DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Programming System Control and I/O Registers: AViiON® 6280 and 8000-8 Series

014−002170−00

AViiON®
PRODUCT LINE

Programming System Control and I/O Registers: AViiON® 6280 and 8000-8 Series

014–002170–00

Rev. 00, December, 1992
Ordering No. 014–002170

Notice

Programming System Control and I/O Registers: AViiON® 6280 and 8000-8 Series
014–002170–00

Revision History:

Original Release – December, 1992

Preface

This manual is aimed toward persons who are either developing a UNIX® operating system to run on an AViiON® 6280 and 8000-8 series systems, or adapting an existing UNIX operating system. This manual may also be used by hardware designers who are designing hardware to be used with an AViiON 6280 or 8000-8 series system.

NOTE: All references to the 6280 series include the 8000-8 series.

Organization of This Manual

This manual contains the following chapters and appendices:

Chapter 1 Architecture

Describes the architecture of the CPU and VIO boards and how this architecture relates to the rest of the system. This includes descriptions of the following: CPU, memory, registers, I/O, address decoding, and busses.

Chapter 2 Addressing

Describes address decoding and how to create address maps; how the CPU addresses system board resources, memory and VME controllers; and how VME controllers address system memory and the global control and status registers.

Chapter 3 Interrupts

Describes interrupts, how they are generated and handled, and all related registers.

Chapter 4 System Control Registers

Describes the system control registers that do not fit in another category.

Chapter 5 Memory

Describes the type of memory available, how to access the memory, and how to detect and process memory errors.

Chapter 6 Programming the Serial and Parallel Interfaces

Describes how to program the system serial ports and parallel port.

Chapter 7 Programming the PIT, CIO, and Real-Time Clock

Describes how to program the programmable interval timer, CIO, and real–time clock.

Chapter 8 The System Control Monitor (SCM)

Describes the System Control Monitor (SCM), including the system calls, the Environment Control Word (ECW), and the SCM commands.

Appendix A Address Map

Defines the addresses of the CPU board registers, VIO board registers, memory, and VME controllers.

Appendix B System Powerup Flowchart

Describes how the system powers up. This appendix includes flowcharts on powerup, reset, initialization, and Programmable Read–Only Memory (PROM)–resident tests.

Appendix C Connectors

Illustrates the CPU and VIO boards and describes the connectors.

Related Documents

This manual refers to the following documents:

Hardware Manuals

Operating AViiON® 6280 and 8000-8 Series Systems (014−002177)

Describes how operate AViiON 6280 and 8000-8 series systems. Contains all operator system control monitor (SCM) information.

Using the AViiON® System Control Monitor (SCM) (014−001802)

Describes how technical users can use the commands and menus of the firmware monitor program to bring up software, control their system environment, and debug programs.

Starting and Testing AViiON® 6000 Series Systems (014−001807)

Explains how to power up the AViiON 6000 series system, run diagnostics, and prepare for your operating system installation. Includes operational, physical, electrical, and environmental specifications for the computer unit.

Software Manuals

Installing the DG/UX™ System (093−701087)

Describes how to install the DG/UX system on AViiON hardware. For system managers.

Managing the DG/UX™ System (093−701088)

Discusses the concepts and tasks related to DG/UX system management, and provides general administration orientation. Explains how to use the sysadm facility. Includes instructions for managing disk resources, user profiles, file systems, printers, tapes drives, and other system features. For system managers and responsible operators; your primary DG/UX reference.

Customizing the DG/UX™ System (093−701101)

Describes how to customize the DG/UX system to your site's needs; includes descriptions of adding user home directories, printers, terminals, third–party packages, operating system clients and secondary releases. For system managers.

Writing a Device Driver for the DG/UX™ System (093–701085)

Describes how to write your own device driver for a DG/UX system running on an AViiON computer. Under the AViiON architecture, drivers must be written to address either a specific device or an adapter that manages secondary bus access to specific devices. This manual address both types of driver.

Other Companies' Manuals

The following manuals are not available from Data General Corporation. To order these documents, contact the appropriate vendor.

Binary Compatibility Standard (BCS) (88open Consortium, Ltd.)

Establishes the standards needed to develop operating systems that promote portability of application code between operating systems.

MC88100 User's Manual, Reduced Instruction Set Computer (RISC) (Motorola)

Describes the Motorola 88100 Central Processing Unit (CPU), including the registers, addressing modes, internal and bus timing, and assembly–language instruction set.

MC88200 User's Manual, Cache/Memory Management Unit (CMMU) (Motorola)

Describes the Motorola 88200 Cache/Memory Management Unit (CMMU), including the CMMU registers, the cache and cache coherency, memory management and user/supervisor space, the Processor bus (Pbus), and the Memory bus (Mbus).

Memory Products Databook (SGS–Thomson)

This databook includes specifications and programming information for the SGS–Thomson MK48T02B 2Kx8 Zeropower™/Timekeeper™ RAM.

The VMEbus Specification (Motorola document number HB212)

Defines the mechanical and electrical specifications, protocols, and terminology of the Versa Modula Europa bus (VMEbus). This interface is used to interconnect data processing, data storage, and peripheral control devices in a closely–coupled hardware configuration.

Z8536 CIO Counter/Timer and Parallel I/O Unit (Zilog)

This data sheet contains specifications and programming information for the Z8536 CIO.

Symbols and Conventions

The following symbol and conventions are used in this manual:

Symbol	Means
RESET*	The asterisk (*) indicates that the signal is asserted Low.
$\overline{\text{RESET}}$	The overbar indicates that the signal is asserted Low.

Convention	Means
SS[3−0]	Numbers within braces designate lines or bits. In this example, SS[3−0] refers to the System Status lines 0 through 3.

All addresses are hexadecimal unless otherwise noted with a subscript.

All data within bit diagrams is binary (the bit diagrams contain binary 1s and 0s.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1–800–DG–HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive *FOCUS* monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1–800–253–3902 or 1–508–443–3330.

End of Preface

Contents

Chapter 1 – Architecture

Chapter 2 – Addressing

Chapter 3 – Interrupts

Chapter 4 – System Control Registers

Chapter 5 – Memory

Chapter 6 – Programming the Serial and Parallel Interfaces

Chapter 7 – Programming the PIT, CIO, and Real–Time Clock

Chapter 8 – The System Control Monitor (SCM)

Tables

Table

Figures

Figure

Chapter 1
Architecture

The AViiON® 6280 series system is an industry–standard computer based on the Motorola Reduced Instruction Set Computing (RISC) architecture. The system uses Data General's DG/UX™ operating system, as well as other Motorola 88000–based operating systems. The system can be configured either as a time–sharing system supporting large numbers of user devices, or as a network server.

The computer chassis contains:

- CPU (Central Processing Unit) boards
- a VIO (VME and I/O Controller) board
- memory boards
- VME (Versa Modula Europa) controller boards
- disk and tape drives
- a power supply.

This chapter describes the system architecture, CPU board architecture, VIO board architecture, memory, I/O, registers, bus arbitration, and buses (Mbus, Vbus, X0bus, X1bus and VMEbus).

System Architecture

As shown in Figure 1–1, the system contains the following:

- CPU (central processing unit) boards
- VIO (VME and I/O Control) board
- memory boards
- VMEbus (Versa Modula Europa bus) controller boards
- Serial ports for system console and modem
- parallel port for Centronics-compatible printer

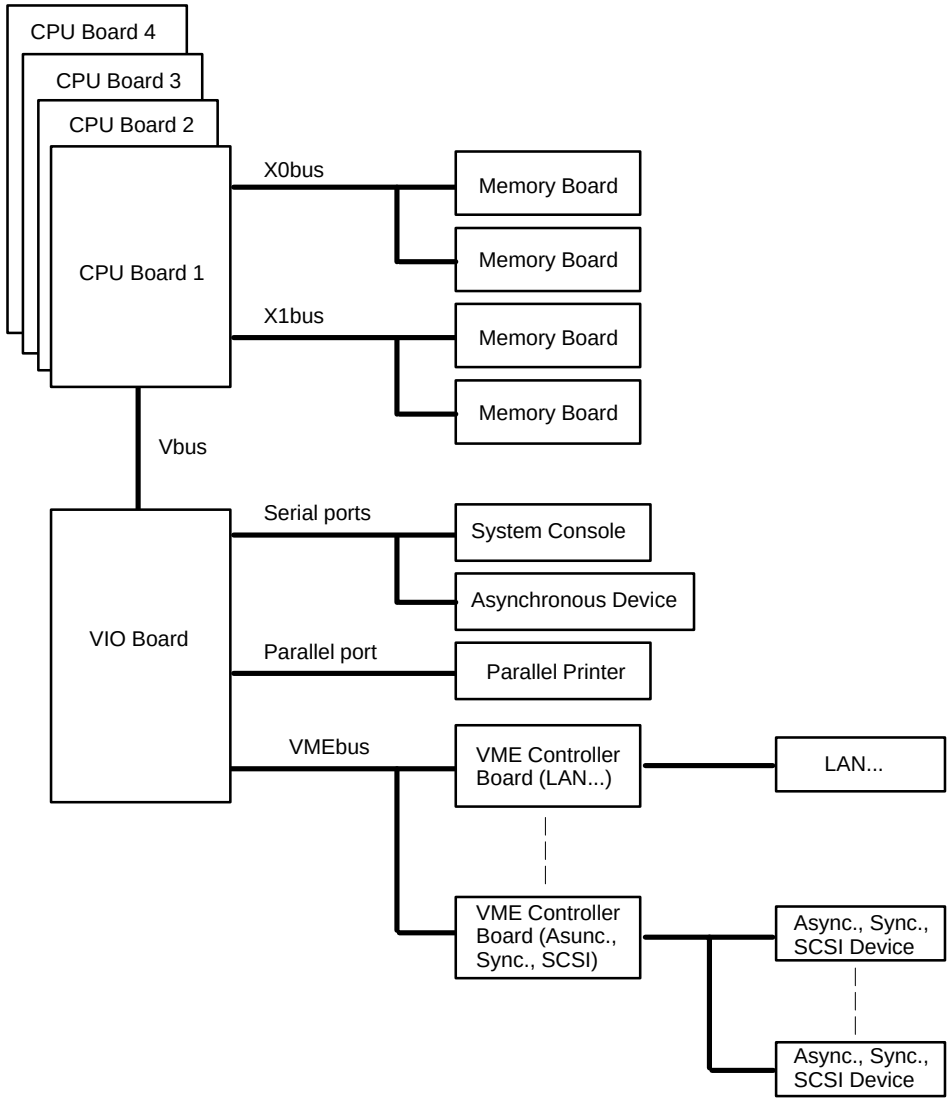


Figure 1–1 System Architecture

CPU Board Architecture

Figure 1–2 illustrates the CPU board architecture.

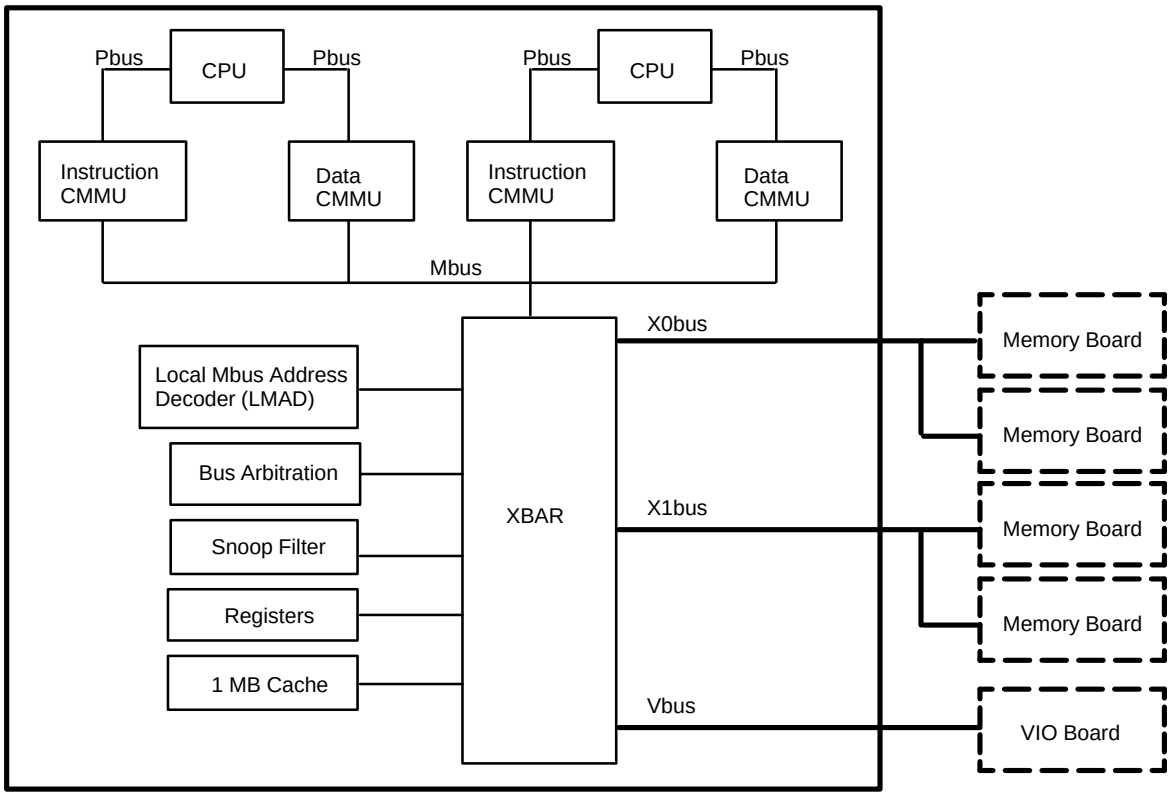


Figure 1–2 CPU Board Architecture

The CPU board uses Motorola MC88100 CPUs and MC88204 CMMUs that implement the Reduced Instruction Set Computing (RISC) architecture. The CPUs and CMMUs are arranged into CPU sets consisting of one CPU and two CMMUs; one instruction CMMU and one data CMMU. This document also refers to the CPUs as Job Processors. On each CPU board, CPU0 is JP0 (job processor 0) and CPU1 is JP1 (job processor 1).

Each CPU communicates with its CMMUs via Processor buses.

Each CMMU has 64Kbytes of cache memory. In addition, the CPU board has 1 MByte of cache memory. Each CMMU translates logical addresses into physical addresses that fall within two logical address ranges (User and Supervisor) of 4 Gbytes each.

The XBAR (cross bar) is an interface between the Mbus, Vbus, X0bus and X1bus. Therefore, it connects the CPU board to the memory boards and VIO board. The X0bus and X1bus connect to memory, and the Vbus connects to the VIO board.

See the following manuals by Motorola for information on the CPU and CMMUs:

- *MC88100 RISC Microprocessor User's Manual*
- *MC88204 Cache/Memory Management User's Manual*

VIO Board Architecture

Figure 1–3 illustrates the VIO board architecture.

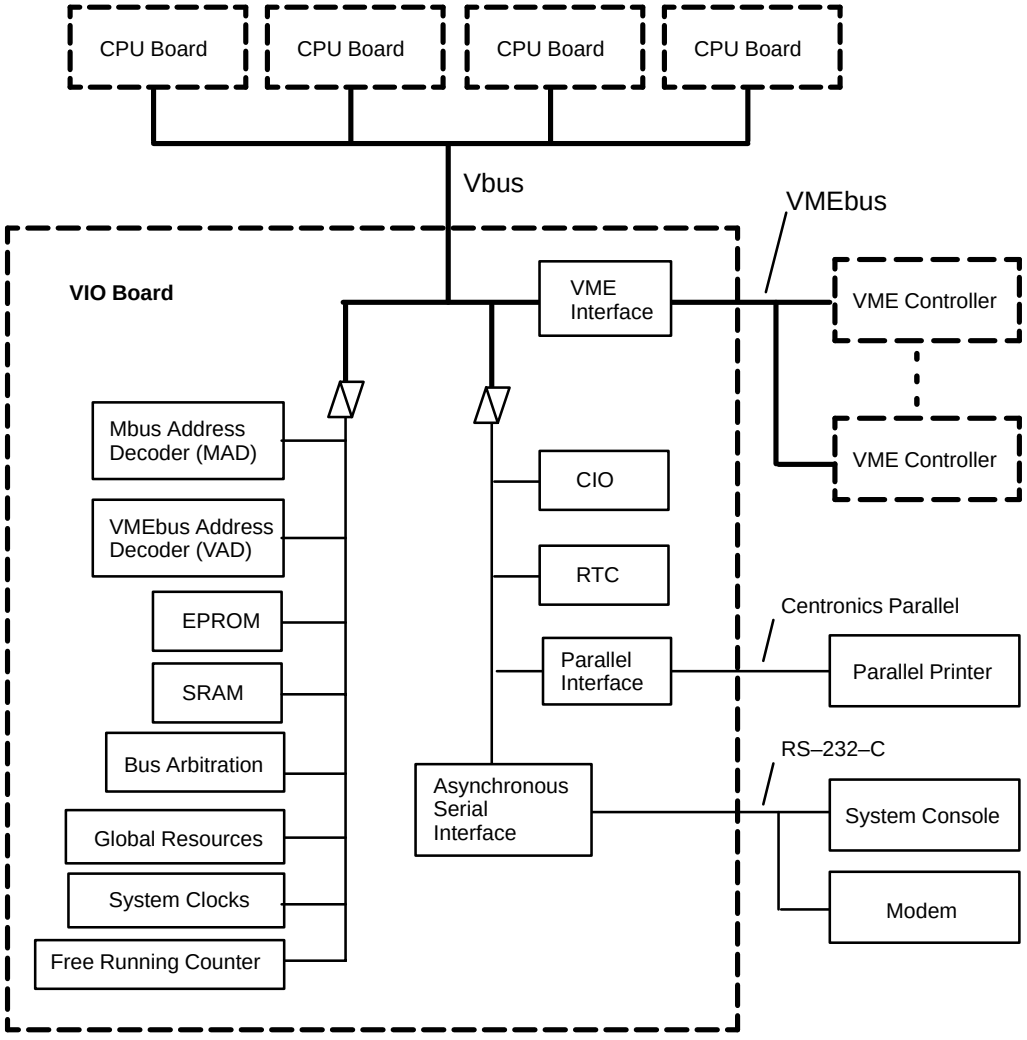


Figure 1–3 VIO Board Architecture

Memory

Dynamic Random–Access Memory (DRAM) is located on the memory boards. All CPUs and many VME controllers have access to the system memory boards. The memory is partitioned by memory maps to regulate access by the CPUs and VME controllers. Also, each memory board has ECC (error checking and correction) logic.

The VIO board has PROM (Programmable Read–Only Memory); the boot code resides in PROM.

The VIO board has NOVRAM (non–volatile RAM) where the system configuration parameters are stored. See Chapter 7, “Programming the Counter/Timer and the Real-Time Clock” for more information on the NOVRAM.

Input/Output

The VIO board has three I/O (Input/Output) interfaces: the VMEbus, serial RS–232–C ports, and a Centronics parallel port. All VME controllers connect to each other and to the VIO board through the VMEbus. The system console and a modem connect to the VIO board via the serial RS–232–C ports. A parallel port allows you to connect a parallel printer using the Centronics interface protocol.

The serial and parallel interfaces use programmable controllers whose registers are mapped into the system address space. The serial ports are bidirectional, but the parallel port is transmit–only. See Chapter 6, “Programming the Serial and Parallel Interfaces” for more information on the serial and parallel ports.

VME controllers are programmed through registers located on each VME controller. These registers are mapped into the system address space. Appendix A, “Address Map,” defines the starting addresses for the VME controllers. The VME interface is bidirectional. The VMEbus is described later in this chapter. For more information on programming a VME–based controller board, see the documentation for that board.

Registers

The CPU boards and VIO board all have programmable registers used for system control and I/O. The CPUs can access all of these registers as well as registers located on the VME controller boards. VME controllers can access only certain global registers located on the VIO board.

Address Decoding

All of the system address space is mapped to control access to the system address space. Both the CPU boards and VIO board have address decoders. The CPU boards each have a MAD (Mbus Address Decoder) that regulates access to CPU board resources and memory. The VIO board has two address decoders, a MAD (Mbus Address Decoder) and a VAD (VMEbus Address Decoder). The MAD regulates access to VIO board resources, memory, and VME space (A16, A24 and A32). The VAD controls access to the VIO and CPU boards by other controllers on the VMEbus. Figure 1–4 roughly illustrates address decoding. Chapter 2, "Addressing," describes address decoding in greater detail.

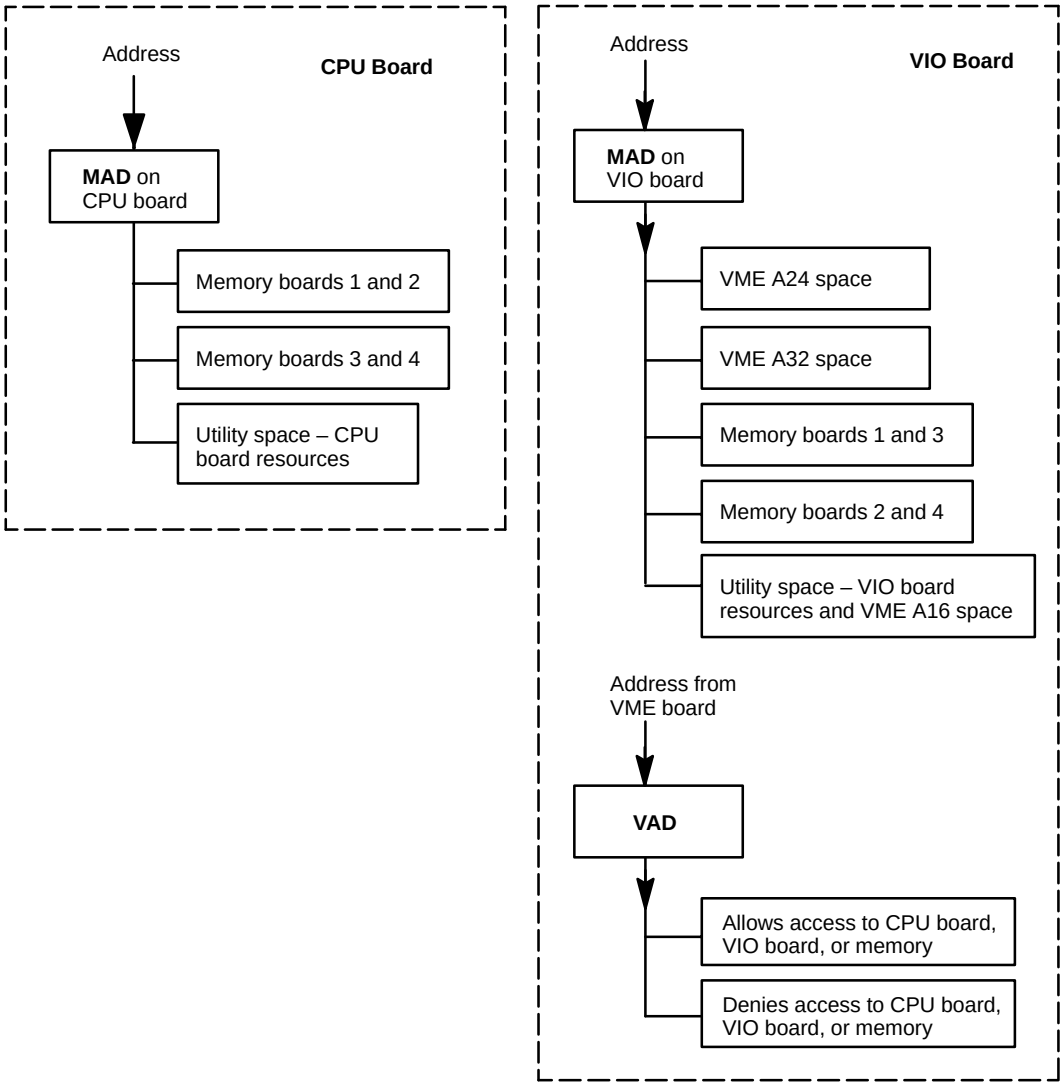


Figure 1–4 Address Decoding

The Mbus

Each CPU board has an Mbus that connects the CPU board resources to each other. The Mbus has a 32–bit multiplexed Data/Address bus and Arbitration/Control lines, as well as other signals defined in Table 1–1.

Table 1–1 Mbus Signals

Signal	Description
Data Transfer:	
AD[31−00]	Address/data
Bus Arbitration:	
BR	Bus request
BA	Bus acknowledge
$\overline{\text{BB}}$	Bus busy
$\overline{\text{AB}}$	Arbitration busy
Control and Status:	
C[6−0]	Control
ST[3−0]	Local status
$\overline{\text{SS}}$[3−0]	System status

When two or more devices try to access the Mbus at the same time, bus arbitration logic determines which device is granted the Mbus.

When accessing the Mbus, VME controllers have higher priority than the CPUs. The CMMUs can be programmed to operate in Fairness mode: the CMMUs will not request access to the Mbus until all other masters who want the bus have been serviced.

The Vbus

The Vbus passes address, data and status between the VIO board and the CPU boards, and within the VIO board. It connects to the Mbus of each CPU board through the Xbar interface of that CPU board. The Vbus is very different from the Mbus, but it is transparent to the programmer, and may be thought of as an extension of the Mbus. Many registers located on the VIO board talk about Mbus characteristics that are actually part of the Vbus.

The X0bus and X1bus

The X0bus and X1bus connect the memory boards to the CPU boards. Again, like the Vbus, these buses are transparent to the programmer, and may be thought of as an extension of the Mbus.

The VMEbus

The Versa Modula Europa bus (VMEbus) connects VME controllers to the VIO and CPU boards. The VME interface, located on the VIO board, arbitrates access to the VMEbus, and decodes addresses from VME controllers to the VIO and memory boards. The VME interface also monitors the ACFAIL line and supplies the 16 MHz VME system clock. Address decode logic, in conjunction with memory maps, determines which accesses across the VME interface are valid. Chapter 2, "Addressing," discusses the memory maps and addressing schemes in detail.

The VIO board accepts words, half–words, bytes, and blocks of data from VME controllers, but transmits only words, half–words or bytes to VME controllers.

See Appendix C "Connectors" and the Motorola manual, *The VMEbus Specification* for more information on the VMEbus.

VMEbus Arbitration

VMEbus arbitration consists of four bus request and bus grant levels; the VMEbus signals associated with these levels are Bus Request ($\overline{\text{BR[3-0]}}$) and Bus Grant ($\overline{\text{BG[3-0]IN}}$ and $\overline{\text{BG[3-0]OUT}}$). Each bus grant level is daisy–chained from VME controller to VME controller; therefore if more than one board requests at a given level, the board closest to the VIO board (i.e. closest to slot 9) will access the VMEbus. When that board releases the VMEbus, the arbitration process repeats with another arbitration cycle. This daisy–chain configuration is illustrated in Figure 1–5. The Motorola manual, *The VMEbus Specification*, describes these signals in greater detail.

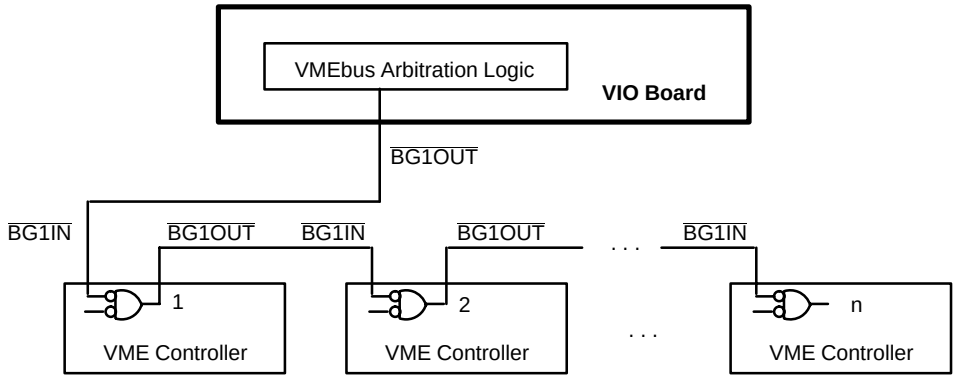


Figure 1–5 VMEbus Grant Daisy–Chain

Accessing the VMEbus

To access the VMEbus, write the desired request level into the VMEbus Request Level (VRL) bits of the Utility Control and Status (UCS) register. This asserts the corresponding bus request bit (BR[3–0]) to the VMEbus. If the VMEbus is free and the CPU is the highest priority requestor, the VIO board will pull the Bus Grant ($\overline{\text{BGnOUT}}$) line High and assert the Bus Busy ($\overline{\text{BBSY}}$) line Low. Otherwise, if the VMEbus is busy, i.e. $\overline{\text{BBSY}}$ is asserted, and the CPU request has a lower priority, the bus arbiter will wait

until the bus is released. When the bus is released, the arbiter will sample the bus lines and grant the request to the highest priority requestor.

VMEbus arbitration is available in two modes: Priority and Round Robin. The mode is selected through the Round Robin (RBN) bit in the Utility Control and Status (UCS) register.

When using the Priority mode, if a controller requests access to the VMEbus while a lower–priority controller is using the bus, the VME arbiter will assert the $\overline{\text{BCLR}}$ signal to request the current VMEbus master to release the VMEbus. Access to the VMEbus is granted to the highest–priority requestor via the Bus Grant ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) lines. Level 3 is the highest priority and level 0 is the lowest priority.

When using the Round Robin mode, the Bus Grant ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) signal is passed through a loop. When a controller receives the bus grant, the controller will either pass the grant on to another controller, or it will use the grant to access the VMEbus, then pass the grant to the next controller. In Round Robin mode, all controllers in the round robin loop have the same priority.

Arbitration Modes

The VIO board accesses the VMEbus using one of three modes: Release Never, Release When Done (RWD) and Release On Request (ROR). These modes are selected using the Release Never (RNV) and VMEbus Release Mode (VRM) bits in the UCS register. VRM selects either Release When Done or Release on Request. These modes and bits are described in greater detail in Chapter 4, “Global Resources.”

The VIO board also implements a Fairness mode if the Fairness (FAIR) bit in the UCS register is set. When the FAIR bit is set, the VIO board will not request the VMEbus if someone else is requesting it from the same level. This passes the bus grant down the bus grant daisy chain ($\overline{\text{BGnIN}}$ and $\overline{\text{BGnOUT}}$) to the next board.

Arbitration Timeout

The VMEbus arbiter limits bus arbitration delays to one second or less. This timing logic can be enabled or disabled through the Enable VMEbus Arbitration Timeout (ETO) bit of the Utility Control and Status (UCS) register. When enabled, the time–out expires if the BBSY signal is not asserted within one second of asserting the Bus Grant (BG*n*) signal. When a VMEbus arbitration timeout occurs, the interrupt logic sets the Arbitration Timeout (ATO) interrupt in the Interrupt Status (IST) register and asserts an interrupt to the CPU.

Transferring Data Over the VMEbus

The VME interface supports word, half–word and byte transfers initiated by either the VMEbus or the Mbus.

A VMEbus error will occur if either a bus time–out occurs because no device responds, or if a VME controller is accessed and receives an address modifier that it cannot handle

(For example, if a controller tries to send a 32–bit data word to a 16–bit controller board, an error will result.)

The VME interface also supports block transfers of words, half–words or bytes initiated by a VME controller. The maximum block transfer initiated by a VME controller is 256 bytes (64 words) while the maximum block transfer on the Mbus is 16 bytes (four words). If a VME controller sends a block larger than four words to memory, the VME interface breaks the block into four–word blocks that the Mbus can handle.

VME–initiated block transfers of consecutive bytes or half–words reduce system performance when compared with full–word data transfers. For byte and half–word data transfers, the memory system performs read–modify–write cycles to merge the new data with the remaining bits of the word and to generate error correction bits. The VME interface provides higher throughput by packing consecutive half–words or bytes, wherever possible, into 32–bit words prior to Mbus transfer.

End of Chapter

Chapter 2
Addressing

This chapter discusses how to address the registers, I/O and memory in your AViiON 6200 series system. It describes address maps and address decoding, how to create or change address maps, and how to verify the address maps. The chapter also explains how the CPUs access VIO resources and system memory, how the CPUs access other VME controllers, and how other VME controllers access system memory and the Global Control and Status (GCS) registers. Each section of this chapter describes related registers.

CAUTION: Much of this chapter, especially the sections that describe address maps, may not be important to you unless you are developing power–up or diagnostic code and need to remap system resources. The system address space is mapped during system powerup, and should not be changed after powerup.

Big–Endian Byte Ordering

The system supports the *big–endian* scheme for ordering bytes in memory. In this scheme, the lower memory bits correspond to the high–order bytes, as shown in Figure 2–1.

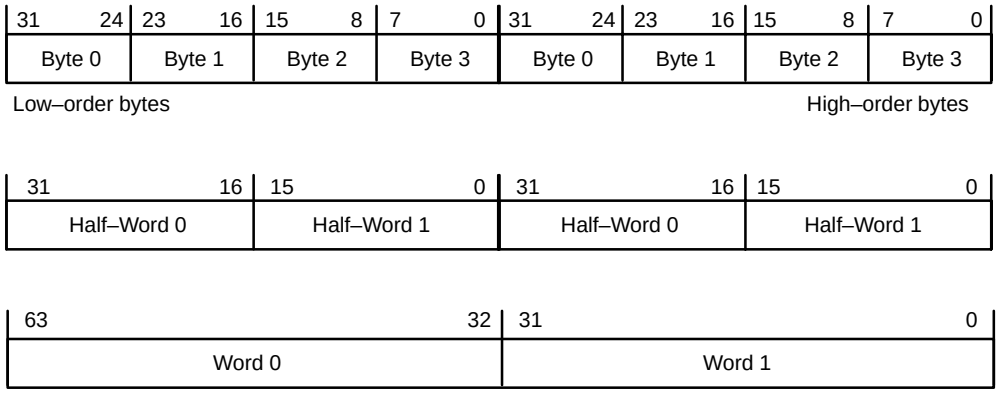


Figure 2–1 Big–Endian Byte Ordering

Memory Maps and Address Decoding

This section discusses how the system address space is mapped to control access to system resources.

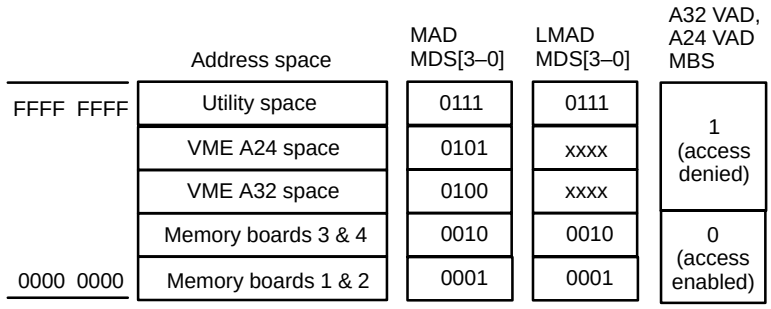
CAUTION: This section may not be important to you unless you are developing power–up or diagnostic code and need to remap system resources. The system address space is mapped during system powerup, and should not be changed after powerup.

The 4–Gbyte system address space is partitioned into 1024 4–Mbyte pages. These 4–Mbyte pages are mapped into larger spaces by address maps, or decoders that enable access to major segments of the system address space. Three address maps, the Mbus Address Decoder (MAD), Local Mbus Address Decoder (LMAD) and VMEbus Address Decoder (VAD), map the system address space. The MAD decodes addresses on the VIO board, and the LMAD decodes addresses on the CPU board. The VAD decodes addresses generated by VME controllers, and allows or inhibits access to the CPU or VIO boards based on these addresses. The MAD, LMAD and VAD are stored in RAM. The MAD has 1024 4–bit locations (Mbus Device Select (MDS[3–0])). Each VAD has 1024 2–bit locations that consist of an Mbus Select (MBS) bit and a VME Snoop Enable (VSE) bit; both bits can be found in the Read VMEbus Address Decoder (RVAD) and Write VMEbus Address Decoder (WVAD) registers. These 1024 locations correspond to the 1024 4–Mbyte pages, and the address decode bits point to pages in the system space.

The contents of each location in the MAD or LMAD may be read via the RMAD (Read Mbus Address Decoder) or RLMAD (Read Local Mbus Address Decoder) register respectively. The contents of each location in the VAD may be read via the RVAD (Read VMEbus Address Decoder) register.

The contents of each location in the MAD or LMAD may be reprogrammed via the WMAD (Write Mbus Address Decoder) or WLMAD (Write Local Mbus Address Decoder) register respectively. The contents of each location in the VAD may be reprogrammed via the WVAD (Write VMEbus Address Decoder) register.

Figure 2–2 illustrates the major segments of address space and the address maps used to enable access to these spaces.



NOTE: The visual spaces in this map are not proportional to the actual address ranges.

Figure 2–2 Address Maps

The utility space contains the system control registers, including all CMMU registers and VME A16 space.

An A32 VME controller (32 address bits) can access any page in memory, but an A24 VME controller (24 address bits) can access only four contiguous pages in memory. The Extended Address (EXTAD) register defines bits 24–31 for A24 controllers.

The Mbus Address Decoders (LMAD and MAD)

Access to system resources is controlled by the LMAD (Local Mbus Address Decoder) and the MAD (Mbus Address Decoder). The LMAD validates addresses on the CPU board, and the MAD validates addresses on the VIO board.

