SRAM /shared CMMU split	A14, A16 split
4	0	1	0	0	A14, A16 split	spv, A16 split
5	0	1	0	1	spv,A16 split	spv, A16 split
6	0	1	1	0	spv only SRAM /shared CMMU split	spv, A16 split
7	0	1	1	1	shared SRAM /shared CMMU split	spv, A16 split
8	1	0	0	0	A14, A16 split	spv only SRAM /shared CMMU split
9	1	0	0	1	spv,A16 split	spv only SRAM /shared CMMU split
A	1	0	1	0	spv only SRAM /shared CMMU split	spv only SRAM /shared CMMU split
B	1	0	1	1	shared SRAM /shared CMMU split	spv only SRAM /shared CMMU split
C	1	1	0	0	A14, A16 split	shared SRAM /shared CMMU split
D	1	1	0	1	spv,A16 split	shared SRAM /shared CMMU split
E	1	1	1	0	spv only SRAM /shared CMMU split	shared SRAM /shared CMMU split
F	1	1	1	1	shared SRAM /shared CMMU split	shared SRAM /shared CMMU split

NOTE: PCNFB maps DATA CMMUs; PCNFA maps CODE CMMUs.

EXTAD Register

The extended address register provides the upper eight M bus address bits when the MVME188A is addressed as a VMEbus A24 slave. This register must be initialized before any local addresses are mapped to VMEbus A24 space. It is unaffected by either SRST or LRST. Bit assignments are shown in Table 4-30.

Table 4-30. VMEbus Slave Access Address Extension Register (EXTAD) Bit Definitions

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
EXTAD31	EXTAD30	EXTAD29	EXTAD28	EXTAD27	EXTAD26	EXTAD25	EXTAD24

Bits 7-0 EXTAD[31:24] WR/rd M bus address bits A24-A31 on VMEbus A24 slave access

EXTAD[31] drives M bus address 31 (BMADR[31]) on VMEbus A24 slave access.

:

:

:

EXTAD[24] drives M bus address 24 (BMADR[24]) on VMEbus A24 slave access.

Unaffected by SRST or LRST.

EXTAM Register

The extended address-modifier register supplies AM0 through AM3 address modifiers to the VMEbus when the MVME188A is bus master. Bit assignments are shown in Table 4-31. (The upper two address modifier bits (AM4 and AM5) are derived from the M bus decoder for the selected page per the following information.) This register is unaffected by SRST or LRST. It should always be initialized before the MVME188A accesses any VMEbus resource.

M Bus Decoder Contents			VMEbus Address Modifier Bits		VMEbus
MDS2	MDS1	MDS0	AM5	AM4	Address Space
1	0	0	0	0	A32 (extended)
1	0	1	1	1	A24 (standard)
1	1	1	1	0	A16 (short)

Table 4-31. VMEbus Master Access Address Modifier Register (EXTAM) Bit Definitions

BIT 3	BIT 2	BIT 1	BIT 0
EXTAM3	EXTAM2	EXTAM1	EXTAM0

Bits 3-0 EXTAM[3:0] WR/rd Defines VMEbus AM3-AM0 on VMEbus master access

EXTAM[3] drives VMEbus AM3 during accesses when MVME188A is master.

:

:

EXTAM[0] drives VMEbus AM0 during accesses when MVME188A is master.

Unaffected by SRST or LRST.

NOTE: AM[5:4] on VMEbus master access are derived from the M bus address decoder SRAM contents. Refer to previous text and *WMAD Register* definition.

WHOAMI Register

The WHOAMI register provides two pieces of information: the configuration of the HYPERmodule and the number of the CMMU that is current M bus master; it is unaffected by SRST or LRST. Bit assignments are shown in Table 4-35.

There are six possible HYPERmodule configurations (twelve possible part numbers) built from three different base circuit boards. Three are fully stuffed versions, two are half populated versions, and one is a quarter populated version. The three PWB types are described in Table 4-32. Their valid configurations are defined in Table 4-33.

