For more information, please call

United States	**1 800 800 7441**
United Kingdom	**0 800 440 440**
Australia	**008 802 677**

Corporate Office
2011 North Shoreline Boulevard
Mountain View, CA 94043

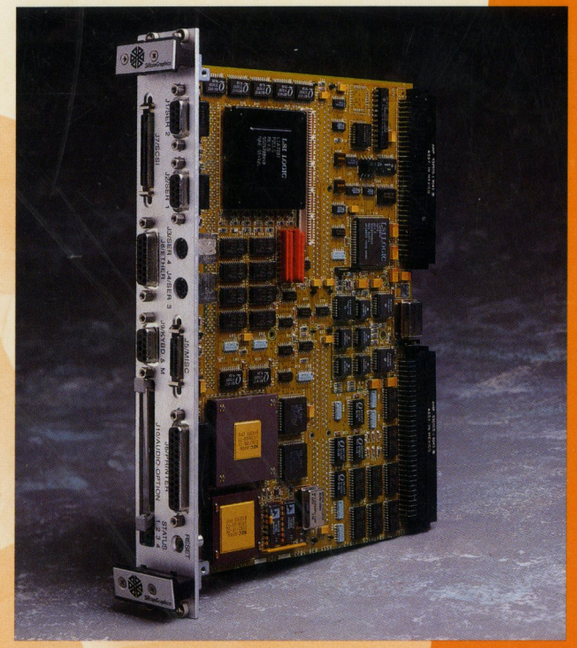
SiliconGraphics
Computer Systems

OSG - V30/35(7/92)

SiliconGraphics
Computer Systems

V30/35

System Integrator's Guide



V30/35 System Integrator's Guide

Document Number 007-5015-020

Contributors

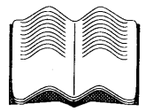
Written by Lawrence Ertel
Illustrated by Keith Granger
Engineering contributions by Bob Abbott, Tore Kellgren, Larry Lewis, Todd Nordland, Edward (Ted) Wilcox, and Ken Williams

V30/35 System Integrator's Guide
Document Number 007-5015-020

Silicon Graphics, Inc.
Mountain View, California

To The Reader

Your V30/35 CPU subsystem came with a standard set of manuals. The set includes the *V30/35 System Integrator's Guide*, the *Personal System Administration Guide*, the *IRIS Software Installation Guide* and the *IRIX Device Driver Programming Guide*. Below you'll find a brief description of each book. For more detailed information on a particular book, please refer to the first chapter in that book.



V30/35 System Integrator's Guide — Explains how to install and maintain your CPU subsystem, install the necessary software, and run interactive diagnostics.

Personal System Administration Guide — Explains how to create and modify a login account, set up your V30/35 as part of a network, back up your work, and more. If you are responsible for managing many workstations or a large network, also see the *IRIX Site Administrator's Guide*.

IRIS Software Installation Guide — Explains how to install Silicon Graphics software products. It covers basic installation procedures, customization techniques, and troubleshooting. Also serves as a complete reference for the installation program, *inst*.

IRIX Device Driver Programming Guide — Explains how to write a device driver for your CPU subsystem.

Contents

Figures

Tables

Chapter 1

Introduction

This guide tells you how to install the V30 and V35 CPU subsystems and the IRIX™ operating system software. It also explains how to operate the interactive diagnostic environment (IDE) hardware diagnostics program. Later chapters provide detailed information about system architecture and the low-level programming interface.

This guide contains the following chapters:

- Chapter 1, "Introduction," explains how to use this guide.
- Chapter 2, "The V30/35 CPU Subsystem," summarizes product features.
- Chapter 3, "Hardware Installation," provides instructions for unpacking and installing the CPU.
- Chapter 4, "Software Installation," provides instructions for installing the IRIX™ operating system.
- Chapter 5, "Diagnostics," provides instructions for operating the IDE diagnostics program.
- Chapter 6, "The VMEbus," gives background information on the VMEbus.
- Chapter 7, "CPU Subsystem Architecture," offers a detailed description of the CPU subsystem.
- Chapter 8, "Low-Level Programming Interface," lists the various registers that provide low-level access to the system.

- Appendix A, "Operating Conditions," gives power requirements of the V30/35 subsystem.
- Appendix B, "Routine Maintenance," provides instructions for changing the clock battery.
- Appendix C, "Pinouts for Peripheral Connectors," outlines the pinouts for all of the V30/35 peripheral connectors.
- Appendix D, "Switches, Jumpers, and Memory Maps," shows you how to locate the V30/35 DIP switches and jumpers and explains how to set them correctly.
- Appendix E, "V30/35 Component Layout," illustrates the front panel and circuit boards used in the V30/35.

Note: For information on installing a graphics subsystem, please refer to the system integrator's guide included with the product.

Conventions

This *V30/35 System Integrator's Guide* uses the following conventions:

- References to other documents are in *italics*.
- References to other chapters and sections within this guide are in quotation marks.
- Step-by-step instructions used to perform a task are shown as numbered sentences. These may be followed by additional instructions, marked with a small box, that further explain the step. For example:

1. Make sure the VMEbus backplane is properly jumpered.
   - If necessary, remove any coverings or enclosures to get access to the rear of the VMEbus backplane. (Some VMEbus backplanes may be jumpered from the front.)
   - With the backplane exposed, replace the single IACK and the four BUS GRANT jumpers from each of the empty slots between the CPU subsystem and your other cards.

- Programming variables are shown in italics.
- Text that appears on screen is shown in a typewriter (courier) font.
- Individual keyboard commands are shown in boldface within angle brackets.
- Keyboard user input is shown in a boldface typewriter (courier) font.

Chapter 2

The V30/35 CPU Subsystem

The V30/35 CPU subsystem provides the processing power of the Silicon Graphics® 4D/30 or 4D/35 Personal IRIS™ workstation for any standard 6U VMEbus card cage with P2 backplane connectors. The V30/35 hardware package consists of a central processing unit (CPU) subsystem which supports multiple optional graphics subsystems. The CPU and graphics subsystems each come prepackaged and ready to install.

The V30/35 is compatible with Silicon Graphics' entire product line of reduced instruction set computing (RISC) systems. It is an ideal solution for applications that require the functionality and performance of a Personal IRIS or IRIS Indigo™ in a VMEbus-based embedded system. The V30/35 is also ideal for hardware configurations that require additional VMEbus expansion.

Like the 4D/30 and 4D/35 workstations, the V30 and V35 CPU subsystems are particularly suited to applications that require intensive visual processing, such as medical imaging, mission planning, and air traffic control. The optional graphics subsystem and on-board I/O controllers use the Silicon Graphics GIO Bus™ to communicate with the CPU, allowing better throughput of data from VMEbus devices.

The V30/35 provides full support for the IRIX operating system (UNIX® System V Release 3 with BSD extensions), as well as the X Window System™ and Motif™ window manager. The Silicon Graphics industry-standard Graphics Library™ can be used to calculate and display three-dimensional objects.

The system can support up to four standard TG - V graphics subsystems with each subsystem monitor displaying multiple windows. The system can also support the Elan/XS - V graphics subsystem; however, it can not support both types of graphics subsystems at the same time.

CPU Subsystem Features

The CPU subsystem assembly consists of two Institute of Electrical and Electronic Engineers (IEEE) Std. 1014-1987 double-height VMEbus EuroCards: a CPU board and an input/output (I/O) interface board. These boards *must* reside in the first two slots as they are serving as the VMEbus system controller (arbiter).

The V35 CPU board features:

- the MIPS® R3000A 32-bit RISC 36 MHz CPU delivering 33 MIPs
- the MIPS R3010A 36 MHz floating point coprocessor delivering 6 MFLOPs

The V30 CPU board features:

- the MIPS R3000A 32-bit RISC 30 MHz CPU delivering 27 MIPs
- the MIPS R3010A 30 MHz floating point coprocessor delivering 4.7 MFLOPs

Both the V30 and V35 CPU boards feature:

- full support for R3000A features including block cache refills and partial cache writes
- a 64 KByte direct mapped instruction cache and a 64 KByte direct mapped, write-through (one-deep write buffer) data cache
- up to 32 MBytes of main memory (using SIMMs), expandable to 96 MBytes with the optional memory expansion board
- one Centronics®-compatible bi-directional parallel port

Both the V30 and V35 I/O interface boards feature:

- a GIO Bus that supports:
  - transfers at sustained rates of up to 100 MB/sec.
  - I/O transfers for up to four TG - V graphics subsystems
  - I/O transfers for a single Elan/XS - V graphics subsystem
- Real Time Clock (RTC)
- four general-purpose RS423 type serial ports with full modem control at a transfer rate of 38.4 Kbaud
- dedicated serial ports for the Personal IRIS keyboard and mouse
- a standard 32-bit VMEbus interface
- a graphics interface (rows P2A and P2C of VMEbus)
- one Ethernet® port with Transmission Control Protocol/Internet Protocol (TCP/IP) support, yielding peak data transfer rates of up to 10 Mbits/sec.
- one single-ended Small Computer System Interface (SCSI) port, with peak data transfer rates of up to 5 MBytes/sec.

Memory Expansion

An optional memory expansion board occupies one additional 6U slot and supports up to 64 MBytes of additional memory in 2 memory banks.

Chapter 3

Hardware Installation

This chapter explains how to install the V30/35 CPU subsystem into the VMEbus card cage. The steps involved include:

1. preparations
2. unpacking
3. installing the CPU subsystem
4. installing additional VMEbus device
5. connecting peripheral devices
6. checking the installation
7. powering up the system

Be sure to read and understand the complete contents of any procedure described in this chapter before attempting it. If something is not clear to you or if you encounter problems, be sure to get help from your support provider.

Note: Your VMEbus system configuration must conform to industry standard VMEbus specifications in order to ensure proper operation of your V30/35 CPU and graphics subsystems.

Service and Support Information

When you purchased your system you may have purchased a support program from either Silicon Graphics, Inc., or a software vendor. Whenever you encounter any problems that you cannot solve using the methods described in this document, contact the organization from which you purchased the support program.

If you did not purchase a support program, contact your sales representative to arrange for support.

Electrostatic Discharge (ESD) Precautions

This equipment is extremely sensitive and may be susceptible to damage caused by Electrostatic Discharge (ESD). ESD is an electrical discharge (spark) caused by the build-up of electrical charge on clothing and other materials.

You must use proper ESD preventative measures:

- Connect a ground strap to your wrist and to the metal frame of the VMEbus card cage.
- You and all the electrical equipment that you handle during this installation must be at ground potential to avoid damage from ESD.
- Keep the boards in the antistatic bags provided.
- Remove a board from its antistatic bag only when you are properly grounded to the card cage with a ground strap.
- Do not use an ohmmeter on the boards.
- The card cage should be powered down and grounded until the installation is complete and ready for testing. The AC power cord provides a ground for the card cage and should be left plugged in unless some other ground is provided.

Electromagnetic Interference (EMI) Compliance

To make this system comply with FCC, VDE, and VCCI emission requirements, follow these guidelines:

- Ensure that the VMEbus chassis and power supply meet FCC, CISPR-22, and VDE specifications.
- Securely tighten screws holding the VMEbus cards.
- Install blanking panels over unused slots.
- Connect all peripheral devices with shielded cables.
- Ensure the internal and external unshielded cables that connect to Input/Output connectors have EMI suppression components such as ferrite cable, cable clamps, filters, etc.

General Guidelines for Installing Hardware

Remember the following guidelines whenever you install or handle the electronic equipment described in this guide:

- In addition to electrostatic discharge, this equipment is also susceptible to damage from physical impact, exposure to excessive temperatures, moisture, solvents, and other conditions known to damage electronic equipment. Please handle with care!
- This equipment has been carefully designed and manufactured to fit properly into the VMEbus card cage. While moderate force is necessary to seat a subsystem in the card cage, you should not have to struggle. If you experience any difficulty whatsoever, remove the subsystem from the card cage and inspect both subsystem and backplane for bent connector pins, obstructions, or other damage. *Do not apply impact or excessive force to seat a subsystem into the card cage; never use tools or other implements to seat a board into a card cage.*
- Use only AC outlets that are properly grounded.

- Make sure that the power supply is adequate for the number of slots being used; typically 30W at 5V for each slot.
- Make sure the card cage has adequate air flow for cooling purposes; use covers and airflow restrictors on all empty slots.

Warning: **Never expose the card cage power supply. Power supplies generate high voltages, *even when not plugged in!* Contact with an exposed or improperly grounded power supply can deliver a severe electrical shock. The power supply must remain grounded and shielded. If you are not certain which coverings prevent access to the power supply, refer to the documentation for your card cage before removing any of its coverings.**

Preparations

Both CPU and graphics subsystems arrive assembled and ready to install. Before you begin, make sure that you have the proper items on hand and that the operating conditions described in Appendix B, "Operating Conditions" have been met.

Before removing the board assemblies from their packaging, make sure that the ambient temperature is between 0°C and 55°C. The work area should be free of excessive dust, airborne fumes, solvents or other possible contaminants, and any other conditions that might result in damage to electronic equipment.

In addition to this guide, you will need the items listed below to install this equipment successfully:

- an electrostatic discharge grounding strap for each person who may need to handle the PC-board assemblies
- a small standard (flat-head) screwdriver
- any other tools needed to remove covers or enclosures surrounding your VMEbus card cage

Note: Before installing the CPU or graphics subsystem, check the ground pins on the P1 and P2 connectors of the slots in which you intend to place it. These pins must be present to ground the subsystem properly.

Power Supply

Be sure your power supply is adequate to support the number of slots in your backplane.

The CPU and subsystem consists of two boards and the graphics subsystems each consist of between two and four boards, depending on the configuration. When installing multiple subsystems, it is especially important to be sure that the power supply is adequate for the number of slots taken up. The VMEbus standard of 30 watts (6 amps at 5 volts) per slot is sufficient. (Refer to Appendix A, "Operating Conditions," for information on power requirements for this CPU.)

Unpacking

While unpacking your V30/35, remember to inspect each assembly and any accompanying materials for damage. When it is not in the card cage, place the new equipment on a clean, stable surface that is safely insulated from electrostatic discharge. If possible, leave the V30/35 board assemblies in their conductive plastic wrappings when they are not in the card cage.