The LMAD partitions the system address space as follows:

- Memory boards 1 (X0bus board 0) and 2 (X1bus board 0)
- Memory boards 3 (X0bus board 1) and 4 (X1bus board 1)
- CPU board resources
- Bus error (an invalid address)

The MAD partitions the system address space as follows:

- Memory boards 1 (X0bus board 0) and 2 (X1bus board 0)
- Memory boards 3 (X0bus board 1) and 4 (X1bus board 1)
- VIO board resources and VME A16
- VME A24
- VME A32
- Bus error (an invalid address)

Loading and Verifying the LMAD

The Data General power–up code automatically loads and verifies the LMAD while the system powers up. The LMAD should not be changed after powerup. If you are developing power–up code or diagnostics and you must rewrite the LMAD, load and verify the LMAD as follows:

1. Clear bit 4 (Mbus Address Decoder Valid (MADV)) of the JPDIAG (Job Processor Diagnostic) register to 0 as follows:

   A. Read the JPDIAG register (see Chapter 4).

   B. Clear bit 4 to 0. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to JPDIAG.

2. Write the decode value(s) to the LMAD. The following code example writes to a range of pages in the LMAD.

```
;           r2 = loop count for number of pages to initialize
;           r3 = MDS (Mbus Device Select) value (see the LWMAD register description)
;           r4 = MPN (Mbus page number), value 0–1023
;           r5 = page size
;           r6 = base address for LWMAD register (0xFFC8)
;           Note:  0x indicates a hexadecimal number.

;initialize registers
or.u        r5,r0,0x0040            ;page size (4MB)
mul         r4,r4,r5                ;form the MPN (Mbus page number)
or.u        r6,r0,0xFFC8            ;write LWMAD high address bits to r6
bsr         init_LMAD               ;execute the init_LMAD subroutine

init_LMAD                           ;loads a single MDS into a range of pages
st          r3,r6,0x8020            ;write MDS to LWMAD register
st          r3,r4,r0                ;write MDS to the MPN
;           CAUTION:  Do not place any instructions between the store instructions!  The
;                     write to MPN must immediately follow the write to LWMAD!
addu        r4,r4,r5                ;increment to the next page
subu        r2,r2,1                 ;decrement the loop counter
bcnd        ne0,r2,init_LMAD        ;loop until 0
```

3. Verify the decode value in LMAD as follows:

   A. Read an MDS value from a location in the LMAD as :

```
;           r3 = MDS (Mbus Device Select) value (see the LWMAD register description)
;           r4 = MPN (Mbus page number), value 0–1023
;           r6 = base address for LWMAD register (0xFFC8)
;           Note:  0x indicates a hexadecimal number.

ld          r3,r6,0x8020            ;load the LRMAD register
ld          r3,r4,r0                ;load the MDS from the MPN to r3
;           CAUTION:  Do not place any instructions between the load instructions!  The
;                     load from MPN must immediately follow the load from LWMAD!
```

   B. Verify the MDS data as desired. You may want to build the above sample into a loop which verifies or writes to a table.

4. Repeat steps 2 and 3 for all locations to be written.

5. Set the MADV bit of the JPDIAG register to 1 as follows:

   A. Read the JPDIAG register.

   B. Set bit 4 to 1. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to JPDIAG.

Loading and Verifying the MAD

The Data General power–up code automatically loads and verifies the MAD while the system powers up. The MAD should not be changed after powerup. If you are developing power–up code or diagnostics that require resetting the MAD, load and verify the MAD as follows:

1. Clear bit 4 (Mbus Address Decoder Valid (MADV)) of the JPDIAG (Job Processor Diagnostic) register to 0 as follows: (this turns off the LMAD)

   A. Read the JPDIAG register (see Chapter 4).

   B. Clear bit 4 to 0. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to JPDIAG.

2. Clear bit 1 (Mbus Address Decoder Valid (MDV)) of the CPU Control and Status (CCS) register to 0 as follows: (this turns off the MAD)

   A. Read the CCS register (see Chapter 4).

   B. Clear bit 1 to 0. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to CCS.

3. Write the location pointer and decode value to the Write Mbus Address Decoder (WMAD) register.

   The data you write into WMAD includes ten bits to select one of the 1024 locations in the MAD, and four bits to store in that location.

4. Verify the decode value in the MAD as follows:

   A. Write the page number into the Mbus Page Number (MPN) bits in the Read Mbus Address Decoder (RMAD) register.

   B. Read RMAD. The Mbus Device Select (MDS) bits contain the decode value.

5. Repeat steps 3 and 4 for all locations to be written.

6. Set the MDV bit of the CCS register to 1 as follows: (this turns on the MAD)

   A. Read the CCS register (see Chapter 4).

   B. Set bit 1 to 1. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to CCS.

7. Set the MADV bit of the JPDIAG register to 1 as follows: (this turns on the LMAD)

   A. Read the JPDIAG register.

   B. Set bit 4 to 1. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to JPDIAG.

When the MDV or MADV bit of the CCS or JPDIAG register is cleared to zero, the MAD or LMAD is turned off. In this state, the upper ten address bits are ignored and all addresses point into the 4–Mbyte utility space, which includes the powerup PROM and the VIO board registers. When MDV or MADV is cleared, the utility space begins at address 0000 0000_{16}. When the MDV or MADV is set to 1, the MAD or LMAD is turned on. This is the normal operating state where the upper ten address bits are decoded by the MAD or LMAD and point to the system address space. When MDV or MADV is set, the utility space is mapped to FFC0 0000_{16}.

The following pages describe the LRMAD, LWMAD, RMAD and WMAD registers.

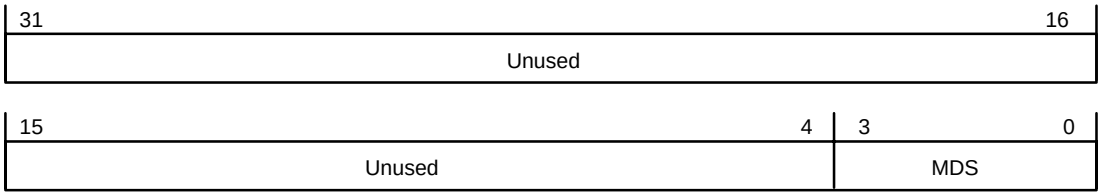
LRMAD Read Local Mbus Address Decoder

FFC8 8024 Read only

The Read Local Mbus Address Decoder (LWMAD) register verifies an address decode value in the local Mbus address decoder. This operation can only be done by processor 0 on each CPU board; processor 1 generates undefined results.

NOTE: LRMAD can be accessed by CPU0, CPU2. CPU4 and CPU6 only.

Resets do not affect LRMAD.



Bit	Mnemonic	Function
31–4	Unused	
3–0	MDS	Mbus Device Select

MDS3	MDS2	MDS1	MDS0	Device Selected
0	0	0	1	Memory 1, X0 and X1
0	0	1	0	Memory 2, X0 and X1
0	1	1	1	Utility Space
0	1	1	0	Bus Error

NOTE: Invalid addresses produce bus errors.

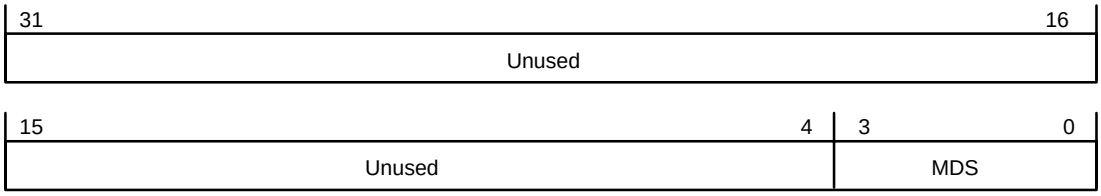
LWMAD Write Local Mbus Address Decoder

FFC8 8020 Write only

The Write Local Mbus Address Decoder (LWMAD) register loads an address decode value into the local Mbus address decoder. This operation can only be done by JP0 (job processor 0) on each CPU board; JP1 generates undefined results.

NOTE: LWMAD can be accessed by JP0 (CPU0, CPU2. CPU4 and CPU6) only.

Resets do not affect LWMAD.



Bit	Mnemonic	Function
31–4	Unused	
3–0	MDS	Mbus Device Select

MDS3	MDS2	MDS1	MDS0	Device Selected
0	0	0	1	Memory 1, X0 and X1
0	0	1	0	Memory 2, X0 and X1
0	1	1	1	Utility Space
0	1	1	0	Bus Error

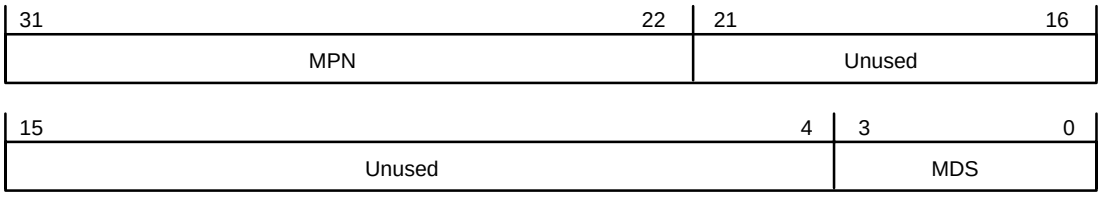
NOTE: Invalid addresses produce bus errors.

RMAD Read Mbus Address Decoder

FFF8 8024 Read/Write

Read Mbus Address Decoder (RMAD) verifies the configuration of the Mbus physical address map.

Resets do not affect RMAD.



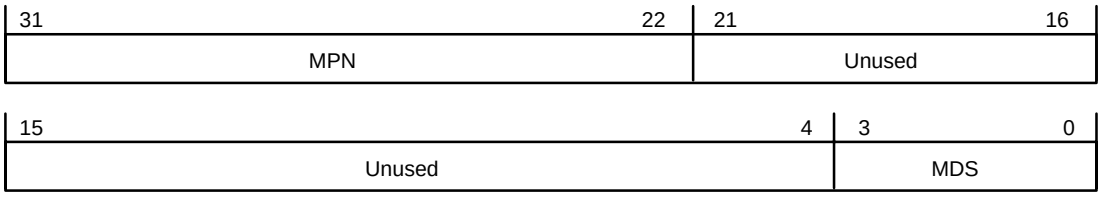
Bit	Mnemonic	Function
31–22	MPN	Mbus Page Number (one of 1024 pages, page size=4 Mbytes) Read/write. Write: The bus master writes the number of the MAD page to be verified in the MPN bits. Read: MPN returns the page number for the Mbus Device Select (MDS) code returned.
21–4	Unused	
3–0	MDS	Mbus Device Select Read only. MDS returns the device select value for the page defined by MPN. MDS3 MDS2 MDS1 MDS0 Device Selected 0 0 0 1 Memory 1, X0 and X1 0 0 1 0 Memory 2, X0 and X1 0 1 0 0 VME A32 Space 0 1 0 1 VME A24 Space 0 1 1 1 Utility Space 0 1 1 0 Bus Error NOTE: Invalid addresses produce bus errors.

WMAD Write Mbus Address Decoder

FFF8 8020 Write only

Write Mbus Address Decoder (WMAD) is used to load an address decode value into the Mbus address decoder.

Resets do not affect WMAD.



Bit	Mnemonic	Function
31–22	MPN[9–0]	Mbus Page Number (one of 1024 pages) The bus master writes the number of the page to be mapped in MPN.
21–4	Unused	
3–0	MDS[3–0]	Mbus Device Select The bus master writes the Mbus device select value in MDS. The MDS values are as follows: MDS3 MDS2 MDS1 MDS0 Device Selected 0 0 0 1 Memory 1, X0 and X1 0 0 1 0 Memory 2, X0 and X1 0 1 0 0 VME A32 Space 0 1 0 1 VME A24 Space 0 1 1 1 Utility Space 0 1 1 0 Bus Error NOTE: Invalid addresses produce bus errors.

The VMEbus Address Decoder (VAD)

VME controllers access memory through the VAD.

Loading and Verifying the VAD

The Data General power–up code automatically loads and verifies the VAD while the system powers up. The VAD should not be changed after powerup. If you are developing power–up code or diagnostics that require resetting the VAD, load and verify the VAD as follows:

1. Clear bit 0 (VMEbus Address Decoder Valid (VDV)) of the CPU Control and Status (CCS) register to 0 (see Chapter 4).

   A. Read the CCS register (see Chapter 4).

   B. Clear bit 0 to 0. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to CCS.

2. Write the VMEbus Address Decoder address and decode parameter to the Write VMEbus Address Decoder (WVAD) register.

   The data you write into WVAD includes one bit to select the VAD (A24 VAD or A32 VAD), ten bits to select one of the 1024 locations in the VAD, and two bits to store the decode value in the selected location.

3. Verify the decode value in the VAD as follows:

   A. Write the page number into the VPN bits in the RVAD register.

   B. Read the RVAD register. The Mbus Select (MBS) bit defines whether or not VME controllers can access the Mbus at this page.

4. Repeat step 2 for all locations to be loaded.

5. Set the VDV bit of the CCS register to 1.

   A. Read the CCS register (see Chapter 4).

   B. Set bit 0 to 1. Do not change any other bits; these are reserved for Data General use only, and may affect the operation of your system.

   C. Write the new value back to CCS.

When the VDV bit of the CCS register is cleared to zero, the VAD is turned off. In this state, the VME controllers cannot access the VIO board resources and memory. When the VDV bit of the CCS register is set to 1, the VAD is turned on. This is the normal operating state where the upper ten address bits are decoded by the VAD and point to a predefined range in the system address space.

RVAD Read VMEbus Address Decoder

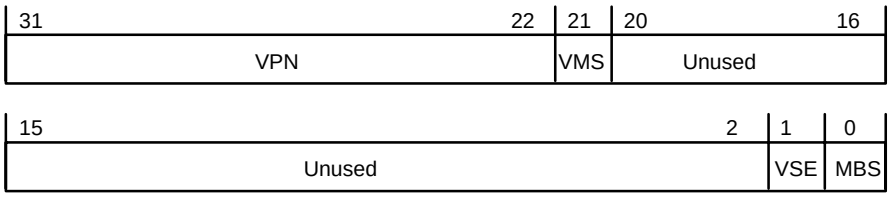
FFF8 802C Read/Write

The Read VMEbus Address Decoder (RVAD) register is used to examine the contents of the VAD. When read, the RVAD register returns the address decode value for the page defined by the VME Page Number (VPN) bits. This value is returned through the Mbus Select (MBS) and VME Snoop Enable (VSE) bits. Before reading RVAD, write the page number and the map select into the VPN and VMS bits of the RVAD register.

Writing to RVAD does not modify the address decoder map. To modify the map, write to WVAD.

Clear the VMEbus Decoder Valid (VDV) bit in the CPU Control and Status (CCS) register to 0 before reading RVAD.

Resets do not affect RVAD.



Bit	Mnemonic	Function
31–22	VPN	VME Page Number (one of 1024 pages) Write only. VPN supplies the page number to which the WVAD register will write the page address.
21	VMS	VME Map Select Write only. VMS selects the map to which the page address will be written. 1 Select VME A24 map. 0 Select VME A32 map.
20–2	Unused	
1	VSE	VME Snoop Enable Read only. 1 Snoop is disabled. 0 Snoop is enabled.
0	MBS	Mbus Select Read only. MBS returns the value of MBS from the VME map defined by VMS and the page defined by VPN[9–0]. 1 Indicates that the VME controller does not have access to that address. 0 Indicates that the VME controller has access to that address.

WVAD Write VMEbus Address Decoder

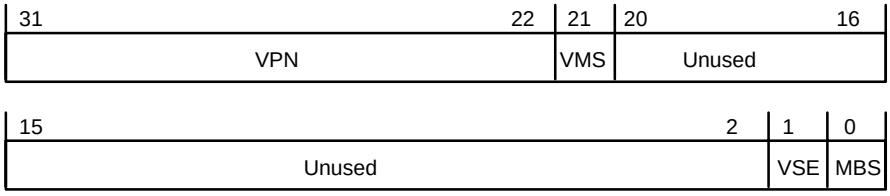
FFF8 8028 Write Only

Write VMEbus Address Decoder (WVAD) loads the VMEbus address decoders. The VMEbus has two decoders: one for the A24 address space and one for the A32 address space.

Clear the VMEbus Decoder Valid (VDV) bit in the CPU Control and Status (CCS) register to 0 before writing to WVAD.

CAUTION: Writing to WVAD will change the page address that RVAD accesses. NEVER write to WVAD between writing to and reading from RVAD.

Resets do not affect WVAD.



Bit	Mnemonic	Function
31–22	VPN	VME Page Number Defines the page number within the VME address decoder. The contents of VSE and MBS are written into the appropriate VAD at this page number.
21	VMS	VME Map Select Selects the VME address decoder to be written to. 1 Selects the VME A24 map. 0 Selects the VME A32 map.
20–2	Unused	
1	VSE	VME Snoop Enable Requests the master CMMU to snoop the VMEbus for a valid Mbus address. 1 Snoop is disabled. 0 Snoop is enabled.
0	MBS	Mbus Select Defines whether the VMEbus has access to the Mbus at the page number defined by the VPN bits. 1 Prevents VME access to the Mbus at that page number. 0 Allows VME access to the Mbus at that page number.

Addressing System Resources

This section describes how a CPU addresses system resources. This addressing process is transparent to the programmer, and this description is intended to assist those who need to reprogram the Mbus Address Decoder.

CAUTION: This section may not be important to you unless you are developing power–up or diagnostic code and need to remap system resources. The system address space is mapped during system powerup, and should not be changed after powerup.

The following numbered steps, in conjunction with Figure 2–3, describe how the CPUs access the system resources.

1. The CPU puts a 32−bit address onto the Mbus.
2. The Mbus address decoder (MAD or LMAD) decodes the upper ten address bits and enables access to a resource such as memory.
3. The 32−bit address points to a location within the selected resource.

Figure 2–3 shows how addresses are decoded.

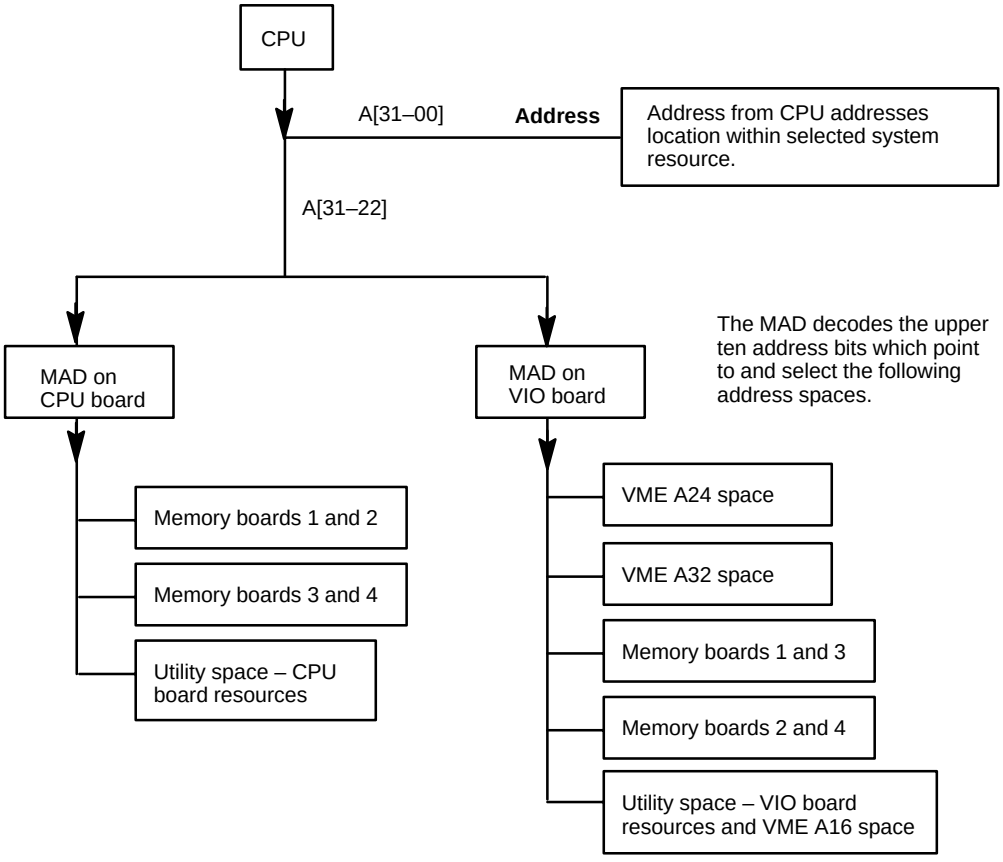


Figure 2−3 Decoding Addresses from a CPU

Addressing a VME Controller

This section describes how to address VME controllers. This addressing process is transparent to the programmer, and much of this description is intended to assist those who need to reprogram the Mbus Address Decoders.

Construct addresses to A16, A24 or A32 VME controllers as follows:

A16 = **FFFF** *nnnn* where *nnnn* is a location within the 64−Kbyte A16 space.

A24 = **FE***nn nnnn* where *nn nnnn* is a location within the 16−Mbyte A24 space.

A32 = *nnnn nnnn* where *nnnn nnnn* is a location within the 4−Gbyte A32 space.

Figure 2–4 illustrates how the VMEbus decodes addresses, and where the address modifier bits come from.

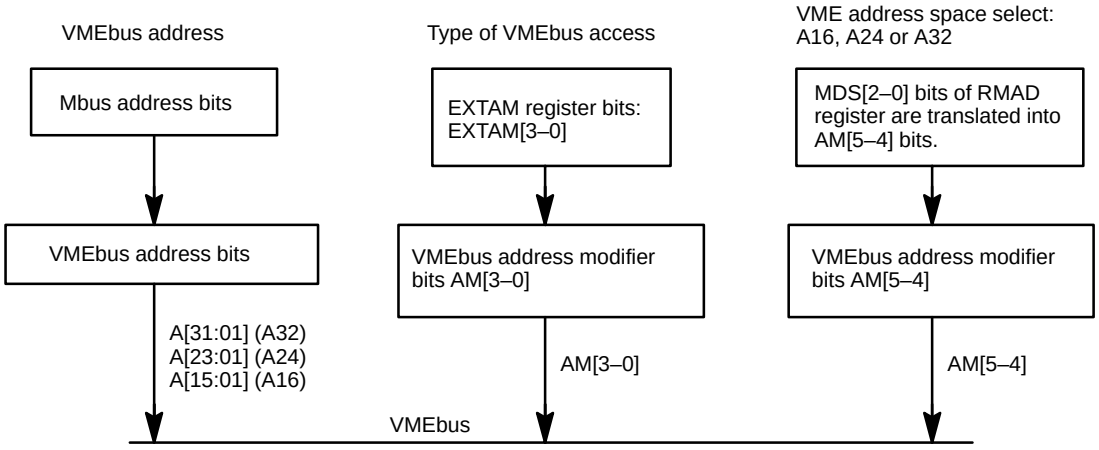


Figure 2−4 Decoding Addresses to the VMEbus

The MDS[3–0] bits, derived from the address, are translated into the address modifier bits AM[5–4]. The address modifier bits inform the VME controllers which VME space the address is written to: A16, A24 or A32 space. See Table 2–1 and Table 2–2 in the description of the EXTAM register that follows for more information about the address modifier bits.

EXTAM Extended Address Modifier

FFF8 8014 Read/write

The extended address–modifier register (EXTAM) supplies the address modifier bits AM[3–0] to the VMEbus when a CPU accesses the VMEbus. The address modifier bits AM[5–0] supply the VME devices with information such as address size, type of access, and identification of the bus master. Address modifier bits AM[5–4] identify the VME address space, and are decoded (by logic) from MDS[2–0] as shown below. AM[5–4] is put on the VMEbus. Table 2–1 illustrates how the address modifier bits (AM[5–4]) are derived from the Mbus device select bits (MDS[2–0]) found in the RMAD register.

Table 2–1 Address Modifier Translation Table

VME space	MDS2	MDS1	MDS0	AM5	AM4
A32	1	0	0	0	0
A24	1	0	1	1	1
A16	1	1	1	1	0

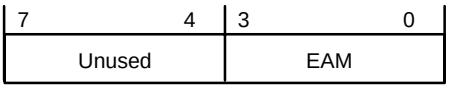
The address modifier defines the type of transfer or access, and is driven onto the VMEbus with each address. Table 2–2 defines the address modifier bits.

Table 2–2 Address Modifiers (Transfer Type)

AM3	AM2	AM1	AM0	Type of access
1	1	1	1	Supervisor block transfer
1	1	1	0	Supervisor program access
1	1	0	1	Supervisor data access
1	0	1	1	User block transfer
1	0	1	0	User program access
1	0	0	1	User data access
NOTE: A user transfer or user access is the same as a non−priveleged access defined in the VMEbus specification.				

Initialize EXTAM before addressing a VME controller.

Resets do not affect EXTAM.



Bit	Mnemonic	Function
7−4	Unused	
3−0	EAM	Extended Address Modifiers Drives the address modifiers AM[3−0] onto the VMEbus when a CPU writes an address to the VMEbus. The address modifier defines the type of transfer or access (see Table 2−2, Address Modifiers).

Addressing System Memory and Registers from a VME Controller

This section describes how VME controllers address system memory and registers. This addressing process is transparent to the programmer, and this description is intended to assist those who need to reprogram the Mbus Address Decoder. The following text and illustrations explain how the CPUs access the system resources.

CAUTION: This section may not be important to you unless you are developing power–up or diagnostic code and need to remap system resources. The system address space is mapped during system powerup, and should not be changed after powerup.

VME controllers have one of three address spaces: A16, A24 or A32 space, and are respectively referred to as A16, A24 or A32 controllers. The number of address lines driven by the controller defines the address space or range available.

A16 controllers have 16 address lines. A16 controllers can access only the Global Control and Status (GCS) registers and other VME controllers.

A24 controllers have 24 address lines. A24 controllers can access assigned system memory only after the CPU has instructed them to do so. When an A24 controller accesses system memory, the EXTAD register appends the upper eight address bits to the A24 address. This enables the A24 device to address the correct pages in memory. When addressing system memory, the address modifier bits AM[3−0] define the type of access: whether the transfer is to user space or to supervisor space, and whether these accesses are block, program or data accesses.

A32 controllers have 32 address lines, therefore their addressing range extends throughout the 4 GB system address space. A32 controllers can access assigned system memory and the GCS registers as well as other VME controllers.

The VME Address Decoders (VAD) process addresses from VME controllers, and if these addresses decode addresses from VME controllers. There are two VADs: one to represent A24 controllers called the A24 VAD, and one for the A32 controllers called the A32 VAD. When a VME controller addresses a system board resource or a location in system memory that it has access to, the appropriate VAD enables access to the location. If the address points to space that is not allocated to the VME controllers, the VAD will not allow access to the resource. This restriction prevents VME controllers from accessing system resources that are not assigned for use by VME controllers.

The GRPAD[7–0] and BDAD[3–0] DIP switches define the base address of the GCS registers.

Figure 2–5 illustrates how VME controllers address system board resources and system memory.

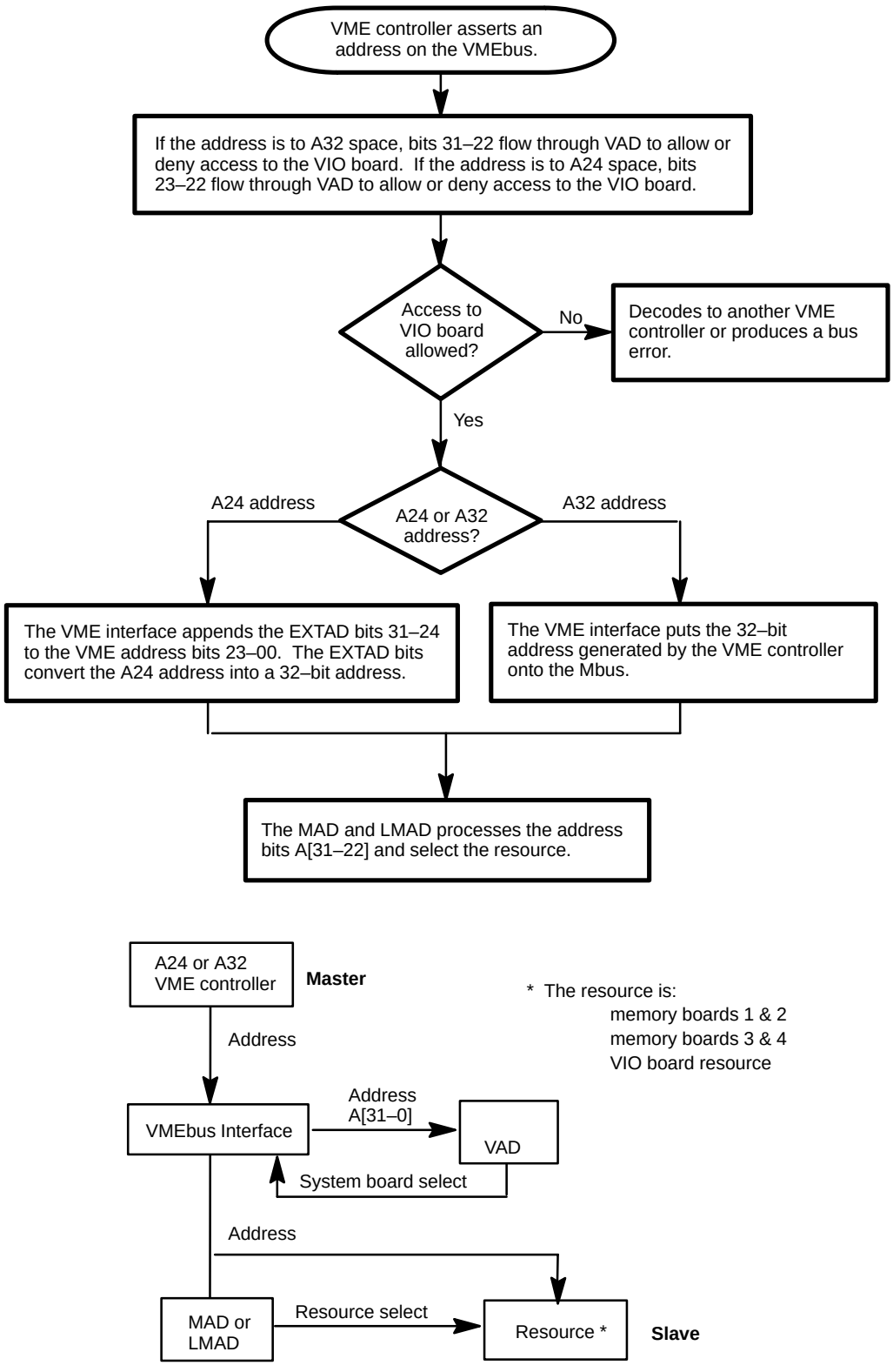


Figure 2–5 Addressing System Resources from a VME Controller

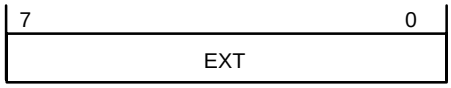
The EXTAD register, whose definition follows, is appended to addresses from A24 VME controllers to generate 32-bit addresses for the Mbus.

EXTAD — Extended Address

FFF8 8010 — Read/write

The extended address register provides the upper 8 Mbus address bits when an A24 VMEbus device accesses the Mbus. EXTAD is loaded during powerup with a base address for VME access to system memory.

Resets do not affect EXTAD.



Bit	Mnemonic	Function
7–0	EXT	Extended Address Supplies the address bits A[31–24] to the Mbus when an A24 VMEbus device accesses the Mbus.

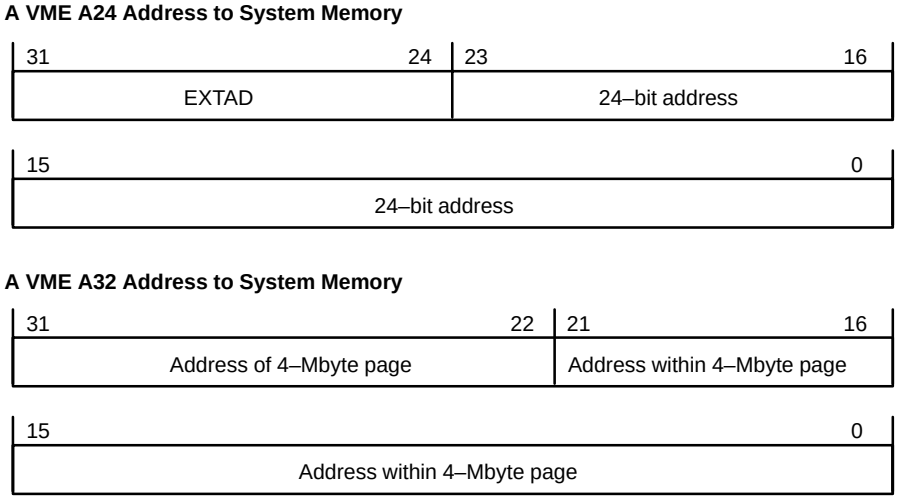


Figure 2–6 Structure of Addresses from VME Controllers to System Memory

End of Chapter

Chapter 3
Interrupts

Interrupts are a means for the system resources to notify a CPU of a condition that needs attention. Each interrupt has an associated interrupt service routine that the CPU executes.

This chapter discusses interrupts, how the interrupting devices interrupt the CPU, and how the CPU handles the interrupts.

CPU Interrupt Registers

This section describes the registers used to interrupt the CPUs and pass interrupt information.

The VIO board has the following registers:

CLRINT	Clear Interrupt
CLRSWI	Clear Software Interrupts
IEN	Interrupt Enable
ISS	Interrupt Source Status
IST	Interrupt Status
SETSWI	Set Software Interrupts

Each *CPU board* has the following registers:

ABTCLR	Clear Abort Interrupt
JPIEN	Job Processor Interrupt Enable
JPIST	Job Processor Interrupt Status
XCLR	Clear Cross Interrupt
XSET	Set Cross Interrupt

The IST, SETSWI, JPIST and XSET interrupts pass interrupt requests to the CPUs, and sefine which interrupt requests are currently asserted. When your interrupt service routine handles an interrupt, be sure to clear that interrupt.

The CLRINT, CLRSWI, and XCLR registers clear some interrupts. If you need to clear an interrupt, check these registers. If the interrupt appears in one of these registers, write to the register as defined in the register description.

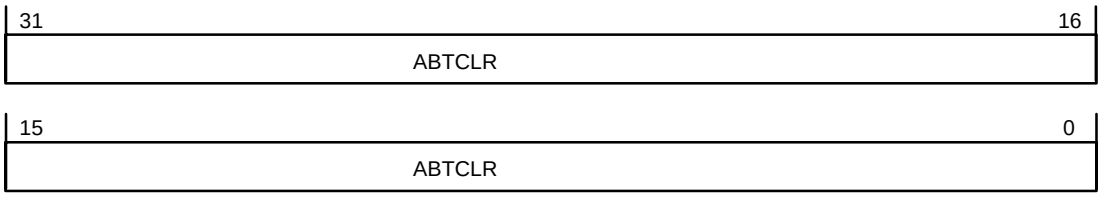
The IEN and JPIEN enable or mask interrupt requests passed through the IST and JPIST registers. To enable or mask an interrupt, write to the appropriate register as defined in the register descriptions that follow.

ABTCLR Clear Abort Interrupt

FFC8 3000 Write only

The Clear Abort Interrupt (ABTCLR) register clears the abort interrupt (bit 1 (ABT) of the JPIST register) to 0 when ABTCLR is either read or written.

NOTE: Each CPU board has an ABTCLR register. The currently active job processor uses the ABTCLR register.



Bit	Mnemonic	Function
31–0	ABTCLR	Clear Abort Interrupt Clears the abort interrupt when either read or written. NOTE: When read, the value returned is undefined. When written, by convention write a 0 even though any value will work.

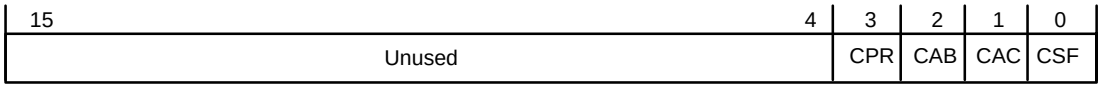
CLRINT Clear Interrupt

FFF8 408C Write Only

The Clear Interrupt (CLRINT) register clears the abort, ac fail, system fail, and printer interrupts in the IST register. The abort interrupt is generated when the abort pushbutton is pressed. The ac fail interrupt is generated when the power supply asserts the $\overline{\text{ACFAIL}}$ line on the VMEbus. The system fail interrupt is generated when any VME controller asserts $\overline{\text{SYSFAIL}}$.

To clear an interrupt request, write a 1 to the appropriate bit in the CLRINT register. For example, to clear a system failure interrupt request, write a 1 into the CSF bit. This clears the System Failure (SF) bit in the Interrupt Status (IST) register.

Resets do not affect CLRINT.



Bit	Mnemonic	Function
15–4	Unused	
3	CPBE	Clear the Printer Buffer (PBE) Interrupt Request. 1 Clears the printer buffer interrupt request. 0 Leaves the printer buffer interrupt request unchanged.
2	CABT	Clear the abort (ABT) Interrupt Request (bit 31 of IST). 1 Clears the ABRT interrupt request. 0 Leaves the ABRT interrupt request unchanged.
1	CACF	Clear the AC fail (ACF) Interrupt Request (bit 30 of IST). 1 Clears the ACF (power fail) interrupt request. 0 Leaves the ACF (power fail) interrupt request unchanged.
0	CSF	Clear the system failure (SF) Interrupt Request (bit 20 of IST). 1 Clears the SF interrupt request. 0 Leaves the SF interrupt request unchanged.

CLRSWI Clear Software Interrupt

FFF8 4084 Write Only

The Clear Software Interrupt (CLRSWI) register clears software interrupts. To clear a software interrupt, set a bit in CLRSWI. This clears the corresponding interrupt in the Interrupt Status (IST) register.

15 8	7	6	5	4	3	2	1	0
Unused	CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0

Bit	Mnemonic	Function
7	CI7	Clear Software Interrupt 7 1 Clears software interrupt 7. 0 Leaves software interrupt 7 unchanged.
6	CI6	Clear Software Interrupt 6 1 Clears software interrupt 6. 0 Leaves software interrupt 6 unchanged.
5	CI5	Clear Software Interrupt 5 1 Clears software interrupt 5. 0 Leaves software interrupt 5 unchanged.
4	CI4	Clear Software Interrupt 4 1 Clears software interrupt 4. 0 Leaves software interrupt 4 unchanged.
3	CI3	Clear Software Interrupt 3 1 Clears software interrupt 3. 0 Leaves software interrupt 3 unchanged.
2	CI2	Clear Software Interrupt 2 1 Clears software interrupt 2. 0 Leaves software interrupt 2 unchanged.
1	CI1	Clear Software Interrupt 1 1 Clears software interrupt 1. 0 Leaves software interrupt 1 unchanged.
0	CI0	Clear Software Interrupt 0 1 Clears software interrupt 0. 0 Leaves software interrupt 0 unchanged.

IEN, IEN0, IEN1, IEN2 and IEN3 — Interrupt Enable

FFF8 403C **IEN** **Write Only**

FFF8 4004 **IEN0**

FFF8 4008 **IEN1**

FFF8 4010 **IEN2**

FFF8 4020 **IEN3**

The Interrupt Enable registers (IEN, IEN0 – IEN3) enable and mask interrupts to the CPU boards. IEN0 – IEN3 enable interrupts to the corresponding CPU board (0 – 3), while IEN enables interrupts to all CPU boards. To enable an interrupt, write a 1 into the corresponding bit in the appropriate interrupt enable register. To mask an interrupt, write a 0 into the corresponding bit in the interrupt enable register. The interrupt enable registers and the interrupt status register are mirror images of each other.