Table 4-32. HYPERmodule PWB Type Summary

PWB Type	Description
1	1 CPU, 2 P buses, 4 CMMUs on each P bus, 2 P bus decoders (code, data)
2	2 CPU, 4 P buses, 2 CMMUs on each P bus, 2 P bus decoders (CPU 0, CPU 1)
3	4 CPU, 8 P buses, 1 CMMU on each P bus, no P bus decoder

Table 4-33. Valid HYPERmodule Configurations

HYPERmodule		#	#	PWB	Stuff	Assigned CMMUs (Code And Data)			
Part Number	Config	CPU	CMMU	Type	Option	CPU0	CPU1	CPU2	CPU3
HM88K-4P128 or HM88K-4P512	0	4	8	3	full	0	1	2	3
HM88K-2P128 or HM88K-2P512	1	2	8	2	full	0,1	2,3	-	-
HM88K-1P128 or HM88K-1P512	2	1	8	1	full	0-3	-	-	-
HM88K-2P64 or HM88K-2P256	5	2	4	3	half	0	1	-	-
HM88K-1P64 or HM88K-1P256	6	1	4	2	half	0,1	-	-	-
HM88K-1P32 or HM88K-1P128	A	1	2	3	quarter	0	-	-	-

The current M bus master is always one of the data CMMUs (because the WHOAMI register is in data space); the CPU number that is accessing the M bus via this CMMU depends on the contents of the MZCNF bits. Refer to Table 4-34.

Table 4-34. Data CMMU Number to CPU Number Translation

WHOAMI Contents		
HYPERmodule Config	CMMU Code	CPU Number
0	1	CPU 0
0	2	CPU 1
0	4	CPU 2
0	8	CPU 3
1	1	CPU 0
1	2	CPU 0
1	4	CPU 1
1	8	CPU 1
2	X	CPU 0 (single processor)
5	1	CPU 0
5	2	CPU 1
6	X	CPU 0 (single processor)
A	X	CPU 0 (single processor)

Table 4-35. WHOAMI Register (WHOAMI) Bit Definitions

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
MZCNF3	MZCNF2	MZCNF1	MZCNF0	DCMMU3	DCMMU2	DCMMU1	DCMMU0

Bits 7-4 MZCNF[3:0] RD only HYPERmodule configuration
The bits indicate the configuration of the HYPERmodule that is currently plugged into the main logic (CPU) board. They are unaffected by SRST or LRST.

Config	MZCNF3	MZCNF2	MZCNF1	MZCNF0	HYPERmodule Configuration
0	0	0	0	0	4 CPU, 8 CMMU, type 3 PWB fully stuffed
1	0	0	0	1	2 CPU, 8 CMMU, type 2 PWB fully stuffed
2	0	0	1	0	1 CPU, 8 CMMU, type 1 PWB fully stuffed
5	0	1	0	1	2 CPU, 4 CMMU, type 3 PWB half stuffed
6	0	1	1	0	1 CPU, 4 CMMU, type 2 PWB half stuffed
A	1	0	1	0	1 CPU, 2 CMMU, type 3 PWB quarter stuffed

Bits 3-0 DCMMU[3:0] RD only Data CMMU that is current M bus master
This register contains a bit indicating which data CMMU is M bus master, encoded as follows. These bits are unaffected by SRST or LRST.

CMMU Code	DCMMU3	DCMMU2	DCMMU1	DCMMU0	M Bus Master
1	0	0	0	1	data CMMU 0
2	0	0	1	0	data CMMU 1
4	0	1	0	0	data CMMU 2
8	1	0	0	0	data CMMU 3

WMAD Register

The write M bus address decoder register is used to set up the M bus physical address map. It is write only, and contains both the page address (upper data bits) and the code for the device to be mapped (lower data bits). Bit assignments are shown in Table 4-36.

Although it is permissible to write to the WMAD register while the MADV bit is set in the CCSR, it is not recommended. The mapping change made by the WMAD write will take effect on the next M bus cycle.