Your packages should contain:

- the CPU subsystem as shown in Figure 3-1
- a single CD-ROM or several installation tapes

Depending on the options you ordered, the installations tapes include the following:

- Execution Only Environment tape, Part 1 (*eoe1*)
- Execution Only Environment tape, Part 2 (*eoe2*)
- optional Development Environment tape (*dev*)

There may also be an additional tape or CD-ROM accompanied by an errata sheet explaining its usage.

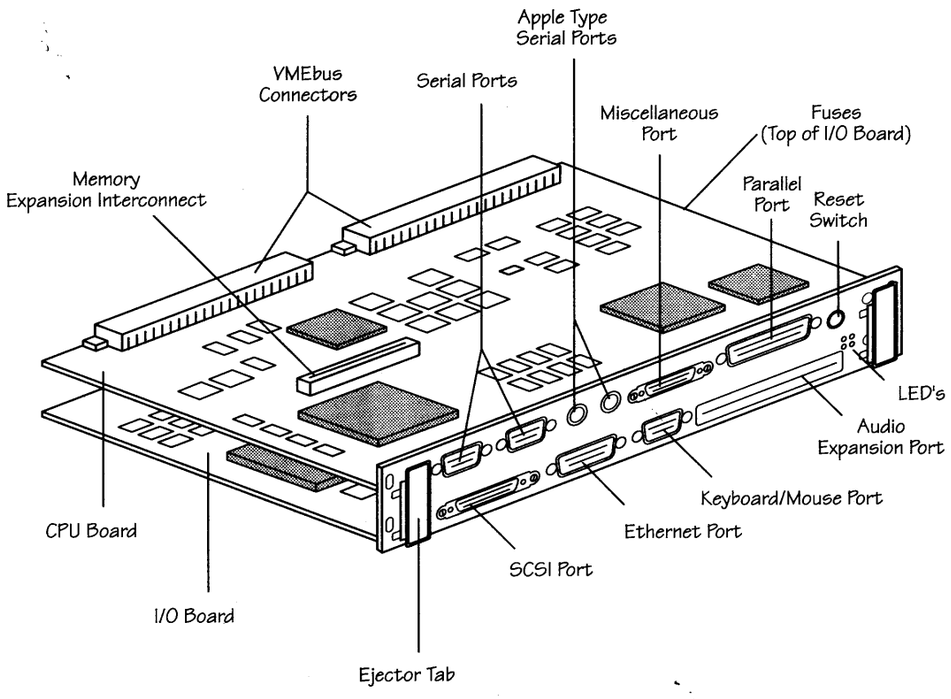


Figure 3-1 The CPU Subsystem

Note: Your V30/35 may include an optional graphics subsystem in a separate package. Instructions on the installation and operation of the graphics subsystem can be found in the system integrator's guide included with the product.

Installing the CPU Subsystem

The CPU subsystem occupies two slots on the VMEbus.

To install the CPU subsystem:

1. Set the switches and jumpers to configure the CPU operation and VMEbus-to-local memory maps.

   The V30/35 CPU subsystem *must* be used as the VMEbus system controller and is set up for this purpose by default with the appropriate mappings for the IRIX operating system. See Appendix D, "Switches, Jumpers, and Memory Maps," for additional information.

2. Verify that the CPU subsystem is configured for your graphics subsystem (if any) with the appropriate oscillator and jumpers.

   For information on the proper configurations, see Appendix D, "Switches, Jumpers, and Memory Maps."

3. Insert the CPU subsystem into the card cage.

   - Remove any coverings or enclosures to access the front opening of the VMEbus card cage. If you are also installing the optional graphics assembly, remove additional coverings or enclosures as needed to get access to the rear of the VMEbus backplane.
   - The CPU must be used as the VMEbus system controller. Remove any boards from the leftmost slots as the CPU subsystem must reside in those slots.
   - Orient the CPU subsystem so that the component side (top) faces toward the right with respect to the front panel. Align each board in its respective slot. (See Figure 3-2.)

- Pull back the captive screws located on the front panel near the ejector tabs so that the screws do not get caught when you slide in the subsystem.

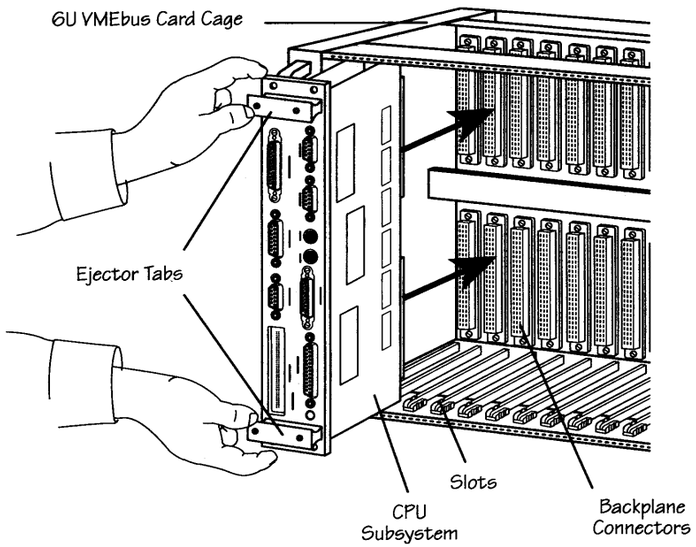


Figure 3-2 Installing the CPU Subsystem

- Slide the subsystem back along the slots and seat it into the backplane with a firm, steady pressure. The subsystem should slide smoothly along the slots and fit snugly into the backplane connectors.
- Secure the subsystem to the card cage using the captive screws located near the extractor tabs at the top and bottom of the faceplate.

4. Make sure the VMEbus backplane is properly jumpered. (See sample configuration in Figure 3-3)
   - If necessary, remove any coverings or enclosures to get access to the rear of the VMEbus backplane. (Some VMEbus backplanes may be jumpered from the front.)
   - With the backplane exposed, replace the single IACK and the four BUS GRANT jumpers from each of the empty slots between the CPU subsystem and your other cards.

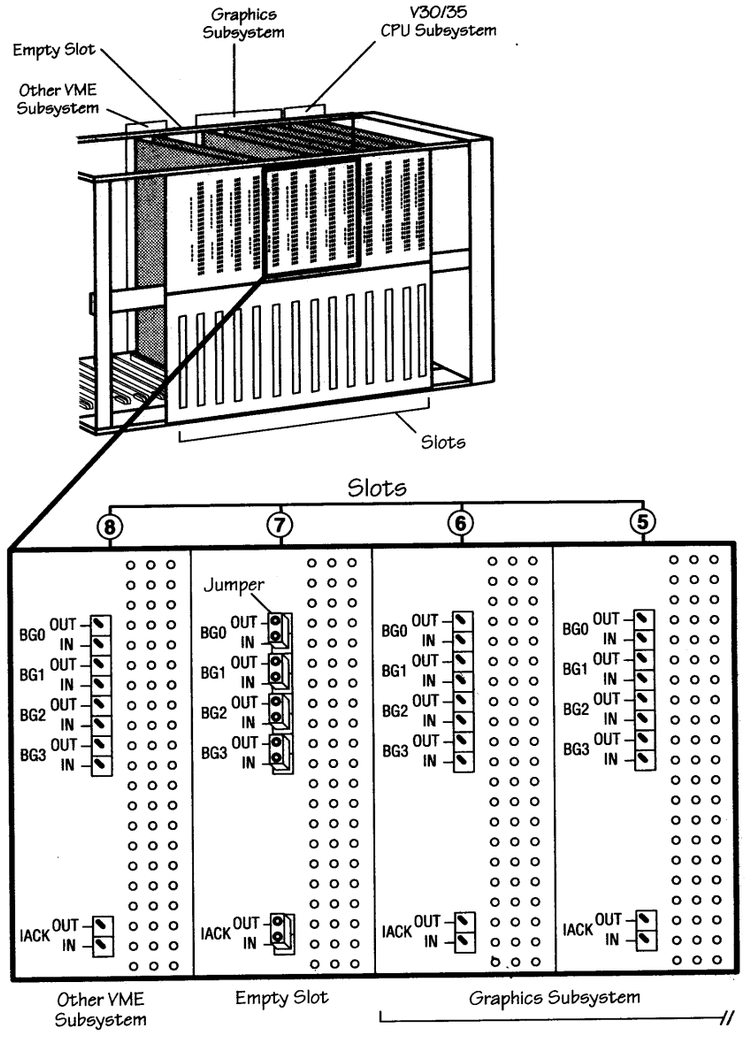


Figure 3-3 Configuring the VMEbus Backplane

Installing the Graphics Subsystem

Keep the backside VMEbus backplane accessible if you are going to install a graphics subsystem.

The graphics subsystem should be installed when the installation of the CPU subsystem is completed and before you load the operating software. Please refer to the system integrator's guide included with your graphics subsystem for additional information.

Installing Additional VMEbus Devices

At this time you can install any additional VMEbus-resident devices by following the manufacturer's instructions.

Installing Airflow Restrictors

Be sure to properly install airflow restrictors in any empty VMEbus slot after you have installed all your VMEbus devices. The airflow restrictors are needed to maintain a sufficient airflow within the card cage.

Connecting Peripheral Devices

With the power to the card cage still off, you are now ready to connect peripheral devices to your V30/35. In order to install the software, you need a SCSI hard disk and a CD-ROM device or a SCSI tape drive.

If you have already installed a graphics subsystem, you also need the following devices to install and boot the IRIX operating system:

- a Silicon Graphics graphics keyboard and mouse
- a high-resolution 1280 by 1024 monitor

If you have not installed a graphics subsystem, you need an ASCII terminal connected to serial port 1 for the power-on tests.

At this time you can also connect the Ethernet and a printer or other Centronics-compatible device. (The Ethernet port may be used to load the software across a network.)

The proper ports and connectors for these peripherals are shown in Figure 3-4. For pinout information, see Appendix C, "Pinouts for Peripheral Connectors."

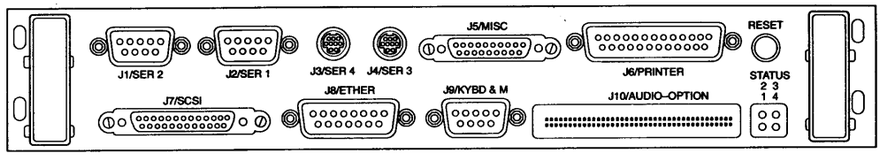


Figure 3-4 Connectors for Peripheral Devices

Supported Devices

The V30/35 works with any Personal IRIS workstation peripheral device sold by Silicon Graphics. SCSI devices not provided by Silicon Graphics as part of its standard product line may require a new driver and are not guaranteed to work with the V30/35.

The Table 3-1 outlines the peripheral devices used by Silicon Graphics with the V30/35.

Note: Silicon Graphics supports only those devices purchased directly from Silicon Graphics. For further information, contact your sales representative. If you have purchased a support program from another vendor, please contact that organization for information on supported peripheral devices.

Mfgr.	Model	Model #	Mfgr. Part #	Description
Hard Drives (each require a minimum of 25 watts)				
Seagate	WREN 6	ST2383N	933003-039	380MB half-height
Seagate	WREN Runner 2	ST4767N	941002-030	760MB
Seagate	WREN 7	ST41200N	939001-040	1200MB
Floppy Drives (each require a minimum of 10 watts)				
Teac			19307393-07	3.5 inch half-height
Tape Drives (each require a minimum of 15 watts)				
Archive	2150S		22300-030	150MB internal half-height
Archive	2150E		21525-125	150MB external
Archive	4320NT		27245-006	1300MB DAT half-height

Table 3-1 Silicon Graphics Peripheral Devices

Checking the Installation

When all of the desired boards are installed, you are ready to inspect the card cage. After verifying that the installation is correct, you can replace any coverings or enclosures to secure the cage. (Depending on the physical layout of your VMEbus card cage, you may need to replace covers or enclosures before connecting your peripheral devices.)

When inspecting the card cage, ensure that:

- Assemblies and boards are seated properly and secured.
- Unoccupied slots between VMEbus cards are correctly jumpered.
- For each graphics subsystem, the GE board is connected to the appropriate CPU subsystem.
- Connections and jumpers are tight and solid.
- Peripheral connectors are securely attached.
- CPU jumpers and switches are properly configured for your environment.
- Air flow inhibitors are properly installed (where appropriate) in unoccupied slots.
- You follow the guidelines for maintaining EMI compliance as outlined under the heading "Electromagnetic Interference (EMI) Compliance" earlier in this chapter.

Once you verify that the hardware is properly installed, you can power up the VMEbus card cage.

Powering Up the System

Connect the card cage and any peripherals that require it to a grounded source of AC power, and switch on the card cage to power up the system.

When the system first comes up, the CPU performs a number of diagnostic tests on itself. The CPU diagnostics are displayed on the ASCII terminal if the *bootmode* remains on its default value, *bootmode=d*.

The CPU diagnostics are not displayed if the *bootmode* has been reset so that *bootmode=c*.

After the CPU diagnostic tests are completed, a series of graphics subsystem tests are run if a graphics subsystem is present. The results of the graphics diagnostic tests appear on an attached graphics monitor if the *console* value remains on its default value, *console=g*.

If no graphics monitor is attached, the results of the graphics test appear on the ASCII terminal. The test results also appear only on the ASCII terminal if the console value has been reset so that *console=d*.

Note: See "Preparing the Standalone Environment" in Chapter 4 for information on setting the *console* and *bootmode* variables.

These tests take a minute or two to complete. In case of error or failure, contact your support representative or the organization from which you purchased your support program. If you have not purchased a support program, contact your sales representative to arrange for support. Do not attempt to correct or repair any boards or components. Refer to Chapter 4, "Diagnostics" for an explanation of the power-on self-tests.

When the self-test completes successfully, you see the following message on the system console:

```
Starting up the system ...

To perform system maintenance instead, press <Esc>
```

Press **<Esc>** to bring up the System Maintenance menu. At this point, the hardware installation is complete; you are ready to install software.

Chapter 4

Installing System Software

This chapter explains how to install system software onto the V30/35 CPU subsystem directly from tape or CD-ROM. The procedure takes about an hour to complete.