The interrupt enable registers are cleared to 0 by system reset and is not affected by local reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	DTI	SI7	SI6	SI5	SI4	IR7	SBM	CIO	SF	IR6	MEM	DI	SHP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	IR5	0	IR4	PBE	IR3	0	LMI	SLP	IR2	0	IR1	SI3	SI2	SI1	SI0

Bit	Mnemonic	Function
31	ABT	Abort Pushbutton 1 Enables the abort pushbutton interrupt. 0 Masks the abort pushbutton interrupt
30	ACF	AC Failure 1 Enables the ac power failure interrupt. 0 Masks the ac power failure interrupt.
29	ATO	VMEbus Timeout 1 Enables the VMEbus timeout interrupt. 0 Masks the VMEbus timeout interrupt.
28	DTI	DUART Timer 1 Enables DUART timer interrupts. 0 Masks DUART timer interrupts.
27	SI7	Software Interrupt 7 1 Enables software interrupt 7. 0 Masks software interrupt 7.
26	SI6	Software Interrupt 6 1 Enables software interrupt 6. 0 Masks software interrupt 6.
25	SI5	Software Interrupt 5 1 Enables software interrupt 5. 0 Masks software interrupt 5.

(continued)

Bit	Mnemonic	Function
24	SI4	Software Interrupt 4 1 Enables software interrupt 4. 0 Masks software interrupt 4.
23	IR7	VME Level 7 Interrupt 1 Enables the $\overline{\text{IRQ7}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ7}}$ interrupt from VME controllers.
22	Unused	
21	CIO	CIO Interrupt 1 Enables the CIO interrupt. 0 Masks the CIO interrupt.
20	SF	System Failure Interrupt 1 Enables the system failure interrupt. 0 Masks the system failure interrupt.
19	IR6	VME Level 6 Interrupt 1 Enables the IRQ6 interrupt from VME controllers. 0 Masks the IRQ6 interrupt from VME controllers.
18	MEM	Memory Interrupt 1 Enables the memory interrupt. 0 Masks the memory interrupt.
17	DI	DUART Interrupt 1 Enables the DUART interrupt. 0 Masks the DUART interrupt.
16, 15	Unused	
14	IR5	VME Level 5 Interrupt 1 Enables the $\overline{\text{IRQ5}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ5}}$ interrupt from VME controllers.
13	Unused	
12	IR4	VME Level 4 Interrupt 1 Enables the $\overline{\text{IRQ4}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ4}}$ interrupt from VME controllers.
11	PBE	Printer Buffer Empty 1 Enables the printer buffer empty interrupt. 0 Masks the printer buffer empty interrupt.
10	IR3	VME Level 3 Interrupt 1 Enables the $\overline{\text{IRQ3}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ3}}$ interrupt from VME controllers.
9	Unused	
8	LM1	Location Monitor Interrupt 1 Enables the location monitor interrupt. 0 Masks the location monitor interrupt.
7	Unused	
6	IR2	VME Level 2 Interrupt 1 Enables the $\overline{\text{IRQ2}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ2}}$ interrupt from VME controllers.
5	Unused	

(continued)

Bit	Mnemonic	Function
4	IR1	VME Level 1 Interrupt 1 Enables the $\overline{\text{IRQ1}}$ interrupt from VME controllers. 0 Masks the $\overline{\text{IRQ1}}$ interrupt from VME controllers.
3	SI3	Software Interrupt 3 1 Enables software interrupt 3. 0 Masks software interrupt 3.
2	SI2	Software Interrupt 2 1 Enables software interrupt 2. 0 Masks software interrupt 2.
1	SI1	Software Interrupt 1 1 Enables software interrupt 1. 0 Masks software interrupt 1.
0	SI0	Software Interrupt 0 1 Enables software interrupt 0. 0 Masks software interrupt 0.

(concluded)

ISS Interrupt Source Status

FFF8 4088 Read Only

The Interrupt Source Status (ISS) register gives the current status of the printer buffer, abort, ac failure and system failure interrupts.

Resets do not affect ISS.

15 4	3	2	1	0
Unused	PBS	ABT	ACF	SF

Bit	Mnemonic	Function
15–4	Unused	
3	PBS	Printer Buffer Status 1 Indicates that the the printer buffer is not full. 0 Indicates that the printer buffer is full.
2	ABT	Abort Button Status 1 Indicates that the abort switch is toggled. 0 Indicates that the abort switch is not toggled.
1	ACF	$\overline{\text{ACFAIL}}$ Status 1 Indicates that an ac fail interrupt request is pending, i.e. the $\overline{\text{ACFAIL}}$ line on the VMEbus is asserted (low). 0 Indicates that there is no ac fail interrupt, i.e. the $\overline{\text{ACFAIL}}$ line on the VMEbus is not asserted (high).
0	SF	$\overline{\text{SYSFAIL}}$ Status 1 Indicates that $\overline{\text{SYSFAIL}}$ is asserted (low) on the VMEbus. 0 Indicates that $\overline{\text{SYSFAIL}}$ is not asserted (high) on the VMEbus.

IST Interrupt Status

FFF8 4040 Read Only

The Interrupt Status (IST) register contains interrupt flags that are set to 1 by devices requesting an interrupt to the CPU. When a device requests an interrupt, the interrupt request signal goes to interrupt logic which sets the corresponding interrupt bit in the IST register. The interrupt logic also interrupts the CPU. When interrupted, the CPU must execute a primary interrupt service routine to find out what kind of interrupt it is. The interrupt service routine may read an interrupt enable register as well as the interrupt status register. If it does read an interrupt enable register, it must compare the two registers to find any enabled interrupt(s). The bits in the interrupt enable registers and the interrupt status register are mirror images of each other.

Resets do not affect IST bits; IST can only be cleared using software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABT	ACF	ATO	DTI	SI7	SI6	SI5	SI4	IR7	SBM	CIO	SF	IR6	MEM	DI	SHP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Unused	IR5	Unused	IR4	PBE	IR3	Unused	LMI	SLP	IR2	Unused	IR1	SI3	SI2	SI1	SI0

Bit	Mnemonic	Function
31	ABT	Abort Pushbutton The operating system can use this to initiate a software reset. 1 Indicates that the abort switch was pressed. 0 Indicates that the abort switch was not pressed.
30	ACF	AC Failure 1 Indicates that an ac power failure has occurred. The ac failure signal originates from the power supply, which is connected to the VMEbus. 0 Indicates that there has not been an ac power failure.
29	ATO	VMEbus Arbiter Time Out 1 Indicates that the VMEbus bus grant has timed out and generated an interrupt. 0 Indicates that the VMEbus bus grant has not timed out.
28	DTI	DUART Timer Interrupt (system console) 1 Indicates that the system console DUART is requesting an interrupt. 0 Indicates that the system console DUART is not requesting an interrupt.
27	SI7	Software Interrupt 7 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.

(continued)

Bit	Mnemonic	Function
26	SI6	Software Interrupt 6 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
25	SI5	Software Interrupt 5 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
24	SI4	Software Interrupt 4 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
23	IR7	VME Level 7 Interrupt A level 7 interrupt ($\overline{IRQ7}$ on the VMEbus) is one of seven possible interrupt requests from VME controllers. $\overline{IRQ7}$ has the highest priority and $\overline{IRQ1}$ has the lowest priority. 1 Indicates that $\overline{IRQ7}$, on the VMEbus, is asserted. 0 Indicates that $\overline{IRQ7}$, on the VMEbus, is not asserted.
22	Unused	
21	CIO	CIO Interrupt 1 Indicates that the CIO requested an interrupt. 0 Indicates that the CIO did not request an interrupt.
20	SF	System Failure Interrupt 1 Indicates that the system failure ($\overline{SYSFAIL}$) line on the VMEbus was asserted. A system failure occurred. 0 Indicates that the system failure ($\overline{SYSFAIL}$) line on the VMEbus was not asserted. There has not been a system failure.
19	IR6	VME Level 6 Interrupt A level 6 interrupt (IRQ6* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ6*, on the VMEbus, is asserted. 0 Indicates that IRQ6*, on the VMEbus, is not asserted.
18	MEM	Memory Interrupt MEM defines whether or not an ECC error has occured. 1 Indicates that a memory error has occurred. 0 Indicates that a memory error has not occurred.
17	DI	DUART Interrupt 1 Indicates that the DUART is requesting an interrupt. 0 Indicates that the DUART is not requesting an interrupt.
16, 15	Unused	
14	IR5	VME Level 5 Interrupt A level 5 interrupt (IRQ5* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ5*, on the VMEbus, is asserted. 0 Indicates that IRQ5*, on the VMEbus, is not asserted.
13	Unused	
12	IR4	VME Level 4 Interrupt A level 4 interrupt (IRQ4* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ4*, on the VMEbus, is asserted. 0 Indicates that IRQ4*, on the VMEbus, is not asserted.

(continued)

Bit	Mnemonic	Function
11	PBE	Printer Buffer Empty 1 Indicates that the parallel printer port buffer is empty. PBE is used when sending more than 512 bytes to the printer. Send 512 bytes, then wait for the PBE interrupt before sending more data. 0 Indicates that the parallel printer port buffer contains data.
10	IR3	VME Level 3 Interrupt A level 3 interrupt (IRQ3* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ3*, on the VMEbus, is asserted. 0 Indicates that IRQ3*, on the VMEbus, is not asserted.
9	Unused	
8	LMI	Location Monitor Interrupt Indicates whether or not a location monitor interrupt is asserted. 1 A location monitor is asserted. 0 A location monitor is not asserted.
7	Unused	
6	IR2	VME Level 2 Interrupt A level 2 interrupt (IRQ5* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ2*, on the VMEbus, is asserted. 0 Indicates that IRQ2*, on the VMEbus, is not asserted.
5	Unused	
4	IR1	VME Level 1 Interrupt A level 1 interrupt (IRQ5* on the VMEbus) is one of seven possible interrupt requests from VME controllers. IRQ7* has the highest priority and IRQ1* has the lowest priority. 1 Indicates that IRQ1*, on the VMEbus, is asserted. 0 Indicates that IRQ1*, on the VMEbus, is not asserted.
3	SI3	Software Interrupt 3 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
2	SI2	Software Interrupt 2 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
1	SI1	Software Interrupt 1 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.
0	SI0	Software Interrupt 0 One of six software interrupts. The interrupt service routine defines the SIn priority levels. 1 Indicates that a software interrupt was generated. 0 Indicates that a software interrupt was not generated.

(concluded)

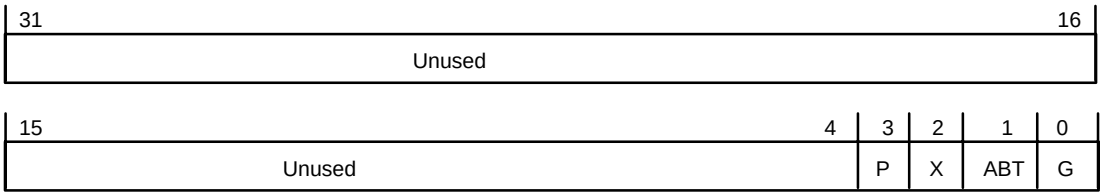
JPIEN Job Processor Interrupt Enable

FFC8 0000 Read/Write

The Job Processor Interrupt Enable register (JPIEN) enables or masks interrupts to the currently active job processor. To enable an interrupt, write a 1; to mask an interrupt, write a 0.

NOTE: Each CPU board has a JPIEN register. The currently active job processor uses the JPIEN register.

Local resets do not clear JPIEN to 0.



Bit	Mnemonic	Function
31–4	Unused	
3	P	PIT Interrupt 1 Enables the PIT interrupt. 0 Disables the PIT interrupt.
2	X	Cross Interrupt 1 Enables the cross interrupt. 0 Disables the cross interrupt.
1	ABT	Abort Pushbutton 1 Enables the abort pushbutton. 0 Disables the abort pushbutton.
0	G	Global Interrupt 1 Enables the global interrupts 0 Disables the global interrupts.

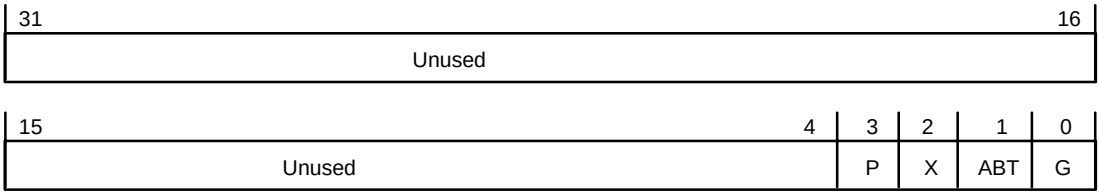
JPIST Job Processor Interrupt Status

FFC8 1000 Read only

The Job Processor Interrupt Status (JPIST) register passes a few interrupts to the currently active job processor, and indicates the status of interrupts.

NOTE: Each CPU board has a JPIST register. The currently active job processor uses the JPIST register.

Resets clear JPIST to 0.



Bit	Mnemonic	Function
31–4	Unused	
3	P	PIT Interrupt 1 Indicates that the PIT has requested an interrupt. 0 Indicates that the PIT has not requested an interrupt.
2	X	Cross Interrupt A cross interrupt is an interrupt asserted by one job processor to any other job processor. See the XSET and XCLR registers. 1 Indicates that a cross interrupt was asserted. 0 Indicates that a cross interrupt was not asserted.
1	ABT	Abort Pushbutton 1 Indicates that the ABORT switch was pressed. 0 Indicates that the ABORT switch was not pressed.
0	G	Global Interrupt 1 Indicates that an interrupt is asserted through the IST register. 0 Indicates that there is no interrupt at the IST register.

SETSWI Set Software Interrupt

FFF8 4080 Write Only

The Set Software Interrupt (SETSWI) register generates user-defined interrupts asserted by software. To generate a software interrupt, set a Set Software Interrupt (SI*n*) bit. This sets the corresponding interrupt in the Interrupt Status (IST) register and asserts the interrupt. Unlike condition–specific interrupts, the software interrupts can be used for any software–defined condition that requires interrupt servicing and has an associated software interrupt service routine.

15 8	7	6	5	4	3	2	1	0
Unused	CI7	CI6	CI5	CI4	CI3	CI2	CI1	CI0

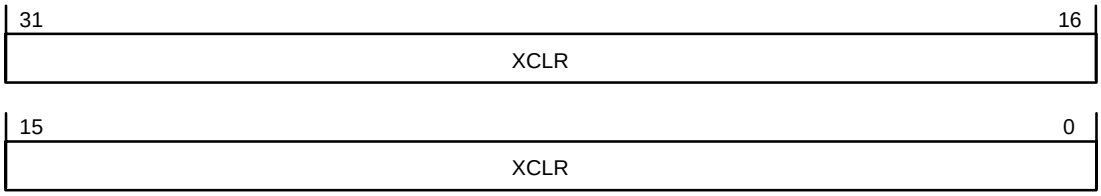
Bit	Mnemonic	Function
7	SI7	Set Software Interrupt 7. 1 Generates a software interrupt 7. 0 Leaves software interrupt 7 unchanged.
6	SI6	Set Software Interrupt 6. 1 Generates a software interrupt 6. 0 Leaves software interrupt 6 unchanged.
5	SI5	Set Software Interrupt 5. 1 Generates a software interrupt 5. 0 Leaves software interrupt 5 unchanged.
4	SI4	Set Software Interrupt 4. 1 Generates a software interrupt 4. 0 Leaves software interrupt 4 unchanged.
3	SI3	Set Software Interrupt 3. 1 Generates a software interrupt 3. 0 Leaves software interrupt 3 unchanged.
2	SI2	Set Software Interrupt 2. 1 Generates a software interrupt 2. 0 Leaves software interrupt 2 unchanged.
1	SI1	Set Software Interrupt 1. 1 Generates a software interrupt 1. 0 Leaves software interrupt 1 unchanged.
0	SI0	Set Software Interrupt 0. 1 Generates a software interrupt 0. 0 Leaves software interrupt 0 unchanged.

XCLR Clear Cross Interrupt

FFC8 2000 Write only

The Clear Cross Interrupt (XCLR) register clears the cross interrupt when either read or written.

NOTE: Each CPU board has an XCLR register.

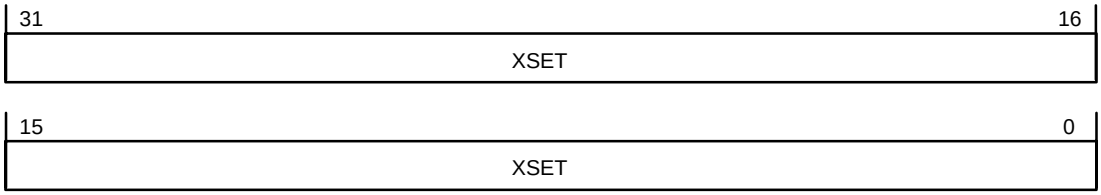


Bit	Mnemonic	Function
31–0	XCLR	Clear Cross Interrupt Clears the cross interrupt when either read or written. NOTE: When read, the value returned is undefined. When written, by convention write a 0 even though any value will work.

XSETn, XSETALL — Set Cross Interrupt

FFF8 B000	**XSET0**	**Write Only**
FFF8 B004	**XSET1**	
FFF8 B008	**XSET2**	
FFF8 B00C	**XSET3**	
FFF8 B010	**XSET4**	
FFF8 B014	**XSET5**	
FFF8 B018	**XSET6**	
FFF8 B01C	**XSET7**	
FFF8 B3FC	**XSETALL**	

The Set Cross Interrupt (XSETn) registers, when written, set cross interrupts to individual or all job processors via the JPIST registers. The JPIEN registers mask or enable the cross interrupt to job processors.



Bit	Mnemonic	Function
31–0	XSET	Set Cross Interrupt Sets the cross interrupt when written.

VME Interrupts

VME controllers assert interrupts via seven general–purpose VME interrupt request lines IRQ[7–1]* located on the VMEbus. The VME Interrupt Request lines (IRQ[7-1]*) are assigned levels of interrupt priority with IRQ7* having the highest priority and IRQ1* having the lowest priority.

When a VME controller asserts one of the seven VME interrupts, the interrupt logic on the VIO board sets the corresponding Interrupt Request (IR*n*) bit in the IST register and asserts the INT line to the CPU boards. A CPU acknowledges one of these interrupts by reading the corresponding VME Interrupt Acknowledge and Vector (VIAV*n*) register. When read, the VIAV*n* register asserts the interrupt acknowledge signals IACK* and IACKOUT* on the VMEbus. These interrupt acknowledge signals trigger the VME controller to write the interrupt vector to the VMEbus. The VME interface logic writes this to the Mbus for the CPU to read. The next section, “Interrupting the CPU,” illustrates how VME controllers assert interrupts, and how to handle these interrupts.

The following two sections describe how VME interrupts pass between VME controllers and CPUs, and how to handle the interrupts.

How a VME Controller Interrupts the CPUs

The following steps and Figure 3–1 illustrate how a VME controller asserts an interrupt.

1. The VME controller asserts the Interrupt Request (IRQn*) line on the VMEbus.
2. Interrupt logic sets the interrupt request level (IRn) bit in the Interrupt Status (IST) register high (1).
3. Interrupt logic asserts the INT line to the CPU to notify the CPU that an interrupt request is pending.

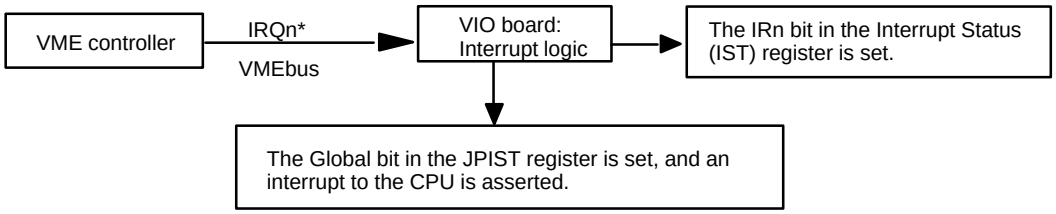


Figure 3–1 VME Controller Asserting an Interrupt

How a CPU Responds to a VME Interrupt

When a CPU responds ts an interrupt, it executes an interrupt service routine that finds, acknowledges and handles the interrupt as follows:

1. Read the appropriate VME Interrupt Acknowledge and Vector (VIAV*n*) register to acknowledge the interrupt and to get the interrupt vector. There are seven VIAV*n* registers, one for each interrupt level. Examine the state of the IACK Bus Error (IBE) bit. If IBE is set, another job processor is servicing the interrupt; disregard the interrupt and return to the previous process.
2. Execute the interrupt service routine located at the vector obtained from the VIAV*n* register.

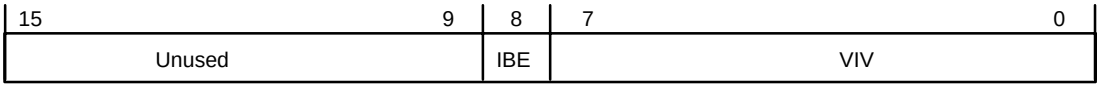
The VIAVn registers are described on the next page.

VIAV*n* VME Interrupt Acknowledge and Vector

FFF8 5004	**VIAV1**	**Read Only**
FFF8 5008	**VIAV2**	
FFF8 500C	**VIAV3**	
FFF8 5010	**VIAV4**	
FFF8 5014	**VIAV5**	
FFF8 5018	**VIAV6**	
FFF8 501C	**VIAV7**	

The CPU acknowledges interrupts and obtains VME interrupt vectors via the seven VME Interrupt Acknowledge and Vector (VIAV*n*) registers. Each of the seven interrupt request levels has a VIAV*n* register associated with it. When a VME controller interrupts the CPU via one of the seven VME interrupt request levels, the CPU acknowledges the interrupt request and obtains the interrupt request vector via the VIAV*n* register. When the CPU reads VIAV*n*, the interrupt logic asserts the interrupt acknowledge lines (IACK* and IACKOUT*) to the VMEbus. The VME controller responds by writing the interrupt vector to the data lines and by asserting the Data Transfer Acknowledge (DTACK*) signal on the VMEbus. If the DTACK* signal is not asserted, the interrupt logic sets the Interrupt Acknowledge Bus Error (IBE) bit in the VIAV*n* register. If IBE is set, the CPU will disregard the interrupt vector. If IBE is set, the CPU disregards the interrupt and returns to its previous process.

Resets do not affect the VIAV*n* registers.



Bit	Mnemonic	Function
15–9	Unused	
8	IBE	IACK Bus Error 1 Indicates that the last VME interrupt acknowledge (IACK) was terminated by a VMEbus error (BERR). 0 Indicates that the last VMEbus IACK cycle was successfully completed. The VIV bits contain a valid interrupt vector.
7–0	VIV	VME Interrupt Vector VIV is the interrupt vector from the VME controller generating the interrupt. NOTE: VIV contains a valid vector only when the IBE bit is cleared.

How to Interrupt a VME Controller

CPUs interrupt VME controllers via the VME Interrupt Request Level (VIRL) register. The CPU supplies the VME controller with an interrupt vector via the VME Interrupt Vector (VIV) register.

To interrupt a VME controller:

1. Write the interrupt vector into the VIV register.
2. Write the interrupt level into the VIRL register.
3. The interrupt logic asserts the appropriate interrupt request line ($\overline{IRQ[6-0]}$) to the VME controllers.

The VME controller acknowledges the interrupt via the IACK* and IACKOUT* lines, then reads the interrupt vector from VIV.

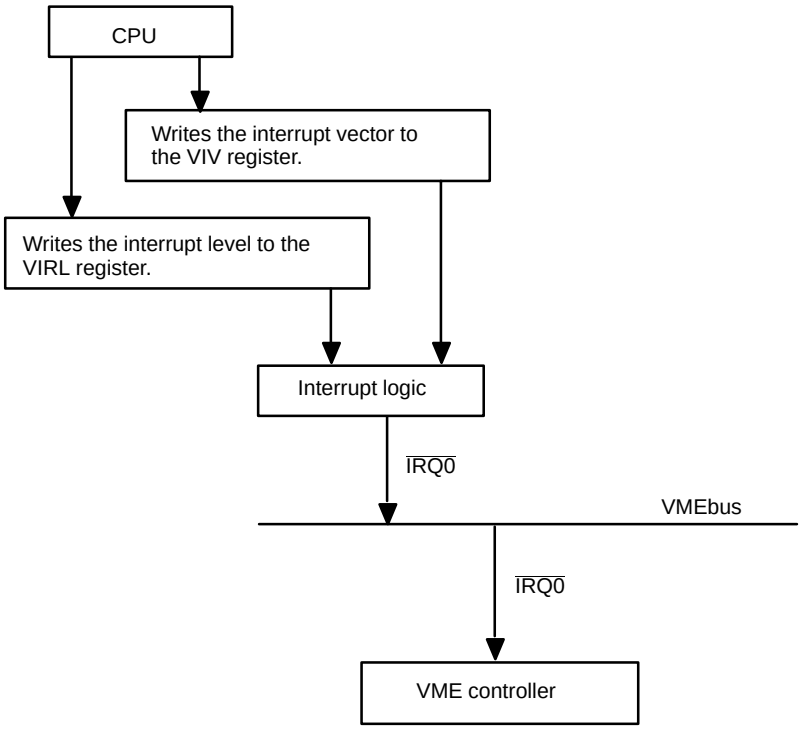


Figure 3–2 CPU Initiating a Level–1 Interrupt to VME Controller

VIRL VME Interrupt Request Level

FFF8 5000 Read/Write

The CPU interrupts VME controllers via the VME Interrupt Request Level (VIRL) register. Interrupt requests include a request level that defines the priority of the interrupt: level–7 interrupts have the highest priority, while level–1 interrupts have the lowest priority. To interrupt a VME controller, the CPU must first write the interrupt vector into the VIV register, then write the binary–coded interrupt level into the VME Interrupt Request Level (VIRL) bits of the VIRL register. A zero value in VIRL indicates that the CPU has not initiated an interrupt request. When the VME controller acknowledges the interrupt, VIRL is reset to zero.

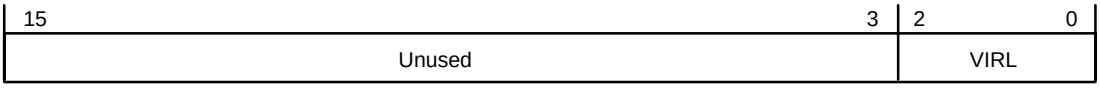
Resets clear VIRL to 0.

NOTE: You cannot write to the VME Interrupt Vector (VIV) register unless VIRL is cleared to 0; the interrupt vector cannot be changed while an interrupt is pending.

NOTE: If a VME controller does not acknowledge an interrupt asserted by a CPU, clear the VIRL bits to 0 to remove the interrupt.

The CPU interrupts a VME controller as follows:

1. The CPU reads VIRL. If VIRL is 0, indicating that there is no interrupt request pending, the CPU writes the interrupt vector into the VIV register. If VIRL has a value between 1 and 7, indicating that an interrupt request is pending, the CPU must wait until the existing VME interrupt completes and the VIRL register is cleared. After VIRL is cleared, the CPU can write the interrupt vector into the VIV register.
2. The CPU writes the interrupt request level into the VIRL bits of the VIRL register.
3. The VME controller acknowledges the interrupt and reads the interrupt vector from the VIV register.



Bit	Mnemonic	Function
15–3	Unused	
2–0	VIRL	VME Interrupt Request Level Initiates an interrupt to a VME controller. The interrupt has a specific level as defined below. Level–7 has the highest priority, and level–1 has the lowest priority.

Bit 2	Bit 1	Bit 0	Interrupt Level
0	0	0	No interrupt request
0	0	1	Level–1 VME interrupt request
0	1	0	Level–2 VME interrupt request
0	1	1	Level–3 VME interrupt request
1	0	0	Level–4 VME interrupt request
1	0	1	Level–5 VME interrupt request
1	1	0	Level–6 VME interrupt request
1	1	1	Level–7 VME interrupt request

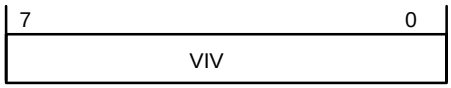
VIV VME Interrupt Vector

FFF8 5020 Read/Write

When a CPU asserts a VME interrupt, the VIV register provides the interrupt vector to the VME controller. The CPU must load the interrupt vector into VIV before initiating a VME interrupt request.

Resets do not affect VIV.

NOTE: You cannot write to VIV unless VIRL is cleared to 0. This prevents you from changing the interrupt vector while an interrupt is pending.



Bit	Mnemonic	Function
7–0	VIV	VME Interrupt Vector Contains the vector of the device that initiated the interrupt. Before a CPU generates an interrupt, it must read the contents of VIRL (must be 0) then write the interrupt vector into VIV. When the VME controller acknowledges an interrupt, if the contents of address bits A[3–0] of the VMEbus match the request level bits in VIRL, the VME controller will put the contents of VIV[7–0] on the VMEbus data bits D[7–0].

End of Chapter

Chapter 4
System Control Registers

This chapter describes the system control registers that do not fall under another specific category such as interrupt registers or addressing registers.

The VIO board has the following registers:

BASAD	Base Address
CCS	CPU Control and Status
ERROR	Error
GLBRES	Global Reset
UCS	Utility Control and Status

Each *CPU board* has the following registers:

JPDIAG	Job Processor Diagnostics
WHOAMI	Job processor (CPU) and CPU board identification

The IST, SETSWI, JPIST and XSET interrupts pass interrupt requests to the CPUs, and sefine which interrupt requests are currently asserted. When your interrupt service routine handles an interrupt, be sure to clear that interrupt.

The following pages describe these registers in alphabetical order.

The IEN and JPIEN enable or mask interrupt requests passed through the IST and JPIST registers. To enable or mask an interrupt, write to the appropriate register as defined in the register descriptions that follow.

The system also has global control and status (GCS) registers described later in this chapter. VME controllers can access these registers; they can not access any other registers on the IOP or CPU boards. The section, “G” describes the following registers and how to use them.

GLOBAL0	Global register 0
GLOBAL1	Global register 1
BRDID	Board ID
GPCSn	General Purpose Control and Status (five registers 0–4)

BASAD Base Address

FFF8 7004 Read Only

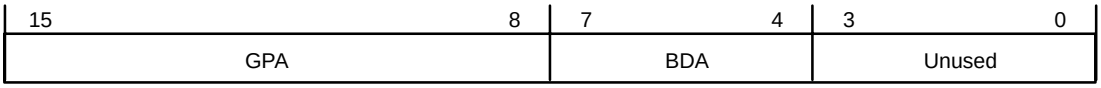
Base Address (BASAD) contains the address entered into the Group Address (GRPAD) and Board Address (BDAD) DIP switches located on the VIO board. These values are stored in the Group Address (GPA) and Board Address (BDA) bits in the BASAD register. Together, they define the three most–significant nibbles of the A16 address that VME controllers must use to access the Global Control and Status (GCS) registers.

To access one of these registers, a VME controller writes the 16–bit address of that register. The VME interface compares address bits A[15–4] with the contents of the BASAD register. If the two addresses are the same, the VME controller can then write to or read from the register. Bits A[3–0] select one of the eight GCS registers.

When a VME controller accesses a GCS register, it does the following within one bus cycle:

1. The VME controller writes the address of the GCS register onto the VMEbus.
2. The VME interface logic compares the value in BASAD with the address from the VME controller. If they are the same address, the VME interface logic passes the address onto the Mbus.
3. The VME controller reads from or writes to the address.

Resets do not affect BASAD.



Bit	Mnemonic	Function
15–8	GPA	Group Address GRPAD contains a copy of the Group Address (GRPAD) switch settings.
7–4	BDA	Board Address BDAD contains a copy of the Board Address (BDAD) switch settings.
3–0	Unused	Not used by BASAD; these bits point to one of the eight GCS registers.

CCS CPU Control and Status

FFF8 8000 Read/Write

The CPU Control and Status (CCS) register, when read, indicates the status of the Mbus Address Decoder (MAD) and the VMEbus Address Decoders (VAD). When you power up your system, the power–up program loads the address decoders and sets the decoder valid bits in the CCS register. Setting the MDV and VDV bits indicates that the address decoders have been loaded and are valid.

System resets clear CCS, local resets do not affect CCS.

CAUTION: Use caution when clearing MDV and VDV. After the Mbus and VMEbus maps are created and verified, MDV and VDV are set. If a problem develops, reset the system to clear the CCS register.

15 2	1	0
Unused	MDV	VDV

Bit	Mnemonic	Function
15–2	Unused	
1	MDV	Mbus Address Decoder Valid Defines whether or not the Mbus Address Decoder (MAD) RAM has been loaded. 1 The Mbus decoder RAM has been loaded. 0 The Mbus decoder RAM has not been loaded.
0	VDV	VMEbus Address Decoder Valid Defines whether or not the VMEbus Address Decoder (VAD) RAM has been loaded. 1 The VMEbus decoder RAM has been loaded. 0 The VMEbus decoder RAM has not been loaded.

CON (Resides on VIO board) Control

FFF8 9000 Read/Write

The Control (CON) register resets the printer and DUART, and sets the printer strobe for use with Centronics–compatible printers.

31	30	29	28	27 23	22	21 16
Reserved	PRE	Reserved	DRE	Reserved	PSP	Reserved

15 0
Reserved

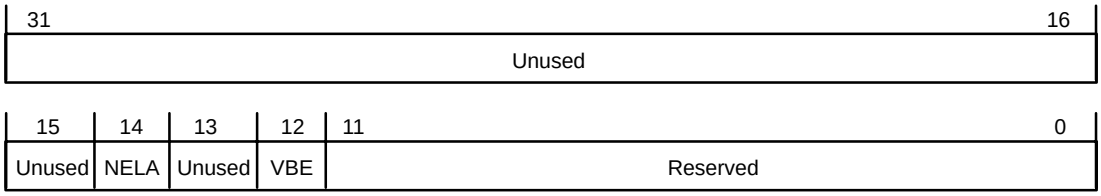
Bit	Mnemonic	Function
31	Reserved	
30	PRE	Printer Reset 1 Reset the printer. 0 No action.
29	Reserved	
28	DRE	DUART Reset 1 Resets the DUARTs. 0 No action.
27–23	Reserved	
22	PSP	Printer Strobe Polarity 1 Centronics 0 Unused
21–0	Reserved	

ERROR Error

FFF8 8004 Read Only

The Error (ERROR) register provides bus error information. The ERROR register is cleared automatically after it is read; errors contained within the ERROR register occurred since the last time the CPU read the register.

System resets clear ERROR, local resets do not affect ERROR.



Bit	Mnemonic	Function
31–15	Unused	
14	NELA	Non–Existent Local Address Indicates whether or not the CPU tried to address a non–existent location. A non–existent location is one that is not mapped into the Mbus Address Decoder (MAD). 1 Indicates that the CPU tried to address a non–existent location. 0 Indicates that all Mbus addresses have been to valid locations. NOTE: When set to 1, NELA generates a bus fault.
13	Unused	
12	VBE	VMEbus Error Indicates whether or not a VMEbus Error (BERR*) occurred in response to a bus request from the VIO board. 1 Indicates that a VMEbus error (BERR*) occurred. 0 Indicates that no VMEbus errors (BERR*) occurred.
11–0	Reserved	

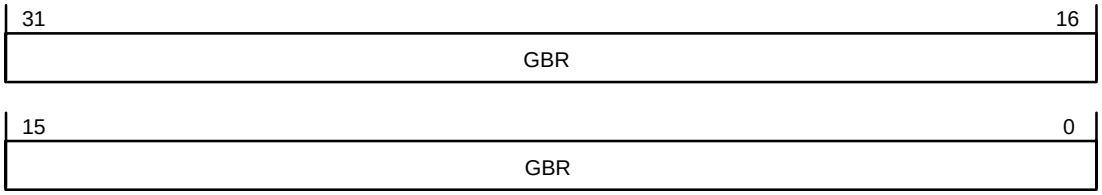
GLBRES Global Reset

FFF8 700C Write Only

The Global Reset (GLBRES) register generates a System Reset (SRST) when written to. GLBRES resets the entire system, including all VME controllers.

CAUTION: Setting any bit in the GLBRES register will reset the entire system.

NOTE: GLBRES does not reset the serial interface. Instead, use the DUART Reset (DRE bit 28) bit of the Control (CON) register as described in 5, “Memory”.



Bit	Mnemonic	Function
31–0	GBR	Global Reset Writing any value to GLBRES resets the system.

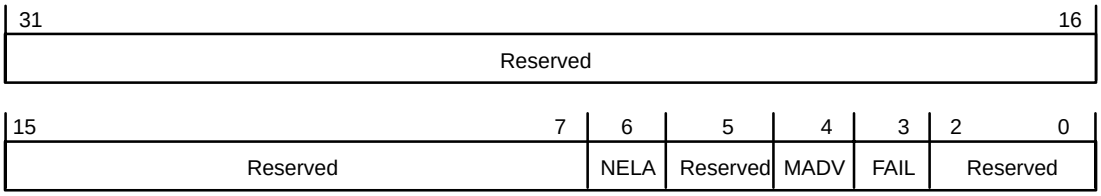
JPDIAG Job Processor Diagnostic and Status

FFC8 6000 Read/Write as noted

The Job Processor Diagnostic and Status (JPDIAG) register provides diagnostic status and control for the CPU board.

Each CPU board has a JPDIAG register.

Resets clear JPDIAG to 0.



Bit	Mnemonic	Function
31–7	Reserved	
6	NELA	Non Existent Local Address (Read/Write) 1 A non–existent local address was referenced. Stays asserted until cleared. To clear NELA, write a 0 to bit 6. 0 Last address exists. Reset to 0.
5	Reserved	
4	MADV	MAD is Valid (Read/Write) 0 MAD is invalid, physical addresses translated to FFCx xxxx. 1 MAD is valid. Reset to 0.
3	FAIL	Fail LED (red) (Read only) 0 Red LED is off. 1 Red LED is on. Reset to 0.
2–0	Reserved	

UCS Utility Control and Status

FFF8 7000 Read/Write

The Utility Control and Status (UCS) register provides status information when read and control of many hardware functions when written to. The UCS register bits can all be read from and written to with the exception of the PUP* bit. If writing to PUP*, you can only set PUP*; you cannot clear PUP*. Only hardware can clear PUP*.