> **CAUTION**
>
> **Extreme caution must be exercised when writing to WMAD; dynamically changing the M bus mapping with the mapper enabled can have disastrous results. This is especially true in the multi-CPU case, where it may be difficult to predict what part of M bus space each processor is using at any given moment.**

Because it is write-only, the WMAD register is unaffected by SRST or LRST.

Table 4-36. Write M Bus Address Decoder Register (WMAD) Bit Definitions

BIT 31	30	29	28	27	26	25	24	23	22	BITS 21-3	2	1	BIT 0
MPN9	MPN8	MPN7	MPN6	MPN5	MPN4	MPN3	MPN2	MPN1	MPN0	X	MDS2	MDS1	MDS0

Bits 31-22 MPN[9:0] WR only Page number (one of 1024 pages: page size = 4MB)
MPN[9:0] is supplied by the M bus master on data bits 31:22.

Bits 21-3 Unused, can be 0 or 1

Bits 2-0 MDS[2:0] WR only M bus device select
This is the value that will be written into the M bus decoder for the page whose number is MPN[9:0].

Table 4-36. Write M Bus Address Decoder Register (WMAD) Bit Definitions (cont'd)

MDS2	MDS1	MDS0	M Bus Device
0	0	0	memory board 0
0	0	1	memory board 1
0	1	0	memory board 2
0	1	1	memory board 3
1	0	0	VMEbus A32
1	0	1	VMEbus A24
1	1	0	none - causes error on access
1	1	1	utility/CMMU/VMEbus A16 space select

RMAD Register

The read M bus address decoder register is used to verify the configuration of the M bus physical address map. It is read/write; on read it returns both the page address (upper 10 data bits) and the code for the device mapped to that page (lower three data bits). On write, an address is written to the upper bits of the RMAD register; this address is used on a subsequent read as a page address. Bit assignments are shown in Table 4-37. The RMAD register is unaffected by SRST or LRST.

Table 4-37. Read M Bus Address Decoder Register (RMAD) Bit Definitions

BIT 31	30	29	28	27	26	25	24	23	22	BITS 21-3	2	1	BIT 0
MPN9	MPN8	MPN7	MPN6	MPN5	MPN4	MPN3	MPN2	MPN1	MPN0	X	MDS2	MDS1	MDS0

Bits 31-22 MPN[9:0] WR/RD Page number (one of 1024 pages: page size = 4MB)

On write, MPN[9:0] is supplied by the M bus master on data bits 31:22, and is written into the read M bus address decoder (RMAD) register. On read, the contents of the RMAD register are echoed in data bits 31:22. Note that the address decoder map is never modified by a write to this register (it is modified by writes to the WMAD register). Unaffected by SRST or LRST.

Bits 21-3 Unused, can be 0 or 1

Table 4-37. Read M Bus Address Decoder Register (RMAD) Bit Definitions (cont'd)

Bits 2-0	MDS[2:0] RD only	M bus device select On write, these bits are unused (can be either 0 or 1). On read, this is the contents of the M bus address decoder for the 4MB page whose number is MPN[9:0]. Unaffected by SRST or LRST. Refer to the WMAD register for a bit description of this field.

WVAD Register

The write VMEbus address decoder register is used to map portions of the local M bus space into the VMEbus address space(s) and to enable snooping of VMEbus accesses. Selected M bus resources can be mapped to respond to the following VMEbus address spaces: A24 only; A32 only; both A24 and A32; neither A24 nor A32 (all with or without snoop enable). WVAD is write only, and contains both the page address (upper 10 data bits), address space map select, and the codes for the space to be mapped (lower 2 data bits). Bit assignments are shown in Table 4-38.

In a multi-CPU environment, only one CPU should be allowed control of the WVAD and RVAD registers. Refer to the caution under the RVAD register description.