Standard software for the V30/35 includes the IRIX Execution Only Environment and the optional IRIS Development Option (ido). The installation process involves loading the system software onto disk using the standalone *inst*(1M) program. For additional background information about the *inst* program and the installation process in general, see the *IRIS Software Installation Guide*.

Alternatively, you can install system software over a network using a tape drive or CD-ROM attached to a remote host. For multiple installations, you can copy the software onto a remote host's disk using *distcp*(1M), and then install it over the network. See the *Personal System Administration Guide* for information on installing software over a network.

Installing Additional Software

You may want to install additional software that you receive from Silicon Graphics or another company that writes application software or device drivers. Always follow the instructions that come with the software to make sure you install new software correctly.

Preparations

To install the IRIX system software from tape or CD-ROM you need:

- this guide
- *IRIS Software Installation Guide*
- a tape drive or CD-ROM device that can be connected to the V30/35
- a partitioned disk drive that holds at least 380 MBytes of data (the drive is already partitioned when you receive it from Silicon Graphics)

  For information on formatting and partitioning disk drives see the *IRIX Site Administrator's Guide*.
- the Execution Only Environment (eoe) CD-ROM or tapes (1 and 2). The CD-ROM contains the entire eoe

  If you are using tapes, tape 1 includes the IDE diagnostics program, the miniroot, and the first portion of the standard system software release; tape 2 includes the remainder of the standard release.
- the (optional) IRIS Development Option (ido) CD-ROM or tape that holds the standard software development tools

Note: You may have received a disk drive that already has the eoe system software installed. If so, you need to install only the optional software before attaching the drive and booting the V30/35 subsystem.

Preparing the Hardware

Before attempting to install system software, make sure that system hardware:

- is properly installed in the VMEbus card cage (as described in Chapter 2)
- passes the power-on self-tests

Figure 4-1 shows one possible system configuration using external SCSI disk and tape drives. Remember to power down the card cage before connecting or disconnecting the drives.

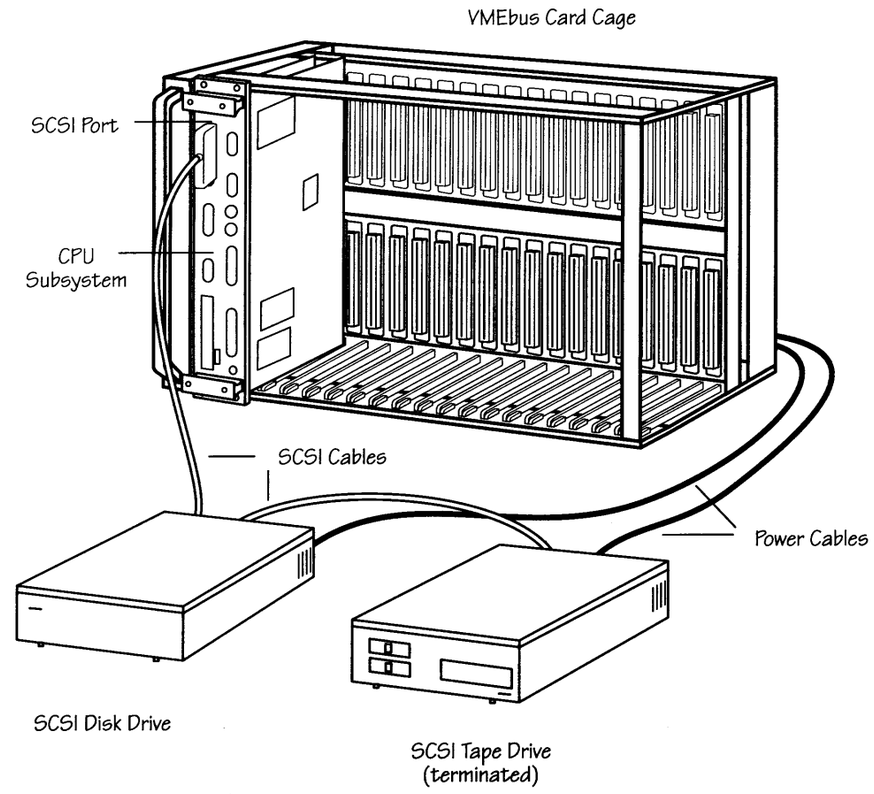


Figure 4-1 Installing Software with External SCSI Tape and Disk Drives

Preparing the Standalone Environment

With the system powered up, make sure that the standalone environment is set up properly. The system gives the message:

```
Starting up the system ...

To perform system maintenance instead, press <Esc>
```

When the message appears, do the following:

1. Press the **<Esc>** key to bring up the System Maintenance menu. The system then displays:

```
System Maintenance Menu

1 Start system

2 Install System Software

3 Run Diagnostics

4 Recover System

5 Enter Command Monitor


Option?
```

2. Enter the command monitor by choosing option 5. The command monitor issues the >> prompt.
3. Make sure that the correct standalone environment is present by giving the *printenv* command. You should see several standalone environment variables listed, including the following:

```
...
bootfile=dksc(0,1,8)sash
...
console=g
...
bootmode=d
...
```


4. The *bootfile* variable determines the device from which the system is to boot. In order to install system software using a local tape drive or CD-ROM device, this variable must have the value shown. If the *bootfile* variable is not correct, use the *setenv* command to assign the correct value to this variable:

```
setenv bootfile dksc(0,1,8)sash
```

Note: For additional information on the *bootfile* variable and other issues pertaining to the PROM monitor see the *IRIX Site Administrator's Guide*.

5. The *console* environment variable indicates that the system console is either the graphics monitor (g or G) or an ASCII terminal (d); this means that the console is attached to the graphics (g) port or the diagnostics (d) port. The graphics monitor is the default setting; however, if you do not have a graphics subsystem in operation, the software automatically switches to an ASCII terminal.

   If you wish to manually set the console variable, type:

```
setenv console g
```

   for a graphics monitor, or type

```
setenv console d
```

   for an ASCII terminal.

   Note: The upper-case graphics monitor variable *G* displays a Silicon Graphics logo in the upper left corner of the screen, the lower-case *g* does not; this is the only difference between the two.

6. The *bootmode* variable determines whether the CPU diagnostics are displayed when the system is booted. The default variable, *bootmode=d*, displays the diagnostics. You can manually set this variable by typing:

```
setenv bootmode d
```

   If you do not want the diagnostics displayed, type:

```
setenv bootmode c
```

7. When these variables are set to the correct values, type `init` to return to the System Maintenance menu. You are now ready to begin installing system software.

Installing the System Software

You can install the system software from a local drive attached directly to the V30/35 subsystem or from a remote drive over a network.

To install the system software:

1. Insert the eoe1 tape into the tape drive, or the CD-ROM into the CD-ROM drive.
2. Choose option 2, "Install System Software," from the System Maintenance menu. When you do, the system displays:

```
Installing System Software ...
Press <Esc> to return to the menu.

Insert the installation tape, then press <enter>:
```

3. Press the **<Enter>** key to load the miniroot and *inst* program. This takes about 10 minutes. When *inst* is installed and ready, the system displays:

```
Inst Main Menu

1. from [source]
2. list [keywords]
3. go
4. install [keywords] [names]
5. remove [keywords] [names]
6. keep [keywords] [names]
7. step [keywords] [names]
8. versions [args]
9. help [topic]
10. admin
11. quit

Inst>
```

4. Choose option 3 to automatically install the entire eoe1 distribution. If you are installing from CD-ROM, skip steps 5 and 6.

5. If you are installing from tape, in about 20 minutes you are asked:

```
Is there more software to install?
```

Answer **y** to this question to proceed with the next tape.

6. You next see the prompt:

```
Insert the next tape, then press <enter>:
```

Replace the eoe1 tape with the eoe2 tape and press **<Enter>**. This tape takes about 15 minutes to install. Repeat this process for the dev tape (about 5 minutes).

7. When you are done, answer **n** when *inst* asks:

```
Is there more software to install?
```

When you answer **n** to this question, *inst* may ask if it should automatically reconfigure the kernel. Answer **y** if so prompted.

The software is now installed.

8. Answer **y** when *inst* asks if you want to start the operating system. This reboots the system.

For additional information on installing the system software, please refer to the *IRIS Software installation Guide*.

Chapter 5

Diagnostics

The V30/35 provides extensive support for testing both CPU and graphics subsystems. The system comes with two sets of diagnostics:

- a basic set of power-on tests ensures that the system can bootstrap and run standalone programs
- the interactive diagnostics environment (IDE), a standalone program that runs in the PROM monitor through the standalone shell (*sash*)

This chapter describes both sets of diagnostics and the various tests they perform. If you need general information about the PROM monitor and the standalone shell, please refer to the *Personal System Administration Guide* that came with your system.

Power-on Tests

Power-on diagnostics are performed automatically whenever the system is powered up. The "Fault" (yellow) light-emitting diode (LED) flashes while these tests are under way. The LED's located on the front panel of the V30/35 are illustrated in Figure 5-1.

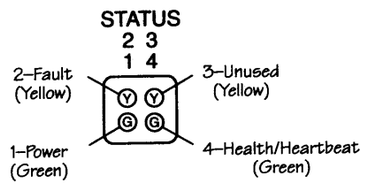


Figure 5-1 Front Panel LED's

The power-on diagnostics are:

- interrupt mask registers (INT2) test
- data and instruction caches test
- SCSI controller 0/devices test
- Duart channel 1 internal loopback test

Output from these tests is displayed only when an ASCII terminal is used as the system console.

Test Failure

When a power-on test fails, the PROM monitor may not come up, or it may come up displaying an error message.

If the memory test detects a bad SIMM module, the failure message displays an IDE alphanumeric symbol to indicate the specific byte that needs replacing as well as the SIMM bank in which it's located. The SIMM bank can be located on either the CPU board or the optional Memory Expansion Board.

Table 5-1 outlines the IDE symbols and the physical location of the byte they represent.

IDE Symbol	Bank	Byte	Board
A1	bank 0	byte 3	CPU board
A2	bank 0	byte 2	CPU board
A3	bank 0	byte 1	CPU board
A4	bank 0	byte 0	CPU board
B1	bank 1	byte 3	memory expansion board
B2	bank 1	byte 2	memory expansion board
B3	bank 1	byte 1	memory expansion board
B4	bank 1	byte 0	memory expansion board
C1	bank 2	byte 3	memory expansion board
C2	bank 2	byte 2	memory expansion board
C3	bank 2	byte 1	memory expansion board
C4	bank 2	byte 0	memory expansion board

Table 5-1 IDE SIMM Failure Symbols.

To locate the correct SIMM module for replacement on the CPU board, refer to Appendix E, "V30/35 Component Layout." To locate the SIMM modules on the optional Memory Expansion Board, please refer to the documentation that came with the memory upgrade.

Using IDE Diagnostics

The interactive diagnostics environment includes comprehensive tests for all of the I/O interface controllers and ports, the graphics subsystem (subsystem 0 only) and its various components, main memory, and other system components.

You can use the IDE program to verify that your system hardware is performing properly or to isolate hardware malfunctions. The IDE program must be loaded into the monitor from disk, from tape, or over the network before you can run it. If you have installed the operating system as described in Chapter 4, the IDE program is on disk and ready to run.

Starting IDE in Terse or Verbose Mode

You can run the IDE program in either terse or verbose mode. To start the diagnostics program in terse mode select option 3, "Run Diagnostics," from the System Maintenance menu. While in terse mode, the program first reports on the hardware configuration and then runs silently, displaying error messages only when a diagnostic fails on an installed board or component.

When IDE runs in terse mode, you cannot interrupt the test suite. You must let it run to completion or reset the system.

You must run IDE in verbose mode if you wish to use it interactively. In verbose mode, you can interrupt the standard test suite at any time by pressing `<Ctrl-C>`; the program then issues the `ide>>` prompt and accepts the commands described in the next section, "IDE Interactive Commands."

To start IDE in verbose mode:

1. Enter the command monitor by selecting option 5 on the System Maintenance menu.
2. Type `ide fe` in response to the command monitor's >> prompt.

In verbose mode, the IDE program reports on each test it attempts, along with the success or failure of that test.

IDE Standard Tests

In either terse or verbose mode, IDE first performs an initial inventory of subsystems that are present (the equivalent of the *hinv* PROM monitor command). The initial output you see looks something like this:

```
Memory size:xx MBytes
Instruction cache size:xxxxx
Data cache size:xxxxx
CPU board:xxxxx
System options:xxxxx
...
Graphics: xxxxx
SCSI disk:dksc(0,1)
```

Once the program starts, it runs a standard test suite. These tests cover all major subsystems and components, taking about 25 minutes to complete.

The standard test suite has two parts. The first part tests the CPU subsystem. The second part tests the graphics subsystem if one is present. Some tests apply only to specific models or options; if IDE attempts to access a piece of hardware that is not supported on this system, IDE reports the device as missing and continues with the next test.

Note: Certain tests take over the graphics display monitor. When these tests are running, the graphics monitor may go black or display various shapes and patterns. This is no cause for alarm. When the tests finish running, the monitor resumes the console display.

CPU tests can fail with any of the following errors. If IDE reports one of these errors, contact your support provider.

```
Failure detected on CPU board.
Failure detected on FPU.
Failure detected on VME device.
Failure detected on SCSI device[s].
```

IDE Interactive Commands

The interactive commands available under IDE allow you to check the operation of the boards and components that make up the system.

The IDE interactive commands are classified by subsystem. Those that pertain to the CPU subsystem are listed below. The IDE commands for the graphics subsystem are listed in the integrator's guide that came with it.

General Commands

The following general commands are used to modify more specific commands pertaining to either the CPU or graphic subsystem:

{cmd; cmd ... }
Command may be grouped between { }.

? Provides on-line help for each IDE command.

exit Exit IDE.

for *[expr1; expr2; expr3] cmd*
Repeat *cmd* while *expr2* is true; similar to the *for* statement in the C programming language.

if *[expr] cmd1 [else cmd2] [fi]*
Execute *cmd1* if *expr* is true, *cmd2* otherwise; *fi* is optional.

repeat *n cm*d
Repeat *cmd* n times.

report *[n]* Set verbose level, where *n* represents a scale of one to four, with four being the most verbose.

wait [*"message"*]
Display the indicated message, if any, and wait for an `<Enter>` to be typed in on the keyboard.

while *[expr] cmd*
Repeat *cmd* while *expr* is true.