15	14	13	12	11	10 9	8	7	6	5	4 3	2	1 0
Unused	PUP	ASF	BIR	VAM	VRL	RNV	FAIR	VRM	ETO	VTS	EWD	WDA

Bit	Mnemonic	Function
15	Unused	
14	PUP	Power−up Initialization Indicator Indicates whether or not an ac failure occured. PUP* is set when a VME controller asserts the ACFAIL line of the VMEbus. 1 Indicates that an ac failure did not occur (ACFAIL* on the VMEbus was not asserted). 0 Indicates that an ac failure occurred (ACFAIL* was asserted). PUP* is not affected by system reset or local reset. NOTE: PUP* is cleared while ACFAIL* is asserted or while the system is powering up. PUP* is set by the CPU after the system powers up. Only the CPU can set PUP*.
13	$\overline{\text{ASF}}$	Assert System Failure When set, asserts a system failure. 1 Leaves system failure unchanged. 0 Asserts a system failure. If Inhibit SYSFAIL* (ISF) of the GLOBAL1 register is set, the VIO board will not assert the SYSFAIL* line on the VMEbus. If ISF is 0, the VIO board will assert the SYSFAIL* line on the VMEbus. Cleared by either system reset or local reset. NOTE: ASF* will light the FAIL LED if asserted (cleared to 0).
12	BIR	Broadcast Interrupt Request Used to assert a broadcast interrupt request over the VMEbus via IRQ1*. 1 Asserts IRQ1*. 0 Does not assert IRQ1*. Cleared by either system reset or local reset. NOTE: IRQ1* is an open−collector line; it can be asserted by another interrupter even when BIR = 0.
11	VAM	VMEbus Arbitration Mode Selects the VMEbus Arbitration Mode, either Round Robin or Priority, used by the VMEbus arbiter. 1 Selects Round Robin bus arbitration. 0 Selects Priority bus arbitration. RBN is cleared by system reset and not affected by local reset.

(continued)

Bit	Mnemonic	Function
10, 9	VRL	VMEbus Request/Grant Level Determines the request and grant level that the VIO board will use to access the VMEbus. VMEbus requests are discussed in detail in *The VMEbus Specification*. VRL Level VMEbus Signals 010 0 BR0*, BG0IN*, BG0OUT* 110 1 BR1*, BG1IN*, BG1OUT* 210 2 BR2*, BG2IN*, BG2OUT* 310 3 BR3*, BG3IN*, BG3OUT* BR*n** Bus Request lines. BG*n*IN* Bus Grant lines (input to boards). Form daisy chain with BGnOUT* lines. BG*n*OUT* Bus Grant lines (output from boards). Form daisy chain with BG*n*IN* lines. VRL is set to level 3 by system reset, and is not affected by local reset. CAUTION: Do not change VRL while a VMEbus request is pending.
8	RNV	Release Never Determines whether or not the CPU releases access to the VMEbus after the CPU has acquired the bus. 1 Once the VIO board has acquired the VMEbus, it will not release control of the bus. To release control of the VMEbus, clear the RNV bit. 0 The VIO board will release control of the VMEbus as defined by the FAIR and VRM bits (see the descriptions that follow). Cleared by system reset and not affected by local reset.
7	FAIR	Fairness Arbitration FAIR defines whether or not the VIO board uses Fairness bus arbitration when accessing the VMEbus. If operating in Fairness mode, the VIO board will not request access to the VMEbus if another request is asserted; the CPU waits until no other VMEbus request is pending. 1 The VIO board uses Fairness arbitration. 0 The VIO board requests access to the VMEbus as needed. Cleared by system reset and not affected by local reset.
6	VRM	VMEbus Release Mode Defines when the VIO board releases control of the VMEbus. 1 The VIO board maintains control of the VMEbus only for the time it takes to complete the transaction. This is Release−When−Done (RWD) mode. 0 The VIO board releases control of the VMEbus when another VME controller requests access to the VMEbus, or when the transaction is completed. This is Release−On−Request (ROR) mode. Cleared by system reset and not affected by local reset.
5	ETO	Enable VMEbus Arbitration Timeout The bus arbiter contains a timer that measures the time between the issuance of a Bus Grant by the bus arbiter and the assertion of a Bus Busy signal (BBSY*) by the responding controller. The ETO bit determines whether or not the VMEbus arbiter will remove the Bus Grant if the controller does not assert Bus Busy (BBSY*) within one second. 1 Enables the one−second VMEbus arbitration timeout. If Bus Busy (BBSY*) is not asserted within one second of the issuance of a Bus Grant, the VME arbiter will remove the Bus Grant. 0 Disables the one−second VMEbus arbitration timeout. Cleared either by system reset or local reset. NOTE: An Arbiter Timeout (ARBTO) interrupt is generated when a bus grant is removed as the result of a bus timeout.

(continued)

Bit	Mnemonic	Function
4, 3	VTS[1, 0]	VMEbus Data Transfer Timeout Select VTS determines the length of time that the Data Strobes (DS0* and DS1*) must be asserted on the VMEbus before the VIO board asserts a Bus Error (BERR*). If the VIO board Location (SBL) bit in the GLOBAL1 register is set to 1, the timer is disabled. VTS Length of Data Strobes 010 32 us 110 64 us 210 128 us 310 Disabled (infinity) VTS is set by system reset and not affected by local reset.
2	EWD	Enable Watchdog Timer The watchdog timer checks for responses to CIO interrupt requests. If the CPU does not respond to a CIO interrupt request within one second, the watchdog timer is set. This results in an action as determined by the WDA[1, 0] bits of this register. (Information on the WDA bits follows.) 1 Enables reactions to watchdog timeouts. When the watchdog timer is enabled and set, an action defined by the WDA bits is performed. 0 Disables reactions to watchdog timeouts. The watchdog timer can be used for other timing functions. EWD is cleared by either system reset or local reset.
1, 0	WDA[1, 0]	Watchdog Action WDA determines what happens when the watchdog timer is set to 1. **Value in WDA** **Action**0 10 The system performs a system reset. 110 The system performs a local reset. 210 The system performs a local reset and hold. 310 The watchdog timer is disabled. WDA is set by system reset and is not affected by local reset.

(concluded)

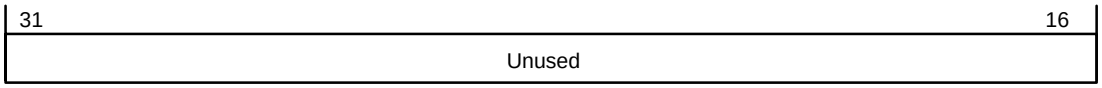
WHOAMI Who Am I

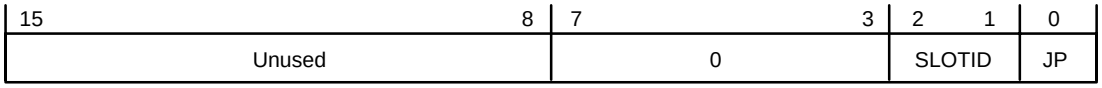
FFC8 7000 Read only

Each Who Am I (WHOAMI) register identifies the position of it's CPU board within the system, as well as which job processor is currently the Mbus master.

Each CPU board has a WHOAMI register.

Resets do not affect WHOAMI.





Bit	Mnemonic	Function
31–8	Unused	
7–3	0	
2, 1	SLOTID	Slot ID Defines which CPU the processor is on. 00 CPU board 0, slot 6 01 CPU board 1, slot 5 10 CPU board 2, slot 4 11 CPU board 3, slot 3
0	JP	Identifies which job processor (CPU) is currently the Mbus master. 0 JP0 1 JP1

Global Control and Status (GCS) Registers

This section describes the following Global Control and Status (GCS) registers:

GLOBALn	Global registers 1 and 2
BRDID	Board ID register
GPCSn	General-Purpose Control and Status registers 0–4

The GCS registers are accessed by VME controllers as well as by CPUs. The CPUs and VME controllers pass information via the GCS registers as follows:

- The BRDID register contains the VIO board identification number.
- The GLOBAL0 register passes the location monitors.
- The GLOBAL1 register handles system fail (SF*) interrupts to the VMEbus, handles local resets, passes SHP and SLP interrupts to the CPUs, and passes the VIO board location to VME controllers.
- The GPCS*n* registers pass user–defined data.

To access the GCS registers, a VME controller writes a 16–bit address of the GCS register and defines the address as an A16 address via the address modifiers AM[5–0]. The three most–significant nibbles of the address must be the same as the value defined by the BASAD register, which is loaded by the Group Address (GRPAD) and Board Address (BDAD) switches on the VIO board. When the VME controller defines and writes an A16 address, the VME interface logic compares the address with the value stored in the Base Address (BASAD) register. If the two addresses are the same, the address is placed on the Mbus. Bits A[3–0] of the address select one of the eight GCS registers.

Unlike the other system registers, the GCS registers are not on word (4–byte) boundaries. If a VME controller attempts to write or read a word, a bus error is returned. If a CPU accesses a word, the GCS logic returns two copies of the register's contents.

CAUTION: When addressing a GCS register from the CPU, do not use the Exchange Memory with Register (XMEM) instruction except via A16 (short I/O) space. This ensures that no other VME masters access the GCS registers between the XMEM load and store operations.

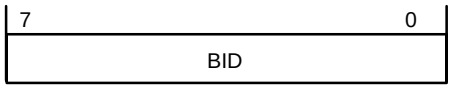
The following pages define the Global Control and Status registers.

BRDID Board ID

FFF8 6005 Read/Write (as noted)

Board ID (BRDID) contains the VIO board identification number. The VMEbus uses BRDID to identify the VIO board.

System resets clear BRDID, local resets do not affect BRDID.

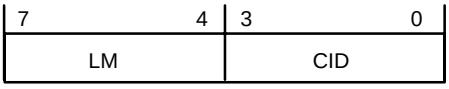


Bit	Mnemonic	Function
7–0	BID	Board Identification VMEbus Read only, Mbus Read/Write During powerup, the CPU writes the VIO board identification into BID. The VMEbus can then read BID to get the VIO board ID.

GLOBAL0 Global Register 0

FFF8 6001 Read/Write (as noted)

Global Register 0 (GLOBAL0) defines the state of the location monitors and the VMEbus interface chip identification number.



Bit	Mnemonic	Function
7–4	$\overline{\text{LM}}$[3–0]	VMEbus Location Monitors Read/Write 1. Defines whether or not the location monitor detected an access. 1 A location monitor access was not detected. 0 A location monitor access was detected. Set to 1 by system reset and not affected by local reset.
3–0	CID	VMEbus/Mbus Chip Identification Number Read only. The CID bits are set to 1111, indicating that the Global Control and Status (GCS) registers use the Motorola Reference Standard. Not affected by either system reset or local reset.

GLOBAL1 Global Register 1

FFF8 6003 Read/Write (as noted)

The GLOBAL1 register sets the Mbus reset, control, and status conditions and informs the CPU of the status of these conditions.

7	6	5	4	3	2	1	0
R&H	SBL	ISF	BDF	0	0	SHP	SLP

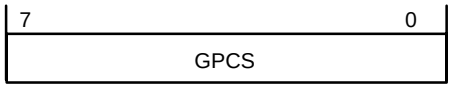
Bit	Mnemonic	Function
7	R&H	Local Reset−and−Hold Read/Write both VMEbus and Mbus. Used to reset the CPU and CMMUs. 1 Resets the CPU and CMMUs. 0 Does not reset the CPU and CMMUs. Cleared by system reset and not affected by local reset.
6	SBL	VIO board Location Read only both VMEbus and Mbus. SBL is hardwired to switch #5 (SCON) of DIP switch S2 (BDAD). When read, SBL must be a 1, indicating that the VIO board is in VME slot 0 and supplying SYSCLK and BCLR to the VMEbus. If SBL is 0, check the SCON switch. 1 Indicates that the SCON switch is on; the VIO board is in VME slot 0 and supplying SYSCLK and BCLR to the VMEbus.. 0 Indicates that the SCON switch is off; the VIO board is not in slot 0 and is not supplying SYSCLK and BCLR to the VMEbus.. Not affected by resets.
5	ISF	Inhibit the SYSFAIL* output to the VMEbus Read/Write both VMEbus and Mbus. Determines whether or not the VIO board will pass a system reset on to the VMEbus (SYSRESET* line). 1 The VIO board will not assert SYSFAIL* on the VMEbus. 0 Assert SYSFAIL* if ASF* of the UCS register is asserted or if the watchdog timer has expired. Cleared by system reset and not affected by local reset.
4	BDF	Board Fail Status Read only both VMEbus and Mbus. Indicates whether the VIO board is initiating a system failure (i.e., it indicates the state of the ASF* bit of the UCS register and the output state of the watchdog timer.) 1 Assert System Failure (ASF*) of the UCS register is asserted (0), or the watchdog timer is set (the timer has expired). 0 ASF* is not asserted, and the watchdog timer output is cleared. Not affected by resets; BDF always reflects the state of ASF* and the watchdog timer output.
3−0	Reserved	Always 0

GPCSn General–Purpose Control and Status

FFF8 6007	**GPCS0**	**Read/Write**
FFF8 6009	**GPCS1**	
FFF8 600B	**GPCS2**	
FFF8 600D	**GPCS3**	
FFF8 600F	**GPCS4**	

The General–Purpose Control and Status (GPCS*n*) registers allow programmer–defined control and status information to be passed from one VME controller to another over the VMEbus. If a VME controller writes control and status into a GPCS*n* register, other controllers can read the GPCS*n* register and use the information. The programmer defines the functions of the bits in these registers.

NOTE: For good programming structure, the GPCS*n* registers should be used only to pass control and status information between VME controllers (including the VIO board).



Bit	Mnemonic	Function
7–0	GPCS	General Purpose Control and Status The programmer defines the function of the GPCS[7-0] bits. A system reset sets GPCS0 to FF. A system reset clears GPCS1 through GPCS4. Local resets do not affect the GPCS*n* registers.

End of Chapter

Chapter 5
Memory

This chapter describes the types of memory available on the CPU board, VIO board and memory boards. All RAM is located on memory boards that contain Error Checking and Correction (ECC) logic. The VIO board has Programmable Read–Only Memory (PROM) which contains boot code. The system configuration parameters are stored in non–volatile RAM (NOVRAM) as described in Chapter 7, “Programming the Counter/Timer and the Real-Time Clock”. Access to memory is regulated by memory maps. CPUs can access all of memory, but VME controllers can access only resources allowed by the VME address map.

Interleaved Memory

Memory is interleaved among the two or four available memory boards. This section describes how memory is interleaved.

Logical addresses on the Mbus bound for system memory are decoded and permuted before being written to the X0bus or the X1bus. Bit 4 of the Mbus address points to either the X0bus (bit 4=0) or the X1bus (bit 4=1). The LMAD decodes the upper ten address bits, and selects either memory board 0 or memory board 1 on the selected bus (X0 or X1). Bit 24 of the Mbus address is written into bit 4, and bits 25–29 are shifted one bit position into bits 24–28. This address is written to the selected bus (X0 or X1) and into the selected memory board on that bus (mem 0 or mem 1). The example below, and Figure 5–1 illustrate interleaving.

EXAMPLE: Assuming you have four 64-Mbyte memory boards, the addresses decode as follows:

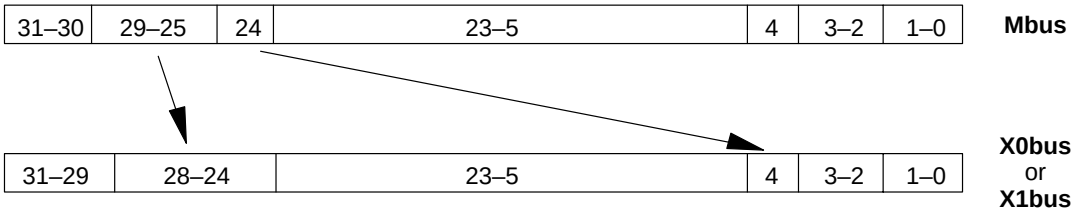
```
0000 0001 0000 0000 0000 0000 0000 0000        Mbus
0000 0000 0000 0000 0000 0000 0001 0000        X0bus
Decodes to X0bus, memory board 0

0000 1000 0000 0000 0000 0000 0000 0000        Mbus
0000 0100 0000 0000 0000 0000 0000 0000        X0bus
Decodes to X0bus, memory board 1

0000 0000 0000 0000 0000 0000 0001 0000        Mbus
0000 0000 0000 0000 0000 0000 0000 0000        X1bus
Decodes to X1bus, memory board 0

0000 1001 0000 0000 0000 0000 0001 0000        Mbus
0000 0100 0000 0000 0000 0000 0001 0000        X1bus
Decodes to X1bus, memory board 1
```

Bit 26 of the Mbus selects mem 0 or mem 1 when have 64MB boards
Bit 27 of the Mbus selects mem 0 or mem 1 when have 128MB boards
Bits 26 and 27 combined select mem 0 or mem 1 when have 192MB boards



Mbus:
Bit 4 selects Xbus (cache block) (0=X0bus, 1=X1bus)
Bits 31–20 are always 0 for system memory.

X0/X1bus:
Bits 23–0 select the address within a bank
Bits 28–24 select the board and wedge
Bits 31–29 are always 0 for system memory.

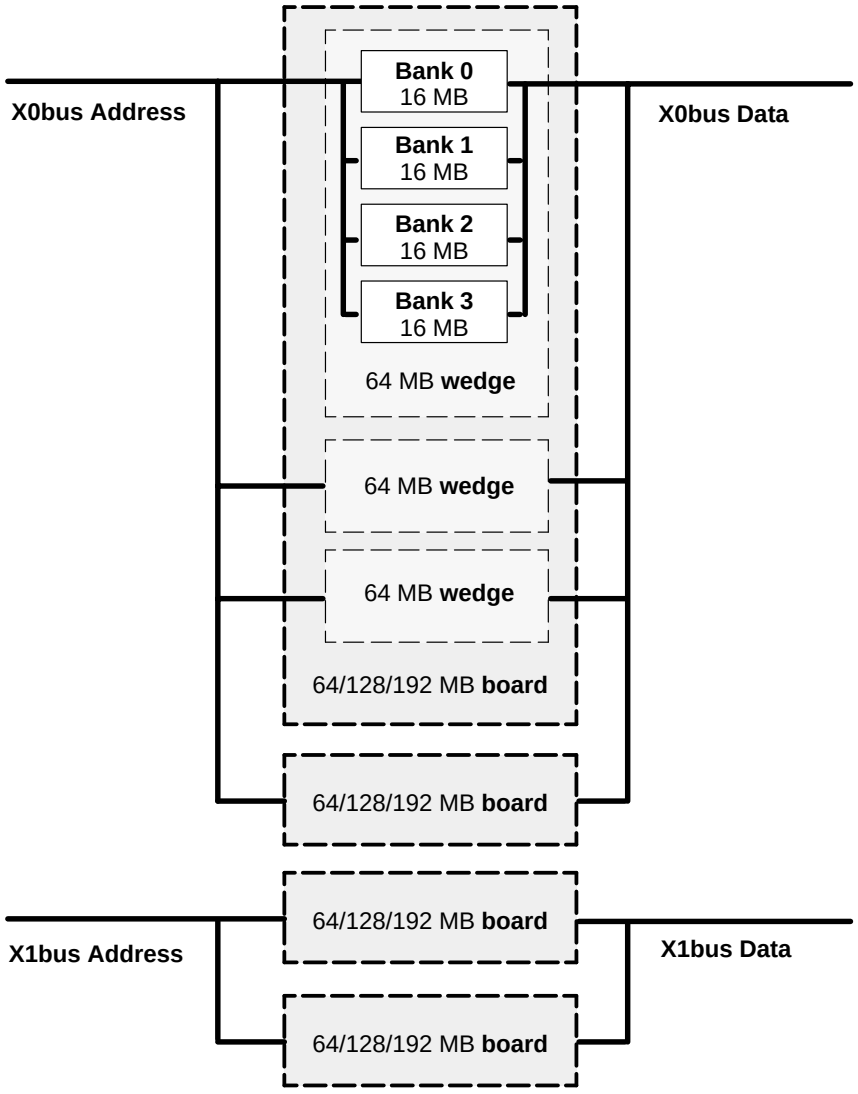


Figure 5–1 Interleaved Memory and Memory Addressing

Reading from Memory

When a device reads data from memory, memory sources the data at a rate of one word per bus cycle until it receives an EOR (End of Request) from the reading device or until the read crosses to a new block of data. If the block crossing occurs first, the memory module asserts End of Data (EOD). Block crossings are described in the "Data Blocks and Block Crossing" section earlier in this chapter.

When data is read, memory examines the data for single–bit and multiple–bit errors. When a multiple–bit error occurs, a bus error is asserted and an exception routine must be run. When a single–bit error occurs, the error correction logic asserts two wait states within which the logic corrects the data and writes the corrected data to the data bus. The error logic also notifies the interrupt logic that a single–bit error has occurred. An interrupt service routine can then update the error log. Single–bit errors are correctable and do not affect address flow. Memory boards do not correct data when written to; they only correct data when read from. The "Error Checking and Correction (ECC)" section later in this chapter describes the error checking process in greater detail.

When a CPU or a VME controller reads a block (4 words) of data from memory, the following occurs:

1. The CPU or VME controller writes the beginning address to memory.
2. The memory module receives the address.
3. If the memory module cannot respond to the address within one clock cycle, the memory module inserts wait states. The data transfer continues when the wait state is removed.
4. The CPU or VME controller reads a word of data from memory.
5. The memory module automatically increments the address to the next word of data, then writes that word of data to the Mbus. This process continues until the address reaches the end of the data block.

Writing to Memory

Memory receives data (in bytes, half–words or words) until it receives an End of Request (EOR) or until a block crossing occurs.

The master writes a word on the data bus and the addressed memory receives the data. The memory module generates ECC bits and writes them to memory with the data. If the data transfer is a byte or half–word, the CPU performs a read–modify–write operation to generate the ECC bits. Read–modify–write operations extend the write cycle.

When a CPU or VME controller writes a block of data to memory, the following occurs:

1. The CPU or VME controller writes the beginning address to memory.
2. The memory module receives the address.

3. If the memory module cannot respond to the address within one clock cycle, it inserts wait states. The data transfer continues when the wait state is removed.
4. The CPU writes a word of data to memory, and the memory module strobes the data into memory.
5. The memory module automatically increments the address to the next word, then strobes that word of data into the memory. This process continues until the address reaches the end of the data block.

Data Blocks and Block Crossing

Data is transmitted as single words, asserting a new address after each word, or as blocks with an address asserted at the beginning of the block. This section describes two data blocks: CMMU and VME data blocks.

CMMU Block

A block (cache line) of data, as defined by the CMMUs, is four 32-bit words of data that when read from or written to memory, are aligned on an even address whose least–significant nibble begins at address 0_{16}. Figure 5–2 illustrates a block of data as defined by the CMMUs.

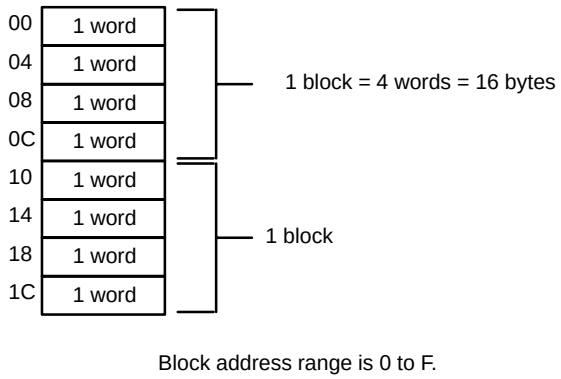


Figure 5–2 CMMU Data Blocks

When a CPU reads a block of data from memory or writes a block of data to memory using an aligned read or write as just described, the read or write will complete within 10 bus cycles. If the read or write crosses from one block to another (called a block crossing), the transfer will require an additional bus cycle.

VME Block

A block of data, as defined by VME controllers, is the largest entity of data that the controller's buffer can receive within a burst read or a burst write. A burst is a transfer of a block of data without CPU intervention; memory automatically increments the address. Each controller establishes its block size and must communicate this size to the master controller.

Cache Coherency

The CMMUs contain cache memory (called cache) as temporary storage to speed up instruction execution. If the cache is turned off, the CPU interacts directly with system memory (called memory). When the cache is turned on, the CPU reads from and writes to memory through the cache. When the cache is turned on, the CMMUs update memory using either Writethrough or Copyback mode. When the cache is operating in Writethrough mode, all modifications to the cache are immediately written to memory. When the cache is operating in Copyback mode, only the first modification of each cache location is written to memory. When memory is read, accurate data is expected.

Cache coherency is a property of the CMMUs and memory to pass correct data when memory is read. If the cache is turned off, or if the cache is on and in Writethrough mode, memory is always current because the CPUs write changed data directly to memory. If the cache is on and updated in Copyback mode, locations in memory may not be current because the changed data is written only to the respective cache, not to the memory.

If the cache is on and in Copyback mode, the CMMUs snoop the Mbus for accesses to locations that have been modified by a CPU. If a modified location is read, the appropriate CMMU updates the location before the read is completed. Figure 5–3 illustrates cache coherency. For a detailed description of cache coherency and Mbus snooping, see either the *MC88200 Cache/Memory Management Unit User's Manual*, or the *MC88204 Cache/Memory Management Unit User's Manual.*

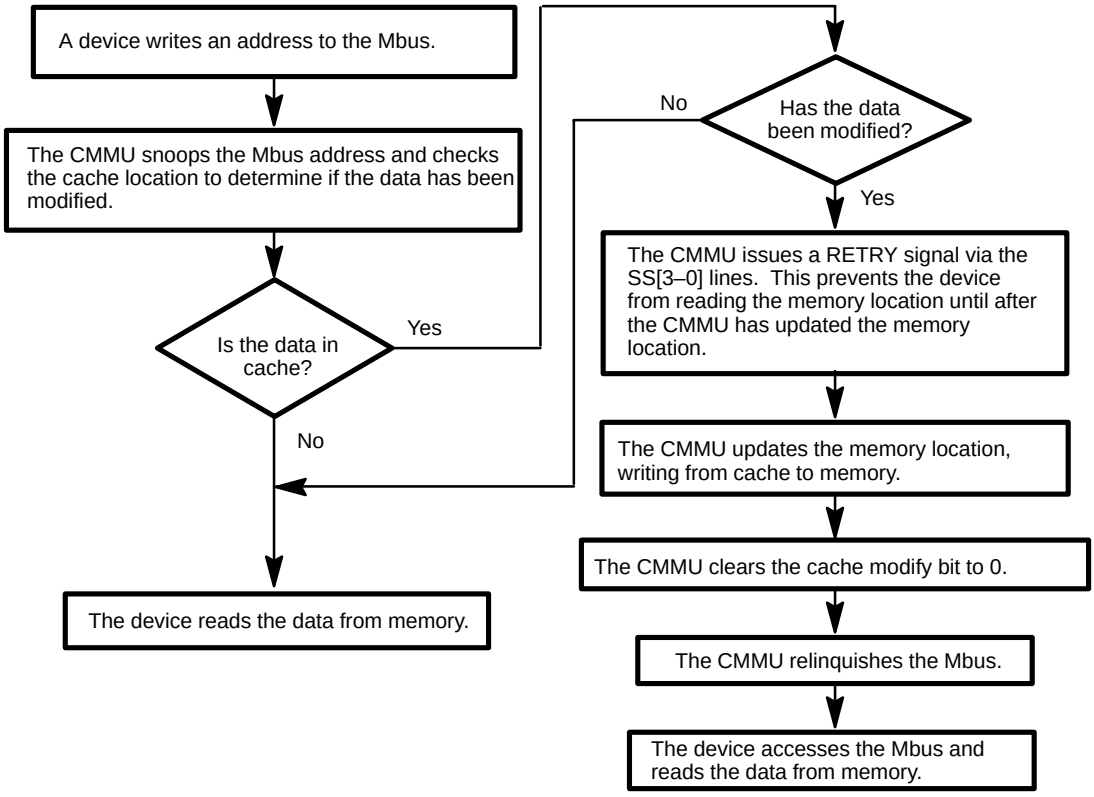


Figure 5–3 Cache Coherency

The System Status Lines

The system status lines (SS[3–0]) on the Mbus pass bus information to or from the CPU. This information includes:

Error Error indicates that a bus error occurred. This means that an invalid address was accessed, a multiple−bit ECC error occurred, or a VMEbus time−out occurred. When an error occurs, the current memory operation is aborted.

Retry The CMMUs generate the Retry signal to indicate that a memory location that is being read has stale data. Retry is used when cache coherency is enabled and the CMMUs contain data that has been modified but not written to memory. When Retry is asserted, the reading device waits for the CMMU to update the memory location before reading the data.

Wait Memory devices assert a Wait when they can not read or write data within one clock cycle; Wait extends the data phase. If memory cannot respond to an address within one clock cycle, the memory module drives the SS1 line Low to insert wait cycles until it is ready to receive or send data.

End of Data Memory asserts an End of Data when a data transfer has completed successfully.

OK OK indicates that the previous bus cycle completed successfully and the computer system is ready to process a new bus cycle.

As you can see from Table 5–1, each status line defines a system condition.

Table 5–1 System Status Lines

SS3	SS2	SS1	SS0	Function
0	X	X	X	Error
1	0	X	X	Retry
1	1	0	X	Wait
1	1	1	0	End of Data (data phase) or OK (address phase)
1	1	1	1	OK (phase completed)

SS3 indicates an Error when Low;
SS2 indicates a Retry when Low;
SS1 indicates a Wait when Low; and
SS0 indicates an End of Data when Low.

An address phase is the period during which an address is written to and stabilized on the bus, and successfully interpreted by the addressed device. A data phase is the time during which data is successfully transferred over the bus.

When all of the system status lines are High, the computer system is OK and ready to start a new bus cycle.

Error Checking and Correction (ECC)

The memory boards each examine data read from RAM for errors, and correct single–bit erors, but not multiple–bit errors. The process of finding and correcting memory errors is called Error Checking and Correction (ECC). When a single–bit error occurs, the interrupt logic interrupts a CPU, which logs the error and runs an interrupt service routine. Multiple–bit errors cannot be corrected.

Each memory board has on–board memory refresh; memory is refreshed every 15.6 ms. If an Mbus transaction is taking place, a memory refresh will occur after the Mbus transaction is completed. While memory is being refreshed, the WAIT status is pending.

All ECC errors are reported to the processors through the IST register's MEM bit. Perform the following steps to clear an ECC error:

1. Shut down all processes that access system memory.
2. Disable the MEM interrupt by clearing the IEN register's MEM bit to 0.
3. Find the error as follows:
   A. Read an ERAx register.
   B. Read the corresponding CONx register.
   C. Check bit 26 (SBE) for a single–bit error. If set, continue with step 4. If cleared, continue with step A for the next register.
4. Build the error offset for the logical memory board as follows:
   A. Shift bits 27–24 left 1 bit.
   B. Copy the value in bit 4 to bit 24.
   C. If the error was found on CON1 or CON3, clear bit 4 to 0.
      If the error was found on CON2 or CON4, set bit 4 to 1.
5. Add the starting address for the logical memory board.
6. Read the location.
7. Execute XMEM to write the value back to memory.
8. Set the GIEN MEM bit to 1 to enable the MEM interrupt.

Registers

This section describes the Control (CON) register and the Error Address (ERA) register. The Control register regulates memory refresh, error detection and correction, memory size, and system diagnostics. The Error Address register contains the address of the first error that occurred since the last time that ERA was read. The following pages describe the CON and ERA registers.

CONx (Resides on memory boards) **Control**

FFF8 9100 **CON1 – X0bus, Board 0** **Read/Write**

FFF8 9110 **CON2 – X0bus, Board 1**

FFF8 9300 **CON3 – X1bus, Board 0**

FFF8 9310 **CON4 – X1bus, Board 1**

The Control (CON) register maintains memory status and control.

31	30	29 27	26	25	24	23 22	21	20 19	18	17	16
CPUCLK	WAITS	Reserved	SBE	Reserved	MBE	LED	DLE	MOD	ERE	COE	DMR

15 13	12 7	6 0
MEM	Reserved	ECB

Bit	Mnemonic	Function
31	CPUCLK	CPU Clock Speed CPUCLK is set to 0 during powerup; do *not* set CPUCLK to 1. 1 20 MHz 0 25 MHz
30	WAITS	Number of Wait States WAITS is set to 0 during powerup; do not set WAITS to 1. 1 3 wait states (Use this setting with 20 MHz CPUCLK) 0 4 wait states (Use this setting with 25 MHz CPUCLK)
29–27	Reserved	
26	SBE	Single Bit Error 1 Single bit error occurred. 0 No error.
25	Reserved	
24	MBE	Multiple Bit Error 1 Multiple bit error occurred. 0 No error.
23, 22	LED	Diagnostic LED Control LED controls the operation of the diagnostic LEDs located on the system board. The RED LED pulses for 200 ms when a correctable error is detected, and remains lit when an uncorrectable error is detected. The YELLOW LED lights when an explicit read/write access is decoded, unless it is overridden by Y=1 or Y=0. The GREEN LED lights with every refresh cycle. Bits Mode 00 Run – the LEDs operate as described above. 01 R=0 – resets the red LED (clears red to 0 (off)) 10 Y=0 – the yellow LED remains off (unlit) 11 Y=1 – the yellow LED remains on (lit)
21	DLE	Diagnostic Latch Enable Enables the latching of ECC bits into the ECC check bits (ECB, bits 6–0) of the CONx register. 1 Enable the latching of ECC bits into the ECB bits. 0 Disable the latching of ECC bits into the ECB bits.

(continued)

Bit	Mnemonic	Function
20, 19	MOD	Diagnostic mode select **Bits** **Mode** 00 Normal operation. 01 Diagnostic generate. 10 Diagnostic detect
18	ERE	Error Enable Enables or disables the reporting of single-bit and multiple-bit errors. 1 Enable ECC error reporting. 0 Disable ECC error reporting. *CAUTION: Never clear this bit except to run a diagnostic memory routine.*
17	COE	Correct Enable 1 Enable ECC error correction. 0 Disable ECC error correction. *CAUTION: Never clear this bit except to run a diagnostic memory routine.*
16	DMR	Disable Memory Refresh 1 Disable DRAM refresh. 0 Enable DRAM refresh. *CAUTION: Never set this bit except to run a diagnostic memory routine.*
15–13	MEM	Memory Size When read, indicates how much memory is on that memory board. To verify the total system memory, read the MEM bits on all of the available CONx registers. **Bits** **Memory Size** 011 64 MBytes 010 128 MBytes 000 192 MBytes
12–7	Reserved	
6–0	ECB	ECC Check Bits

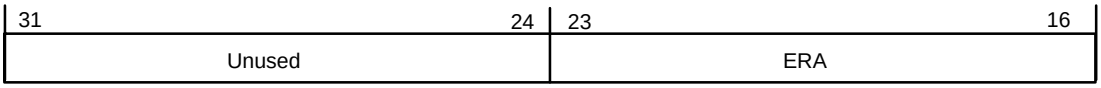
(concluded)

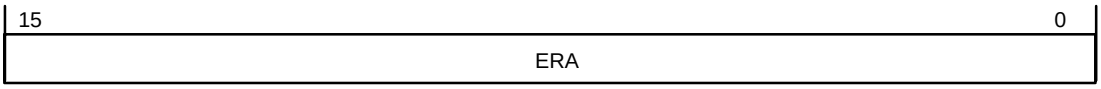
ERAx (Resides on memory boards) **Error Address**

FFF8 9104	**ERA1 – X0bus, Board 0**	**Read only**
FFF8 9114	**ERA2 – X0bus, Board 1**	
FFF8 9304	**ERA3 – X1bus, Board 0**	
FFF8 9314	**ERA4 – X1bus, Board 1**	

The Error Address (ERA) register contains the address of the first error that occurred since the last time that a CPU read the CONx register.

NOTE: ERAx will not store a new error address until the existing address has been read. If more than one error occurs between reads, only the first error address is stored; subsequent error addresses are not recorded.





Bit	Mnemonic	Function
31–24	Unused	
23–0	ERA	Error address Contains bits [25–2] of the address of a single–bit or multiple–bit error. This error is the first error that occurred since the last time that a CPU read CONx. Intervening error addresses are not latched into ERA.

End of Chapter

Chapter 6
Programming the Serial and Parallel Interfaces

This chapter describes the serial and parallel interfaces, their respective ports, and how to program the interfaces.

The Serial and Parallel Interfaces

The system board has two RS−232−C serial asynchronous ports, and one Centronics−compatible parallel port. One serial port is a terminal port for the system console, while the other has modem lines and can accept a modem, terminal or printer. The serial ports can transmit and receive data, while the parallel port is limited to transmitting data only. Figure 6−1 illustrates the serial and parallel interfaces.

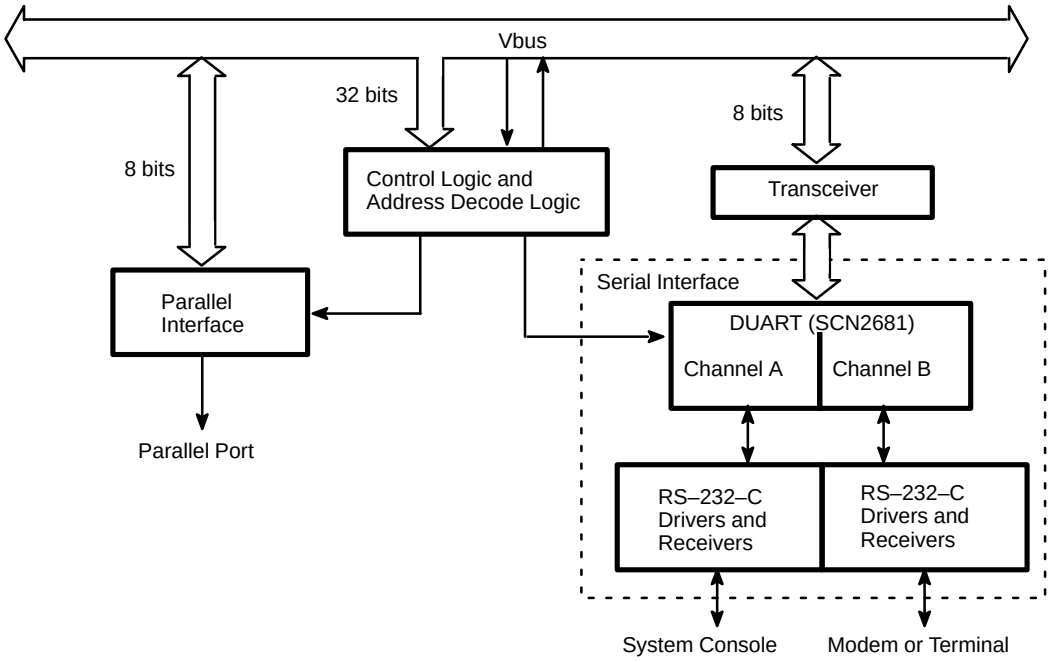


Figure 6−1 The Serial and Parallel Interfaces

Serial Interface

The serial interface consists of a Signetics SCN2681 Dual Asynchronous Receiver/Transmitter (DUART). The DUART has two channels, channel A for a system console and channel B for a modem or terminal. For descriptions of the signals, see Appendix C. The DUART clock input is 3.6864 Mhz.