The address and data paths used to access the WVAD register are the same as the ones used by the decoder when it is enabled. Consequently, the VADV bit in the CCSR *MUST BE CLEAR* before any write to the WVAD register. If VADV is not clear, multiple data and address path buffers are simultaneously active on the same lines. This results in erroneous data being written to the WVAD register; it may also result in damage to the hardware.

Because it is write-only, the WVAD register is unaffected by SRST or LRST.

CAUTION

The VADV bit of the CCSR must be 0 before any WVAD register access. If not, erroneous data is written to the WVAD register, and the hardware may also be damaged.

Table 4-38. Write VMEbus Address Decoder Register (WVAD) Bit Definitions

BIT 31	30	29	28	27	26	25	24	23	22	21	BITS 20-2	1	BIT 0
VPN9	VPN8	VPN7	VPN6	VPN5	VPN4	VPN3	VPN2	VPN1	VPN0	VASP	X	VSE*	VBDSEL*

Bits 31-22 VPN[9:0] WR only Page number (one of 1024 pages: page size = 4MB)
VPN[9:0] is supplied by the M bus master on data bits 31:22.

Bit 21 VASP WR only VMEbus address space map select
This bit selects the map to be written (there are two separate 1024 page maps, one for A24 and one for A32).
1 : VMEbus A24 space map.
0 : VMEbus A32 space map.

Bits 20:2 Unused, can be 0 or 1

Bit 1 VSE* WR only VMEbus snoop enable

Bit 0 VBDSEL* WR only VMEbus slave board select
These bits will be written into the VMEbus decoder for the page whose number is VPN[9:0] and whose map is defined by VASP. The MVME188A VMEbus slave response is defined as follows.

VSE*	VBDSEL*	VMEbus Slave Response
0	0	VMEbus slave space enabled, snoop enabled
0	1	No VMEbus slave response
1	0	VMEbus slave space enabled, snoop disabled
1	1	No VMEbus slave response

RVAD Register

The read VMEbus address decoder register is used to verify the configuration of the MVME188A VMEbus slave address map. It is read/write; on read it returns the code for the address space mapped to that page (lower 2 data bits); the upper 30 bits are always "1". On write, an address (VPN[9:0]) and a map select bit (VASP) are written to the upper bits of the RVAD register; the address is used on a subsequent read as a page address in the map corresponding to the VASP bit. Bit assignments are shown in Table 4-39.

The address and data paths used to access the RVAD register are the same as the ones used by the decoder when it is enabled. Consequently, the VADV bit in the CCSR *MUST BE CLEAR* before any *READ* or *WRITE* to the RVAD register. If VADV is not clear, multiple data and address path buffers will be simultaneously active on the same lines. This will result in erroneous data being written to or read from the RVAD register; it may also result in damage to the hardware.

> **CAUTION**
>
> **There is only one register for holding the page address to the VMEbus address decoder SRAM. A write to the WVAD register will change the location (page address) that the RVAD register accesses and vice-versa. NEVER allow a write to WVAD to occur between the write of an address to RVAD and the subsequent read of the map code from RVAD, or the map code returned will most likely be incorrect. For the same reason, no writes to RVAD should be allowed to occur between the write of an address to WVAD and the subsequent read of the map code from RVAD.**

Because there is only one address register, if the last write to the WVAD is a write of a page you wish to verify, it is not necessary to write to the RVAD; simply read from the RVAD. The address is loaded by the WVAD write, making a write to RVAD unnecessary.

Note that in a multi-CPU environment, protocol should allow only one CPU at a time to access the RVAD/WVAD registers. This prevents the occurrence of the above-mentioned "interleaved writes" while configuring or verifying the contents of the VMEbus address mapper.

The RVAD register is unaffected by SRST or LRST.