CPU Subsystem Tests

audio DSP audio diagnostics (run only if the audio option is present).

buffoff Turn off message buffering.

Messages are not displayed when running IDE on a graphics terminal. *buffon* stores the messages which are then displayed when IDE is completed and the graphics terminal is automatically reset as a console window; otherwise, messages are discarded.

buffon Turn on message buffering.

cache2 Instruction/data caches data test.

cache3 Instruction/data caches tag test.

cache4 Instruction/data caches random refill test.

cdsio1 CDSIO data loopback test.

cdsio2 CDSIO interrupt test.

checksum *range*
Calculate checksum for the specified range of addresses. A range can be specified in one of two forms:

start:end	starting address up to but not including the ending address
start#count	starting address up through count words

Addresses can be specified in either hexadecimal or decimal, with the following scaling abbreviations:

K	1024 bytes
P	4096 bytes (page)
M	1048576 bytes (10242 bytes)

clock Real-time clock counters carry test.

cpudsp_sram
CPU DSP SRAM data test (runs only if audio option is present).

dram [**–p**] [*range*]
Walking bit and address uniqueness tests:

–p	Perform parity test instead
range	Same as for checksum command; must be greater than 4MB

dsp56 DSP560001 diagnostics.

duart [–ie*X*]
DUART internal/external loopback test:

X	# of DUART port tested (0 – 6)
–i	Internal loopback (the default)
–e	External loopback

emfail Produce "Electronics Module" failure message.

enet1 Ethernet data/address port register test.

enet2 [**–ie**] Ethernet internal/external loopback test:

–i	Internal loopback (the default)
–e	External loopback

finflgs Test the DK3 finish flags.

fpu Run floating-point unit diagnostics.

help_mem Print memory diagram.

hinv Hardware inventory (same as *hinv* PROM monitor command).

hpc1 HPC1 registers and fifos.

igallop [**–R** | **–W**] [**–c** | **–r**]
Instruction cache galloping column/row test:

–R	Read only
–W	Read-write
–c	Column test
–r	Row test

initgr Initialize graphics board (includes microcode downloading - runs only if graphics subsystem is present).

int2 Init2 interrupt mask registers.

ldram [**–p**]
Lower (0-4Mbyte) DRAM walking bit and address uniqueness tests:

–p	Perform parity test instead.

led 0 | 1 | 2 Turn health LED on or off.

memtest CPU memory test.

nvram1 EEPROM address test.

nvram3 EEPROM data test.

nvram4 EEPROM ID test.

nvram5 EEPROM pin signals test.

parity Parity test.

print1 Centronics printer connection test.

print2 Centronics printer interface DMA test.

print3 Centronics printer loop back test.

resetcons
Reset graphics console (runs only if graphics subsystem is present).

scope [memory | icache | dchace | scsi | enet | rtc | duart0 | duart1 | duart2]
CPU scope loop.

sdma Stride DMA test.

scsi Test SCSI interface controller.

test *subcmd*

Memory tests; *subcmd* is one of:

data [–bhwiuf] *range*
Walking bit test

parity [–bhwiuf] *range*
Parity test

addr [–wiuf] *range*
Address uniqueness test

wire [–gmnpv] index page frame
Set up address mapping for addresses in the ranges 0x0–0x7fffffff and 0xc0000000–0x0fffffff

Arguments to the data, parity, and addr subcommands are:

–b	Load and store bytes.
–h	Load and store halfwords.
–w	Load and store words (the default). (The –b, –h, and –w options do not apply to address ranges for other commands such as checksum, dram, or ldram.)
–i	Invert the binary sense of the test; use walking zero instead of walking one; use complemented address uniqueness instead of uncomplemented
–u	Repeat test until an error occurs
–f	Continue repeating (forever), despite errors

range a range of addresses taking one of two forms:

start_addr:end_addr
(Specifies a range from start_addr through,but not including, end_addr.)

start_addr#count
Specifies a range from start_addr through that address plus the number of units specified by count; unit size is determined by **–b, —h, or –w.**

An address can be specified in either hexadecimal or decimal, with the following scaling abbreviations:

K	1024 units
P	4096 units (page)
M	10242 units

For instance, the command:

```
test addr 0xa01M#0x1M
```

performs an address uniqueness test over the one megaword (4Mbyte) range starting at address 0xa0100000

Arguments to the **wire** subcommand are:

–g	Global mapping (don't compare TLBPIDs).
–m	Allow writes at virtual addresses given by *page*.
–n	Do not look in cache after translating an address.
–p	Purge page map of entries.
–v	Mark map entry as valid for address translations.
index	A map entry index between 0 and 63.
page	A page number (five hexadecimal digits).
frame	A page frame number (five hex. digits) into which addresses beginning with page's five digits are to be translated.

For instance, the following commands set up a walking bit test on two pages of mapped memory using pages 0 and 1:

```
test wire -pvm 0 0 0x18
test wire -vm 1 10x19
test data -b 0#2p
```

timer Timer frequency test.

tlb Translation lookaside buffer (TLB) tests.

utlb UTLB miss exception test.

vme1 VME interrupt test using ESDI disk controller.

vme2 VME DMA test using ESDI disk controller.

xscsi Exercise all SCSI disk drives found.

Chapter 6

The VMEbus

The VMEbus is an asynchronous bus supporting a large variety of computer peripherals and CPUs. Originally designed in Europe as a primary computer bus for the Motorola MC68000 series microprocessor, the VMEbus is used today as a peripheral bus by many manufacturers, including Silicon Graphics.

The V30/35 uses the VMEbus as a peripheral bus; however, not all *modes* of the VMEbus are supported.

This chapter provides an overview of VMEbus definitions and functionality. It is not intended to be used as a VMEbus specification, but rather as a source of general information about the VMEbus and its capabilities.

Note: VMEbus functionality supported by the V30/35 is described in Chapter 7, "CPU Subsystem Architecture."

VMEbus History

With the introduction of the MC68000 family of processors in the early 1980's, Motorola introduced the VERSAbus which matched the different address and data modes of the MC68000. However, the VERSAbus suffered from some important design flaws: the card sizes and connectors to the bus were very large and awkward.

Meanwhile, Motorola offices in Germany had developed a different bus for the MC68000. The sizes of the cards, form factors, and the bus connectors were considered better than those designed for the VERSAbus. The form factor from Europe (EuroCard) was married with the electrical specifications of the VERSAbus and the VMEbus (Versa Module Eurocard) was born.

Since that time the VMEbus has had two revisions and is now at Revision C. The IEEE has given the number of IEEE-P1404-1987 for the VMEbus. The IEC number is IEC-821BUS.

Note: As the *V30/35 System Integrator's Guide* goes to print Revision D of the VMEbus is anticipated but has not yet been released. Several new capabilities expected in Revision D will be referred to in this chapter.

VMEbus Signals

The signals of the VMEbus can be divided into 4 groups:

- data transfer bus (DTB)
- arbitration of the DTB
- interrupt bus
- utility bus

The DTB performs data transfers and defines the protocols necessary to deal with access timing, address space, and data size considerations. The arbitration of the DTB is based on a MASTER/SLAVE concept and supports several bus scheduling algorithms. The interrupt bus is prioritized into seven levels. The utility bus handles power and control lines such as ACFAIL and SYSRESET.

Figure 6-1 illustrates the VMEbus signal pathways.

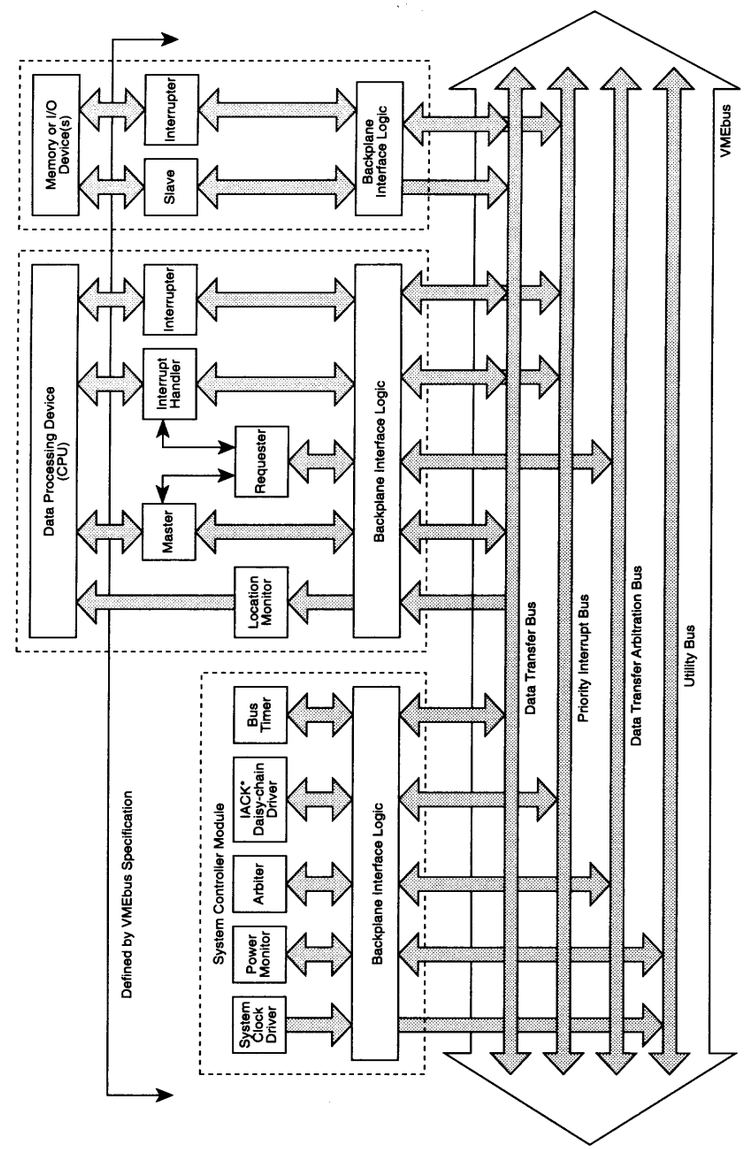


Figure 6-1 VMEbus Block Diagram

DTB Bus

The VMEbus uses a master/slave concept. The MASTER initiates data transfers. Only the MASTER actually moves data across the bus. The SLAVE is passive and responds to the MASTER.

The DTB is a non-multiplexed bus consisting of:

- 24 or 32 address lines
- 16 or 32 data lines
- two data strobe lines
- an address strobe line
- a write line
- a long word line
- a data transfer acknowledge line

The DTB supports three address widths: A32, A24, and A16. The address space sizes are 4 GBytes, 16 MBytes and 64 KBytes respectively.

These address spaces are referred to as "extended" for A32, "standard" for A24, and "short" for A16. The DTB also uses six address modifier lines which determine the access type.

Note: Rev D of the VMEbus supports A64 using data lines.

Modifier Code	Function
0x3f	A24 supervisory block transfer
0x3e	A24 supervisory program access
0x3d	A24 supervisory data access
0x3b	A24 non-privileged block transfer
0x3a	A24 non-privileged program access
0x39	A24 non-privileged data access
0x2d	A16 supervisory access
0x29	A16 non-privileged access
0x0f	A32 supervisory block transfer
0x0e	A32 supervisory program access
0x0d	A32 supervisory data access
0x0b	A32 non-privileged block transfer
0x0a	A32 non-privileged program access
0x09	A32 non-privileged data access

Table 6-1 VMEbus Address Modifiers

The VMEbus supports two operation modes: supervisory and non privileged. The two modes allow a multi-tasking operating system to have access protection by employing the concept of protected memory regions. Program text and data sections could be separated and bus signals would produce an error if an incompatible access occurred. The VMEbus modifier codes reflect this concept with supervisory and non-privileged access along with block, program, and data access.

The DTB supports three data path sizes: D8, D16, and D32 which provide 8 bits, 16 bits and 32 bits of data transfer in each address space.

Note: Revision D of the VMEbus will support A64 providing 64 bits of address space. The additional 32 bits are gained by using the data lines (multiplexing) to present the address.

The DTB also has the ability to access non-aligned bytes and do three byte (tri-byte) transfers.

The DTB supports read, write, address only, and read-modify-write operations. Read-Modify-Write allows a MASTER to read data, modify it and write it back without releasing the bus.

Since the VMEbus is asynchronous, a non-responsive SLAVE could hang the bus, so the system controller usually incorporates a watch dog timer to timeout the bus causing a bus error. A mismatch between data transfer size of the SLAVE and the MASTER can also cause a bus error. In either case, the bus error (BERR*) line is asserted.

Note: A * indicates that low level logic is true.

Descriptions of the various DTB cycles are outlined in the following subsections. These descriptions are provided as general reference information; they are not intended to provide specific VMEbus protocols and specifications.

Write Cycle

The write cycle begins when the MASTER gains control of the bus. The MASTER places the address and the address modifier on the address bus to indicate the type of access and address space (i.e. A32, A24, or A16). The address strobe (AS*) is asserted to indicate a stable address.

The MASTER specifies data direction, in this case driving WRITE* low. The MASTER places the data on the data bus and then asserts the data strobes DS0* and DS1*. The combination of DS0*, DS1*, A01, and LWORD* determines the data size (i.e. D32, D16, or D8). (Note that DS0*, DS1*, A01, and LWORD are used to select which byte location(s) within the 4 byte word are accessed during the data transfer.)

The SLAVE reads the data and responds by driving the DTACK* line low. The MASTER then releases the address lines, modifier line, data lines, and LWORD* line and drives DS0*, DS1* and AS* to high. Finally, when the SLAVE receives DS0*, DS1*, and AS*, it releases DTACK* to complete the cycle. If a mismatch in the data transfer size or other errors occur, the SLAVE asserts BERR* and the bus error terminates the cycle.

Read Cycle

The read cycle is the same as the write cycle except the MASTER drives WRITE* high to indicate a read cycle and the SLAVE places the data on the data bus in response to the data strobes (DS0*and DS1*). The SLAVE asserts DTACK* when the data is stable and the MASTER reads it. The MASTER then releases the address lines and drives DS0*, DS1 and AS* high. Finally, the SLAVE receives these lines and releases the data lines and DTACK* to complete the cycle.