Parallel Interface

The parallel interface transmits data to a parallel slave device using the Centronics protocol. The parallel interface has a 512−Kbyte buffer, therefore you can transmit as many as 512 characters at a time. When the 512−character printer buffer fills, the interface clears the Printer Buffer Empty (PBE) bit in the Interrupt Status (IST) register. When the printer buffer empties, the interface sets the PBE bit, and the CPU can send more data. The parallel port pin assignments are defined in Appendix C.

Programming the Serial Interface

The serial I/O consists of two I/O channels within one DUART. These channels must be configured via the DUART registers before they receive or send data. To program the DUART, write to its command registers. To get DUART status information, read the DUART's status registers.

Changing the contents of a register while the DUART is transmitting or receiving data may cause errors. For example, changing the number of bits per character while transmitting data may cause the transmission of an incorrect character. Do not write to the Mode registers, the Clock Select registers, or the Output Port Configuration register while the receivers and transmitters are enabled. Also, do not write to the Auxiliary Control register while the counter/timer is enabled (running).

CAUTION: Software must not issue sequential reads and/or writes to the DUART registers because of the slow setup timing from read/write inactive to read/write active specified by Signetics. You should put two nonoperative instructions between register read and write instructions.

The rest of this section describes the following tasks:

- Programming the DUART registers.
- Initializing the serial interface.
- Resetting the serial interface.

Registers

The serial interface registers are accessed as words and are aligned on word boundaries. Each channel has its own Mode, Command, Clock Select, and Status registers.

All registers are mapped to addresses FFF8 2000_{16} through FFF8 $203C_{16}$ in the system's address space. Table 6−1 lists the registers and their addresses.

Table 6–1 Serial Interface Registers

Address	Register (A is the serial port; B is the mouse port)	Type of Register
FFF8 2000	Mode Registers A (MR1A, MR2A)	read/write
FFF8 2004	Status Register A (SRA) Clock Select Register A (CSRA)	read write
FFF8 2008	Command Register A (CRA)	write [1]
FFF8 200C	Receive Holding Register A (RHRA) Transmit Holding Register A (THRA)	read write
FFF8 2010	Input Port Change Register (IPCR) Auxiliary Control Register (ACR)	read write
FFF8 2014	Interrupt Status Register (ISR) Interrupt Mask Register (IMR)	read write
FFF8 2018	Counter/Timer Upper Register (CTU, CTUR)	read/write
FFF8 201C	Counter/Timer Lower Register (CTL, CTLR)	read/write
FFF8 2020	Mode Registers B (MR1B, MR2B)	read/write
FFF8 2024	Status Register B (SRB) Clock Select Register B (CSRB)	read write
FFF8 2028	Command Register B (CRB)	write !
FFF8 202C	Receive Holding Register B (RHRB) Transmit Holding Register B (THRB)	read write
FFF8 2030	Reserved	
FFF8 2034	Input Port Output Port Configuration Register (OPCR)	read write
FFF8 2038	Start Counter Command Set Output Port Bits Command	unused unused
FFF8 203C	Stop Counter Command Reset Output Port Bits Command	unused unused

[1] Do not read CRA or CRB. Reading these registers may hang the asynchronous line. (If you use this line for the system console, the system may hang and require a reset.)

Initializing the Serial Interface

No special initialization process is necessary.

Resetting the Serial Interface

The serial interface is reset either during powerup or via the DUART Reset (DRE bit 28) bit of the Control (CON) register as described in Chapter 5, “Memory”. Parts of the DUART can be reset via the Command registers (CRA and CRB) as described in the following register descriptions.

Serial Interrupts

The DUART may generate several types of interrupts, but it has only one interrupt line which sets the DUART Interrupt (DI) bit in the Interrupt Status (IST) register. When the CPU acknowledges the interrupt, it reads the DUART’s Interrupt Status Register (ISR) and handles the interrupt as needed. Any of the following conditions can generate an interrupt.

- Channel A or B transmitter is ready.
- Channel A or B receiver is ready, or its FIFO buffer is full.
- Channel A or B detected a Break Character.
- The input port changed state.

The programmer can mask serial interrupts via the DUART’s Interrupt Mask Register (IMR). When an interrupt is masked, the DUART will not set the bit in the ISR, and consequently will not assert the INT line to the IST register for that interrupt.

ACR Auxiliary Control

FFF8 2010 Write Only

The Auxiliary Control register (ACR) selects the baud rate used and the Counter/Timer mode, as well as enabling or disabling input port change interrupts for input ports 0−3.

7	6 4	3	2	1	0
BRG	CTM	EIP3CI	EIP2CI	EIP1CI	EIP0CI

Bit	Mnemonic	Function
7	MIE	Master Interrupt Enable Enables or disables all CIO interrupt requests by disabling the INT* line. 1 CIO interrupt requests are enabled or disabled by their respective interrupt enable bits. 0 CIO interrupt requests are disabled. MIE overrides all individual interrupt enable bits.
6−4	CTM	Counter/Timer Mode Selects a mode and clock source for the counter/timer.

CTM[2]	CTM[1]	CTM[0]	Mode	Clock Source
0	0	0	Counter	External (IP2)
0	0	1	Counter	TxCA (1x, channel A)
0	1	0	Counter	TxCB (1x, channel B)
0	1	1	Counter	Crystal or external clock
1	0	0	Timer	External (IP2)
1	0	1	Timer	External (IP2) div. by 16
1	1	0	Timer	Crystal or external clock
1	1	1	Timer	Crystal or external clock divided by 16

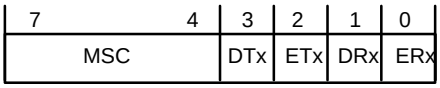
3	EIP3CI	Enable Input Port 3 Change Interrupt 0 Changes in state of IP3 does not affect the input port change (IPC) bit of the interrupt status register (ISR). 1 Changes in state of IP3 will set the IPC bit in ISR.
2	EIP2CI	Enable Input Port 2 Change Interrupt 0 Changes in state of IP2 does not affect the input port change (IPC) bit of the interrupt status register (ISR). 1 Changes in state of IP2 will set the IPC bit in ISR.
1	EIP1CI	Enable Input Port 1 Change Interrupt 0 Changes in state of IP1 does not affect the input port change (IPC) bit of the interrupt status register (ISR). 1 Changes in state of IP1 will set the IPC bit in ISR.
0	EIP0CI	Enable Input Port 0 Change Interrupt 0 Changes in state of IP0 does not affect the input port change (IPC) bit of the interrupt status register (ISR). 1 Changes in state of IP0 will set the IPC bit in ISR.

CRA, CRB — Command

FFF8 2008 — CRA — Write Only
FFF8 2028 — CRB

The Command registers (CRA and CRB) set up parameters for channels A and B.

CAUTION: Do not read CRA and CRB. Reading them may hang the line. If the system console is on this line, the system may hang and require a reset.



Bit	Name	Function
7–4	MSC	Miscellaneous Commands

Bits	Function
0000	No command.
0001	Resets the mode register pointer to point to MR1.
0010	Resets the channel's receiver. Disables the receiver and flushes the FIFO.
0011	Resets the channel's transmitter.
0100	Clears the channel's received break, parity error, framing error, and overrun error bits in the channel's Status Register by clearing the bits to 0.
0101	Reset channel break change interrupt. Clears the channel's break detect change bit in the Interrupt Status Register (ISR) by clearing the bit to 0.
0110	Start break. Forces the channel's TxD output low (spacing). If the transmitter is empty, the start of the break condition is delayed up to two bit times. If the transmitter is active, the break begins when character transmission is completed. If a character is in the channel's Transmit Holding Register (THR), the start of the break is delayed until that character, or any others loaded subsequently, are transmitted. To accept this command, the transmitter must be enabled.
0111	Stop break. The channel's TxD line goes high within two bit times, and remains high for one bit time before the next character is transmitted.
1000	Asserts the RTS (request to send) signal (at the DB25 connector) high.
1001	Clears the RTS signal low.
1010	Not used.
1011	Not used.
1100	Not used.
1101	Not used.
1110	Stops the DUART oscillator, suspending all functions requiring this clock. The contents of all registers are saved. Software should disable the transmitter and receiver before issuing this command. This command is in Command Register A (CRA) only.
1111	Resets the Power Down mode to start DUART oscillator running again. This command is in Command Register A (CRA) only.

(continued)

Bit	Mnemonic	Function
3	DTx	Disable Transmitter 0 No action, transmitter state not changed. 1 Disables the transmitter.
2	ETx	Enable Transmitter 0 No action, transmitter state not changed. 1 Enables the transmitter.
1	DRx	Disable Receiver 0 No action, receiver state not changed. 1 Disables the receiver.
0	ERx	Enable Receiver 0 No action, receiver state not changed. 1 Enables the receiver.

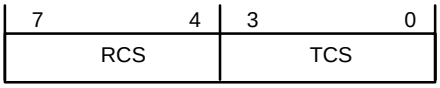
(concluded)

CSRA, CSRB — Clock Select

FFF8 2004 **CSRA** **Read/Write**

FFF8 2024 **CSRB**

The Clock Select registers (CSRA, CSRB) contain clock receive and transmit data.



Bit	Mnemonic	Function
7–4	RCS	Receiver Clock Select Programs the receiver clock for channel A (CSRA) or channel B (CSRB).
3–0	TCS	Transmitter Clock Select Programs the transmitter clock for channel A (CSRA) or channel B (CSRB).

You can select two sets of baud rates using bit 7 of the Auxiliary Control register in conjunction with the Clock Select registers. By programming these registers, you select the clock source of the baud rate generator (BRG) and the actual baud rate generated. Table 6–2 lists the possible baud rates.

Table 6–2 Baud Rate Generator Characteristics (crystal or clock = 3.6864 MHz)

CSR Bits [7:4] or [3:0]	Baud Rate BRG = 0	BRG = 1	16x Clock (KHz)	Error (%)
0000	50	75	0.8 (1.2)	0 (0)
0001	110	110	1.759	−0.069
0010	134.5	134.5	2.153	0.059
0011	200	150	3.2 (2.4)	0 (0)
0100	300	300	4.8	0
0101	600	600	9.6	0
0110	1,200	1,200	19.2	0
0111	1,050	2,000	16.756 (32.056)	−0.26 (0.175)
1000	2,400	2,400	38.4	0
1001	4,800	4,800	76.8	0
1010	7,200	1,800	115.2 (28.8)	0 (0)
1011	9,600	9,600	153.6	0
1100	38,400	19,200	614.4 (307.2)	0 (0)
1101	Timer	Timer		
1110	IPx=16x	IPx=16x		
1111	IPx=1x	IPx=1x		

NOTE: BRG is the baud rate generator select located in the ACR register (bit 7). This selects one of the two sets shown above.

CTUR, CTLR — Counter/Timer

FFF8 2018 **CTUR** **Read/Write**

FFF8 201C **CTLR**

The Counter/Timer Upper Register (CTUR) and Counter/Timer Lower Register (CTLR) together form a 16−bit counter. (These registers are write−only; a read of these addresses is actually a read of the physical counter/timer.)

The CTUR bits are defined as follows:

7	6	5	4	3	2	1	0
C15	C14	C13	C12	C11	C10	C9	C8

Bit	Mnemonic	Function
7−0	C15−C8	Counter/Timer Bits 15−8

The CTLR bits are defined as follows:

7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Bit	Mnemonic	Function
7−0	C7−C0	Counter/Timer Bits 7−0

IMR Interrupt Mask

FFF8 2014 Write Only

The Interrupt Mask Register (IMR) masks and enables interrupts generated by the DUART. When an interrupt is masked via IMR, if the masked interrupt condition occurs, the DUART will not set the corresponding bit in the ISR, and therefore will not assert the interrupt line to the system board's Interrupt Status (IST) register.

7	6	5	4	3	2	1	0
MIPC	MBKB	MSOFB	MTxRB	MCTR	MBKA	MSOFA	MTxRA

Bit	Mnemonic	Function
7	MIPC	Input Port Change Interrupt Mask 0 Mask the input port change interrupt. 1 Enable the input port change interrupt.
6	MBKB	Channel B Break Interrupt Mask 0 Mask the channel B break interrupt. 1 Enable the channel B break interrupt.
5	MSOFB	Channel B State of FIFO Interrupt Mask 0 Mask the channel B state of FIFO interrupt. 1 Enable the channel B state of FIFO interrupt.
4	MTxRB	Transmitter B Ready Interrupt Mask 0 Mask the transmitter B ready interrupt. 1 Enable the transmitter B ready interrupt.
3	MCTR	Counter Ready Interrupt Mask 0 Mask the counter ready interrupt. 1 Enable the counter ready interrupt.
2	MBKA	Channel A Break Interrupt Mask 0 Mask the channel A break interrupt. 1 Enable the channel A break interrupt.
1	MSOFA	Channel A State of FIFO Interrupt Mask 0 Mask the channel A state of FIFO interrupt. 1 Enable the channel A state of FIFO interrupt.
0	MTxRA	Transmitter A Ready Interrupt Mask 0 Mask the transmitter A ready interrupt. 1 Enable the transmitter A ready interrupt.

IPCR Input Port Change

FFF8 2010 Read Only

The Input Port Change register (IPCR) indicates when the input ports change state as well as reporting the current state of the ports.

7	6	5	4	3	2	1	0
IP3SC	IP2SC	IP1SC	IP0SC	IP3S	IP2S	IP1S	IP0S

Bit	Mnemonic	Function
7	IP3SC	IP3 State Change 0 Channel B carrier detect (CD) did not change state. 1 Channel B CD changed state.
6	IP2SC	IP2 State Change 0 Channel A carrier detect (CD) did not change state. 1 Channel A CD changed state.
5	IP1SC	IP1 State Change 0 Channel B clear to send (CTS) did not change state. 1 Channel B CTS changed state.
4	IP0SC	IP0 State Change 0 Channel A clear to send (CTS) did not change state. 1 Channel A CTS changed state.
3	IP3S	IP3 Current State 0 Channel B carrier detect (CD) is asserted. 1 Channel B CD is not asserted.
2	IP2S	IP2 Current State 0 Channel A carrier detect (CD) is asserted. 1 Channel A CD is not asserted.
1	IP1S	IP1 Current State 0 Channel B clear to send (CTS) is asserted. 1 Channel B CTS is not asserted.
0	IP0S	IP0 Current State 0 Channel A clear to send (CTS) is asserted. 1 Channel A CTS is not asserted.

ISR Interrupt Status

FFF8 2014 Read Only

The Interrupt Status register (ISR) provides the status of all interrupt sources. When an interrupt condition occurs, the representative bit in the ISR register is set. If the corresponding bit in the interrupt mask register is set, the DUART asserts the interrupt request (INTRN) line, which sets the DUART interrupt request (DIR) bit in the interrupt status (IST) register of the system board. When this happens, the system board interrupt logic asserts the INT line to the CPU, and the CPU services the interrupt.

7	6	5	4	3	2	1	0
IPC	BKB	SOFB	TxRB	CTR	BKA	SOFA	TxRA

Bit	Mnemonic	Function
7	IPC	Input Port Change 0 The input ports have not changed state. 1 One of the four input ports has changed state. NOTE: The contents of IPC are valid only if the ACR register bits [3−0] have been set to enable input port changes to be registered into IPC.
6	BKB	Channel B Change in Break 0 The channel B receiver did not detect a break. 1 The channel B receiver detected a break.
5	SOFB	Channel B State of FIFO SOFB is programmed to respond to one of two conditions by the receiver ready interrupt select (RRIS) bit of the mode register (MR1B). If RRIS is set to 1, SOFB reacts to the state of the receiver, whether it is full or ready to receive data. If RRIS is cleared to 0, SOFB reacts to the state of the FIFO receive buffer, whether full or not full. 0 Receiver B is not ready to receive data, or the FIFO buffer is not full. 1 Receiver B is ready to receive data, or the FIFO buffer is full.
4	TxRB	Channel B Transmitter Ready 0 The transmit holding register is full, the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.
3	CTR	Counter/Timer ReadyCTR is programmed to respond to one of two modes, the Counter mode or the Timer mode, by the Counter/Timer mode (CTM) bits of the Auxiliary Control register (ACR). 0 The counter has not reached its terminal count, or the timer has not reached the second zero state. 1 The counter has reached its terminal count, or the timer has reached its second zero state. In the timer mode, CTR is set every other time that the counter reaches zero, not every time that the timer reaches zero.
2	BKA	Channel A Change in Break 0 The channel A receiver did not detect a break. 1 The channel A receiver detected a break.
1	SOFA	Channel A State of FIFO SOFA is programmed to respond to one of two conditions by the receiver ready interrupt select (RRIS) bit of the mode register (MR1A). If RRIS is set to 1, SOFA reacts to the state of the receiver A, whether it is full or ready to receive data. If RRIS is cleared to 0, SOFA reacts to the state of the FIFO receive buffer, whether full or not full. 0 Receiver A is not ready to receive data, or the FIFO buffer is not full. 1 Receiver A is ready to receive data, or the FIFO buffer is full.
0	TxRA	Channel A Transmitter Ready 0 The transmit holding register is full, or the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.

MR1A, MR1B Mode Registers

FFF8 2000 **MR1A** **Read/Write**

FFF8 2020 **MR1B**

The Mode registers (MR1A and MR1B) are accessed via independent auxiliary pointers. The pointer is set to MR1x (x is channel A or B) on a reset or by issuing a Reset Pointer command in the appropriate command register. Any read or write of the Mode Register while the pointer is at MR1x switches the pointer to MR2x where it remains until reset as described above.

7	6	5	4 3	2	1 0
RTSE	RRIS	EM	PM	PT	CF

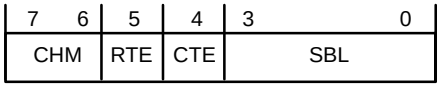
Bit	Mnemonic	Function
7	RTSE	Request to Send Enable 0 Disables the request to send (RTS) signal. 1 Enables the RTS signal.
6	RRIS	Receiver Ready Interrupt Select 0 If either RxA or RxB is ready to receive data, the corresponding state of FIFO (SOFA or SOFB) bit of the interrupt status register (ISR) is set. 1 When the FIFO buffer fills, the corresponding state of FIFO (SOFA or SOFB) bit of the interrupt status register (ISR) is set.
5	EM	Error Mode 0 Character 1 Block
4, 3	PM	Parity Mode 00 With parity 10 No parity 01 Force parity 11 Multidrop
2	PT	Parity Type 0 Even 1 Odd
1, 0	CF	Character Format (bits per character) 00 5 01 6 10 7 11 8

MR2A, MR2B Mode Registers

FFF8 2000 **MR2A** **Read/Write**

FFF8 2020 **MR2B**

The Mode registers (MR2A and MR2B) are accessed via independent auxiliary pointers. The pointer is set to MR1x (x is channel A or B) on a reset or by issuing a Reset Pointer command in the appropriate command register. Any read or write of the Mode Register while the pointer is at MR1x switches the pointer to MR2x, where it remains until reset as described above.



Bit	Mnemonic	Function
7, 6	CHM	Channel Mode 00 Normal 01 Auto−echo 10 Local loop 11 Remote loop
5	RTE	Request to Send Enable 0 Disable RTS. 1 Enable RTS.
4	CTE	Clear to Send Enable 0 Disable CTS. 1 Enable CTS.
3−0	SBL	Stop Bit Length 0 0.563 4 0.813 8 1.563 C 1.813 1 0.625 5 0.875 9 1.625 D 1.875 2 0.688 6 0.983 A 1.688 E 1.983 3 0.750 7 1.000 B 1.750 F 2.000

OPCR Output Port Configuration

FFF8 2034 Write Only

The Output Port Configuration register (OPCR) is used to configure the output ports OP2–OP7.

7	6	5	4	3 2	1 0
OP7	OP6	OP5	OP4	OP3	OP2

Bit	Mnemonic	Function
7	OP7	Output Port 7 Not used.
6	OP6	Output Port 6 Not used.
5	OP5	Output Port 5 Not used.
4	OP4	Output Port 4 Not used.
3, 2	OP3	Output Port 3 (port B DTR) 00 Inverts the signal at OP3. 01 C/T OUTPUT 10 TxCB(1x) 11 RxCB(1x)
1, 0	OP2	Output Port 2 (port A DTR) 00 Inverts the signal at OP2. 01 TxCA(16x) 10 TxCA(1x) 11 RxCA(1x)

SRA, SRB — Status

FFF8 2004 — SRA — Read Only
FFF8 2024 — SRB

The Status registers (SRA, SRB) provide the status of various DUART errors or conditions.

7	6	5	4	3	2	1	0
BRK	FE	PE	OE	TxE	TxRDY	FFL	RxRDY

Bit	Mnemonic	Function
7	BRK	Break 0 No error. 1 The channel received an all−zero character without a stop bit. Inhibits further entries into the FIFO until the RxD line returns to the mark state.
6	FE	Framing Error 0 No error. 1 The last character received did not have a stop bit.
5	PE	Parity Error 0 No error. 1 The last character received had the incorrect parity.
4	OE	Overrun Error 0 No error. 1 One or more characters in the received data stream were lost. NOTE: BRK, FE, PE and OE are cleared by writing 0x40 into the appropriate command register.
3	TxE	Transmitter Empty 0 The transmit register(s) have data, or the transmitter is disabled. 1 Both the transmit holding register and the transmit shift register are empty.
2	TxRDY	Transmitter Ready 0 The transmit holding register has data, or the transmitter is disabled. 1 The transmit holding register is empty and ready to be loaded with a character.
1	FFL	FIFO Full 0 The receive FIFO is not full; it has room for more data. 1 The receive holding register (FIFO) is full.
0	RxRDY	Receiver Ready 0 The receive holding register is empty. 1 The receive holding register (FIFO) has received a character for the CPU to read.

Programming the Parallel Interface

This section explains how to transmit data through the parallel interface.

PARALLEL — Parallel Port

FFF8 A000 — Read/Write

The parallel interface enables you to send parallel bytes of data to a printer. The interface has a 512−Kbyte buffer, therefore you can transmit as many as 512 characters at a time. When sending a file that exceeds 512 bytes, send the first 512 bytes, then wait for a Printer Buffer Empty (PBE) interrupt before sending the next 512 bytes. The interface generates the interrupt through bit 11 (PBE) of the Interrupt Status (IST) register. To clear the interrupt, write to the Clear Interrupt (CLRINT) register.

7 4	3	2	1	0
Reserved	PRP	SEL	Res	RDY

Bit	Mnemonic	Function
7−4	Reserved	
3	PRP	Printer Protocol Indicates the protocol being used. 1 Centronics. 0 Unused
2	SEL	Printer Select Indicates whether or not the printer is online. 1 The printer is online. 0 The printer is not online.
1	Reserved	
0	RDY	Printer Ready Indicates whether or not the printer is ready to print. 1 The printer is ready to print. 0 The printer is not ready to print.

Configuring the Parallel Interface

The parallel interface can communicate with a Centronics−compatible printer. To select the Centronics protocol, set the Printer Strobe Polarity bit (PSP, bit 22) of the Control register (CON) to 1 as described in 5, “Memory”.

CAUTION: Use caution when writing to the Control (CON) register; check all bits first.

End of Chapter

Chapter 7
Programming the PIT, CIO, and Real–Time Clock

This chapter describes the Zilog Z8536 Counter/Timer and Parallel I/O (CIO) device, the SGS–Thomson MK48T02B 2Kx8 Zeropowert/Timekeepert RAM chip, and the programmable interval timer (PIT).

The Real–Time Clock (RTC) and Nonvolatile RAM (NOVRAM) are parts of the SGS–Thomson MK48T02B. The section entitled "Programming the Real–Time Clock (RTC) and Nonvolatile RAM (NOVRAM)" describes the RTC and NOVRAM and how to program these devices. For more information on the Real–Time clock and the NOVRAM, see the SGS–Thomson *Memory Products Databook*.

Each processor has a PIT (Programmable Interval Timer) that it can use to generate time intervals.

The CIO section describes the programmable CIO registers, how the system board uses the CIO controller, and how to program the CIO. For more information on the CIO, see the *Z8536 CIO Counter/Timer and Parallel I/O Unit* data sheet by Zilog.

Programming the Real–Time Clock (RTC) and Nonvolatile RAM (NOVRAM)

The Real–Time Clock (RTC) and Nonvolatile RAM are contained within an SGS–Thomson MK48T02B 2Kx8 Zeropower™/Timekeeper™ RAM chip. The RAM within this chip is nonvolatile for over 3 years of operation without power. Real–time clock accuracy is within 1 minute per month at room temperature (25_C). The device provides a programmable register for calibrating the timekeeper accuracy. In addition to the timekeeper functions, this chip provides 2040 bytes of NOVRAM used for diagnostics and system configuration/boot information.

For a detailed description of the Real–Time clock and the NOVRAM, see the SGS–Thomson *Memory Products Databook*.

Real–Time Clock Registers

The Real–Time Clock (RTC) registers are 8–bit registers in the timekeeper RAM. Software can adjust the RTC by writing the desired time and date to these registers. These registers are realigned on word (32–bit) boundaries and occupy the last eight locations of the 2048 words reserved for the NOVRAM. Table 7–1 defines the registers and their addresses.

Table 7–1 Real–Time Clock Registers

Register	Address	Range	Format Example
Control	FFF8 1FE0	—	40
Seconds	FFF8 1FE4	00–59	41
Minutes	FFF8 1FE8	00–59	25
Hour	FFF8 1FEC	00–23	16
Day	FFF8 1FF0	01–07	05
Date	FFF8 1FF4	01–31	17
Month	FFF8 1FF8	01–12	08
Year	FFF8 1FFC	00–99	89

The Control register (FFF8 $1FE0_{16}$) allows you to read, set, or calibrate the RTC. Before reading or writing the RTC registers, you must prevent the clock from updating the registers by setting the appropriate bit: to read data set bit 6 of the Control register to 1; to write data set bit 7 to 1. Halting the register updating does not affect the Time of Boot (TOB) clock's timekeeping. Resetting the read bit to 0 continues updating of the registers; resetting the write bit to 0 transfers the register values to the RTC and continues the timekeeping operations. The date/time values occupy the low–order bits of the registers with any unused bits set to 0.

To calibrate the RTC, write a calibration value into bits 0–5 of the Control register. Bit 5 is the sign bit: 0 indicates negative calibration, slowing the clock down; 1 indicates positive calibration, speeding the clock up. Bits 0–4 contain a binary calibration value from 0–31. Each value modifies a number of minutes in each 64–minute cycle of the

clock. A value of 000012 modifies the first 2 minutes by 1 second per minute. A value of 001102 modifies the first 12 minutes by 1 second per minute.

NOVRAM Addresses

The NOVRAM contains 2040 bytes of nonvolatile data for Data General diagnostics and system configuration and boot information. The NOVRAM occupies two 4–kilobyte pages in the Mbus utility space. Each NOVRAM byte is word aligned in Mbus address space, and is accessed as the lowest–order byte (bits 7–0) of each word. Table 7–2 defines the NOVRAM addresses.

Table 7–2 NOVRAM Addresses

Address	NOVRAM Byte Number	Comments
FFF8 1FDC . . . FFF8 0400	2039 . . . 0256	General use (not write−protected)
FFF8 03FF . . . FFF8 0200	0255 . . . 0128	Reserved (write−protected)
FFF8 01FF . . . FFF8 0000	0127 . . . 0000	Reserved (not write−protected)

Programming the Programmable Interval Timer (PIT)

Each processor has a Programmable Interval Timer (PIT) that it can use to generate time intervals. This section describes the PIT registers.

PIT_DATA PIT Counter Value

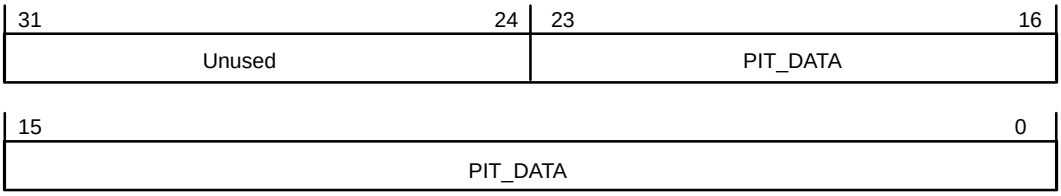
FFC8 4000 PIT_DATA Read/Write

The PIT counter value register (PIT_DATA) initializes the PIT counter. The counter is an up–counter; when the counter overflows bit 23, it sets the P bit in JPIST to 1. The value written to PIT_DATA must be the 2's complement of the desired time period. The counter is clocked at 25 MHz/256 (10.24 μsec per tick).

NOTE: Each job processor (CPU) has its own PIT_DATA register.

NOTE: The counter will continue to count after overflowing bit 23.

Local resets clear PIT_DATA to 0; the counter is disabled because PIT_SC is also reset.



Bit	Mnemonic	Function
31–24	Unused	
23–0	PIT_DATA	PIT Counter Value When written, inputs the initial value of the PIT counter. When read, supplies the current value of the PIT counter.

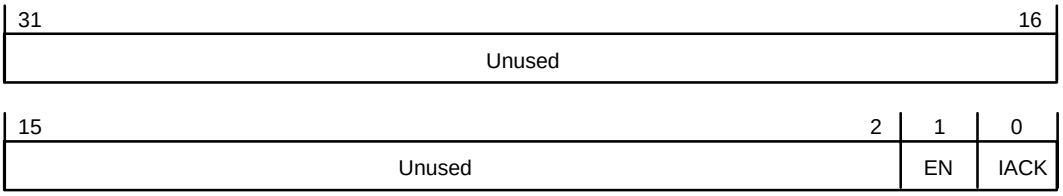
PIT_SC PIT Status and Command

FFC8 5000 PIT_SC Read/Write as noted

The PIT Status and Command (PIT_SC) register enables interrupts to a processor.

NOTE: Each processor has its own PIT_SC register.

PIT_SC is reset to 0 by local resets.



Bit	Mnemonic	Function
31–2	Unused	
1	EN	Enable PIT Counter (Read/Write) 1 Enable counter. 0 Disable counter.
0	ICLR	Clear the PIT Interrupt (Write only) 1 Clear the PIT overflow interrupt in JPIST. 0 Do not clear PIT overflow interrupt.

Programming the CIO

This section describes the features of the counter/timer and parallel I/O (CIO) device, how it connects to the system board, and to program it.

Features of the CIO

The Zilog Z8536 is a Counter/Timer and Parallel I/O (CIO) device. The CIO contains three counter/timers, one for each of the three ports, A through C. Each counter/timer consists of a 16–bit downcounter, a 16–bit Time Constant (TC) register, a 16–bit Current Count (CC) register, and two 8–bit registers for control and status. The counter/timers connect to the outside world through I/O ports. The parallel I/O portion is used only as a means to pass signals to and from the counter/timers.

Using the CIO

The Zilog Z8536 is a Counter/Timer and Parallel I/O (CIO) device. The CIO contains counter/timers and parallel I/O ports. Figure 7–1 illustrates the CIO, and Table 7–3 defines the CIO lines. Figure 7–2 illustrates the counter/timer jumpers to use when connecting the counter/timer inputs and outputs.

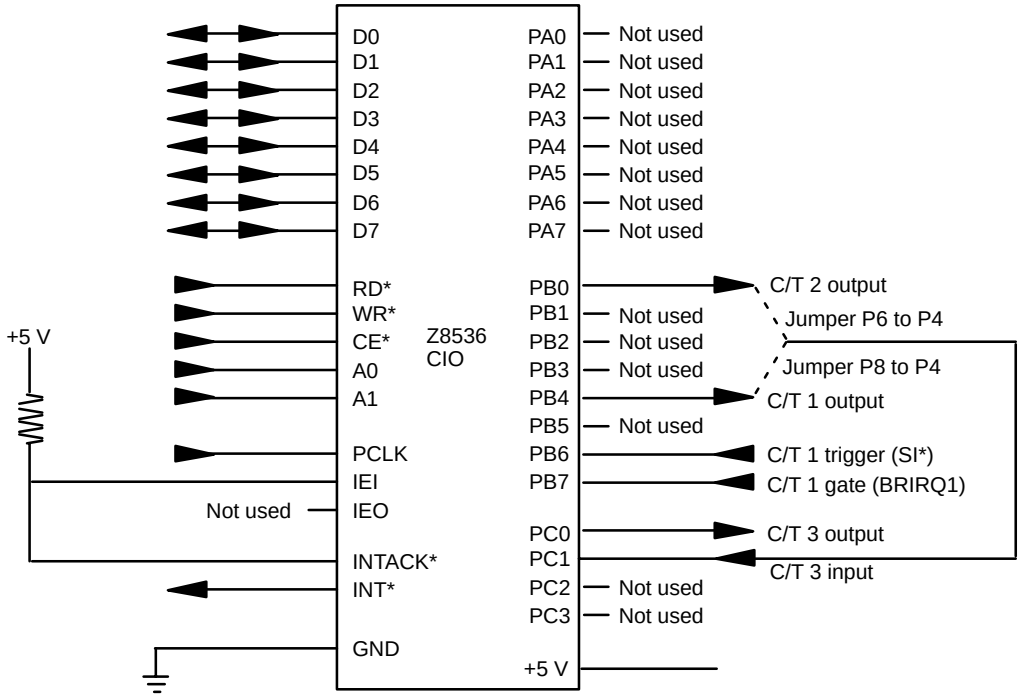


Figure 7–1 The Zilog Z8536 CIO

Table 7–3 The Zilog Z8536 CIO Lines

Signal	Description	Signal	Description
Data Bus		**Port B**	
D[7–0]	Data lines	PB[7]	C/T 1 gate input (from BRIRQ1)
Control, Timing and Reset		PB[6]	C/T 1 trigger input (speed indicator (SI) from modem)
RD*	Read enable	PB[5]	Not used.
WR*	Write enable	PB[4]	C/T 1 out (to C/T 3 input if jumper P8 to P4)
A[1, 0]	Address lines	PB[3–1]	Not used.
CE*	Chip enable	PB[0]	C/T 2 out (to C/T 3 input if jumper P6 to P4)
Interrupt		**Port C**	
INT*	Interrupt request	PC[3, 2]	Port C, lines 2 & 3. (Not used.)
INTACK*	Interrupt acknowledge	PC[1]	C/T 3 input (from C/T 1 output or C/T 2 output)
IEI	Interrupt enable input	PC[0]	C/T 3 output (to XYLINX XC3090 pin 230)
IEO	Not used.		
Port A			
PA[7–0]	Not used.		

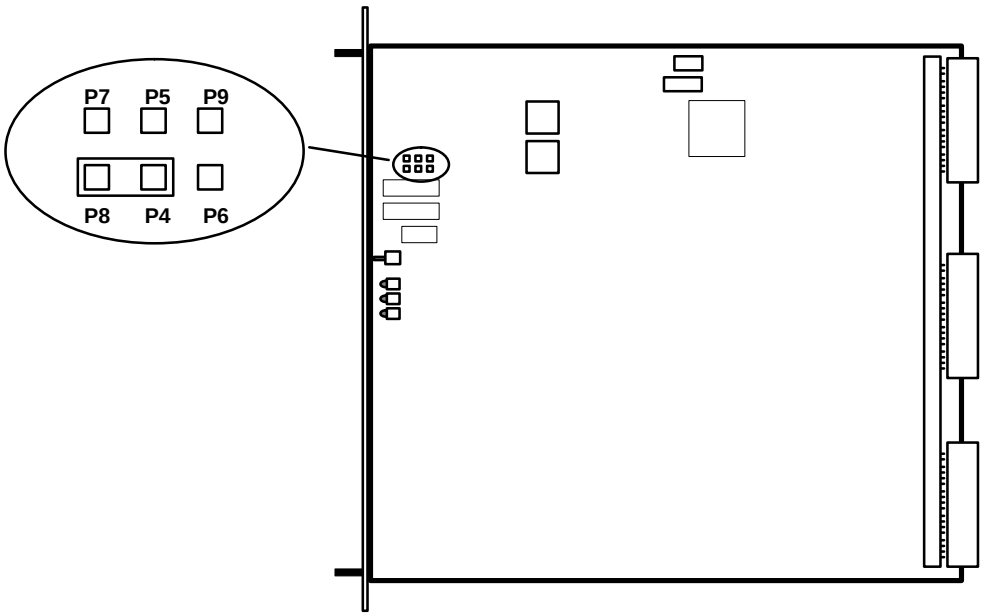


Figure 7–2 The Counter/Timer Jumpers – VIO Board

Programming the Counter/Timers

To use the CIO as a counter/timer, a programmer must set–up and start the counter/timer, then wait for an interrupt from the CIO indicating that the countdown has reached zero.

The CIO has programmable registers used to control the CIO and obtain status from the CIO. Table 7–4 identifies the CIO registers and their 6–bit pointers.

Table 7–4 CIO Registers

Register	Pointer	Register	Pointer
Master Control Registers		**Port Specification Registers**	
Master Interrupt Control	00	Port Mode Specification A	20
Master Configuration Control	01	Port Mode Specification B	28
Counter/Timer Registers		Port Handshake Specification A	21
Counter/Timer 1 Command & Status	0A	Port Handshake Specification B	29
Counter/Timer 2 Command & Status	0B	Port Command and Status A	08
Counter/Timer 3 Command & Status	0C	Port Command and Status B	09
Counter/Timer 1 Mode Specification	1B	**Bit Path Definition Registers**	
Counter/Timer 2 Mode Specification	1C	Data Path Polarity A	22
Counter/Timer 3 Mode Specification	1D	Data Path Polarity B	2B
Counter/Timer 1 Current Count (MSB)	10	Data Path Polarity C	05
Counter/Timer 1 Current Count (LSB)	11	Data Direction A	23
Counter/Timer 2 Current Count (MSB)	12	Data Direction B	2C
Counter/Timer 2 Current Count (LSB)	13	Data Direction C	06
Counter/Timer 3 Current Count (MSB)	14	Special I/O Control A	24
Counter/Timer 3 Current Count (LSB)	15	Special I/O Control B	2C
Counter/Timer 1 Time Constant (MSB)	16	Special I/O Control C	07
Counter/Timer 1 Time Constant (LSB)	17	**Port Data Registers**	
Counter/Timer 2 Time Constant (MSB)	18	Port A Data	0D
Counter/Timer 2 Time Constant (LSB)	19	Port B Data	0E
Counter/Timer 3 Time Constant (MSB)	1A	Port C Data	0F
Counter/Timer 3 Time Constant (LSB)	1B	**Pattern Definition Registers**	
Interrupt Vector Registers		Pattern Polarity A	25
Interrupt Vector Register A	02	Pattern Polarity B	2D
Interrupt Vector Register B	03	Pattern Transition A	26
Interrupt Vector Register C/T	04	Pattern Transition B	2E
Current Vector Register	1F	Pattern Mask A	27
		Pattern Mask B	2F

Access the control registers as follows:

1. Write the 6−bit pointer to the target register into the pointer register located at address FFF8 300C.
2. Read from or write to the target register.