> **CAUTION**
>
> **The VADV bit of the CCSR must be 0 before any RVAD register access (refer to the CAUTION above).**

Table 4-39. Read VMEbus Address Decoder Register (RVAD) Bit Definitions

BIT 31	30	29	28	27	26	25	24	23	22	21	BITS 20-2	1	BIT 0
VPN9	VPN8	VPN7	VPN6	VPN5	VPN4	VPN3	VPN2	VPN1	VPN0	VASP	X	VSE*	VBDSEL*

Bits 31-22 VPN[9:0] WR only Page number (one of 1024 pages: page size = 4MB)
On write, VPN[9:0] is supplied by the M bus master on data bits 31:22, and is written into the read VMEbus address decoder (RVAD) register. On read, these bits are always "1". Note that the address decoder map is never modified by a write to this register (it is modified by writes to the WVAD register).
Unaffected by SRST or LRST.

Bit 21 VASP WR only VMEbus address space map
On write, this bit selects the map to be written (there are two separate 1024 page maps, one for A24 and one for A32). On read, this bit is always "1".
1 : VMEbus A24 space map select.
0 : VMEbus A32 space map select.

Bits 20:2 Unused, can be 0 or 1

Bit 1 VSE* RD only VMEbus snoop enable

Bit 0 VBDSEL* RD only VMEbus slave board select
On write, these two bits are ignored (can be either "0" or "1"). On read, these bits reflect the contents of the VMEbus decoder for the page whose number is VPN[9:0] and whose map is defined by VASP. Refer to the WVAD register for a description of these two bits.

APPENDIX A
EIA-232-D INTERCONNECTIONS

The EIA-232-D Standard is the most widely used interface between terminals and computers or modems, and yet it is not fully understood. This is because all the lines are not clearly defined, and many users do not see the need to conform for their applications. A system should easily connect to any other. Many times designers think only of their own equipment, but the state-of-the-art is computer-to-computer or computer-to-modem operation.

The EIA-232-D Standard was originally developed by the Bell System to connect terminals via modems. Therefore, several handshaking lines were included. In many applications these are not needed, but since they permit diagnosis of problems, they are included in many applications.

Table A-1 lists the standard EIA-232-D interconnections. To interpret this information correctly it is necessary to know that EIA-232-D is intended to connect a terminal to a modem. When computers are connected to computers without modems, one of them must be configured as a terminal and the other as a modem. Because computers are normally configured to work with terminals, they are said to be configured as a modem. Also, the signal levels must be between +3 and +15 volts for a high level, and between -3 and -15 volts for a low level. Any attempt to connect units in parallel may result in out of range voltages and is not allowed by the EIA-232-D specifications.

Table A-1. EIA-232-D Interconnections

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
1		Not used.
2	TXD	TRANSMIT DATA - data to be transmitted is furnished on this line to the modem from the terminal.
3	RXD	RECEIVE DATA - data which is demodulated from the receive line is presented to the terminal by the modem.
4	RTS	REQUEST TO SEND - RTS is supplied by the terminal to the modem when required to transmit a message. With RTS off, the modem carrier remains off. When RTS is turned on, the modem immediately turns on the carrier.
5	CTS	CLEAR TO SEND - CTS is a function supplied to the terminal by the modem which indicates that it is permissible to begin transmission of a message. When using a modem, CTS follows the off-to-on transition of RTS after a time delay.
6	DSR	DATA SET READY - data set ready is a function supplied by the modem to the terminal to indicate that the modem is ready to transmit data.
7	SIG-GND	SIGNAL GROUND - common return line for all signals at the modem interface.
8	DCD	DATA CARRIER DETECT - sent by the modem to the terminal to indicate that a valid carrier is being received.
9-14		Not used.
15	TXC	TRANSMIT CLOCK - this line clocks output data to the modem from the terminal.

Table A-1. EIA-232-D Interconnections (cont'd)

PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
16		Not used.
17	RXC	RECEIVE CLOCK - this line clocks input data from a terminal to a modem.
18,19		Not used.
20	DTR	DATA TERMINAL READY - a signal from the terminal to the modem indicating that the terminal is ready to send or receive data.
21		Not used.
22	RI	RING INDICATOR - RI is sent by the modem to the terminal. This line indicates to the terminal that an incoming call is present. The terminal causes the modem to answer the phone by carrying DTR true while RI is active.
23		Not used.
24	TXC	TRANSMIT CLOCK - Same as TXC on pin 15.
25	BSY	BUSY - A positive EIA signal applied to this pin causes the modem to go off-hook and make the associated phone busy.