Unaligned Transfer (UAT) Cycles

Unaligned transfer cycles occur in the same way as the read and write cycles described above. Data bytes for unaligned transfers are selected by the DS0*, DS1*, A01, and LWORD* combination. Unaligned transfers include:

- double byte (1-2)
- triple byte (0-2)
- triple byte (1-3)

Read Modify Write (RMW) Cycle

The Read-Modify-Write cycle starts as a normal read cycle. When the SLAVE asserts DTACK* the MASTER only releases the data strobes (DS0*-DS1*) while the assertion on address strobe (AS*) is maintained. This indicates a RMW cycle. The MASTER then, after modification, writes the data back with a normal write cycle.

Block Transfer (BLT) Cycles

The block transfer cycle starts as a normal read or write cycle. When the first data transfer is complete the MASTER does not allow address strobe (AS*) to go high. Instead it repeatedly drives data strobes(s) (DS0* and DS1*) low in response to the data transfer acknowledge (DTACK*) from the SLAVE. It then transfers data to and from sequential memory locations in ascending order in blocks of up to 256 bytes for D32.

Note: Revision D of the VMEbus will support D64 providing data transfers of up to 2048 byte blocks.

Address Only (AO) Cycles

An Address Only cycle allows slow peripherals to prefetch data.

This case is like the write or read cycle except no data is transferred and the data strobes are never asserted. Only the address and address strobes are used. The MASTER releases the bus without receiving a BERR* or DTACK*. This allows the MASTER's local memory decoder to work in parallel with the SLAVES's address decoder and thereby, in some cases, speed up the transfer.

Bus Arbitration

The VMEbus is a multi-master system and supports bus arbitration to:

- Prevent simultaneous use of the bus by two masters.
- Schedule requests for multiple masters for optimum use of the bus.

There are four bus request lines BR0* to BR3* with BR3* used as the highest priority. When the bus is granted to a particular request level (0-3,) the bus grant signal is daisy-chained down the backplane via the BGxIN and BGxOUT* (where x is 0-3).The bus grant for a particular level "x" is brought into the card via BGxIN* and is sent on to the nearest neighbor (the next slot to the right) via BGxOUT. If a MASTER requires the bus and is a requester at level "x", it does not pass BGxOUT* but instead asserts the bus busy line (BBSY*). This daisy chain system means that for each bus request level, the requesting MASTER closest to the arbiter (system controller) gains control of the bus.

The MASTER will relinquish the bus based on a policy of Release On Request (ROR) or Release When Done (RWD). ROR requires the MASTER to relinquish the bus only when another request is pending. RWD causes the MASTER to release the bus when it has finished its current operation. This policy is built into the VMEbus MASTER.

Modern VMEbus MASTERS often offer the ability to select RWD or ROR.

The policy of ROR or RWD can impact system performance. ROR MASTERs tend to hold the bus and monitor the bus request lines (BR0*-BR3*) even if they don't need the bus. ROR is a good policy for MASTERs which make frequent bus accesses and do not want to spend the time arbitrating for the bus on each access. This of course means that other MASTERs in the system will normally find the bus busy and will have to wait for arbitration by the system controller.

RWD MASTERs tend to release the bus to allow rearbitration on each bus access. This policy is good for MASTERs which may make fewer bus accesses or with systems where multiple MASTERs need more equal access to the bus.

There are many algorithms that can be used to arbitrate the bus; only two are addressed here: round robin arbitration and priority arbitration. Round Robin (RRS) arbitration assigns the bus on a rotating priority basis. When the bus is granted to the requester on the bus request line BRx*, the highest priority for the next arbitration is assigned to bus request line BR(x-1)*. The bus is rotated from BR3* through BR0* and then back to BR3* again.

Prioritized (PRI) arbitration assigns the bus according to a fixed priority scheme where each of the four bus request lines has a priority from highest (BR3*) to lowest (BR0*). This means the higher priority devices could potentially "lock-out" lower priority devices.

Another priority scheme which is a subset of PRI is single level (SGL). SGL only accepts request on BR3* and relies on the daisy-chain (BG3IN*, BG3OUT) as the arbitration mechanism for the requests. Many system controllers offer the ability to select between RRS and PRI or other arbitration schemes.

Interrupt Handling

The VMEbus allows seven prioritized interrupt request levels (IRQx). These are IRQ1* to IRQ7* with IRQ1* the lowest and IRQ7* the highest. Once an interrupt is asserted and the bus is granted to a handler, a 3 bit code, which identifies the interrupt level being acknowledged, is placed on address bits 1-3 and IACK* and AS* are asserted. The handler also indicates the status data size (D32, D16, or D8) via the data strobes. IACK* runs the full length of the backplane and is connected to IACKIN* on slot 1. When driven low the IACKIN* line causes the daisy-chain driver located in slot 1 to propagate a falling edge down the backplane via the interrupt acknowledge daisy-chain (IACKIN* AND IACKOUT*).

Each board compares the interrupt level with the interrupt level it asserted and IACKIN*. If the board does not have an interrupt outstanding or does not match the interrupt level, the IACKIN* is passed to the next board via IACKOUT*. If the interrupting board matches the interrupt level, it does not pass on IACKOUT* and places a status on the data bus.

The size of the status word is determined by the interrupt handler. Note that even though the VMEbus definition allows D8, D16, or D32 size status, most systems only use D8. Even if an interrupting board can use D16 or D32 it must respond to the size declared by the interrupt handler or declare a bus error (BERR*).

VMEbus boards have two methods of interrupt acknowledge:

- release on acknowledge of interrupt
- release on register access.

With release on acknowledge, the interrupting device de-asserts the IRQ when the controller acknowledges the interrupt. That is, when the interrupting device transfers its interrupt vector, the assertion on IRQ is released.

With release on register access, the interrupting device does not remove its IRQx request until a register on the board has been accessed or modified. Therefore, after the interrupt vector has been transferred, the device still asserts IRQx.

Utility Bus

The utility bus supplies power, system reset signals, system clock, power fail signals, and I/O pins. The VMEbus specifies voltages of: +5V, +12V, -12V, and a standby +5V.

The system clock is a 16 MHz clock available via SYSCLK*. This is supplied by the system controller. This clock does not synchronize the bus. The VMEbus is asynchronous.

The SYSFAIL* signal allows any board to report a failure to the system controller by asserting this line. In the same vein, ACFAIL* signal is generated when the main power supply goes down. (This signal may not give much warning.)

The SYSRESET* is asserted upon system startup by the system controller and may be used to reset any cards in the VMEbus chassis.

VMEbus Specifications

There are three sizes of card, known as form factors, for the VMEbus. These are 3U, 6U, and 9U cards. These form factors indicate the physical size and the number of VMEbus connectors. Note that extended VMEbus, 32 address and data lines, requires a 6U or 9U form factor.

A "U" or "unit" is equivalent to 1.75 inches and measures the height of the front panel. The actual size of a 3U (single height) card is 3.93 inches (100mm) high and 6.30 inches (160mm) deep. A 6U (double height) card is 9.18 inches (233.35mm) high and 6.30 inches (160mm) deep. A 9U card is 14.43 inches (366.70 mm) high and 15.73 inches (400 mm) deep.

The standard VMEbus supports 24 address lines and 16 data lines and requires only the P1 connector. The extended VMEbus supports 32 address lines and 32 data lines and requires P1 and part of connector P2 (row B). Rows A and C of the P2 connector are user-definable.

VMEbus Pinouts

Each pin assignment name is a mnemonic for its activity. The * indicates that a low logic level is true. The pin assignments for the P1 and P2 connectors are outlined in Table 6-2 and 6-3. Table 6-4 defines the pin assignment names.

Pin	Row A	Row B	Row C
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL	D10
4	D03	BG0IN*	D11
5	D04	BGOUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK	A17
22	IACKOUT*	SERDAT*	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12

Table 6-2 VMEbus P1 Pin Assignments

Pin	Row A	Row B	Row C
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5VSTDBY	+12V
32	+5V	+5V	+5V

Table 6-2 (continued) VMEbus P1 Pin Assignments

Pin	Row A	Row B	Row C
1		+5V	
2		GND	
3		RESERVED	
4		A24	
5		A25	
6		A26	
7		A27	
8		A28	
9		A29	
10		A30	
11	U	A31	U
12	S	GND	S
13	E	+5V	E
14	R	D16	R
15		D17	
16	D	D18	D
17	E	D19	E
18	F	D20	F
19	I	D21	I
20	N	D22	N
21	E	D23	E
22	D	GND	D
23		D24	
24		D25	
25		D26	
26		D27	

Table 6-3 VMEbus P2 Pin Assignments

Pin	Row A	Row B	Row C
27	U D	D28	U D
28	S E	D29	S E
29	E F	D30	E F
30	R I	D31	R I
31	N	GND	N
32	E	+5V	E
	D		D

Table 6-3 (continued) VMEbus P2 Pin Assignments

Pin Name	Definition
D00 - D31	Data lines. These lines are tri-state and are not defined until the data strobes (DS0* and DS1*) are asserted by the MASTER.
A00 - A31	Address lines. These lines are tri-state and are not defined until the address strobe (AS*) is asserted by the MASTER.
AM0 - AM5	Address modifier lines. Asserted by the MASTER and indicates the type of data transfer to take place. VME SLAVEs look at the lines to determine if they will respond and what type of response will be made.
DS0* - DS1*	Data Strobe lines. Asserted by the MASTER and indicates stable data on the data bus.
AS*	Address strobe. Asserted by the MASTER and indicates a stable address is present on the address lines.
BR0* - BR3*	Bus request lines. MASTER request a busy bus via these prioritized levels.
BG0IN* - BG3IN*	Bus grant in. (daisy chained)
BG0OUT* - BG3OUT*	Bus grant out. (daisy chained)
BBSY*	Bus busy.
BCLR*	Bus clear. (Hint to bus master to relinquish the bus, VME MASTERs are not required to comply)
IRQ1* - IRQ7*	Interrupt request lines.
IACK*	Interrupt acknowledge. Asserted by MASTER to indicate the VME interrupt level to be serviced.
IACKIN*	Interrupt acknowledge in. (daisy chained)
IACKOUT*	Interrupt acknowledge out. (daisy chained)
DTACK*	Data transfer acknowledge. Asserted by SLAVE to indicate a successful bus transfer.
WRITE*	Write/Read select with right = low logic level
LWORD*	Indicates long word transfer. (D32)

Table 6-4 VMEbus Pin Assignment Names

Pin Name	Definition
SYSCLK	16Mhz system clock. (Does not control bus timing.)
SERCLK	Serial data clock.
SERDAT*	Serial data line.
BERR*	Bus error line.
SYSFAIL*	Indicates a card has failed.
ACFAIL	AC power failure notify line.
SYSRESET*	Reset signal for VME bus.

Table 6-4 (continued) VMEbus Pin Assignment Names

Chapter 7

CPU Subsystem Architecture

The heart of the V30/35 subsystem is a 36 MHz (30 MHz in the V30) MIPS R3000A CPU with an R3010A FPU, a 64 KByte instruction cache, and a 64 KByte data cache connected on a local high-bandwidth bus.

The V30/35 VMEbus I/O characteristics described later in this chapter are identical to those of the 4D/30 and 4D/35 Personal IRIS.

CPU Subsystem

The CPU subsystem contains three functional sections:

- The processor core, which contains the CPU and FPU.
- Main memory, which contains DRAM and supporting circuitry.
- The I/O system, which contains peripheral ports and hardware designed to read incoming data, as well as manage incoming and outgoing data.

Figure 7-1 illustrates the three sections in a block diagram.

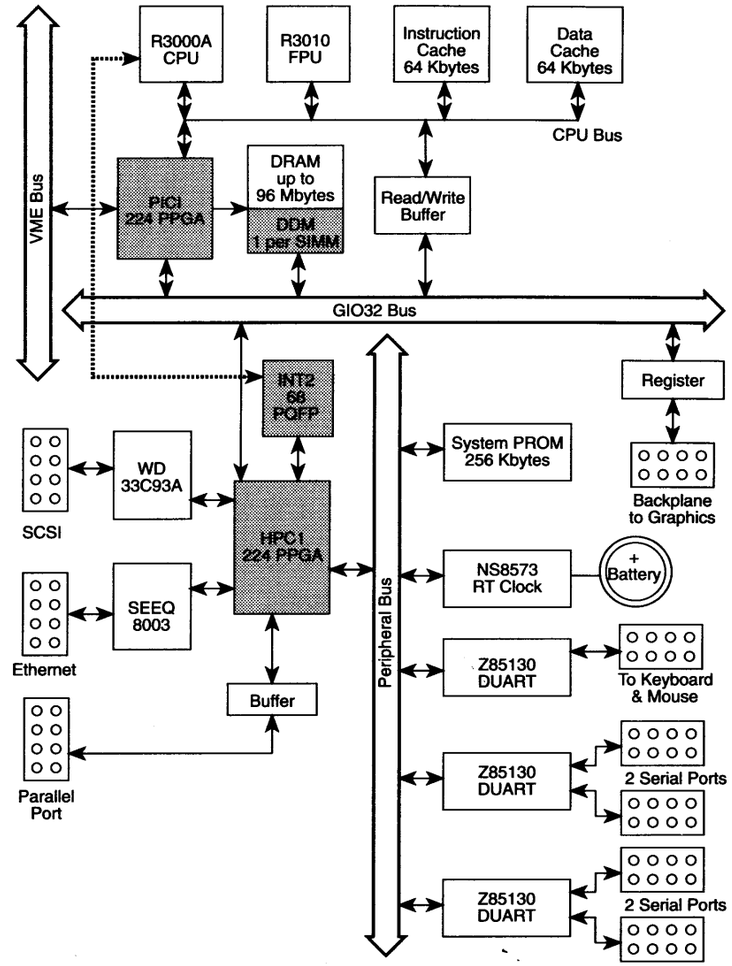


Figure 7-1 CPU Board Block Diagram

Note: PPGA is an acronym for "plastic pin grid array" and PQFP is an acronym for "plastic quad flat pack" in the above block diagram.