When a countdown reaches zero, the CIO sets the Interrupt Pending (IP) bit in the appropriate Counter/Timer Command and Status (CTCS) register and asserts the INT* line. The system board interrupt logic processes this interrupt signal and sets the CIO Interrupt bit (CIO) in the Interrupt Status (IST) register. If another countdown reaches zero while a CIO interrupt is still pending (i.e. the IP bit is set) the CIO sets an internal error flag. When the pending interrupt completes, the error flag forces both the IP bit and the Interrupt Error (ERR) bit set. Figure 7–3 illustrates how the CIO counter/timers are configured.

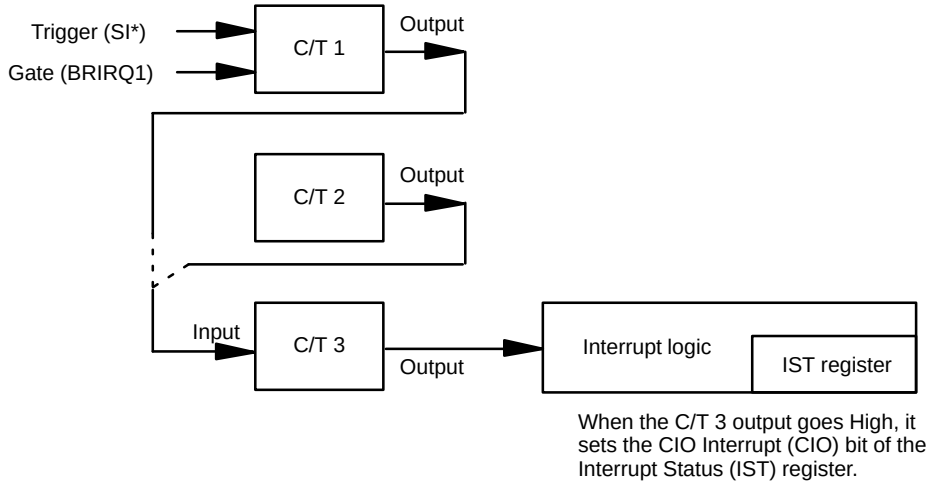


Figure 7–3 The Counter/Timers

CIO Registers

This section describes the CIO registers. The types of CIO registers include master control registers, counter/timer registers, interrupt vector registers, port specification registers, port data registers, and pattern definition registers.

Master Control Registers

This section describes the master control registers: Master Interrupt Control (MIC) and Master Configuration Control (MCC).

MIC — Master Interrupt Control

Pointer 00 — Read/Write

The Master Interrupt Control (MIC) register defines what type of interrupt request is pending. The interrupt vector is located in the Current Vector (CV) register. The MIC register also enables or disables the interrupt output line (INT*).

7	6	5	4	3	2	1	0
MIE	DLC	NV	PAV	PBV	CTV	RJA	RESET

Bit	Mnemonic	Function
7	MIE	Master Interrupt Enable Enables and disables all CIO interrupt requests by disabling the INT* line. 1 CIO interrupt requests are enabled or disabled by their respective interrupt enable bits. 0 CIO interrupt requests are disabled. MIE overrides all individual interrupt enable bits.
6	DLC	Disable Lower Chain Not used.
5	NV	No Vector Enables and disables the output of interrupt vectors when read by an interrupt service routine. 1 Disables the output of an interrupt vector when read. 0 Enables the output of an interrupt vector when read.
4	PAV	Port A Vector Enable Not used.
3	PBV	Port B Vector Enable Enables and disables the port B interrupt vector. 1 Enables the port B interrupt vector. 0 Disables the port B interrupt vector.
2	CTV	Counter/Timer Vector Enable Enables and disables the counter/timer interrupt vector. 1 Enables the counter/timer interrupt vector. 0 Disables the counter/timer interrupt vector.
1	RJA	Right Justified Address Justifies the address with either A0 or A1. 1 Right justify (A0 is loaded from AD0). 0 Shift left one bit (A0 is loaded from AD1).
0	RESET	Reset When RESET is set to 1, the CIO resets.

MCC Master Configuration Control

Pointer 01 Read/Write

The Master Configuration Control (MCC) register enables or disables the ports and timers.

7	6	5	4	3	2	1 0
PBE	CT1E	CT2E	CT3E	PLC	PAE	LC

Bit	Mnemonic	Function
7	PBE	Port B Enable Enables and disables port B. 1 Enables port B. 0 Disables port B.
6	CT1E	Counter/Timer 1 Enable Enables and disables counter/timer 1. 1 Enables counter/timer 1. 0 Disables counter/timer 1.
5	CT2E	Counter/Timer 2 Enable Enables and disables counter/timer 2. 1 Enables counter/timer 2. 0 Disables counter/timer 2.
4	CT3E	Counter/Timer 3 and Port C Enable Enables and disables counter/timer 3 and port C. 1 Enables counter/timer 3 and port C. 0 Disables counter/timer 3 and port C.
3	PLC	Port Link Control Links ports A and B. 1 Ports A and B are linked. 0 Ports A and B operate independently. *CAUTION: PLC must be set to zero.*
2	PAE	Port A Enable *CAUTION: Set PAE to zero.*
1, 0	LC[1, 0]	Counter/Timer Link Controls Links the counter/timer 1 output to counter/timer 2. **LC[1] LC[0] Function** 0 0 The C/Ts are independent. 0 1 C/T 1's output gates C/T 2. 1 0 C/T 1's output triggers C/T 2. 1 1 C/T 1's output is C/T 2's count input.

Counter/Timer Registers

This section describes the Counter/Timer registers.

CTCS Counter/Timer Command and Status

Pointer 0A **C/T 1** **Read/Partial Write**

Pointer 0B **C/T 2**

Pointer 0C **C/T 3**

The Counter/Timer Command and Status (CTCS) registers indicate the status of the counter/timers and pass commands to the counter/timers. Each counter/timer has an associated CTCS register with a pointer as shown above.

7	6	5	4	3	2	1	0
IUS	IE	IP	ERR	RCC	GCB	TCB	CIP

Bit	Mnemonic	Function
7	IUS	Interrupt Under Service (Read only) Indicates whether or not the Interrupt Pending (IP) interrupt is being serviced. 1 The IP interrupt is being serviced. 0 The IP interrupt is not being serviced.
6	IE	Interrupt Enable (Read/Write) Enables or masks the Interrupt Pending (IP) interrupt. 1 Enables the IP interrupt. When IP is set, the INT* line to the IST register of the system board will be asserted. 0 Masks the IP interrupt. The IP bit may be set, but the INT* line to the IST register will not be asserted.
5	IP	Interrupt Pending (Read only) IP is set when an event requiring interrupt servicing occurs. When IP is set, the CIO drives the INT* line Low. When read, IP indicates whether or not an interrupt is pending. 1 The counter has reached its terminal count, and the interrupt is pending. 0 No interrupt is pending for this counter/timer.
4	ERR	Interrupt Error (Read only) Indicates that an interrupt occurred while a previous interrupt was still pending. 1 Indicates that an interrupt error has occurred; i.e. an interrupt occurred while a previous interrupt was still pending. 0 No interrupt is pending for this counter/timer.
3	RCC	Read Counter Control (Write only) RCC can be set only. When set, RCC freezes the valueso that the CPU can read the counter/timer and get a valid count. When the CPU reads the counter/timer, RCC is cleared and the counter/timer countdown resumes.

(continued)

Bit	Mnemonic	Function
2	GCB	Gate Command Bit (Read/Write) Enables and disables the output of interrupt requests from the CIO. 1 Enables interrupt requests. 0 Disables interrupt requests.
		NOTE: The gate command bit must be set for the CIO to generate interrupts.
1	TCB	Trigger Command Bit (Write only) Writing a 1 to TCB loads the downcounter.
0	CIP	Count In Progress (Read only) Indicates whether or not the counter is counting down. 1 The downcounter has been loaded, and the countdown has not reached zero. 0 The downcounter has reached zero.

(concluded)

CTMS Counter/Timer Mode Specification

Pointer 1B **C/T 1** **Read/Write**

Pointer 1C **C/T 2**

Pointer 1D **C/T 3**

The Counter/Timer Mode Specification (CTMS) registers set various Counter/Timer parameters such as the method for triggering the counter/timer, the countdown rate, and whether or not the counter output is enabled.

7	6	5	4	3	2	1 0
C/SC	EOE	ECE	ETE	EGE	REB	DCS

Bit	Mnemonic	Function
7	C/SC*	Continuous/Single* Cycle Controls the operation of the counter when the count reaches zero. 1 Continuous cycle. When the count reaches zero, the time constant is reloaded and the countdown resumes. 0 Single cycle. When the count reaches zero, the countdown sequence stops.
6	EOE	External Output Enable Enables and disables the external output from the respective counter/timer. 1 Enables the external output. 0 Disables the external output.
5	ECE	External Count Enable When the downcounter uses the Timer mode to count down, the ECE bit determines the rate of the count down. 1 The downcounter counts down at the PCLK frequency. 0 The downcounter counts down at half the PCLK frequency.
4	ETE	External Trigger Enable Enables and disables the external trigger input to the respective counter/timer. 1 Enables the external trigger. 0 Disables the external trigger. NOTE: ETE is used for C/T 1 only; C/T 2 and C/T 3 do not use ETE.
3	EGE	External Gate Enable Enables and disables the external gate input to the respective counter/timer. 1 Enables the external gate. 0 Disables the external gate. *CAUTION: EGE must be set to zero for C/T 2 and C/T 3. EGE is used for C/T 1 only.*
2	REB	Retrigger Enable Bit Defines whether or not the counter/timer reacts to triggers initiated while the counter is counting down. If REB is set, the corresponding counter/timer reloads the downcounter and restarts the countdown sequence. 1 Reload and restart the counter/timer when triggered. 0 Do not respond to triggers.
1, 0	DCS	Output Duty Cycle Selects an output signal mode for the counter/timer as follows: **DCS[1] DCS[0] Function** 0 0 Pulse output 0 1 One−shot output 1 0 Square−wave output 1 1 Do not specify output

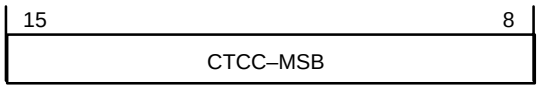
CTCC Counter/Timer Current Count

Pointer 10	**C/T 1, MSB**	**Read only**
Pointer 11	**C/T 1, LSB**	
Pointer 12	**C/T 2, MSB**	
Pointer 13	**C/T 2, LSB**	
Pointer 14	**C/T 3, MSB**	
Pointer 15	**C/T 3, LSB**	

The Counter/Timer Current Count (CTCC) registers contain the current value of their respective downcounter. To accurately read a CTCC register, the related counter/timer must be stopped. Each counter/timer has an associated CTCC register with a pointer as shown above.

CTCC Most–Significant Byte (MSB) Registers

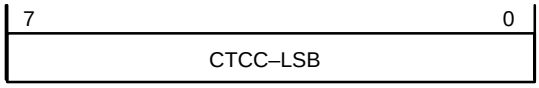
The CTCC MSB registers contain the most–significant byte of the 16–bit current count.



Bit	Mnemonic	Function
15–8	MSB	Most–Significant Byte Contains the current countdown value of the most–significant byte (MSB) of the associated counter/timer.

CTCC Least–Significant Byte (LSB) Registers

The CTCC LSB registers contain the least–significant byte of the 16–bit current count.



Bit	Mnemonic	Function
7–0	LSB	Least–Significant Byte Contains the current countdown value of the least–significant byte (LSB) of the associated counter/timer.

CTTC Counter/Timer Time Constant

Pointer 16	**C/T 1, MSB**	**Read/Write**
Pointer 17	**C/T 1, LSB**	
Pointer 18	**C/T 2, MSB**	
Pointer 19	**C/T 2, LSB**	
Pointer 1A	**C/T 3, MSB**	
Pointer 1B	**C/T 3, LSB**	

The Counter/Timer Time Constant (CTTC) registers contain the 16–bit time constants used by their respective counter/timers as an initial value when counting down to zero. Each counter/timer has an associated CTTC register with a pointer as shown above.

CTTC Most–Significant Byte (MSB) Registers

The CTTC MSB registers contain the most–significant byte of the 16–bit time constant.

Bit	Mnemonic	Function
15–8	MSB	Most–Significant Byte Contains the most–significant byte (MSB) of the initial time constant from which the counter/timer will count down.

CTTC Least–Significant Byte (LSB) Registers

The CTTC LSB registers contain the least–significant byte of the 16–bit time constant.

Bit	Mnemonic	Function
7–0	LSB	Least–Significant Byte Contains the least–significant byte (LSB) of the initial time constant from which the counter/timer will count down.

Interrupt Vector Registers

This section describes the interrupt vector registers.

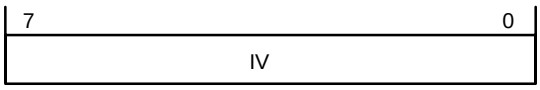
IV — Interrupt Vector

Pointer 02 **Port A (not used)** **Read/Write**

Pointer 03 **Port B**

Pointer 04 **Counter/Timers**

The Interrupt Vector (IV) registers contain the vector(s) for interrupt service routines. The CIO interrupt service routine reads the IV register to access the appropriate secondary interrupt service routine (ISR).

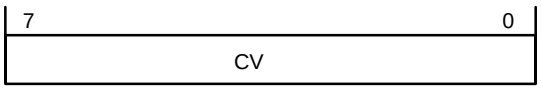


Bit	Mnemonic	Function
7–0	IV	Interrupt Vector Provides the interrupt vector to the CIO interrupt service routine. The CIO ISR executes the secondary ISR pointed to by the IV bit.

CV Current Vector

Pointer 1F Read only

The Current Vector (CV) register contains the interrupt vector for the current CIO interrupt request with the highest priority. The primary interrupt service routine executes the secondary interrupt service routine located at this vector.



Bit	Mnemonic	Function
7–0	CV	Current Vector CV contains the interrupt vector for the next CIO interrupt request to execute, i.e., the CIO interrupt with the highest priority.

Port Specification Registers

This section describes the port specification registers.

PMS — Port Mode Specification

Pointer 20 **Port A (not used)** **Read/Write**

Pointer 28 **Port B**

The Port Mode Specification (PMS) registers define the operating mode of port B.

7 6	5	4	3	2 1	0
PTS	ITB	SB	IMO	PMS	LPM

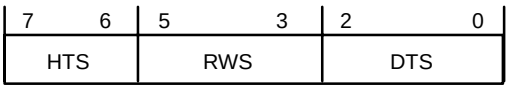
Bit	Mnemonic	Function
7, 6	PTS	Port Type Select Defines whether port B is an input port or an output port. **PTS[1] PTS[0] Function** 0 0 Bit port 0 1 Input port 1 0 Output port 1 1 Bidirectional port NOTE: If PTS is set for bidirectional port, program the data direction (DD) register to individually whether the lines of the port are input or output.
5	ITB	Interrupt on Two Bytes Defines whether or not the CIO waits until both input buffers for the port are filled to generate an interrupt request. If ITB is set to 1, the input port is not cleared until both bytes of data have been read. 1 The CIO does not generate an interrupt request until both input buffers are filled. 0 The CIO generates an interrupt request when one input buffer is filled.
4	SB	Single Buffered Mode Defines whether the port is treated as having one input buffer or two input buffers. 1 The port is treated as having one 1−byte input buffer. 0 The port is treated as having two 1−byte input buffers.
3	IMO	Interrupt on Match Only Interrupts the CPU only if a pattern is successfully matched. The PMS bit defines the mode used to match the patter
2, 1	PMS	Pattern Mode Specification PMS defines modes used to match patterns. **PMS[1] PMS[0] Function** 0 0 Disable pattern match 0 1 AND mode 1 0 OR mode 1 1 OR priority encoded vector mode.
0	LPM	Latch on Pattern Match LPM is in effect when the port is operating in Bit mode. If the bit pattern received by the port matches the bit pattern defined by the pattern registers (PP, PT and PM), LPM
	DTE	Deskew Time Enable DTE is in effect when the port is operating in Handshake mode. DTE enables or disables the deskew timer. When set to 1, DTE enables the deskew timer.

PHS Port Handshake Specification

Pointer 21 **Port A (not used)** **Read/Write**

Pointer 29 **Port B**

The Port Handshake and Specification (PHS) registers define the type of handshaking used by ports A and B.



Bit	Mnemonic	Function
7, 6	HTS	Handshake Type Specification See the *Z8536 CIO Counter/Timer and Parallel I/O Unit* data sheet by Zilog. **HTS[1] HTS[0] Function** 0 0 Interlocked handshake 0 1 Strobed handshake 1 0 Pulsed handshake 1 1 Three−wire handshake
5−3	RWS	REQUEST/WAIT* Specification See the *Z8536 CIO Counter/Timer and Parallel I/O Unit* data sheet by Zilog. **HTS[2] HTS[1] HTS[0] Function** 0 0 0 REQUEST/WAIT* disabled 0 0 1 Output WAIT* 0 1 1 Input WAIT* 1 0 0 Special request 1 0 1 Output request 1 1 1 Input request
2−0	DTS	Deskew Time Specification See the *Z8536 CIO Counter/Timer and Parallel I/O Unit* data sheet by Zilog.

PCS Port Command and Status

Pointer 08 Port A (not used) Read/Partial Write

Pointer 09 Port B

The Port Command and Status (PCS) registers provide status–information–for and control–of ports A and B.

7	6	5	4	3	2	1	0
IUS	IE	IP	ERR	ORE	IRF	PMF	IOE

Bit	Mnemonic	Function
7	IUS	Interrupt Under Service Indicates that the interrupt request is being serviced and is not completed. IUS is typically used within a daisy chain of two or more interrupting devices to prevent lower−priority devices from generating an interrupt request while an interrupt from the device with higher priority is being handled. This feature is not used in this system because the CIO is not configured within an interrupt daisy chain. 1 A CIO interrupt request is being serviced. 0 The CPU is not servicing any CIO interrupt.
6	IE	Interrupt Enable Enables or masks port C interrupt requests. 1 Port C interrupt requests are enabled. 0 Port C interrupt requests are masked.
5	IP	Interrupt Pending Indicates that an interrupt request is pending. When IP is set, the CIO asserts the INT* line which, through interrupt logic on the system board, sets the CIO Interrupt (CIO) bit in the Interrupt Status (IST) register. 1 An interrupt request is pending. 0 No interrupt requests are pending.
4	ERR	Interrupt Error (Read only) When set, ERR indicates that an interrupt error occured.
3	ORE	Output Register Empty (Read only) When set, ORE indicates that output register (buffer) is empty.
2	IRF	Input Register Full (Read only) When set, IRF indicates that the input register (buffer) is full.
1	PMF	Pattern Match Flag (Read only) When set, PMF indicates that the pattern has been matched.
0	IOE	Interrupt On Error See the *Z8536 CIO Counter/Timer and Parallel I/O Unit* data sheet by Zilog.

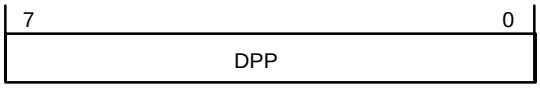
DPP — Data Path Polarity

Pointer 22 — **Port A (not used)** — **Read/Write**

Pointer 2B — **Port B**

Pointer 09 — **Port C (LSBs only)**

The Data Path Polarity (DPP) registers define whether or not the data ports' output signals are inverted.



Bit	Mnemonic	Function
7–0	DPP	Data Path Polarity Defines whether or not the data port signals are logically inverted. 1 The signal is inverted. 0 The signal is not inverted.

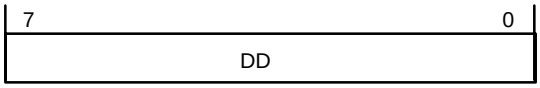
DD — Data Direction

Pointer 23 — **Port A (not used)** — **Read/Write**

Pointer 2C — **Port B**

Pointer 06 — **Port C**

The Data Direction (DD) register defines the direction (receive or transmit) of each line of the data ports A, B and C.



Bit	Mnemonic	Function
7–0	DD	Data Direction Defines the direction (input or output) of each line of the data port. 1 Input. 0 Output.

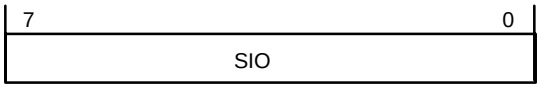
SIO — Special I/O Control

Pointer 24 **Port A (not used)** **Read/Write**

Pointer 2C **Port B**

Pointer 07 **Port C**

The Special I/O Control (SIO) registers define how the I/O is electrically configured.



Bit	Mnemonic	Function
7–0	SIO	Special I/O Control Defines whether or not the input bits are configured as a 1's catcher, and whether the outputs are configured as open drain. 1 Output with open drain, or input with 1's catcher. 0 Normal input or output.

Port Data Registers

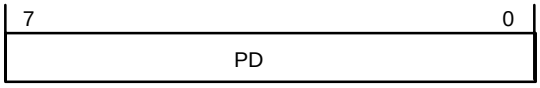
This section describes the Port Data registers.

PD — Port Data

Pointer 0D **Port A (not used)** **Read/Write**

Pointer 0E **Port B**

The Port Data (PD) register passes data to and from the corresponding port A or port B. The PD register consists of three registers: the input data register, the output data register, and the buffer register. These registers temporarily store data that was received by or that will be transmitted through the corresponding port.

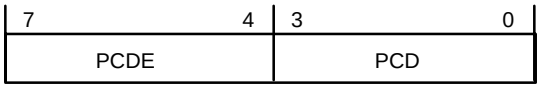


Bit	Mnemonic	Function
7–0	PD	Port Data When read, PD returns the contents of its input data register. When written, PD stores the data in the output data register to be sent through the corresponding port.

PCD Port C Data

Pointer 0F Read/Write

The Port C Data (PCD) register passes data to and from port C. The PCD register consists of two registers: the input data register and the output data register. These registers temporarily store data that was received by or that will be transmitted through port C.



Bit	Mnemonic	Function
7–4	PCDE	Port C Data These bits enable and mask bits 3–0. Bit 7 enables or masks bit 3, bit 6 affects bit 2, and so on. 1 Masks the corresponding bit. 0 Enables the corresponding bit.
3–0	PCD	Port C Data When read, PCD returns the contents of its input data register. When written, PCD stores the data in the output data register to be sent through port C.

Pattern Definition Registers

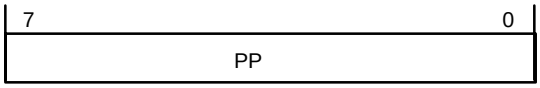
This section describes the pattern definitions registers.

PP — Pattern Polarity

Pointer 25 **Port A (not used)** **Read/Write**

Pointer 2D **Port B**

The Pattern Polarity (PP) registers define the patterns that will generate interrupts.



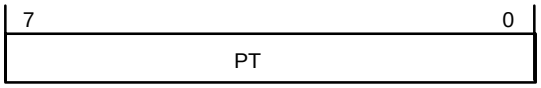
Bit	Mnemonic	Function
7–0	PP	Pattern Polarity Defines the polarity of the pattern that will generate an interrupt when matched. 1 One 0 Zero

PT — Pattern Transition

Pointer 26 **Port A (not used)** **Read/Write**

Pointer 2E **Port B**

The Pattern Transition (PT) registers define whether or not the patterns are detected as a transition from one state to another (i.e. 1 to 0, or 0 to 1) or as a steady state (0 or 1). The state or transition of each bit is defined by the Pattern Polarity (PP) register.



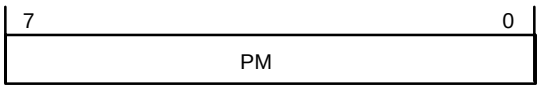
Bit	Mnemonic	Function
7–0	PT	Pattern Transition Defines whether the patterns are matched as a steady state or as a transition. 1 Transition 0 Steady state

PM — Pattern Mask

Pointer 27 **Port A (not used)** **Read/Write**

Pointer 2F **Port B**

The Pattern Mask (PM) registers mask or enable the pattern bits of the corresponding Pattern Polarity (PP) registers.



Bit	Mnemonic	Function
7–0	PM	Pattern Mask Masks and enables each pattern bit of the Pattern Polarity (PP) register. 1 Masks the corresponding pattern bit. 0 Enables the corresponding pattern bit.

End of Chapter

Chapter 8
The System Control Monitor (SCM)

Overview

The System Control Monitor (SCM) is the interface to AViiON® RISC–based computer hardware. The SCM is a firmware monitor program that tests and manages the system at powerup and maintains control until the operating system takes over. The SCM resumes control whenever the operating system is halted.

Several SCM commands, described later in this chapter, were designed for system programming, diagnostic development and program debugging.

NOTE: For SCM information not included in this chapter, see "*Starting AViiON® 6280 Series Systems*" (014–002178).

The SCM Prompt

The SCM prompt appears when all processors (CPUs) are halted.

In single–processor systems, the prompt is:

```
SCM>
```

In multiprocessor systems, the prompt displays the number (n) of the attached job processor:

```
Jp#n/SCM>
```

The SCM prompt may be changed via the **Prompt** command. On multiprocessor systems, the attached processor can be changed via the **Attach** command. These commands are described later in this chapter.

To access the SCM during powerup, press <**Ctrl–C**> before the operating system boots. If the operating system is already running, bring it down before running the SCM.

The SCM prompt appears when:

- The boot fails or is interrupted
- The user halts the operating system

- A command break sequence halts active processors
- The hardware reset or power switch is pressed
- The operating system encounters an unsupported breakpoint or interrupt

Exceptions are described in Motorola's book *MC88100 RISC Microprocessor User's Manual*.

Halting the Operating System

If an operating system is running, first shut it down; then halt the system. The SCM prompt will display.

If running the DG/UX™ operating system, shut it down as follows:

```
# cd /
# shutdown
```

NOTE: The **shutdown** command can be modified to specify whether to bring down software immediately, or to provide a period of time for users to log out. See the DG/UX documentation.

Then, halt the system and display the SCM prompt, as follows:

```
# halt –q
SCM>
```

CAUTION: Halting the system while the operating system or other software is running may result in lost or corrupted data. In a multiprocessor environment, all active processors (those started with a ***.START*** *system call) must receive the* ***.HALT*** *system call.*

If you don't have a DG/UX™ operating system, see the documentation for your operating system.

Resetting the System

To reset the system, either press the reset switch or turn the power off and on.

CAUTION: Always try to shut down the operating system before resetting the computer. Resetting the hardware while the operating system or other software is running may result in lost or corrupted data.

System Configuration Menu

To display the View or Change System Configuration menu, enter the following command line at the SCM prompt:

```
SCM> F ↵
```

```
View or Change System Configuration

1    Change default system boot path
2    Setup dual−initiator SCSI IDs
3    Change console parameters
4    Change modem port parameters
5    View system configuration
6    Change VME A24 configuration
7    Return to previous screen

Enter choice(s) −>
```

To select a sub–menu, type the menu number followed by a carriage return (enter).

To exit a menu, select the last item on the menu or press New Line at the prompt without selecting an item.

Changes made to the SCM become the new *default* parameter; however, the changes are not in effect until the system is powered up or reset. To reset the computer, either press the reset switch or issue the **Reset** command.

NOTE: To restore the original system configuration defaults, type Ctrl–I. Ctrl–I intializes the system console port to the following factory defaults: 9600 baud, 8 data bits, no parity, ANSI character set, enabled flow control. Also, Ctrl–I initializes the following system defaults: U.S. English keyboard language, Block transfer mode for SCSI tape drives, and 1200 baud for the mouse or modem port.

Environment Control Word

The Environment Control Word (ECW) (see Table 8–1) is part of the Change Test Parameters menu, which displays the state of the ECW bits and allows you to toggle the states. These changes do not affect the operating system or application programs; the ECW defines the test environment *in the SCM only* unless read or changed using the **.TECW** (Toggle Environment Control Word) system call.

Modify the ECW as follows:

1. While in the View or Change System Configuration menu, select "Change testing parameters."

   The system displays the following screen.

```
ECW Bit        Function                      State
-------------------------------------------------

0              Reserved                   – Disabled
1              Loop on error              + Enabled
2              Output to console          + Enabled
3              Percent failure            – Disabled
4              Print pass messages        + Enabled
5              Output to printer          – Disabled
6              Disassembler               + Enabled
7              Print subtest message      + Enabled
8              Report all                 – Disabled
9              Halt on error              – Disabled
10             Enable error logging       – Disabled
11             Continue on exception      – Disabled
12             Reserved                   – Disabled
13             Page mode                  – Disabled
14             Reserved                   – Disabled
15             Operator present           – Disabled
16–31          Reserved                   – Disabled

Select bit(s) to toggle –>
```

2. To toggle the parameter states, enter the bit number(s) and then press New Line. Separate bits by either a space or a comma.

NOTE: Bit 10 can only be toggled via the **.TECW** system call. You cannot toggle it via the Change Testing Parameters menu.

To Exit, press New Line *without* entering a number at the prompt.

Table 8–1 Environment Control Word

Bit	Function	State at Powerup
0	Reserved	0
1	Loop on error 0 Disables testing when program encounters an error. 1 Continues testing when program encounters an error.	0
2	Output to console 1 Directs program output to the system console.	1
3	Percent failure 0 Disables reporting of this error. 1 Enables reporting of percent of errors after looping (errors per total number of loops). Note that bit 1 (loop on error) must also be enabled.	0
4	Print pass messages 0 Disables printing of message. 1 Enables printing of messages to the system console after each test pass completes.	1
5	Output to printer 0 Disables output to printer. 1 Enables program output to the default printer port.	0
6	Disassembler 0 Disables display. 1 Enables displaying an additional output field that contains the mnemonics of memory address contents.	1
7	Print subtest message 0 Disables printing message. 1 Enables printing subtest messages to the system console.	1
8	Report all 0 Print brief messages to the system console. 1 Print verbose messages to the system console.	0
9	Halt on error 1 Enables halting the program after an error and returning the SCM prompt.	0
10	Enable error logging 0 Disables error logging. 1 Enables recording all errors in system error log.	0
11	Continue on exception 0 Displays the exception and exits to the SCM. 1 Continues when an exception occurs.	0
12	Reserved	0
13	Page mode 0 Disables Page mode. 1 Enables displaying output on the system console one screen (page) at a time	1
14	Reserved	0
15	Operator Present 0 Runs tests with default settings. 1 Each test prompts the user.	0
16–31	Reserved	0

System Calls

System calls manipulate CPU registers; the operating system can pass control to and from the SCM using these system calls.

System calls provide:

- Access to input/output devices.
- System configuration information.
- Panic and error reporting.

Execute a system call as follows:

1. Load the registers listed in the argument column of Table 8–2. Load the values shown in the argument column.

2. Set CR7, the Vector Base Register (VBR), to the VBR specified by the PROM during powerup (PROM VBR). If the value in CR7 is different from the PROM VBR, save the value in CR7 before writing the PROM VBR.

NOTE: During powerup, the PROM VBR is written to CR7.

3. Execute the Trap on bit clear instruction: **tb0 0,R0,496**

Specify bit 0 of R0. 496 is an offset to the VBR the PROM system call. Because bit 0 of register R0 is always 0, this instruction always executes the trap routine.

Table 8–2 System Calls

System Call	Argument(s)	Data Returned	Function
.BANNER	r9=113	r2=Pointer to string	Returns pointer to system banner string.
.CHAR	r9=0	r2(LSB)=ASCII character	Waits for an ASCII character from the default input port, reads it, and returns the character in the least significant byte of register 2. (A null indicates a break.)
.CHFLOW	r9=116 r2=0 or non−0 r3=Value	r2=Flow control flag	Reads or writes the character flow control (XON/XOFF) flag. If r2 = 0 initially, then r2 will contain the current flag value. If r2 = non−0 initially, then stores value from r3. An r3 value of F816 indicates flow control enabled; any other value indicates disabled.
.CHKSUM	r2=Pointer to data r3=Byte count r9=68	r2=Checksum	Performs data checksum test and returns the value in r2. (Adds all the bytes and complements the result.)
.CHSTAT	r9=5	r2(LSB)=Default input status	Polls the standard input port for character status and returns this value in the least significant byte of r2.
.COMMID	r9=114	r2=Pointer to address	Returns pointer to Ethernet address.
.COPYRIGHT	$R9=119_{16}$	R2=Pointer to copyright string	Returns a pointer to the prom copyright string.

.CPUCONFIG	R2=Pointer to buffer where config structure will be placed
	$R9=107_{16}$
		R2=0	Returns the CPU and cache configuration. More specifically the PROM call returns the PROM call revision, the number of CPUs in the system, the IGANG and DGANG values, a bit mask indicating which ddress bits the hardware uses to decode between CMMUs in a multiple ICache and/or Dcache system for a particular address and 1 word reserved for future second level cache information.
			
			The Cpu Configuration structure is:
			
			struct cpu_configuration
			{
			int version_number = 0;
			int number_of_cpus;
			short igang;
			short dgang;
			int icache_decode_mask;
			int dcache_decode_mask;
			}
			
			The version number identifies how the rest of the structure is defined.
			
			igang and dgang indicate how many CMMUs are on the instruction and data sides, respectively. The decode_mask entries indicate which logical address bits are used to enable each cache. For example a system with an igang count of 2 would have 1 bit set in the decode mask. If that bit is clear in the logical address then the 1st icache is enabled otherwise if its set then the 2nd is enabled.

.CPUID	r9=102	r2=CPU ID	Returns the CPU ID.

Note: If r2 is set to 1, an error has occurred.

(continued)

Table 8–2 System Calls

System Call	Argument(s)	Data Returned	Function

.DATETIME	R2=0 to read
	1 to set
	R3=Pointer to
	date/time structure
	R9=200_{16}
		R2=0, or 1 for error	Reads or sets the date and time. This is required because the dynamic password must be cleared if the date changes. An error will be returned on a set for any invalid values (ie. 30 days in Feb.).
			The date/time structure is:
			struct date_time
			{
			int year;
			int month;
			int day;
			int hour;
			int minute;
			int second;
			int day_of_week; /* not needed for set − will be computed */
			}
			All values are handled as binary integers, PROM will take care of any conversions that the clock module needs. The years run from 0 thru 99. January is month 1. The day_of_week is 1 for Sunday through 7 for Saturday and is not required when setting the date.

.GMT	R2=0 to read 1 to set R3=New value (if R2=1) R9=209_{16}	R2=GMT offset value	Reads or writes the GMT offset value. This value is the time in minutes that the local timezone is from GMT time. The valid range of times is −720 to +720 minutes.
.GTLINE	r2=32−bit string buffer address r9=2	r2=String length	Reads a character string of 256 characters or less from the standard input port, echoes them, and places the character string in the buffer address in r2. Terminator characters are: \n = New Line, \l = Carriage Return, and \f = Formfeed. The SCM screen edit control functions are supported.
.HALT	r9=63	None	Halts the user program and enters the SCM.
.HOSTID	r9=107	r2=Pointer to host ID	Returns to r2 a pointer to 4−byte binary host ID data.
.INVALID	r9=112 r2=JP# or −1	None	Invalidates the instruction cache (Icache). If r2 = a JP number, then only that JP Icache is invalidated. If r2 = −1, then all JP Icaches are invalidated.
.JPSTART	r2=JP# to start r3=Starting address r9=100	r2=Status	Starts another processor (JP#) after an initial boot (used only in multiprocessor systems). The status returned to r2 is 0 Start successful 1 Illegal or missing JP 2 Single JP configuration 3 JP not halted 4 JP does not respond
Note: If r2 is set to 1, an error has occurred.			
.MSIZE	r9=103 r2=0 or non−0	r2=Top of memory	Returns top of memory to r2. If r2 = 0 initially; then r2 will contain top of physical memory. If r2 = non−0 initially, then r2 will contain top of user memory.

(continued)

Table 8–2 System Calls

System Call	Argument(s)	Data Returned	Function
.NBLOCAL	r9=115 r2=0 or non−0 r3=Value	r2=LAN port number	Reads or writes the LAN port number. If r2 = 0 initially, then r2 will contain the LAN port number. If r2 = non−0 initially, then stores value from r3.
.OCHAR	r9=20 r2(LSB)=ASCII character	r2=0	Prints the value in the least significant byte of r2 to the standard output device.
.OCRLF	r9=26	r2=0	Prints a Carriage Return/line feed to the standard output device.
.PANIC	r9=110	r2=Error code	Halts the user program, returns an error code to r2, and enters the SCM.
.POLLKEY	r9=5	r2=Key hit	Returns an indication of whether or not a key was pressed. If r2 = 0, no key was pressed. If r2 = non−0, a key was pressed.
.PTLINE	r9=21 r2=32−bit address of string	r2=0	Prints the character string pointed to by the address in r2 to the standard output device. Does not return until it encounters the null terminator in the string. Note that this call allows five additional arguments and uses the C **printf** characteristics.
.REBOOT	r9=101 r2=0 or Pointer to boot path	None	Resets and reinitializes the workstation, initializes the boot time registers, and enters the boot menu. If r2 = 0, the call uses the default boot path. If r2 = non−0, the call uses the pointer in r2.
.REVNUM	r9=104	r2=Revision number	Returns PROM revision to r2 in the format: bit 31 (if 1), engineering revision; bits 30−16, major revision number; bits 15−0, minor revision number. For example, 80050002 = Rev E05.02 30000 = Rev 3.0
.STDIO	r9=70	r2=I/O device number	Returns the standard input and output ports. Device number values are 0 Serial input and output 1 Serial input/serial and graphics output 2 Keyboard input/graphics output
.SYSID	$R9=31_{16}$	R2=System ID	Returns the system ID. This ID is unique for each machine.
.TECW	r9=108 r2=0 or non−0 r3=New ECW value	r2=ECW	Returns or sets Environment Control Word (ECW). If r2 = 0, returns ECW to r2. If r2 not = 0, writes r3 value to ECW. Table 8−1 lists the ECW bit values, functions, and default states at powerup.