NOTES: 1. High level = +3 to +15 volts. Low level = -3 to -15 volts.

2. EIA-232-D is intended to connect a terminal to a modem. When computers are connected to computers without modems, one of the computers must be configured as a modem and the other as a terminal.

A

EIA-232-D INTERCONNECTIONS

There are several levels of conformance that are appropriate for typical EIA-232-D interconnections. The bare minimum requirement is the two data lines and a ground. The full version of EIA-232-D requires 12 lines and accommodates automatic dialing, automatic answering, and synchronous transmission. A middle-of-the-road approach is illustrated in Figure A-1.

One set of handshaking signals frequently implemented are RTS and CTS. CTS is used in many systems to inhibit transmission until the signal is high. In the modem application, RTS is turned around and returned as CTS after 150 microseconds. RTS is programmable in some systems to work with the older type 202 modem (half duplex). CTS is used in some systems to provide flow control to avoid buffer overflow. This is not possible if modems are used. It is usually necessary to make CTS high by connecting it to RTS or to some source of +12 volts such as the resistors shown in Figure A-1. It is also frequently jumpered to an MC1488 gate which has its inputs grounded (the gate is provided for this purpose). Another signal used in many systems is DCD. The original purpose of this signal was to tell the system that the carrier tone from the distant modem was being received. This signal is frequently used by the software to display a message like CARRIER NOT PRESENT to help the user to diagnose failure to communicate. Obviously, if the system is designed properly to use this signal, and it is not connected to a modem, the signal must be provided by a pullup resistor or gate as described before (see Figure A-1). Many modems expect a DTR high signal and issue a DSR. These signals are used by software to help prompt the operator about possible causes of trouble. The DTR signal is used sometimes to disconnect the phone circuit in preparation for another automatic call. It is necessary to provide these signals in order to talk to all possible modems (see Figure A-1). Figure A-1 is a good minimum configuration that almost always works. If the CTS and DCD signals are not received from the modem, the jumpers can be moved to artificially provide the needed signal. Figure A-2 shows a way that an EIA-232-D connector can be wired to enable a computer to connect to a basic terminal with only three wires. This is because most terminals have a DTR signal that is ON and can be used to pullup the CTS, DCD and DSR signals. Two of these connectors wired back-to-back can be used. It must be realized that all the handshaking has been bypassed and possible diagnostic messages do not occur. Also, the TX and RX lines may have to be crossed since TX from a terminal is outgoing but the TX line on a modem is an incoming signal.