Three internal buses provide communications within the CPU subsystem:

- The CPU bus, which connects the CPU, FPU, cache control, and bus control hardware.
- The GIO32bus, which is the main system bus connecting the processor core, main memory, I/O system, VMEbus, and graphics port.
- The Peripheral Bus, which connects the peripheral ports and other I/O components.

The CPU bus and the GIO32 bus have separate clocks and run at different speeds so that each part runs at maximum capability, and the CPU and other chips can be upgraded independently as technology improves. ASICs aid communication between systems and the GIO32 bus:

- The PIC1 ASIC is the GIO32bus arbiter, providing an interface between the processor core and the GIO32 bus, and acting as memory controller, allowing direct memory access (DMA) by devices other than the CPU.
- The HPC1 ASIC interfaces peripheral I/O and other devices on the peripheral bus, connecting them to the GIO32 bus.

Processor Core

The processor core is centered around the CPU. It includes the FPU for floating point calculations and a CPU memory cache to store frequently accessed instructions and data. It uses two custom chips: the PIC1 ASIC, to give the CPU access to memory and the GIO32 bus, and the INT2 ASIC, to deliver interrupts to the CPU.

MIPS R3000A CPU

The MIPS R3000A CPU, is a RISC (Reduced Instruction Set Computer) chip that runs at 36 MHz (30 MHz in the V30). Its instruction set is simple, well-defined and, combined with a fast cycle time, is an excellent target for optimizing compilers.

The R3000A has a wide range of memory addresses divided between user-addressable virtual memory and kernel memory. A fully associative Translation Lookaside Buffer (TLB) assists virtual memory addressing. It contains 64 entries, each of which map a 4 kilobyte page, and has controls for read/write access and process identification. In addition, the V30/35 provides a single, uniform 2 Gigabyte virtual address space for user processes. Each virtual address is tagged with a 6-bit process ID to form unique virtual addresses for up to 64 user processes. All user-mode addresses are mapped through the TLB to physical memory locations.

The architecture of the V30/35 is designed to fully implement all R3000A capabilities, including block cache refills, instruction streaming, and partial cache writes. All instructions and addresses are 32 bits wide. These design considerations, combined with the CPU's five-stage pipeline design and a high-bandwidth memory interface, help achieve an execution rate that approaches one instruction per cycle.

Bit and Byte Numbering

The MIPS processor is big-endian for byte-numbering within the word. Bit-numbering, however, is little-endian by convention. Thus, the least-significant bit is bit 0, whereas the least-significant byte is byte 3. All DMA transfers proceed in increasing order of byte address; therefore the most significant byte is the first byte written in the 32-bit word. Figure 7-2 illustrates bit and byte numbering.

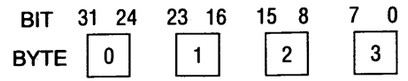


Figure 7-2 Bit and Byte Numbering

For detailed information about the R3000A CPU, refer to *MIPS RISC Architecture; R2000/R3000*, Prentice Hall, ISDN 0-13-584749-4.

MIPS R3010A FPU

The MIPS R3010A FPU, the floating point co-processor, connects to the CPU via the CPU bus. It assists the CPU by performing both 32-bit single-precision and 64-bit double-precision IEEE standard floating point operations.

The MIPS R3010A RISC FPU operates in conjunction with the R3000A CPU to provide seamless integration of fixed and floating-point operations. All floating-point operations conform to the ANSI/IEEE Std. 754-1985 *IEEE Standard for Binary Floating-Point Arithmetic.*

Instruction and Data Caches

The instruction and data caches, each a 64 KByte section of SRAM, greatly increase the operating speed of the CPU by providing faster access than would be possible directly to main memory. The instruction cache holds instructions frequently used by the CPU; the data cache holds frequently-used operands (data).

When the CPU requires an instruction or an operand, it checks the caches first, alternating access to the two caches during each CPU cycle. If the CPU finds the instruction or operand in the cache (a cache hit), it retrieves the information and continues at full speed. If the CPU doesn't find the instruction or operand (a cache miss), it finds the information in main memory, and transfers its block into the appropriate cache, adding stall cycles to compensate for the slower access time of main memory (see Figure 7-3).

The V30/35 uses a write-through scheme in the data cache to ensure that writes made to the cache are also written to the corresponding page in main memory. It also supports a technique for caching instructions called instruction streaming. Instruction streaming, which works only in the instruction cache, allows the CPU to bring instructions into the cache and begin to execute the first of them while the rest of the block is still being written to the cache.

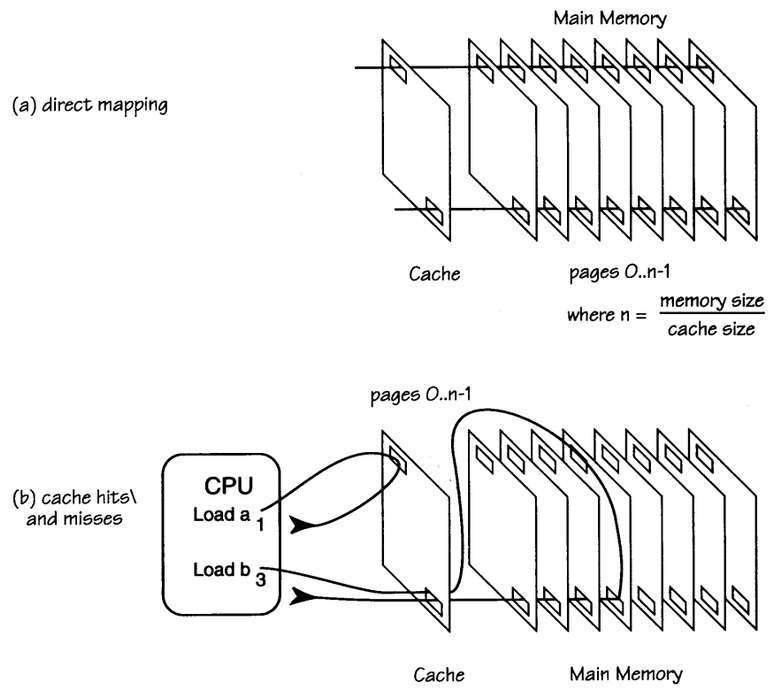


Figure 7-3 Cache Operation

PIC1 ASIC

The PIC1 ASIC, the Processor Interface Controller, is a custom Silicon Graphics chip connected to the CPU via the CPU bus. It is also connected to the GIO32 bus (the system bus), and has address and control lines connected to main memory. It performs five main functions:

- It provides an interface between main memory and the CPU core.
- It acts as a system arbiter for the GIO32 bus.
- It serves as a DMA (Direct Memory Access) controller for all memory requests from the graphics system and any other devices on the GIO32 bus.
- It provides single-word accesses for the CPU to GIO32 bus devices and to the graphics system.
- It manages the VMEbus to GIO32bus interface.

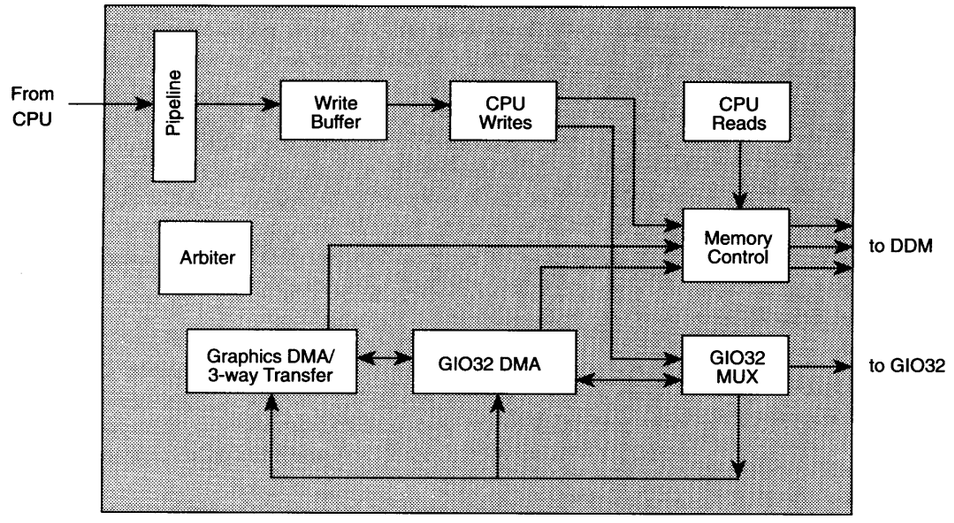


Figure 7-4 PIC1 ASIC

The PIC1 ASIC supplies the write buffer that compensates for the different access times of the cache and main memory. It also implements DMA, including the special three-way transfer, and provides arbitration for the GIO32 bus.

The PIC1 controls main memory through address and control lines coming from its memory control and connected to the DDM ASIC chips in memory. The memory controller synchronously supports R3000A CPU accesses for block cache filling and instruction streaming, and performs parity checking during cache refills.

As part of the V30/35's write-through cache architecture, the PIC1 incorporates a FIFO-based write buffer to increase the write performance of the CPU. The CPU writes at cache speed at one end, while the PIC1 flushes it at a slower rate to the system bus or memory at the other end. The FIFO is eleven levels deep, which means that up to eleven address/data pairs can be waiting to be written without stalling the CPU.

To accommodate CPU writes to different parts of the system, the write buffer holds three basic types of write transactions: DRAM writes, graphics writes, and GIO32 bus writes. The type is determined by decoding the address. Address/data pairs are queued in the order that they are issued by the CPU, regardless of type. Two state machines in the PIC1 are responsible for flushing the write buffer. One runs on the CPU clock and flushes main memory writes; the other is clocked by the GIO32 bus clock and performs writes to the GIO32 bus and to the graphics address space. The PIC1 empties the write buffer whenever it gains access to the system bus. In addition, CPU reads from memory are blocked if the write buffer is not empty.

The PIC1, as GIO32 bus arbiter, controls the GIO32 bus. It guarantees a fixed upper bound on the bus acquisition latency by allocating time slots for each bus requester, and it provides a pre-emption mechanism that gives priority to real-time I/O devices – very important for devices such as the Ethernet port and the audio subsystem.

The Graphics DMA controller section of the PIC1 supports direct memory access for the graphics system, allowing support for a variety of advanced graphics functions such as rectangular DMA, linked-list descriptor arrays, and hardware synchronization of DMA transfers. It also includes a three-way transfer engine, which is a "short-span" or "small block"' DMA mode for transferring smaller blocks of data without invoking the operating system.

The GIO32 DMA controller supports DMA for any other devices on the GIO32 bus, including those that are connected through the HPC1 chip.

The GIO32 MUX provides direct access for the CPU to read and write any GIO32 address.

INT2 ASIC

The INT2 ASIC, the interrupt and timer logic chip, is a custom Silicon Graphics chip connected to the CPU through 6 interrupt lines. It is also directly connected to any devices on the system that may request interrupts, including the HPC1 ASIC in the I/O system. The CPU can read INT2's registers, including timer information. INT2 provides interrupts to the CPU and provides timing for software-driven interrupts. Its major functional blocks, as shown in Figure 7-5, are as follows:

- CPU initialization logic, which establishes the operating mode of the R3000A by sensing the state of the interrupt inputs to the CPU during the last four clock cycles of the reset sequence. INT2 provides interrupts to establish characteristics for the data and instruction caches, including block refill sizes, feature selections such as instruction streaming and partial word store capability.
- The system timer, which reads an external 10 MHz clock, divides it by 10 to produce a 1 microsecond signal, and then uses it to generate software-selectable time intervals and associated interrupts.
- The interrupt multiplexer, which sends interrupts to the CPU and the DSP. The R3000A CPU has eight level-triggered interrupt inputs with the first three reserved for software. Level 3 is reserved for the FPU. Level 4 provides the VMEbus interrupts and is also connected to I/O devices on the GIO32 bus: SCSI, Ethernet, parallel port, and the serial port. Level 5 is used for power failure, the DSP, and by requests for vertical retrace interrupts (part of video signal timing). Levels 6 and 7 reflect timer interrupts. And level 8 reports addressing and parity errors.

- The output port, which is a general-purpose 8-bit output register. The five low-order bits of the register hold system configuration information and may be used to drive LEDs. The three high-order bits are unused and reserved for future uses. The V30/35 currently has four LEDs which use this port to show power-up status and console health.

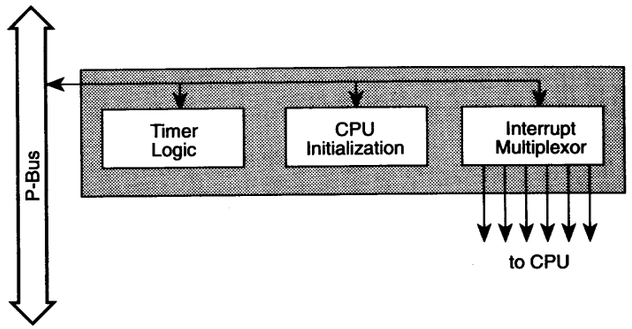


Figure 7-5 INT2 ASIC

Main Memory

Main memory, which is controlled by the PIC1 chip in the processor core, is designed to provide access to large amounts of extremely fast DRAM to the system. The basic boardset holds from 8 to 32 MBytes of physical memory using one bank of DRAM. The memory option board provides an additional 2 banks of physical memory for up to 64 MBytes of additional system memory. Each bank can hold either 8 MB (four 2-MB SIMMs), 16 MB (four 4-MB SIMMs), or 32MB (four 8-MB SIMMs).

Each V30/35 custom SIMM contains a DDM chip that provides the interface between the DRAM and GIO32 bus. Two state machines are used, one on the CPU clock, the other on the GIO32 clock. One or the other is activated based on the access mode.

To increase the DRAM speed, the V30/35 uses custom SIMMs with on-board 2-way interleave controllers (the DDM ASIC) directly under the control of the PIC1 chip. Each DDM ASIC provides extremely high memory bandwidth and a large memory capacity. PIC1 provides a translation algorithm that can sense the base address and size of each physical memory bank and store them in PIC registers when the machine is powered on. In this way, any assortment of 8 or 32 MB memory banks can be used to present a contiguous address space to the CPU.