Note: If r2 is set to 1, an error has occurred.

(concluded)

SCM Subroutines

In addition to system calls, the System Control Monitor (SCM) supports hardwired entry points to subroutines in Table 8–3 (accessible with a **jsr** instruction containing the appropriate entry point).

Table 8–3 SCM Subroutines

Entry Point (Hex)	Subroutine	Argument	Description
1000	putchar to stdio	r2=char	Outputs the character in r2.
1004	getchar from stdio	r2=char	Returns a character to r2.

SCM Commands

The System Control Monitor (SCM) has commands to control and debug programs, boot media, and change system configuration parameters.

A command line consists of one valid SCM command and possibly one or more arguments.

If a command is issued incorrectly, the SCM displays a general message (ex. *Invalid command, Requires argument(s), Invalid argument(s))* and then returns the prompt.

Type a maximum of 80 characters per command line. You do not have to type the entire command name; the SCM accepts the first letter of a command.

SCM commands and arguments are *not* case–sensitive, with the exception of device specification arguments to the **BOOT** command, which must be lowercase.

The SCM supports several keyboard control characters. Table 8–4 describes control sequences used to edit command lines, interrupt and exit from SCM commands, and restore configuration parameters.

Address and Data Conventions

The address arguments used in SCM commands consist of two 16–bit, hexadecimal words. You may omit leading zeros. SCM commands support physical and logical address arguments according to the current mode of the memory management unit (MMU). When address translation is on, logical addresses map to physical addresses.

The assembler assumes data is decimal unless there is a dollar sign ($) preceding the data to indicate hexadecimal. The disassembler returns hexadecimal data. The SCM displays data output mnemonics and accepts data input mnemonics by default, since both the assembler and disassembler are enabled by default.

With address translation on, the SCM displays the contents of a memory location using the following format:

```
Memory logicaladdress – physicaladdress  /  data – mnemonic
```

For example, if the contents of memory location 10 is $5555FFFF_{16}$ the SCM displays the following:

```
Memory 00000010 – 00000010 / 5555FFFF – xor.u   r10 r21 $FFFF
```

The data mnemonic includes the opcode (xor.u in the example above), the registers pointed to by the first 16–bit word of the 32–bit address (r10 and r21), and the hexadecimal data in the second word of the address (FFFF).

Table 8–4 SCM Line Editing and Keyboard Control Commands

Keyboard Entry	Function
↵[1]	Completes the current input line, begins execution of command input, and returns the SCM prompt.
<**Ctrl–A**>	Recalls and displays the last command string you entered at the SCM prompt.
<**Ctrl–C**> [2]	Interrupts execution of an SCM command and returns the SCM prompt. This is a polled interrupt; some procedures complete before they break. If you do not have an auto−repeat keyboard, execute the Ctrl−C sequence repeatedly until you see the SCM prompt.
<**Ctrl–I**>	Restores default configuration parameters to the following factory settings. System console port: 9600 baud, 8 data bits, no parity, ANSI character set, enabled flow control, U.S. English keyboard language. Parallel printer: Centronics interface. SCSI tape drives: block transfer mode. Mouse or modem port: 1200 baud. Also restores video timing parameters. Enter <**Ctrl–I**>, wait until you hear one beep; then, enter **1** if you have a 70 Hz. monitor, or enter **2** for a 60 Hz. monitor.
<**Ctrl–P**> [3]	Displays the current state of the Environment Control Word (ECW).
<**Ctrl–Q**> [4]	Resumes SCM output display that was suspended with the Ctrl−S sequence.
<**Ctrl–S**> [4]	Suspends SCM output display until you resume it with the Ctrl−Q sequence.
<**Ctrl–U**>	Erases the current line of text, from the left of the cursor to the SCM prompt.

1 The New Line and Enter keys have special functions when used with the **EXAMINE** command..

2 Only functions as an interrupt to SCM functions.

3 Execute this sequence only while in the SCM.

4 Only works when Flow Control protocol is enabled within the SCM.

Table 8–5 SCM Commands

Command	Description	Function
. (period)	Displays processor status	Debugging
Attach	Specifies attached processor	Program control, system operation
Boot	Starts system from bootstrap device	See installation book.
Continue	Restarts attached processor	Program control, debugging
Display	Shows register file contents	Debugging
Examine	Open or edit contents of registers and memory locations	Debugging
Format	Displays View or Change Configuration menu	See installation book.
Help	Lists valid SCM commands	System operation, debugging program control
Initialize	Writes data to a range of memory	Debugging
Locate	Searches memory for a data pattern	Debugging
Move	Duplicates a memory block in new location	Debugging
Onestep	Executes the next program instruction	Debugging, program control
Prompt	Changes text of SCM prompt	See installation book.
Reset	Initializes system to power−up state	System operation
Start	Begins processor at specified address	Program control, system operation
Trap	Views or inserts breakpoints	Debugging
Untrap	Removes breakpoints	Debugging
View	Displays a range of memory	Debugging
Write	Inserts data in one memory location	Debugging
Zloader	Starts s−record loader utility	See installation book.

The command descriptions follow this format:

COMMAND–NAME

Command description.

COMMAND *argument [argument]*

NOTE: The minimal mnemonic for each command is always the *first letter* of the command name.

Description

Describes the command.

Arguments

Defines the arguments.

None	The command accepts no arguments.
argument	The command requires the argument defined here.
[argument]	The argument is optional; do not include brackets.

Related Commands

Lists closely related SCM commands.

Related Messages

Lists messages that may appear after invoking the command.

Examples

Examples of the command.

. (period)

Display job processor status.

. (period)

Description

The **.** (period) command displays the status of the attached job processor and its program registers.

The status includes the following:

- PSR (Processor Status Register)
  The processor state of the program that was last running on the attached processor. The PSR is the value in Control Register 1 (cr1).
- XPC (Execute Program Counter)
  The contents of the program counter (PC) of the program that caused entry into the SCM. XPC is from Control Register 4 (cr4).
- DCSH (Data Cache Enable/Disable)
  The state of the user program data cache before entering the SCM. Y indicates enabled, and N indicates disabled.
- DMMU (Data Memory Management Unit Enable/Disable)
  The state of the user program data MMU before entering the SCM. Y indicates enabled, and N indicates disabled.
- ICSH (Instruction Cache Enable/Disable)
  The state of the user program instruction cache before entering the SCM. Y indicates enabled, and N indicates disabled.
- IMMU (Instruction Memory Management Unit Enable/Disable)
  The state of the user program instruction MMU before entering the SCM. Y indicates enabled, and N indicates disabled.

Arguments

None

Related Commands

Attach	Specify the attached processor in multiprocessor systems.
Continue	Resume program execution at the program counter value of the currently attached processor.
Onestep	Execute the next single instruction of a program and then display trace information.

Related Messages

None

Examples

- Display processor status on single processor system.

```
SCM> .
PSR      XPC      DCSH      DMMU     ICSH  IMMU
A00003F2 FFC039DE N         N          N     N
```

- Display processor status information on multiprocessor system.

```
Jp#0/SCM> .
Jp#0/PSR XPC      DCSH      DMMU     ICSH  IMMU
A00003F2 FFC039DE N         N          N     N
```

ATTACH

Specify active job processor.

ATTACH *[jp#]*

Description

Attach is only valid for multiprocessor systems. It allows you to attach the SCM to a specified job processor for subsequent operations (in other words, it makes a particlar processor active). Unattached processors remain in an idle state. The SCM prompt indicates which job processor is currently attached (Jp#n/SCM> where n is the number of the attached job processor). By default, Jp#0 is the attached processor after powerup.

Attach returns the currently attached processor if used without an argument.

Arguments

[jp#]	The number of the job processor to attach.

Related Commands

. (period)	Display the status of the attached processor.
Continue	Resume program execution at the program counter value of the currently attached processor.
Prompt	Change the SCM prompt text.
Start	Execute a program from a specified address.

Related Messages

Jp#n attached

Examples

- Attach SCM to job processor #1.

```
Jp#0/SCM> A 1 ↵
Jp#1 attached
```

- Display the currently attached job processor.

```
Jp#1/SCM> A ↵
Jp#1 attached
```

BOOT

Starts the system from a specified device.

BOOT *[device][file]*

NOTE: For more information on the Boot command, see "*Starting AViiON® 6280 Series Systems" (DG # 014–002178).*

Description

BOOT resets the system hardware; then it loads a bootstrap program from a device specified in the *dev* argument. The argument format conforms to the common object file format (coff) defined in section 4 of the Binary Compatibility Standard (BCS).

When you use **BOOT** *without* an argument, the SCM boots from a default boot path. If the default boot path is not initialized or not valid, the SCM tries to find a valid bootstrap file according to a hardcoded probe sequence.

Arguments

device([controller([ctrlr_num,ctrlr_id])],[unit],[lun])
The device specification (first–stage boot) consists of a mnemonic that identifies the type of device and names the device driver, plus optional parameters that provide additional information to fully specify that device.

[ld:][/dir...]file Specifies an executable filename or an Internet host address, used and defined by the bootstrap program in a second–stage boot sequence.

Related Commands

FORMAT Displays the View or Change System Configuration menu. You can display the Change Boot Parameters menu from this menu.

Related Messages

Booting from ...

Unable to load boot file ...

Examples

1. Boot the default system boot path.

```
Jp#0/SCM> b ↵
```

2. Boot from file #0 on the first tape drive (SCSI ID#4).

   ```
   Jp#0/SCM> b st(dgsc(),4) ↵
   (Default third parameter and second stage boot: specifies
   st(dgsc(),4,0)0 )
   ```

3. Boot from the third file on the tape in the second tape drive (SCSI ID#5).

   ```
   Jp#0/SCM> b st(dgsc(),5)2 ↵
   ```

4. Boot your DG/UX operating system kernel to run level 3.

   ```
   Jp#0/SCM> b sd(dgsc()root:/dgux -3 ↵
   ```

5. Boot AViiON System Diagnostics (the program file **diags** located in the directory called **stand** on the logical disk **usr**) from the default system disk.

   ```
   Jp#0/SCM> b sd(dgsc(),0)usr:/stand/diags↵
   ```

6. Boot any executable file called **bootfile** in the root directory on the second SCSI disk (SCSI ID#1).

   ```
   Jp#0/SCM> b sd(dgsc(),1)root:/bootfile ↵
   ```

7. Boot from the host at Internet address 128.111.5.6 on a VLC Ethernet LAN.

   ```
   Jp#0/SCM> b hken()128.111.5.6: ↵
   ```

CONTINUE

Restart attached processor.

CONTINUE *[trace–count]*

Description

Continue resumes program execution at the address stored in the program counter (NPC) of the attached processor, for the number of instructions specified in the optional trace count argument. The NPC is CR5 (Control Register 5). When you use the command without the trace count argument, system control passes completely to the continued program.

Arguments

[trace–count]	The number of instructions to execute (in hexadecimal). The system displays the address, data, and mnemonic (in that order) after each instruction, then halts, displays the processor status, and returns the SCM prompt.

Related Commands

. (period)	Display the status of the attached processor, including the value of the NPC.
Attach	Specify the attached processor in multiprocessor systems.
Display	Show the contents of all register files.
Start	Execute a program from a specified address.

Related Messages

None

Examples

- Resume program execution at the NPC value; leave the SCM.

```
SCM> C ↵
```

- Resume program execution at the current PC value, display trace information after the next three instructions; then halt the processor and return to the SCM.

```
SCM> C 3 )
Trace       00000010       5555FFFF xor.u    r10 r21 $ffff
Trace       00000014       00000000 xmen.bu  r0 r0 $0
Trace       00000018       00000000 xmen.bu  r0 r0 $0

 PSR      XPC      NPC       FPC      DCSH  DMMU   ICSH  IMMU
 A0000000 0000001A 0000001E  00000022  Y     Y      N     N

SCM>
```

DISPLAY

Show contents of register files.

DISPLAY

Description

Display shows the contents of general register files (r), control register files (cr), and floating–point control registers (fcr) in the attached processor. Without an argument, only general register files are displayed.

Arguments

None

Related Commands

Attach Specify the attached processor in multiprocessor systems.

Examine Display and/or change specified registers or memory.

Related Messages

None

Examples

Display the current content of all register files.

```
SCM> D ↵

  r00 = 00000000  r01 = 00066B58  r02 = FFC00000   r03 = 00000000
  r04 = 0000000C  r05 = FFF00000  r06 = 0016E570   r07 = 0016B340
  r08 = 00003230  r09 = 00000063  r10 = 00000998   r11 = FFFFFFA1
  r12 = 00000000  r13 = 00000000  r14 = 00000000   r15 = 00000000
  r16 = 00000000  r17 = 00000000  r18 = 00000000   r19 = 00000000
  r20 = 00000000  r21 = 00000000  r22 = 00000000   r23 = 00000000
  r24 = 0016E570  r25 = 0000002C  r26 = 000E0000   r27 = 000F0000
  r28 = 00010000  r29 = 00000000  r30 = 017DEFD0   r31 = 017DEFC8
 cr00 = FFF8113C cr01 = 800003FB cr02 = 800003f0  cr03 = 00000000
 cr04 = 0006009A cr05 = 0006009E cr06 = 000600A2  cr07 = FFC00000
 cr08 = 0000403F cr09 = 0000403F cr10 = 017DEF40  cr11 = 00000000
 cr12 = 017DEF58 cr13 = 017DEF4C cr14 = 00000000  cr15 = 00000000
 cr16 = 00000000 cr17 = 00066B58 cr18 = FFC00000  cr19 = 00000000
 cr20 = 00000000fcr00 = 00000000fcr01 = 00000000 fcr02 = 00000000
fcr03 = 00000000fcr04 = 00000000fcr05 = 000048A2 fcr06 = 00000000
fcr07 = 00000000fcr08 = F8004808fcr62 = 00000001 fcr63 = 00000000
```

EXAMINE

Open and optionally change the contents of selected register or memory locations.

EXAMINE [*[M] address* | *H address* | *Q address* | *Rnumber* | *CRnumber* | *FCRnumber*]

Description

Examine opens and displays the contents of a specified memory address or register file. After opening a memory location, you can use special key functions described in Table 8–6 to enter new data, open the next or previous memory location, or return to the SCM prompt without making changes.

CAUTION: There are no restrictions to the areas of memory you can modify; modifying system control registers or NOVRAM locations could halt the system or destroy necessary system information. Ctrl–C will not recover data already modifed.

Use one argument per command line to specify whether you want to open a memory location or register file, and whether to view memory locations in word (32 bit), half–word (16 bit), or quarter–word (8–bit, or single byte) increments.

Without an argument, the **Examine** command opens and displays the last memory addressed examined. After the system halts or the first time after powerup, the SCM opens memory location 0 when you use the command without an argument.

Table 8–6 Special Key Functions for EXAMINE Command

Standard PC Keyboard	DASHER Keyboard	Function With EXAMINE Command
Escape	New Line or Break ESC	Writes data if entered; then closes the memory location or register and returns the SCM prompt.
Enter	Carriage Return	When a register file is open, closes the register and returns the SCM prompt. When a memory location is open, writes data if entered, closes the current memory location, and opens the next location.
Shift–6	Shift–6	When a register file is open, closes the register and returns the SCM prompt. When a memory location is open, writes data if entered, closes the current memory location, and opens the previous location.

Argument

[[M] address] The 32–bit (single word) memory location you want to examine or modify. Typing **M** is optional. If you type a number without a **M**, **H**, **Q**, **R**, **CR**, or **FCR** before it, the SCM interprets the number as a 32–bit memory address. Any address is masked to be a true 32–bit word (bits 0 and 1 are cleared).

[H address] The 16 bits (half–word) of memory you want to examine or modify. You must type **H**, followed by a space and the memory location. Any address is masked to be a true half–word (bit 0 is cleared).

[Q address] The 8 bits (one byte) of memory you want to examine or modify. You must type **Q**, followed by a space and the memory location. No address masking.

[Rnumber] The CPU register file you want to examine or modify. You must type **R**. Valid entries are R0 through R31.

[CRnumber] The control register you want to examine or modify. You must type **CR**. Valid entries are CR0 through CR20.

[FCRnumber] The floating point register you want to examine or modify. You must type **FCR**. Valid entries are FCR0 through FCR8, FCR62, or FCR63.

Related Commands

Display Show the contents of all register files.

Related Messages

None

Examples

NOTE: The examples in this section include keyboard characters for a standard IBM PC AT–compatible keyboard.

- Display the contents of memory address 1000 without modifying. Disassembler and MMU are disabled in this example.

```
SCM> E M 1000 ↵     or    SCM> E 1000 ↵
Memory 00001000 / 58670004 <Esc>
SCM>
```

- Display the contents of memory address 1000 without modifying. Disassembler and MMU are enabled in this example.

```
SCM> E 1000 ↵
Memory 00001000 – 00001000/12345678 – ld.d r17 r20 $5678
Esc
SCM>
```

- Display without modifying the contents of last memory address examined.

```
SCM> E ↵
Memory 00001000 – 00001000/12345678 – ld.d r17 r20 $5678
Esc
SCM>
```

- Display without modifying the contents of register file r03.

```
SCM> E R3 ↵   or   SCM> E R03 ↵
R03 = 00000000
SCM>
```

- Display without modifying the contents of control register file cr01.

```
SCM> E CR01 ↵   or   SCM> E CR1 ↵
r03 = 80000F3B ↵
SCM>
```

- Enter data 12345678 in memory address 1000; then exit from the command and return to the SCM.

```
SCM> E M1000 ↵   or   SCM>E 1000 ↵
Memory 00001000 / 58670004  12345678 <Esc>
SCM>
```

- Display the contents of memory address 1000; deposit data 12345678 in memory address 1004; then return to the SCM.

```
Jp#0/SCM> E 1000 ↵
Memory 00001000 / 12345678 <CR>
Memory 00001004 / 58670004   12345678 <Esc>
SCM>
```

- Display the contents of floating–point control register file fcr62; then deposit data 5555FFFF into fcr62.

```
SCM> E FCR62 ↵
fcr62 = 00000000   5555FFFF ↵
SCM>
```

- Display the first 16 bits at location 1000.

```
SCM> E H 1000 ↵
Memory 00001000 / 1234 <Esc>
SCM>
```

- Display the first 8 bits at location 1000.

```
SCM> E Q 1000 ↵
Memory 00001000 / 12 <Esc>
SCM>
```

FORMAT

Displays Configuration Menus.

FORMAT

Description

Format displays the “View or Change System Configuration” menu.

To return to the SCM prompt, press New Line or select the last item in the “View or Change System Configuration” menu.

Arguments

None

Related Commands

Boot Boot a device or display the Change Boot Parameters menu.

Related Messages

None

Examples

None

HELP

Display available SCM commands.

HELP

Description

Displays an alphabetical list of the minimal mnemonic for valid SCM commands, the arguments each command accepts, and a brief command description.

Arguments

None

Related Messages

None

Examples

None

INITIALIZE

Write specified data to a range of memory.

INITIALIZE *data beg–addr end–addr*

Description

The **Initialize** command writes specified data to a range of memory addresses that starts at the specified 32–bit beginning address and ends at the 32–bit ending address.

CAUTION: There are no restrictions to areas of memory you can initialize. Initializing system control registers or NOVRAM locations could halt the system or destroy necessary data. Use the Ctrl–C sequence to exit from the command during processing.

Arguments

data	Content written to each location of the memory range you are initializing.
beg–addr	First location in the range of memory.
end–addr	Last location in the range of memory.

Related Commands

Move	Duplicate the contents of a specified source range and write it to a specified destination range.
View	Display the contents of a range of memory.
Write	Write data to a single memory location.

Examples

Initialize memory by writing 5555FFFF to a range of memory beginning at address 0 and ending at 10. Then view the memory range to see the results.

```
Jp#1/SCM> I 5555FFFF 0 10 ↵

Jp#1/SCM> VIEW 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000004  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000008  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 0000000C  /  5555FFFF – xor.u         r10  r21  $FFFF
Memory 00000010  /  5555FFFF – xor.u         r10  r21  $FFFF
```

LOCATE

Find a specified data pattern in memory.

LOCATE *data [beg–addr] [end–addr]*

Description

Locate searches memory for a specified data pattern; then displays the address and contents of each location in which it finds the data. You can omit the address arguments and search all of physical memory, specify a starting address only, or specify a range of memory with starting and ending address arguments.

NOTE: A search through all of memory could take several hours! To stop a search before it completes, use Ctrl–C.

Arguments

data	Pattern in memory for which the system searches.
[beg–addr]	Location in memory where system begins searching. If you specify only one address argument, the system interprets it as a starting address.
[end–addr]	Location in memory where system stops searching.

Related Commands

Move	Duplicate the contents of a specified source range and write it to a specified destination range.
View	Display the contents of a range of memory.

Related Messages

None

Examples

- Locate each occurrence of data pattern 01234567 in a range of memory beginning at 0 and ending at 1000; then display the address of each occurrence.

```
Jp#1/SCM> L 01234567 0 1000 ↵
Memory 00000800  /  01234567 – ld.d r17  r20  $4567
```

- Locate each occurrence of data pattern 01234567 in all of main memory.

```
Jp#1/SCM> L 01234567 ↵

Memory 00000800  /  01234567 – ld.d r17  r20  $4567
Memory 00001200  /  01234567 – ld.d r17  r20  $4567
```

MOVE

Duplicate a block of memory.

MOVE *count source–addr dest–addr*

Description

The **Move** command copies the block of data that begins at the source address and ends after the specified count of 32–bit words; then moves the copy into a block of the same size starting at the destination address.

CAUTION: There are no restrictions to areas of memory into which you can move data. Overwriting data in system control registers or NOVRAM locations could halt the system or destroy necessary data. Use the Ctrl–C sequence to exit from the command during processing.

Arguments

count	Hexadecimal number of 32–bit words from the beginning address to the end of the block to be moved. There is no maximum or minimum block size.
source–addr	Memory location at the beginning of the range to be moved.
dest–addr	Memory location at the beginning of the range the block is moving to.

Related Commands

Initialize	Write data to a range of memory.
Locate	Search memory for a data pattern.
View	Display the contents of a range of memory.

Related Messages

None

Examples

- Move the data block beginning at memory address 0 and spanning 4F words to destination address 8000.

```
Jp#1/SCM> M 4F 0 8000 ↵
```

- Move the block that begins at 0 and spans 5 words to destination address 400. First, view the range of memory from locations 0 through 10.

```
Jp#1/SCM> VIEW 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000004  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000008  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 0000000C  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000010  /  5555FFFF – xor.u       r10 r21 $FFFF
```

  Next, move the block.

```
Jp#1/SCM> M 5 0 400 ↵
```

  Finally, view the contents of the block of memory from locations 400 through 410.

```
Jp#1/SCM> V 400 410 ↵

Memory 00000400  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000404  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000408  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 0000040C  /  5555FFFF – xor.u       r10 r21 $FFFF
Memory 00000410  /  5555FFFF – xor.u       r10 r21 $FFFF
```

ONESTEP

Execute next step of a program.

ONESTEP *[trace–count]*

Description

The **Onestep** command begins program execution at the address stored in the program counter (CR5) in increments of one instruction. It displays trace information after each instruction executes. With no argument, the system executes the single program instruction stored in CR5; then halts and displays status information.

Arguments

[trace–count] Hexadecimal number of instructions to be executed one by one (in single steps) before the processor halts.

Related Commands

. (period) Display the status of the attached processor.

Continue Resume program execution at the program counter value of the currently attached processor.

Start Execute a program from a specified address.

Related Messages

None

Examples

- Execute the instruction pointed to by the program counter.

```
SCM> 0 ↵

PSR      XPC      DCSH      DMMU     ICSH   IMMU
A0000000 FF001602 Y         N          N      N
```

- Execute three instructions one by one, beginning with the instruction pointed to by the PC.

```
SCM> 0 3 ↵

Trace       00000010       5555FFFF xor.u    r10 r21 $FFFF
Trace       00000014       00000000 xmen.bu  r0 r0 $0
Trace       00000018       00000000 xmen.bu  r0 r0 $0

PSR      XPC      DCSH      DMMU     ICSH  IMMU
A0000000 0000001A N         N          N     N
```

PROMPT

Changes SCM prompt text prefix.

PROMPT *[new–prompt]*

Description

Prompt changes the default SCM prompt to a specified ASCII string. This can be useful to uniquely identify multiple systems. The right bracket symbol (>) appears after the text prefix; if you change the prompt text to a null text string, your prompt is the right bracket symbol.

NOTE: Multiprocessor systems add the text string Jp#n/ (n = the number of the attached job processor) to the default prompt text.

Arguments

[new–prompt] Text string of ASCII characters to replace the prompt. The ASCII string can have as many as 1510 characters. There are no character or symbol restrictions.

Related Commands

Attach Specify the attached job processor in a multiprocessor system.

Related Messages

Argument(s) out of range

Examples

1. Display the current SCM prompt; then change it to **AV5000**.

```
Jp#0/SCM> P ↵
Jp#0/SCM>
Jp#0/SCM> P AV5000 ↵
Jp#0/AV5000>
```

RESET

Restore system to power–up state.

RESET

Description

Initializes system elements (excluding memory) to their original power–up state. Unlike a *cold reset* (power applied to the system), a *warm reset* (initiated by software, the **Reset** command, or a Reset switch) does not initialize memory or run power–up diagnostics.

*CAUTION: Be careful not to enter **R** at the SCM prompt accidentally. You cannot use Ctrl–C or an SCM command to recover.*

NOTE: In systems with Motorola 88204 CMMUs, RESET does not reset the serial interface; reset this serial interface via the DUART Reset (DRE bit 28) bit of the Control (CON) register as described in 5, “Memory”.

Arguments

None

Related Commands

Boot Boot a device.

Related Messages

System Reset

Examples

- Reset the system (processors, keyboard interface, graphics interface, etc.).

```
SCM> R ↵

PSR      XPC      DCSH     DMMU     ICSH  IMMU
A00003F2 FFC039DE N        N          N     N
```

START

Start job processor at specified address.

START *address [trace–count]*

Description

The **Start** command begins executing a program at the main memory address specified. The operating system or user program resumes system control unless you use the *trace–count* argument.

Arguments

address	Memory location at which the processor starts executing.
[trace–count]	The system displays the address, data, and mnemonic (in that order) after executing the hexadecimal number of instructions you specify with this argument. Then the system halts and the monitor displays status information.

Related Commands

Boot	Boot a device.
Continue	Resume program execution at the program counter value of the currently attached processor.
Onestep	Execute the next single instruction of a program and then display trace information.

Related Messages

None

Examples

- Start processor executing at address 398F0.

```
SCM> S 398F0 ↵
```

- Start processor executing at address 398F0 with a trace count of 3.

```
SCM> S 398F0 3 ↵

Trace       000398F0       5555FFFF xor.u    r10 r21 $FFFF
Trace       000398F4       00000000 xmen.bu  r0 r0 $0
Trace       000398F8       00000000 xmen.bu  r0 r0 $0

PSR      XPC      DCSH       DMMU    ICSH  IMMU
A0000000 000398FA Y          Y         N     N
```

TRAP

View or insert breakpoints.

TRAP *[address] ...*

Description

Insert a breakpoint at the specified address. You can insert up to 20_{10} breakpoints. The command does not allow duplicate breakpoints.

With no argument, **Trap** displays a list of all current breakpoints.

NOTE: The SCM implements breakpoints with the trap exception; hence, the command names **Trap** and **Untrap**.

Arguments

[address]	Memory location at which you want to insert a breakpoint.
...	The memory location of each subsequent breakpoint (when you insert more than one breakpoint per command line).

Related Commands

Untrap	Delete breakpoint(s)

Related Messages

None

Examples

- Insert a breakpoint at addresses 5000 and 77000.

```
SCM> T 5000 77000 ↵

Breakpoint # 1 – 00005000  Set
Breakpoint # 2 – 00077000  Set
```

- Display the current list of breakpoints and their addresses.

```
SCM> T ↵

Breakpoint # 1 – 00005000
Breakpoint # 2 – 00077000
```

No response indicates there are no current breakpoints.

UNTRAP

Delete breakpoints.

UNTRAP *[breakpoint] ...*

Description

The **Untrap** command removes the breakpoint number(s) specified, or removes all current breakpoints with no argument. To view current breakpoint numbers, use the **Trap** command.

Arguments

breakpoint	Number of the breakpoint you want to remove. (A decimal number 1 through 20.)
...	Numbers of each additional breakpoint number you want to remove.

Related Commands

Trap	Insert a breakpoint at specified address, or display the current breakpoints.

Related Messages

Invalid breakpoint

Examples

- Find the three current breakpoints; then remove two of them.

```
SCM> T  ↵

Breakpoint # 1 – 00005000
Breakpoint # 2 – 00077000
Breakpoint # 3 – 000C6001

SCM> U 1 3 ↵

Breakpoint # 1 deleted
Breakpoint # 3 deleted
```

- Remove all current breakpoints.

```
SCM> U  ↵
```

VIEW

Display a range of memory.

VIEW *beg–addr [end–addr]*

Description

The **View** command displays the contents of a specified block of contiguous memory. If you use the command with only a beginning address, the SCM displays all of memory from the specified address to the top of memory address.

NOTE: There are no restrictions to areas of memory you can specify. Viewing large blocks of memory could take hours, or cause memory exceptions. Use the Ctrl–C sequence to exit before completing the operation.

From left to right, the system displays the logical memory address, the physical memory address (only when MMU is enabled), the contents of the address, and the instruction mnemonic.

Arguments

beg–addr	First memory address in the range you want to view.
[end–addr]	Last memory address in the range you want to view.

Related commands

Initialize	Write data to a range of memory.
Locate	Search memory for a data pattern.
Move	Duplicate the contents of a specified source range and write it to a specified destination range.
Write	Write data to a single memory location.

Examples

- View the contents of a block of contiguous memory beginning at location 0 and ending at location 00000010.

```
SCM> V 0 10 ↵

Memory 00000000  /  5555FFFF – xor.u     r10 r21 $FFFF
Memory 00000004  /  5555FFFF – xor.u     r10 r21 $FFFF
Memory 00000008  /  5555fFFF – xor.u     r10 r21 $FFFF
Memory 0000000C  /  5555FFFF – Xor.u     r10 r21 $FFFF
Memory 00000010  /  5555FFFF – xor.u     r10 r21 $FFFF
```

WRITE

Insert data in one memory location.

WRITE *[H] [Q] address data*

Description

Write writes data to a specified address.

CAUTION: Write executes immediately; you cannot exit using Ctrl–C. There are no write restrictions; although writing data to system control registers or NOVRAM could halt the system or destroy necessary data.

Arguments

[H]	Address and data are 16–bits.
[Q]	Address and data are 8–bits.
address	Address to write data.
data	Hexadecimal data or assembler command.

Related Commands

Examine	Display and/or change specified registers or memory.

Related Messages

None

Examples

- Write 32 bits (one word) of data (01234567) to address 800.

  SCM> **W 800 01234567** ↵

- Write 16 bits (one half–word) of data (1234) to address 800.

  SCM> **W H 800 1234** ↵

- Write 8 bits (one byte) of data (12) to address 800.

  SCM> **W Q 800 12** ↵ or SCM> **W B 800 12** ↵

ZLOADER

Start the s–record load utility.

ZLOADER

Description

Zloader executes a program that downloads files in Motorola s–record format (S3/S7 type) to memory from a device connected to the system console port. The SCM recognizes s–records only after you use the **Zloader** command.

An s–record is a file that contains an unlimited number of records; each record has a maximum of 256 bytes. Servers use s–records to download files serially. The loader utility copies s–records from the serial port to system memory. The SCM reads information appended to individual s–records, stores each record at locations specified in the s–record header, and then verifies checksums. Information appended to the last s–record notifies the SCM when the entire file has been sent.

For this command to function, you must configure the system to use an s–record utility and a server must send s–record files. If you are not familiar with the s–record loader utility, if the system is not configured to receive s–records, or if the server is not sending, press the Ctrl–C sequence to exit from the **Zloader** command.

NOTE: The system will pause until it receives the last s–record in a file, or until you execute a Ctrl–C command sequence to exit to the SCM prompt.

Arguments

None

Related Commands

None

Related Messages

`DLL Started` DLL (downline loader) is an s–record load utility

`DLL Done`

Examples

None

End of Chapter

Appendix A
Address Map

The following table is a map of addressable locations within AViiON 6280 series systems.

Table A–1 Address Map

Resource	Address
User Space (3.996 Gbytes)	0000 0000 – FFBF FFFF
System Memory (DRAM) 768 MB maximum	0000 0000 – 2FFF FFFF
NOTE: All system memory must be contiguous..	
A32 space	FC00 0000 – FDFF FFFF
A24 space (16 Mbytes)	FE00 0000 – FEFF FFFF
A32 space (12 Mbytes)	FF00 0000 – FFBF FFFF
Utility Space (4 Mbytes)	FFC0 0000 – FFFF FFFF
EPROM (256 Kbytes)	FFC0 0000 – FFC7 FFFF
JPIEN	FFC8 0000
JPIST	FFC8 1000
XCLR	FFC8 2000
ABTCLR	FFC8 3000
PIT_DATA	FFC8 4000
PIT_SC	FFC8 5000
JPDIAG	FFC8 6000
WHOAMI	FFC8 7000
LWMAD	FFC8 8020
LRMAD	FFC8 8024
SRAM (128 Kbytes)	FFE0 0000 – FFE1 FFFF

(continued)

Table A–1 Address Map

Resource	Address
CMMU	
System Control Registers	
CMMU ID (IDR)	FFF0 X000
System Command (SCR)	FFF0 X004
System Status (SSR)	FFF0 X008
System Address (SAOR)	FFF0 X00C
System Control (SCTR)	FFF0 X104
Local Registers	
Local Status (PFSR)	FFF0 X108
Local Address (PFAR)	FFF0 X10C
Area Pointers	
Supervisor Area (SAPR)	FFF0 X200
User Area (VAPR)	FFF0 X204
BATC Write Pointers	
Port 0 (BWP0)	FFF0 X400
Port 1 (BWP1)	FFF0 X404
Port 2 (BWP2)	FFF0 X408
Port 3 (BWP3)	FFF0 X40C
Port 4 (BWP4)	FFF0 X410
Port 5 (BWP5)	FFF0 X414
Port 6 (BWP6)	FFF0 X418
Port 7 (BWP7)	FFF0 X41C
Cache Diagnostic Port	
Data Port 0 (CDP0)	FFF0 X800
Data Port 1 (CDP1)	FFF0 X804
Data Port 2 (CDP2)	FFF0 X808
Data Port 3 (CDP3)	FFF0 X80C
Tag Port 0 (CTP0)	FFF0 X840
Tag Port 1 (CTP1)	FFF0 X844
Tag Port 2 (CTP2)	FFF0 X848
Tag Port 3 (CTP3)	FFF0 X84C
Set Status (CSSP)	FFF0 X880

Where X=
0 CMMU0 – CPU0, Data CMMU
1 CMMU1 – CPU0, Instruction CMMU
2 CMMU2 – CPU1, Data CMMU
3 CMMU3 – CPU1, Instruction CMMU

during a reset, base address is FFF7 Xxxx:, X=
8 CMMU2 – CPU1, Data CMMU
9 CMMU3 – CPU1, Instruction CMMU
A CMMU0 – CPU0, Data CMMU
B CMMU1 – CPU0, Instruction CMMU

NOTE: During powerup, add 0007 0000 to the CMMU register addresses.

For information on the CMMU registers, see the *MC88200 User's Manual*.

Table A–1 Address Map

Resource	Address
Real−Time Clock Registers	
Control	FFF8 1FE0
Seconds	FFF8 1FE4
Minutes	FFF8 1FE8
Hour	FFF8 1FEC
Day	FFF8 1FF0
Date	FFF8 1FF4
Month	FFF8 1FF8
Year	FFF8 1FFC
DUART Registers	
Read Registers / Write Registers	
MR1A, MR2A / MR1A, MR2A	FFF8 2000
SRA / CSRA	FFF8 2004
Reserved / CRA	FFF8 2008
RHRA / THRA	FFF8 200C
IPCR / ACR	FFF8 2010
ISR / IMR	FFF8 2014
CTU / CTUR	FFF8 2018
CTL / CTLR	FFF8 201C
MR1B, MR2B / MR1B, MR2B	FFF8 2020
SRB / CSRB	FFF8 2024
Reserved / CRB	FFF8 2028
RHRB / THRB	FFF8 202C
Reserved / Reserved	FFF8 2030
Input port / OPCR	FFF8 2034
Start Counter / Set Output Bits	FFF8 2038
Stop Counter / Reset Output Bits	FFF8 203C
CIO Registers	
Port C Data Register	FFF8 3000
Port B Data Register	FFF8 3004
Port A Data Register	FFF8 3008
Control Registers	FFF8 300C
IEN0	FFF8 4004
IEN1	FFF8 4008
IEN2	FFF8 4010
IEN3	FFF8 4020
IEN	FFF8 403C
IST	FFF8 4040
SETSWI	FFF8 4080
CLRSWI	FFF8 4084
ISS	FFF8 4088
CLRINT	FFF8 408C
VIRL	FFF8 5000
VIAV1	FFF8 5004
VIAV2	FFF8 5008
VIAV3	FFF8 500C
VIAV4	FFF8 5010

(continued)

Table A–1 Address Map

Resource	Address
VIAV5	FFF8 5014
VIAV6	FFF8 5018
VIAV7	FFF8 501C
VIV	FFF8 5020
GLOBAL0	FFF8 6001
GLOBAL1	FFF8 6003
BRDID	FFF8 6005
GPCS0	FFF8 6007
GPCS1	FFF8 6009
GPCS2	FFF8 600B
GPCS3	FFF8 600D
GPCS4	FFF8 600F
UCS	FFF8 7000
BASAD	FFF8 7004
GLBRES	FFF8 700C
CCS	FFF8 8000
ERROR	FFF8 8004
EXTAD	FFF8 8010
EXTAM	FFF8 8014
WMAD	FFF8 8020
RMAD	FFF8 8024
WVAD	FFF8 8028
RVAD	FFF8 802C
CON	FFF8 9000
CON1 – (X0 bus, MEM0)	FFF8 9100
ERA1 – (X0 bus, MEM0)	FFF8 9104
CON2 – (X1 bus, MEM0)	FFF8 9110
ERA2 – (X1 bus, MEM0)	FFF8 9114
CON3 – (X0 bus, MEM1)	FFF8 9300
ERA3 – (X0 bus, MEM1)	FFF8 9304
CON4 – (X1 bus, MEM1)	FFF8 9310
ERA4 – (X1 bus, MEM1)	FFF8 9314
Parallel Printer Port	FFF8 A000
XSET0	FFF8 B000
XSET1	FFF8 B004
XSET2	FFF8 B008
XSET3	FFF8 B00C
XSET4	FFF8 B010
XSET5	FFF8 B014
XSET6	FFF8 B018
XSET7	FFF8 B01C
XSETALL	FFF8 B3FC
VME A16 space (64 Kbytes)	FFFF 0000 – FFFF FFFF

(concluded)

End of Appendix

Appendix B
System Powerup Flowchart

This appendix illustrates how the system hardware powers up, including power−up tests, indicators, initialization and reset.