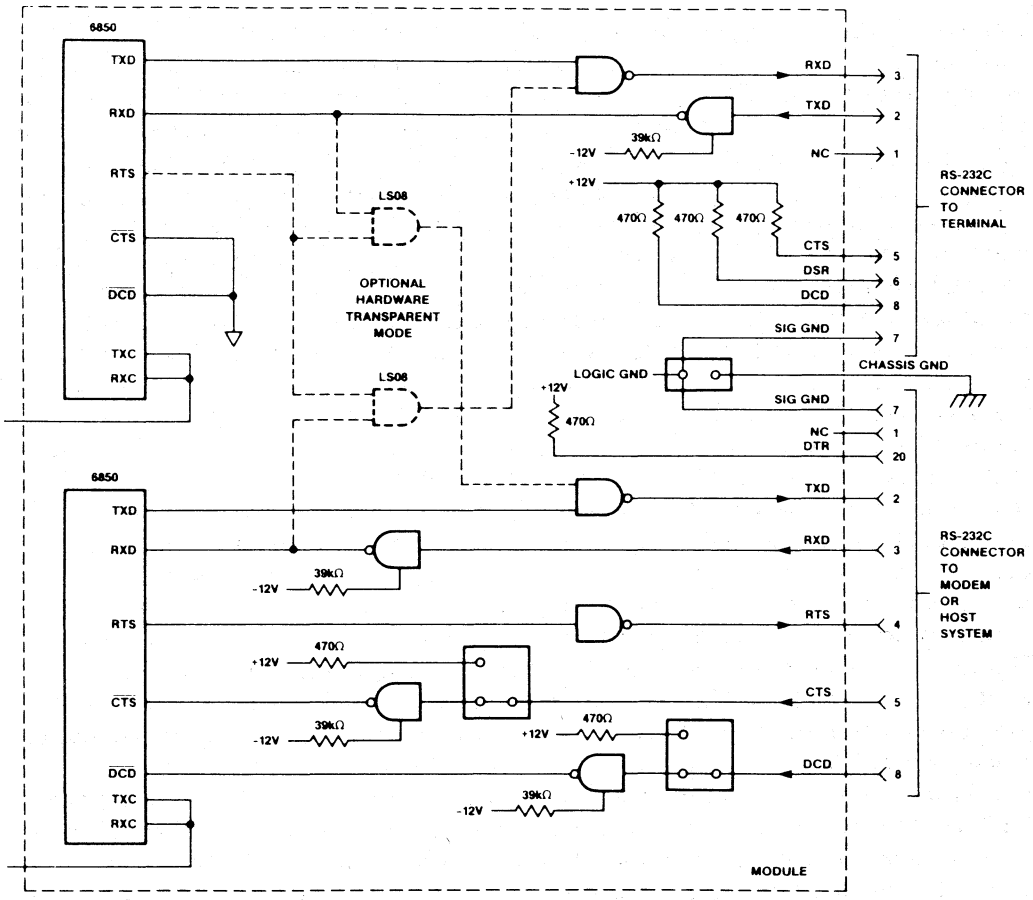


Figure A-1. Middle-of-the-Road EIA-232-D Configuration

Another subject that needs to be considered is the use of ground pins. There are two pins labeled GND. Pin 7 is the SIGNAL GROUND and must be connected to the distant device to complete the circuit. Pin 1 is the CHASSIS GROUND, but it must be used with care. The chassis is connected to the power ground through the green wire in the power cord and must be connected to the chassis to be in compliance with the electrical code. The problem is that when units are connected to different electrical outlets, there may be several volts difference in ground potential. If pin 1 of the devices are interconnected with a cable, several amperes of current could result. This not only may be dangerous for the small wires in a typical cable, but could result in electrical noise that could cause errors. That is the reason that Figure A-1 shows no connection for pin 1. Normally, pin 7 should only be connected to the CHASSIS GROUND at one point and, if several terminals are used with one computer, the logical place for that point is at the computer. The terminals should not have a connection between the logic ground return and the chassis.

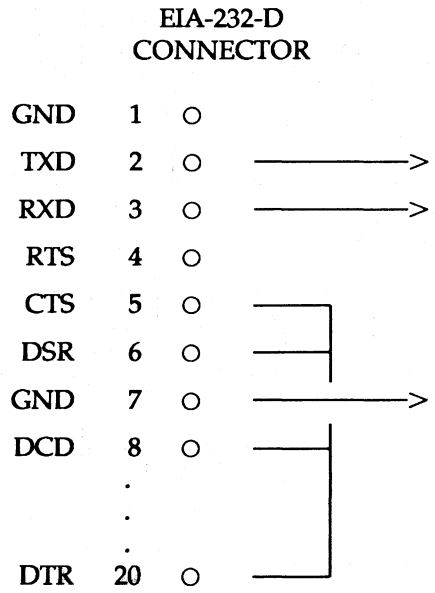


Figure A-2. Minimum EIA-232-D Connection

INDEX

When using this index, keep in mind that a page number indicates only where referenced material begins; it may extend to the following page or pages.

C

D

E

F

G

R

S

T

U

V

W

X

Z