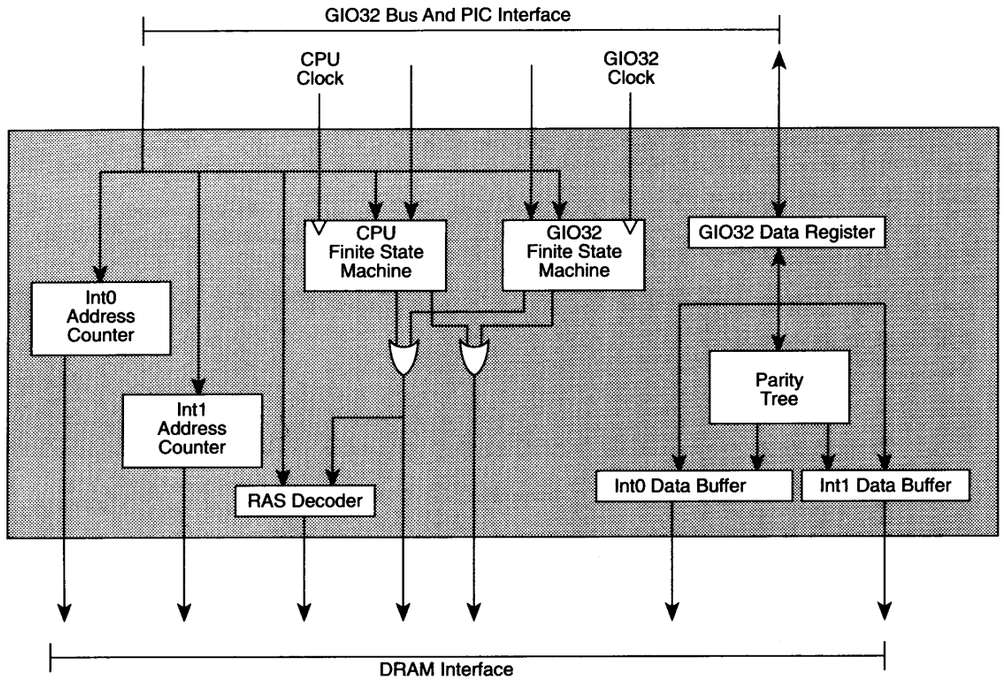


Figure 7-6 DDM ASIC

I/O System

The I/O system ties together a variety of I/O ports and the chips that drive them, a system clock, system PROM for booting up, and static RAM. It uses a peripheral bus to transfer some I/O data, and also uses a custom chip – the HPC1 ASIC – to interface between the system and the GIO32 bus.

The HPC1 ASIC, the High Performance Peripheral Controller, is a custom Silicon Graphics chip that connects to the GIO32 bus, the peripheral bus, and directly to several of the I/O ports. It is the heart of the I/O system, and quickly transfers data between main memory and a rich collection of peripheral devices, at the peak rates of such devices. It uses minimum bandwidth on the GIO32 bus, freeing the bus for other data transfers.

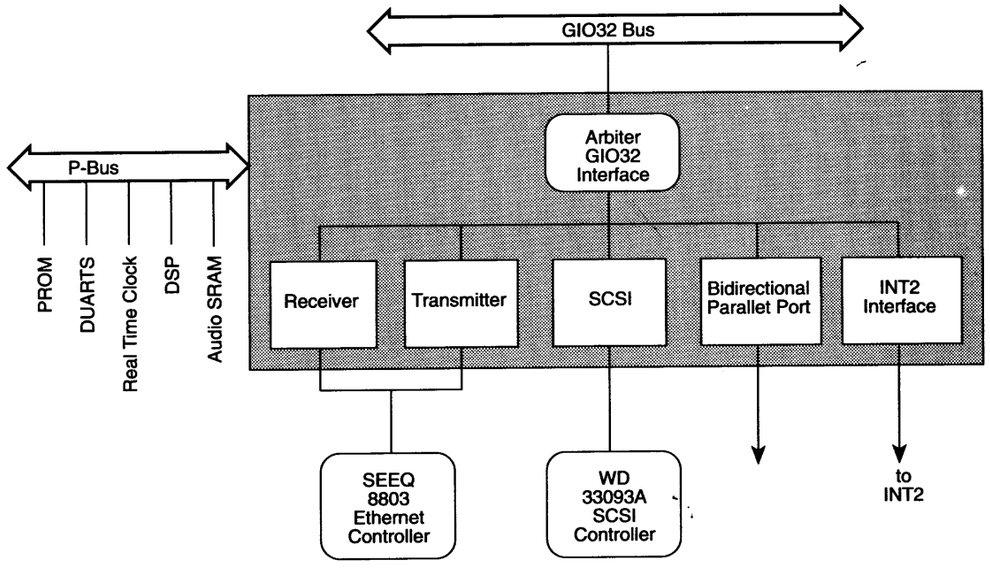


Figure 7-7 HPC1 ASIC

The HPC1 permits fast data interchange between peripheral devices and main memory without involving the CPU, improving both CPU and peripheral performance. For each peripheral device, the HPC1 provides an independent FIFO data buffer, and supports DMA to main memory through the GIO32 bus and the PIC1 ASIC with a burst performance of 100 MBytes per second. It interfaces to the serial ports, audio system, and other devices through the peripheral bus.

Ethernet Port

The Ethernet interface consists of an Ethernet port supported by a controller that is connected directly to the HPC1 ASIC. The port can be connected to either a thick or a thin wire Ethernet. The HPC1 supplies the logic required to retransmit packets when collisions occur and to manage the interface's 32-byte FIFO buffer. When the HPC1 receives a packet, it interrupts the CPU after it writes the packet into memory. When transmitting, it interrupts the CPU when a packet is successfully sent or when 16 transmission attempts have all failed.

Note: The V30/35 uses a SEEQ 8003 microchip as the Ethernet controller. For additional information, please consult the microchip specifications.

SCSI-II

The SCSI-II interface connects to the external SCSI port on the front panel, supported by a SCSI controller connected directly to the HPC1 ASIC. The HPC1 uses a 64-byte FIFO buffer and a controlling state machine to transfer data between up to 7 SCSI devices and memory at rates exceeding 4 MBytes per second, all while using minimum GIO32 bus bandwidth.

Note: The V30/35 uses a Western Digital 33C93A microchip as the SCSI controller. For additional information, please consult the microchip specifications.

Parallel Port

The parallel port interface provides a Centronics parallel port to connect printers, plotters, scanners, and other similar devices. The port is connected to the HPC1 ASIC, where the HPC1 provides a 64-byte FIFO buffer it uses to transfer data between main memory and the parallel port at up to 2.5 MBytes/sec.

Peripheral Bus (P-Bus)

The Peripheral Bus (P-Bus) is a 20-bit address, 16-bit data bus used by the HPC for additional peripheral support. It connects the boot PROMs, a real-time clock, the timer, four serial ports controlled by DUARTs, and the audio subsystem DSP, which serves as the bus master.

Serial Ports

The serial interface consists of six serial ports, each pair controlled by a DUART chip that connects to the Peripheral Bus. Two ports are for keyboard and mouse; two are available as external Apple Macintosh type connectors that current software uses as RS232 ports. The other two ports are standard 9 pin D-subs supporting RS232 at a transfer rate of up to 38.4 KBaud.

Note: The V30/35 uses the Zilog 85130 DUART microchip as the serial port controller. For additional information, please consult the microchip specifications.

GIO32 Bus

The GIO32 is the V30/35's main system bus, and is designed for high speed data transfers. It connects the main systems of the V30/35: the processor core, main memory, the I/O systems, and the graphics subsystem. It is a 32-bit synchronous, multiplexed address/data, burst mode bus that operates at a minimum of 25 MHz, clocked independently of the CPU. The bus protocol supports data transfers at a maximum sustained rate of one word per clock.

This bus is especially targeted to support fast pixel moves between the CPU and the graphics subsystem's GRX bus.

The GIO32 bus in the V30/35 helps the system run over three times as efficiently as previous Silicon Graphics systems. The HPC1 ASIC, which provides a buffer between I/O devices and the GIO32 bus, adds another three-fold improvement in system performance by operating at full-speed bursts. It never needs more than 5% of the bus bandwidth, and thus leaves most of the bus capacity free for the CPU, graphics, and other subsystems.

VMEbus Interface

The V30/35 CPU subsystem VMEbus interface is identical to that of the 4D/30 and 4D/35 Personal IRIS; however, the V30/35 CPU subsystem provides DIP switches for extra functionality to manipulate A24 and A32 Slave accesses (see section below).

The cache parity implementation precludes the use of the data cache for VME addresses. This is because cache loads copy local memory parity. Since the VMEbus is not parity checked, no parity is generated on the cached VME references. Thus, subsequent references to a cached VME address would result in cache parity errors.

For more general information on VMEbus functions, see Chapter 6, "The VMEbus Overview."

Note: Information in this chapter on the V30/35 VMEbus interface applies only to a specific version of the CPU subsystem. The version ID numbers can be located on the CPU and I/O boards by referring to Appendix E, "V30/35 Component Layout." The I/O board version ID number should be 030-5024-002. The CPU board version ID number should be 030-5023-002. If any other version ID numbers appear on either board, please contact your support representative or sales representative for further information.

Local (V30/35) to VMEbus Memory Access

(VMEbus Master)

The CPU acts as a master on the VMEbus by performing read or write cycles to the VME address ranges specified in the physical address space (see Table 7-1). The CPU is capable of generating VME A32:24:16, D32:16:8, and IACK D8 cycles. The CPU is also capable of generating Read-Modify-Write (RMW) cycles, but is not capable of generating Block Transfers (BLT) or Unaligned Transfers (UAT).

Address Range	Size	Usage
0x 00 00 00 00 – 0x 0f ff ff ff	256 MB	Local memory
0x 10 00 00 00 – 0x 1b ff ff ff	192 MB	VME A32, D8:16:32, extended non-privileged data access (0x1 forced as highest nibble in target VMEbus address)
0x 1c 00 00 00 – 0x 1c ff ff ff	16 MB	VME A24, D8:16:32, standard supervisory data access
0x 1d 00 00 00 – 0x 1d 0f ff ff	1 MB	VME A16, D8:16:32, short supervisory data access
0x 1d 10 00 00 – 0x 1d 1f ff ff	1 MB	VME A16, D8:16:32, short non-privileged data access
0x 1d 20 00 00 – 0x 1d ef ff ff	13 MB	Unused (but maps into either VME A16 or VME IACK)
0x 1d f0 00 00 – 0x 1d ff ff ff	1 MB	VME IH(1-7), D8:16, IACK cycles
0x 1e 00 00 00 – 0x 1e ff ff ff	16 MB	VME A24, D8:16:32, standard non-privileged access

Table 7-1 V30/35 VMEbus Master Address Map

The CPU is a release when done (RWD) Master and always requests the bus on bus request level 3 (BR3*).

VMEbus-to-Local (V30/35) Memory Access

(VMEbus Slave)

The CPU supports access from VME bus masters to local memory in single cycle mode or block (BLT) mode under both A32 and A24 addressing. Unaligned transfers (UAT) are not accepted or responded to. Although Read-Modify-Write cycles are accepted, local GIO devices can access memory between the read and write access (atomic access to memory is not supported).

The single cycle and block addressing are as follows:

- A32 addressing provides a VMEbus Master access to all local memory in both privileged and non-privileged modes. Three DIP switches are provided to map the local memory to any 256 MByte boundary within the lower 2 GBytes of the 4 GByte address space. The three DIP switches correspond to A30, A29, and A28. The V30/35 CPU subsystem responds when the incoming address bits (A30:28) equal the DIP switch settings. Note the incoming address must always have A31 = 0.
- Using A24 addressing in either privileged or non-privileged mode, VMEbus masters can access the lower 8 MBytes of local memory; programmable 4 MByte windows are also available in non-privileged mode only. A DIP switch is provided to disable the A24 slave response so that the V30/35 slave responds to A32 access only.

For information how to set the DIP switches, please see Appendix D.

VMEbus requests receive equal priority as graphics DMA requests from the GIO bus. The GIO bus arbitration uses the Round Robin algorithm.

Accesses to physical addresses above 0x3fffffff or local memory addresses beyond the maximum size of 96 MBytes are illegal; they are reported as bus errors on reads and interrupts on writes.

A read/write to an address between the end of local memory and the end of the 256 MByte window (0x0fffffff) is also illegal.

VMEbus Arbiter

The VMEbus arbiter is a Single Level Arbiter (SGL) monitoring bus request line 3 (BR3). The CPU subsystem therefore relies on the daisy chain mechanism for arbitration. The CPU must always be the arbiter and provide the system controller functions; therefore it must reside in slot 1.

Caution: Use of BR0:2 is not supported.

VMEbus System Clock

The CPU subsystem provides a high-drive 16 MHz System Clock on the VME back plane.

VMEbus Watchdog Timer

The V30/35 provides a 32us timeout for all transactions.

CPU Address Space

Table 7-2 and 7-3 show the capabilities of the V30/35 as a VMEbus master. To generate the physical addresses presented in the table, one can use Kseg1 addresses. Kseg1 provides 512 MBytes of uncached, unmapped virtual address space starting at 0xa0000000. The processor directly maps Kseg1 onto the first 512 Mbytes of physical space.

For example:
0x a0 00 00 00 maps to 0x 00 00 00 00
and
0x b0 00 00 00maps to 0x 10 00 00 00

Note: 0x b0 00 00 00 = (0x a0 00 00 00 +0x 10 00 00 00)

Please refer to the Silicon Graphics publication *Guide to Writing Device Drivers* for additional information.

Address Range	Size	Usage
0x 00 00 00 00 – 0x 0f ff ff ff	256 MB	Local memory
0x 10 00 00 00 – 0x 1b ff ff ff	192 MB	VME A32, D8:16:32, extended non-privileged data access (0x1 forced as highest nibble in target VMEbus address)
0x 1c 00 00 00 – 0x 1c ff ff ff	16 MB	VME A24, D8:16:32, standard supervisory data access
0x 1d 00 00 00 – 0x 1d 0f ff ff	1 MB	VME A16, D8:16:32, short supervisory data access
0x 1d 10 00 00 – 0x 1d 1f ff ff	1 MB	VME A16, D8:16:32, short non-privileged data access
0x 1d 20 00 00 – 0x 1d ef ff ff	13 MB	Unused
0x 1d f0 00 00 – 0x 1d f fff ff	1 MB	VME IH(1-7), D8 IACK cycles
0x 1e 00 00 00 – 0x 1e ff ff ff	16 MB	VME A24, D8:16:32, standard non-privileged access
0x 1f 00 00 00 – 0x 1f bf ff ff	12 MB	Local I/O (see Table 7-4)
0x 1f c0 00 00 – 0x 1f ff ff ff	4 MB	Boot PROM

Table 7-2 Global Address Map

Table 7-3 shows three address ranges (3.5 GBytes) that are not accessible using unmapped addresses (Kseg0 or Kseg1). The MIPS chip used in the V30/35 strips off the top three bits when using a Kseg0 or Kseg1 address and therefore precludes easy access to the top 3.5 GBytes of address space.