A cold start is when the system is starting up from a complete powerdown. A cold start includes a powering−up phase during which the system voltages are generated and stabilize. A warm start is when a system is starting up from a system reset; the power supply was not shut down.

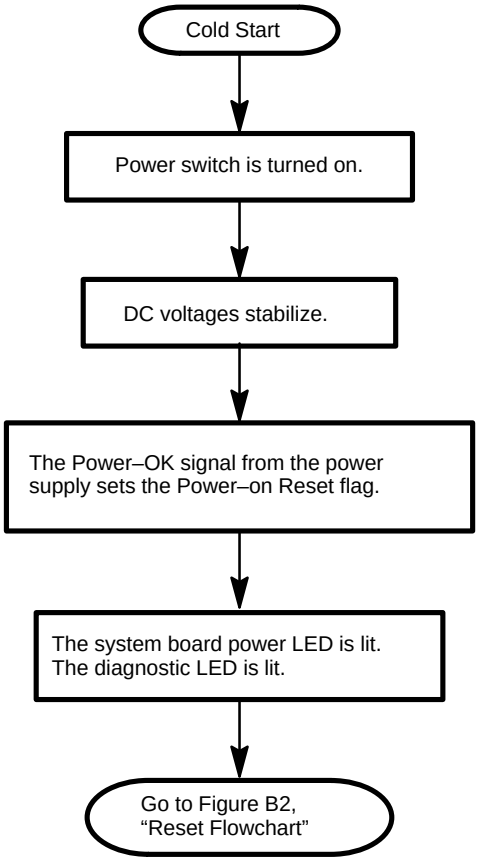


Figure B–1 Initial Powerup Flowchart

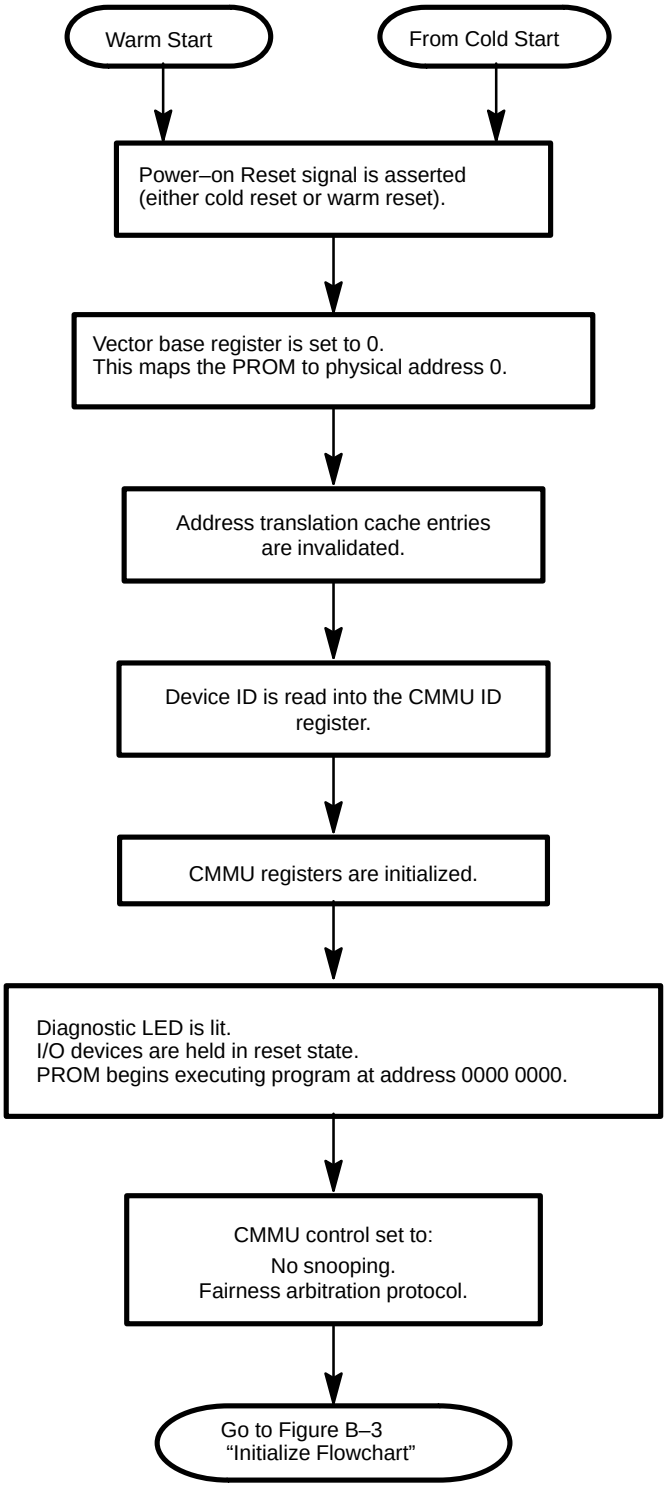


Figure B–2 Reset Flowchart

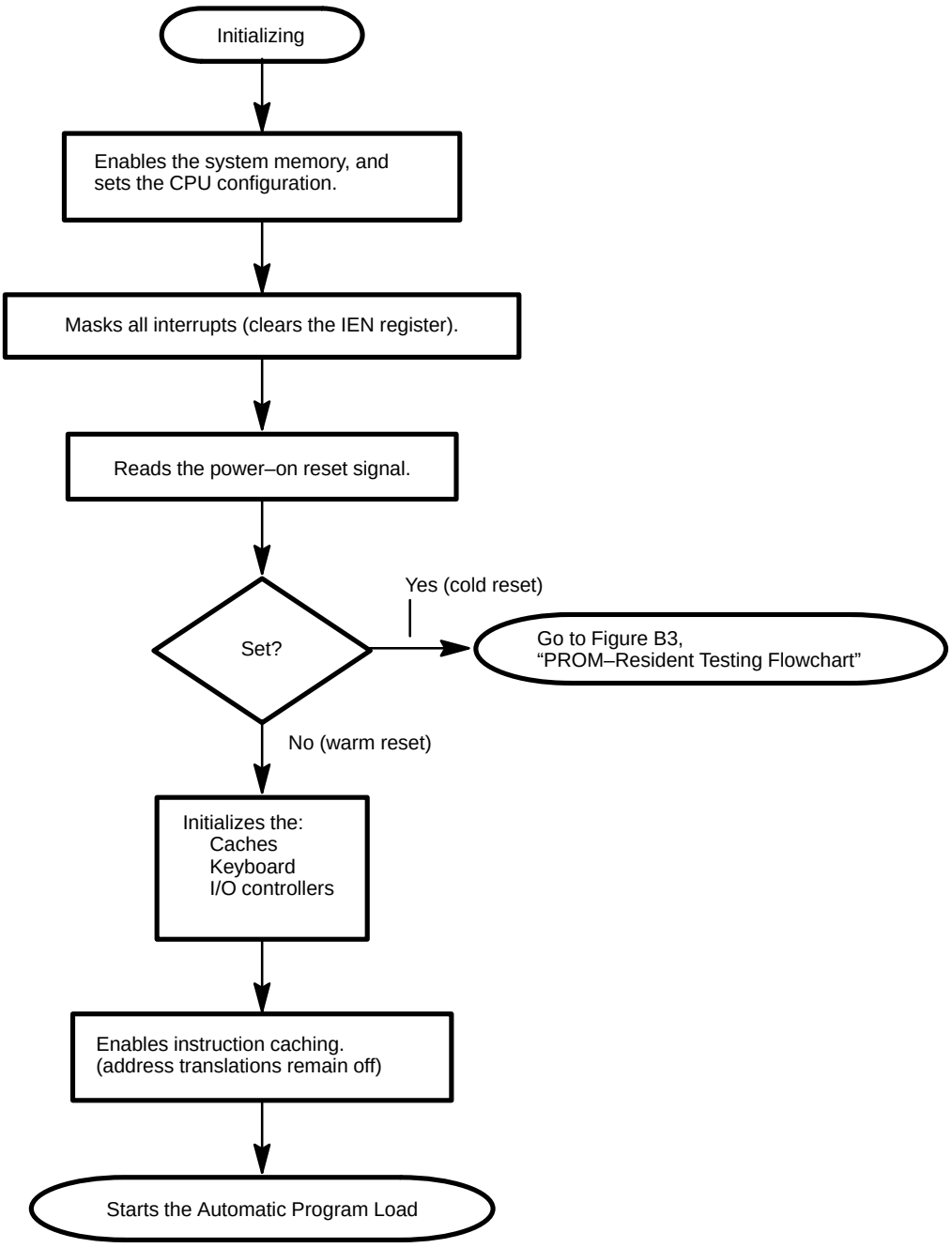


Figure B–3 Initialize Flowchart

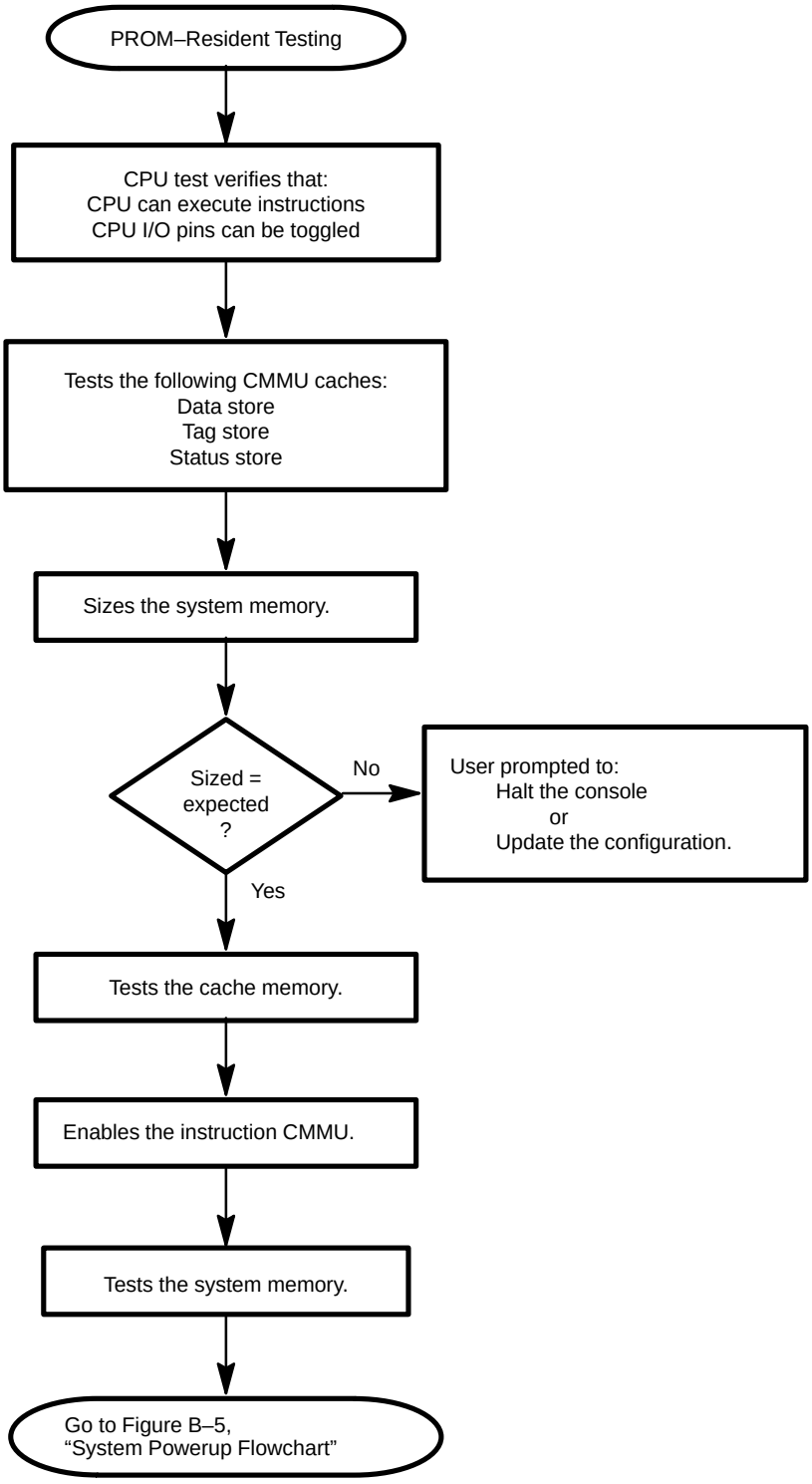


Figure B–4 PROM–Resident Testing Flowchart

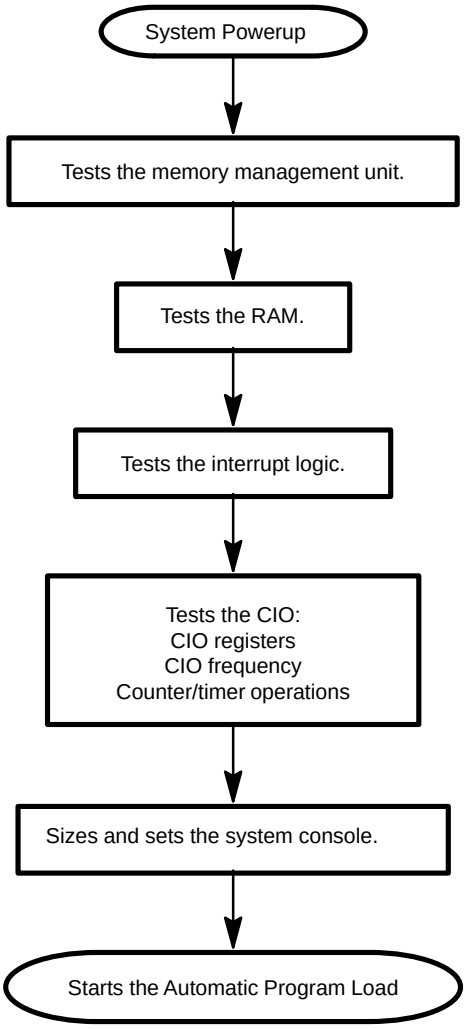
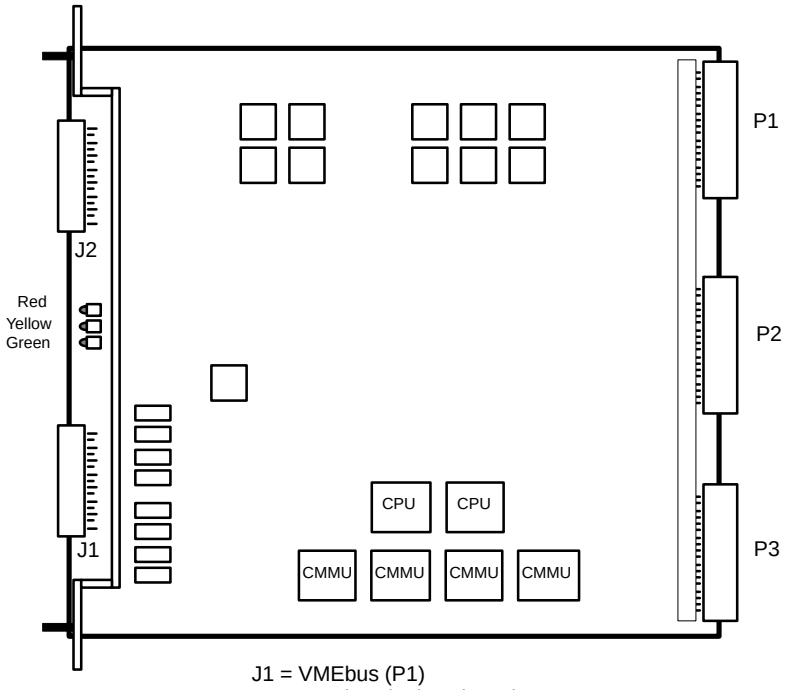


Figure B–5 System Powerup Flowchart

End of Appendix

Appendix C
Connectors

This appendix identifies the signals on the CPU board and VIO board connectors.



J1 = VMEbus (P1)
J2 = VMEbus (P2) and PExbus
J3 = PExbus
J4 = RS–232–C ports
J5 = Parallel port

Figure C–1 CPU Board

P1

Table C−1 describes connector P1 on the CPU board.

Table C–1 CPU Board Connector P1: Vbus, X0bus, and X1bus

Row A		Row B		Row C	
1	SLOTID[1]	33	$\overline{\text{BR0\_X0}}$	65	SLOTID[0]
2	$\overline{\text{BR1\_X0}}$	34	$\overline{\text{BR\_X0}}$	66	$\overline{\text{BR2\_X0}}$
3	BA_M2X0_D	35	$\overline{\text{BR3\_X0}}$	67	$\overline{\text{BRV\_X0}}$
4	GROUND	36	$\overline{\text{BR0\_X1}}$	68	BA_V2X0_D
5	$\overline{\text{BR1\_X1}}$	37	$\overline{\text{BR\_X1}}$	69	$\overline{\text{BR2\_X1}}$
6	BA_M2X1_D	38	$\overline{\text{BR3\_X1}}$	70	$\overline{\text{BRV\_X1}}$
7	GROUND	39	$\overline{\text{BR0\_V}}$	71	BA_V2X1_D
8	$\overline{\text{BR1\_XV}}$	40	$\overline{\text{BR\_V}}$	72	$\overline{\text{BR2\_V}}$
9	GROUND	41	$\overline{\text{BR3\_V}}$	73	GROUND
10	$\overline{\text{ANYCPUVBG}}$	42	$\overline{\text{BRV\_V}}$	74	$\overline{\text{VMEBUSY}}$
11	GROUND	43	V_AD[0]	75	ABRT
12	GINTR	44	V_AD[1]	76	$\overline{\text{SYSRESET}}$
13	GROUND	45	V_AD[2]	77	$\overline{\text{PEND\_OR}}$
14	−−−−	46	V_AD[3]	78	BA_M2V_D
15	GROUND	47	V_AD[4]	79	V_AD[5]
16	−−−−	48	V_AD[6]	80	−−−−
17	GROUND	49	V_AD[7]	81	V_AD[8]
18	−−−−	50	V_AD[9]	82	−−−−
19	GROUND	51	V_AD[10]	83	V_AD[11]
20	V_AD[12]	52	GROUND	84	V_AD[13]
21	V_AD[14]	53	V_AD[15]	85	V_AD[16]
22	V_AD[17]	54	$\overline{\text{V\_MEM\_UTIL\_EN}}$	86	V_AD[18]
23	V_AD[19]	55	GROUND	87	V_AD[20]
24	V_AD[21]	56	$\overline{\text{V\_MEM\_EN[1]}}$	88	−−−−
25	V_AD[22]	57	$\overline{\text{V\_MEM\_EN[2]}}$	89	LRST
26	V_AD[23]	58	−−−−	90	GROUND
27	V_AD[34]	59	V_AD[25]	91	V_AD[26]
28	V_AD[27]	60	V_AD[28]	92	V_AD[29]
29	$\overline{\text{X0\_SFST[2]}}$	61	$\overline{\text{X1\_SFST[2]}}$	93	V_AD[30]
30	$\overline{\text{X0\_SFST[1]}}$	62	$\overline{\text{X1\_SFST[1]}}$	94	V_AD[31]
31	−−−−	63	GROUND	95	+12V
32	+5V	64	+5V	96	−12V

P2

Table C−2 describes connector P2 on the CPU board.

Table C–2 CPU Board Connector P2: Vbus, X0bus, and X1bus

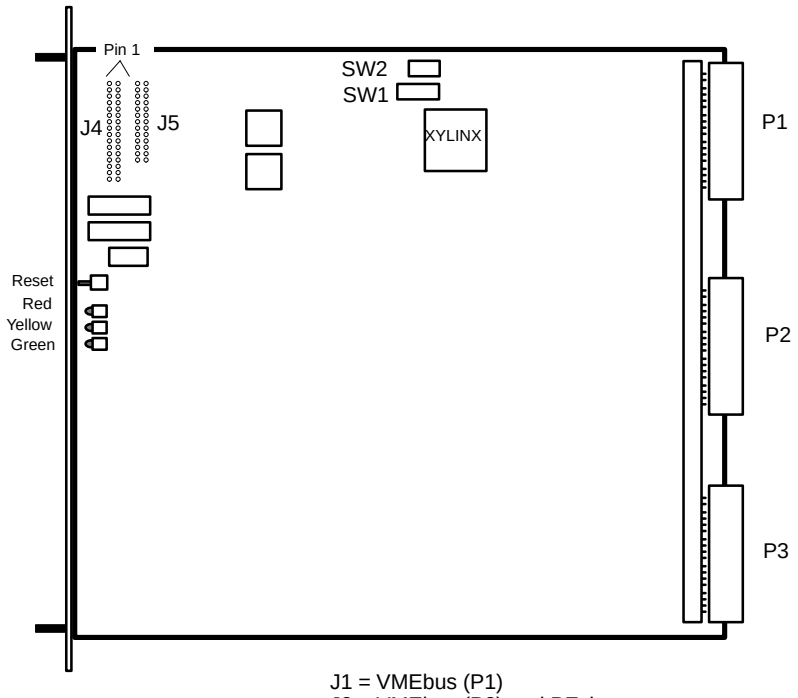
Row A		Row B		Row C	
1	$\overline{\text{XO\_MEM\_UTIL\_EN}}$	33	+5V	65	$\overline{\text{X0\_MEM\_ST[0]}}$
2	V_VIOST[1]	34	GROUND	66	GROUND
3	$\overline{\text{X0\_MEM}}$	35	$\overline{\text{X1\_MEM\_UTIL\_EN}}$	67	$\overline{\text{V\_XBARST[1]}}$
4	GROUND	36	$\overline{\text{X1\_MEM\_ST[0]}}$	68	GROUND
5	V_VIOST[2]	37	$\overline{\text{X1\_MEM\_ST[1]}}$	69	+5V
6	$\overline{\text{V\_XBARST[2]}}$	38	+5V	70	GROUND
7	V_VIOST[3]	39	$\overline{\text{X1\_MEM\_ST[3]}}$	71	$\overline{\text{X0\_MEM\_ST[3]}}$
8	$\overline{\text{V\_XBARST[3]}}$	40	+5V	72	GROUND
9	GROUND	41	+5V	73	+5V
10	$\overline{\text{X0\_SS[0]}}$	42	$\overline{\text{X1\_SS[0]}}$	74	GROUND
11	+5V	43	+5V	75	$\overline{\text{X1\_SS[2]}}$
12	$\overline{\text{X0\_SS[2]}}$	44	GROUND	76	GROUND
13	$\overline{\text{X0\_SS[3]}}$	45	+5V	77	$\overline{\text{X1\_SS[3]}}$
14	GROUND	46	+5V	78	GROUND
15	$\overline{\text{V\_BB}}$	47	+5V	79	$\overline{\text{CPU\_CLK}}$
16	+5V	48	+5V	80	GROUND
17	V_VIOST[0]	49	$\overline{\text{X1\_BB}}$	81	$\overline{\text{X0\_BB}}$
18	$\overline{\text{V\_XBARST[0]}}$	50	−−−−	82	GROUND
19	GROUND	51	V_C[0]	83	+5V
20	X0_C[0]	52	X1_C[0]	84	GROUND
21	X0_C[1]	53	X1_C[1]	85	V_C[1]
22	X0_C[2]	54	GROUND	86	GROUND
23	X0_C[3]	55	X1_C[2]	87	V_C[2]
24	GROUND	56	X1_C[3]	88	V_C[3]
25	X0_C[4]	57	X1_C[4]	89	V_C[4]
26	X0_C[5]	58	X1_C[5]	90	V_C[5]
27	X0_C[6]	59	X1_C[6]	91	V_C[6]
28	$\overline{\text{X0MEM\_EN[1]}}$	60	$\overline{\text{X1\_MEM\_EN[1]}}$	92	GROUND
29	GROUND	61	$\overline{\text{X0\_MEM\_EN[2]}}$	93	$\overline{\text{X1\_MEM\_EN[2]}}$
30	$\overline{\text{FAIL}}$	62	+5V	94	+5V
31	$\overline{\text{X0\_SS[1]}}$	63	GROUND	95	$\overline{\text{X1\_SS1}}$
32	$\overline{\text{X0\_MEM\_SS1}}$	64	+5V	96	$\overline{\text{X1\_MEM\_SS1}}$

P3

Table C−3 describes connector P3 on the CPU board.

Table C–3 CPU Board Connector P3: Vbus, X0bus, and X1bus

Row A		Row B		Row C	
1	+5V	33	X0_AD[0]	65	X1_AD[0]
2	X1_AD[1]	34	X0_AD[1]	66	GROUND
3	X1_AD[2]	35	X0_AD[2]	67	GROUND
4	+5V	36	X0_AD[3]	68	X1_AD[3]
5	X1_AD[4]	37	X0_AD[4]	69	GROUND
6	X1_AD[5]	38	X0_AD[5]	70	GROUND
7	+5V	39	X0_AD[6]	71	X1_AD[6]
8	X1_AD[7]	40	X0_AD[7]	72	GROUND
9	X1_AD[8]	41	X0_AD[8]	73	GROUND
10	+5V	42	X0_AD[9]	74	X1_AD[9]
11	X1_AD[10]	43	X0_AD[10]	75	GROUND
12	X1_AD[11]	44	X0_AD[11]	76	GROUND
13	+5V	45	X0_AD[12]	77	X1_AD[12]
14	X1_AD[13]	46	X0_AD[13]	78	GROUND
15	X1_AD[14]	47	X0_AD[14]	79	GROUND
16	+5V	48	X0_AD[15]	80	X1_AD[15]
17	X1_AD[16]	49	X0_AD[16]	81	GROUND
18	X1_AD[17]	50	X0_AD[17]	82	GROUND
19	+5V	51	X0_AD[18]	83	X1_AD[18]
20	X1_AD[19]	52	X0_AD[19]	84	GROUND
21	X1_AD[20]	53	X0_AD[20]	85	GROUND
22	+5V	54	X0_AD[21]	86	X1_AD[21]
23	X1_AD[22]	55	X0_AD[22]	87	GROUND
24	X1_AD[23]	56	X0_AD[23]	88	GROUND
25	+5V	57	X0_AD[24]	89	X1_AD[24]
26	X1_AD[25]	58	X0_AD[25]	90	GROUND
27	X1_AD[26]	59	X0_AD[26]	91	GROUND
28	+5V	60	X0_AD[27]	92	X1_AD[27]
29	X1_AD[28]	61	X0_AD[28]	93	GROUND
30	X1_AD[29]	62	X0_AD[29]	94	GROUND
31	+5V	63	X0_AD[30]	95	X1_AD[30]
32	X1_AD[31]	64	X0_AD[31]	96	GROUND



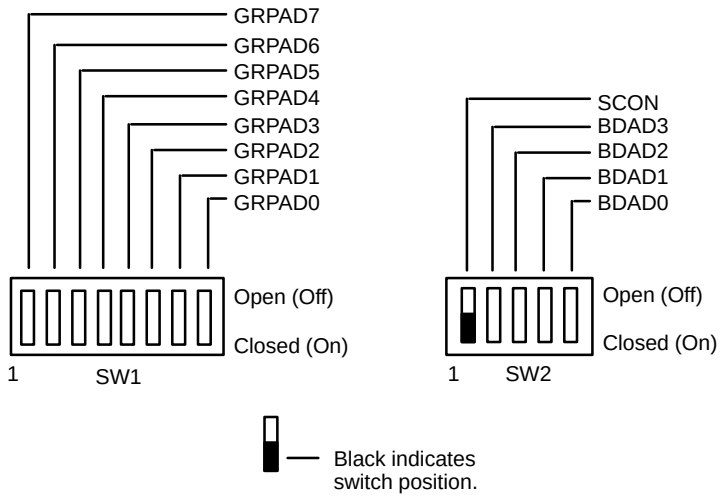


Figure C–2 VIO Board

P1: VMEbus

Table C−4 describes the VMEbus connector P1.

Table C–4 Connector J1: VMEbus

Row A VMEbus		Row B VMEbus		Row C VMEbus	
1	D00	33	$\overline{\text{BBSY}}$	65	D08
2	D01	34	$\overline{\text{BCLR}}$	66	D09
3	D02	35	$\overline{\text{ACFAIL}}$	67	D10
4	D03	36	$\overline{\text{BG0IN}}$	68	D11
5	D04	37	$\overline{\text{BG0OUT}}$	69	D12
6	D05	38	$\overline{\text{BG1IN}}$	70	D13
7	D06	39	$\overline{\text{BG1OUT}}$	71	D14
8	D07	40	$\overline{\text{BG2IN}}$	72	D15
9	GND	41	$\overline{\text{BG2OUT}}$	73	GND
10	SYSCLK	42	$\overline{\text{BG3IN}}$	74	$\overline{\text{SYSFAIL}}$
11	GND	43	$\overline{\text{BG3OUT}}$	75	$\overline{\text{BERR}}$
12	$\overline{\text{DS1}}$	44	$\overline{\text{BR0}}$	76	$\overline{\text{SYSRESET}}$
13	$\overline{\text{DS0}}$	45	$\overline{\text{BR1}}$	77	$\overline{\text{LWORD}}$
14	$\overline{\text{WRITE}}$	46	$\overline{\text{BR2}}$	78	AM5
15	GND	47	$\overline{\text{BR3}}$	79	A23
16	$\overline{\text{DTACK}}$	48	AM0	80	A22
17	GND	49	AM1	81	A21
18	$\overline{\text{AS}}$	50	AM2	82	A20
19	GND	51	AM3	83	A19
20	$\overline{\text{IACK}}$	52	GND	84	A18
21	$\overline{\text{IACKIN}}$	53	Reserved	85	A17
22	$\overline{\text{IACKOUT}}$	54	Reserved	86	A16
23	AM4	55	GND	87	A15
24	A07	56	$\overline{\text{IRQ7}}$	88	A14
25	A06	57	$\overline{\text{IRQ6}}$	89	A13
26	A05	58	$\overline{\text{IRQ5}}$	90	A12
27	A04	59	$\overline{\text{IRQ4}}$	91	A11
28	A03	60	$\overline{\text{IRQ3}}$	92	A10
29	A02	61	$\overline{\text{IRQ2}}$	93	A09
30	A01	62	$\overline{\text{IRQ1}}$	94	A08
31	−12V	63	+5VSTBY	95	+12V
32	+5V	64	+5V	96	+5V

P2: VMEbus and Vbus

Table C−5 describes the VMEbus and Vbus connector P2.

Table C–5 Connector J2: VMEbus and Vbus

Row A Vbus		Row B VMEbus		Row C Vbus	
1	$\overline{\text{BRV\_X0}}$	33	+5V	65	$\overline{\text{BRV\_X1}}$
2	V_VIOST1	34	GND	66	GND
3	$\overline{\text{BRV\_V}}$	35	Reserved	67	$\overline{\text{V\_XBARST1}}$
4	GND	36	A24	68	GND
5	V_VIOST2	37	A25	69	$\overline{\text{MEM\_UTIL\_EN}}$
6	$\overline{\text{V\_XBARST2}}$	38	A26	70	GND
7	V_VIOST3	39	A27	71	$\overline{\text{ANYCPU\_VBG}}$
8	$\overline{\text{V\_XBARST3}}$	40	A28	72	GND
9	GND	41	A29	73	$\overline{\text{X1\_MEM\_INTR}}$
10	GINTR0	42	A30	74	GND
11	GINTR1	43	A31	75	$\overline{\text{X0\_MEM\_INTR}}$
12	GINTR2	44	GND	76	GND
13	GINTR3	45	+5V	77	$\overline{\text{CPU0\_CLK}}$
14	GND	46	D16	78	GND
15	$\overline{\text{V\_BB}}$	47	D17	79	$\overline{\text{CPU1\_CLK}}$
16	$\overline{\text{X1\_BB}}$	48	D18	80	GND
17	V_VIOST	49	D19	81	$\overline{\text{X0\_BB}}$
18	$\overline{\text{V\_XBARST0}}$	50	D20	82	GND
19	GND	51	D21	83	$\overline{\text{CPU2\_CLK}}$
20	V_C0	52	D22	84	GND
21	V_C1	53	D23	85	$\overline{\text{CPU3\_CLK}}$
22	V_C2	54	GND	86	GND
23	V_C3	55	D24	87	$\overline{\text{X0\_B\_CLK}}$
24	GND	56	D25	88	$\overline{\text{X0\_C\_CLK}}$
25	V_C4	57	D26	89	GND
26	V_C5	58	D27	90	$\overline{\text{X1\_B\_CLK}}$
27	V_C6	59	D28	91	$\overline{\text{X1\_C\_CLK}}$
28	$\overline{\text{MEM1\_EN}}$	60	D29	92	GND
29	GND	61	D30	93	$\overline{\text{MEM2\_EN}}$
30	$\overline{\text{FAIL}}$	62	D31	94	$\overline{\text{PEND\_OR}}$
31	$\overline{\text{MARGIN\_SEL}}$	63	GND	95	MARGIN_CLK
32	ABRT	64	+5V	96	$\overline{\text{VBUSY}}$

P3: Vbus

Table C−6 describes the Vbus connector P3.

Table C–6 Connector J3: Vbus

Row A Vbus		Row B Vbus		Row C Vbus	
1	+5V	33	V_AD_00	65	GND
2	+5V	34	V_AD_01	66	GND
3	+5V	35	V_AD_02	67	GND
4	+5V	36	V_AD_03	68	GND
5	+5V	37	V_AD_04	69	GND
6	+5V	38	V_AD_05	70	GND
7	+5V	39	V_AD_06	71	GND
8	+5V	40	V_AD_07	72	GND
9	+5V	41	V_AD_08	73	GND
10	+5V	42	V_AD_09	74	GND
11	+5V	43	V_AD_10	75	GND
12	+5V	44	V_AD_11	76	GND
13	+5V	45	V_AD_12	77	GND
14	+5V	46	V_AD_13	78	GND
15	+5V	47	V_AD_14	79	GND
16	+5V	48	V_AD_15	80	GND
17	+5V	49	V_AD_16	81	GND
18	+5V	50	V_AD_17	82	GND
19	+5V	51	V_AD_18	83	GND
20	+5V	52	V_AD_19	84	GND
21	+5V	53	V_AD_20	85	GND
22	+5V	54	V_AD_21	86	GND
23	+5V	55	V_AD_22	87	GND
24	+5V	56	V_AD_23	88	GND
25	+5V	57	V_AD_24	89	GND
26	–––––	58	V_AD_25	90	–––––
27	–––––	59	V_AD_26	91	–––––
28	–––––	60	V_AD_27	92	–––––
29	–––––	61	V_AD_28	93	–––––
30	–––––	62	V_AD_29	94	–––––
31	–––––	63	V_AD_30	95	–––––
32	–––––	64	V_AD_31	96	–––––

J4: RS–232–C Ports

Table C–7 describes connector J4 of the VIO board.

Table C–7 Connector J4: RS–232–C Ports

Odd Pins		Even Pins		
1	Ground	2		Port A – System Console
3	TXD	4		
5	RXD	6		
7	RTS	8		
9	CTS	10		
11	–––––	12	Mouse power	
13	Ground	14	DTR	
15	DCD	16		
17	–––––	18		
19	–––––	20		
21	–––––	22		
23	–––––	24		
25	–––––			
		26	CG	Port B – Modem or Terminal
27	–––––	28	TXD	
29	–––––	30	RXD	
31	–––––	32	RTS	
33	–––––	34	CTS	
35	–––––	36	DSR	
37	–––––	38	SG	
39	DTR	40	DCD	
41	–––––	42		
43	RI	44		
45	SI	46		
47	–––––	48		
49	–––––	50		

J5: Parallel Port (Centronics)

Table C−8 describes connector J5 of the VIO board.

Table C–8 Connector J5: Parallel Port

Odd Pins		Even Pins	
1	$\overline{\text{DATA\_STROBE}}$	2	GND
3	D0	4	GND
5	D1	6	GND
7	D2	8	GND
9	D3	10	GND
11	D4	12	GND
13	D5	14	GND
15	D6	16	GND
17	D7	18	GND
19	$\overline{\text{ACK}}$	20	GND
21	BUSY	22	–––––
23	–––––	24	–––––
25	SLCT	26	$\overline{\text{RESET}}$
27	GND	28	READY
29	–––––	30	–––––
31	–––––	32	–––––
33	–––––	34	–––––
35	–––––	36	DEMAND
37	ONLINE	38	–––––
39	$\overline{\text{RESET}}$	40	–––––

End of Appendix

Index

Within the index, a range of page numbers indicates the reference spans those pages.

A

B

C

D

E

F

G

I

K

L

M

N

P

R

S

V

W

X

Z

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   A. MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   B. Send your order form with payment to: Data General Corporation
   ATTN: Educational Services/TIPS G155
   4400 Computer Drive
   Westboro, MA 01581–9973

   C. TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   A. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   B. Check or Money Order – Make payable to Data General Corporation. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
Over 200 Items	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$0–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	**SHIP TO:** (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME_____	COMPANY NAME_____
ATTN:_____	ATTN:_____
ADDRESS_____	ADDRESS (NO PO BOXES)_____
CITY_____	CITY_____
STATE_____ ZIP_____	STATE_____ ZIP_____

Priority Code _____ (See label on back of catalog)

_____ _____ _____ _____ _____
Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

	ADD
☐ UPS	
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0–$149.99	0%
$150–$499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	–
SUB TOTAL	
Your local* sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is_____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number | Expiration Date

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508–870–1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation (“DGC”) provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC’s net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC’s sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC’S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC’S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC’s acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Programming System Control and I/O Registers: AViiON® 6280 and 8000-8 Series

014−002170−00

Cut here and insert in binder spine pocket