Address Range	Size	Usage
0x 20 00 00 00 – 0x 2f ff ff ff	256 MB	VME A32, D8:16:32, extended non-privileged data. (0x0 forced as highest nibble in target VMEbus address). Not directly accessible when running IRIX.
0x 30 00 00 00 – 0x 3f ff ff ff	256 MB	VME A32, D8:16:32, extended supervisory data access (0x0 forced as highest nibble in target VMEbus address). Not directly accessible when running IRIX.
0x 40 00 00 00 – 0x ff ff ff ff	3 GB	Unused. Causes read bus error or write interrupt. Not directly accessible when running IRIX.

Table 7-3 Normally Inaccessible Addresses

Tables 7-4 shows the local I/O segment address map and Table 7-5 shows the HPC address map.

Address Range	Size	Usage
0x 1f 00 00 00 – 0c 1f 3f ff ff	4 MB	Graphics board memory
0x 1f 40 00 00 – 0x 1f 9f ff ff	6 MB	Slot 0/1 address space
0x 1f a0 00 00 – 0x 1f af ff ff	1 MB	Local PIC1 registers
0x 1f b0 00 00 – 0x 1f bf ff ff	1 MB	HPC and associated peripheral I/O devices (see Table 7-5)

Table 7-4 Local I/O Segment Map

Address Range	Usage
0x 1f b8 00 00 – 0x 0f b8 00 7f	Ethernet registers
0x 1f b8 00 00 – 0x 1f b8 00 9f	SCSI registers
0x 1f b8 00 a0 – 0x 1f b8 00 bf	Parallel port
0x 1f b8 01 00 – 0x 1f b8 01 1f	Ethernet external registers
0x 1f b8 01 20 – 0x 1f b8 01 24	SCSI external registers
0x 1f b8 01 34	Parallel external register
0x 1f b8 01 80 – 0x 1f b8 01 ba	Peripheral bus internal register
0x 1f b8 01 c0 – 0x 1f b8 01 ff	INT2
0x 1f b8 0d 00 – 0x 1f b8 0d 2f	Duart 0-2 (second LS digit signifgies DUART number)
0x 1f b8 0e 03 – 0x 1f b8 0e 7f	Watch RTC
0x 1f bc 00 00 – 0x 1f bd ff ff	Modem SRAM
0x 1f be 00 00 – 0x 1f bf ff ff	Audio DSP SRAM

Table 7-5 HPC Address Map

Chapter 8

Low-Level Programming Interface

This chapter lists the address, data and control registers used by the CPU, VMEbus, and the various I/O interface controllers. This information is provided for use in developing or porting operating systems.

CPU, Memory, Boot PROM, VME Registers

The registers outlined in Table 8-1 can be used to control operation of the CPU, main memory, boot PROM, and VMEbus.

Name	Address	R/W	Description	Bits
CPUCTRL	0x 1f a0 00 00	R/W	CPU control	16
RSTCONFIG	0x 1f a0 00 04	R	System mode	8
SYSID	0x 1f a0 00 08	R	System identity and status	16
MEMCFG0	0x 1f a1 00 00	R/W	Memory config register 0	32
MEMCFG1	0x 1f a1 00 04	R/W	Memory config register 1	32
REFTIM	0x 1f a1 01 00	R/W	Refresh timer/divider	32
PARERR	0x 1f a1 02 00	R	Parity error register	8
CLERERR	0x 1f a1 02 10	W	Clear parity error registers/ INT5	

Table 8-1 CPU, Main Memory, Boot PROM, and VMEbus Registers

Name	Address	R/W	Description	Bits
CPUADR	0x 1f a1 02 04	R	CPU cache refill parity error address	6
GIOADR	0x 1f a1 02 08	R	DMA slave/VME slave error address	6
BURST	0x 1f a2 00 08	R/W	Arbiter burst register	8
DELAY	0x 1f a2 00 0c	R/W	Arbiter delay register	8
3WAYTRIGGER	0x 1f a7 00 00	R/W	3-way transfer trigger data	32
3WAYSTART	0x 1f a8 00 04	R	3-way transfer start address	32
3WAYMASK	0x 1f a8 00 08	R/W	3-way address mask	29
3WAYSUBS	0x 1f a8 00 0c	R/W	3-way address substitution register	29
VMERMW	0x 1f a9 00 00	W	Set VME RMW flag	
DABR	0x 1f aa 00 00	R/W	Graphics DMA descriptor array base addr	32
BUFADR	0x 1f aa 00 04	R	Graphics stride/width register	32
WIDTH	0x 1f aa 00 08	R	DMA setup register	16
DESTADR	0x 1f aa 00 0c	R	Graphics operand address	32
STRIDE	0x 1f aa 00 10	R	DMA address stride and line count	32
STRTDMA	0x 1f aa 01 00	W	Start DMA	

Table 8-5 (continued) CPU, Main Memory, and Boot PROM Registers

CPUCTRL

CPUCTRL, the primary CPU control register, resides at address 0x1fa00000 and uses the following bit assignments. All 16 bits are cleared to zero by *init*(1M).

15	14	13	12	11	10	9	8
GRR\	DOG	BAD	ARB	SLA	PAR	SIN	GR1\
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VTE	GIOFBT	GSE	GDE	IBE	DBE	X	RFE

RFE	Refresh enable
DBE	Data block refill enable
IBE	Instruction block refill enable
GSE	Graphics DMA sync enable (enables GFX.DMASYNC\)
GIOFBT	GIO fast bus time out, for test vectors
VTE	VMEbus timer enable
GDE	Graphics DMA complete interrupt enable
GR1\	GR1 compatibility mode when set to 0
SIN	SYSINIT – Force the RESETOUT\ line to be asserted. Note that RESETOUT\ line causes the INIT\ line to be asserted which resets the entire IP12 including the MIPS processor. The effect of setting this bit is similar to powering the system up and down.
PAR	Enable parity checking and error reporting on reads from the local memory. When cleared parity errors are ignored.
SLA	Slave – Allow VMEbus masters to access the IP12 memory as a slave. Otherwise all VMEbus access attempts are ignored.
ARB	Clear to enable the VME arbiter in PIC1 for a single external request level. Set to disable PIC1 VME arbiter (for CPU with external arbiter)
BAD	GENBAD – Force bad parity to be written into local memory.

DOG — ENABWDOG – Set to high to enable watchdog timer. If enabled for the full timeout period, the timer activates the WDOG signal which forces initialization of the VMEbus and IP6. Set ENABWDOG to low to reset the watchdog timer. While ENABWDOG is low, timeout is disabled (forced to this state after system alert).

GRR\ — Clear to assert reset to graphics subsystem.

RSTCONFIG

This register resides at address 0x1fa00004. It is set during the reset sequence and contains a subset of the W,X,Y,Z reset cycle mode bits of the R3000. The 8 variables described below are made available to the HPC1 and INT2 via the peripheral bus (through appropriate pullups and pulldowns). The HPC1 will echo these values on the GIO bus during reset so that PIC1 can latch them on the trailing edge of reset. The only bits in this register which actually affect PIC1 operation are the instruction and data block refill sizes.

This register uses the following bit assignments:

7	6	5	4	3	2	1	0
BDRV	STP	NCSH\	ISTRM	IB1	IB0	DB1	DB0

BDRV — Bus-drive ON

STP — Store partial enable

NCSH\ — R3000 no cache mode

ISTRM — Instruction streaming enable

IB1, IB0 — Instruction block size: 32, 16, 8, 4 (used by PIC1)

DB1, DB0 — Data block size: 32, 16, 8, 4 (used by PIC1)

SYSID

This register resides at 0x1fa00008 and is the CPU status register. It is a read only register that provides status and configuration information.

This register uses the following bit assignments:

8	7	6	5	4	3	2	1	0
REV2	REV1	REV0	VMERMW	X	DMAIDLE	DMAERR	X	FPRESS

REV2 — PIC1 chip revision bit 2.

REV1 — PIC1 chip revision bit 1.

REV0 — PIC1 chip revision bit 0.

VMERMW — Asserted during a VME read-modify-write cycle.

DMAIDLE — Asserted at the end of DMA transfer.

DMAERR — Asserted if the DMA terminated in an error condition.

FPPRES — Floating point co-processor present indicator.

MEMCFG0, MEMCFG1

MEMCFG0 resides at 0x1fa10000 and MEMCFG1 resides at 0x1fa10004. They are used for programming the size and location of each memory bank. These registers default to 0 on power-up and must be set to the appropriate value based on the physical memory module configuration.

The format of these registers is shown below.

MEMCFG0

31 30 29 28	27 26 25 24	23 22	21 20 19 18 17 16
X	MSIZE0	X	BASE0
15 14 13 12	11 10 9 8	7 6	5 4 3 2 1 0
X	MSIZE1	X	BASE1

MEMCFG1

31 30 29 28	27 26 25 24	23 22	21 20 19 18 17 16
X	MSIZE2	X	BASE2
15 14 13 12	11 10 9 8	7 6	5 4 3 2 1 0
X	MSIZE3	X	BASE3

MSIZEn This nibble sets the size of module n
0000 – 4 MByte
0001 – 8 MByte
0011 – 16 MByte
0111 – 32 MByte
1111 – 64 MByte
(Note: Other settings are invalid and have undefined effects.)

BASEn This 6-bit field specifies the high order bits of the base address of module n. Bits 27 to 22 of the address are compared with bits 5 to 0 of the BASE register.

REFTIM

REFTIM resides at 0x1fa10100. The refresh clock divider and timer generate a variety of slow timing signals within PIC1. The clock divider produces a refresh request every 64usec. If the refresh is enabled (the RFE is set in CPUCTRL) then a refresh cycle follows (after a bus acquisition delay).

The entire counter is accessible at the REFTIM address.

PARERR, CLERERR

PARERR resides at 0x1fa10200 and CLERERR resides at 0x1fa10210. The IP12 checks parity during reads from its local memory. When a parity error occurs the parity error register is loaded with information on which byte(s) had a parity error. The register also has sticky bits that identify the kind of parity errors that have occurred. The parity error register is cleared by writing to CLRERR.

Bit assignments in the parity register are as follows:

Bits	7	6	5	4
Parity Bits	3	2	1	0
Bits	3	2	1	0
Access Types	VME	CPU	DMA	GDMA

CPUADR, GIOADR

CPUADR resides at 0x1fa10 04 and GIOADR resides at 0x1fa10208.

Each error address register is loaded when the corresponding bus master receives a parity error. The CPU should use the flags in PARERR to decide which error address to look at.

GIO Bus Operation

The GIO bus is a synchronous, multiplexed address-data, burst mode bus operating at 25 Mhz. The GIO bus interconnects the HPC1, PIC1, and memory, and also provides a connection to the graphic subsystems through a set of registered transceivers. The two levels of registers on either end of the board-to-board interconnect cable add two clocks of latency to transfers between the CPU and the graphics subsystem. The bus protocols are designed to support transfer of data at a maximum sustained rate of one word per clock in spite of the four clock round-trip delay between boards, by assuming that each board can buffer at least two words when the pipeline stalls.

For additional information, please consult the GIO Bus specifications.

Name	Address	R/W	Description	Bits
BURST	0x 1f a2 00 08	R/W	Arbiter burst register	8
DELAY	0x 1f a2 00 0c	R/W	Arbiter delay register	8

Table 8-2 GIO Bus Registers

Graphics Channel

The graphics interface provides a DMA channel for transfers between the graphics subsystem and the system main memory. The graphics interface is capable of running in TG or ELAN mode. In ELAN mode the GIO Bus DMA features are enabled.

For additional information, please consult the GIO Bus specifications.

Name	Address	R/W	Description	Bits
DABR	0x 1f aa 00 00	R/W	Graphics DMA descriptor array base addr	32
BUFADR	0x 1f aa 00 04	R	Graphics stride/width register	32
WIDTH	0x 1f aa 00 08	R	DMA setup register	16
DESTADR	0x 1f aa 00 0c	R	Graphics operand address	32
STRIDE	0x 1f aa 00 10	R	DMA address stride and line count	32
STRTDMA	0x 1f aa 01 00	W	Start DMA	

Table 8-3 Graphics Channel Registers

The following registers are provided to work the DMA interface:

0-27= DABR (27:0)
28 - 31 = X

Note that the DABR hold a full physical address of the base of the table.

Channel Registers

The descriptor is copied into the channel registers shown below by the DMA channel as it chains through the descriptor array (except for the BURST field which is loaded statically by the CPU). DMA bursts are regulated by the burst and delay length register, discussed previously.

The channel register format is as follows:

BUFADR (31:0)							
SINC	UNUSED (30:28)	GFXSTRIDE (27:16)	L	LINC	MODE1	MODE2	WIDTH (11:0)
GRXADR (31:0)							
UNUSED (31:28)		STRIDE (27:16)	LINE COUNT (15:0)				
NEXTDESC (31:0)							

L	1 indicates end of list
SINC	Graphics stride increment enable bit
LINC	Graphics line increment enable bit
GFXSTRIDE	Stride for graphics address domain
MODE (1 : 0)	DMA mode: 00 dma read 10 dma write 01 accumulation dma
WIDTH	Width in bytes of DMA scan line
STRIDE	Host address offset for rectangular DMA
LINECOUNT	Counter for scan lines
NEXTDESC	Pointer to the next graphics descriptor.

3-Way Transfer

The three way transfer is a mechanism for moving blocks of data to the graphics subsystem without invoking the operating system kernel. Since a kernel call implies a significant software overhead, the three way transfer is especially effective for small transfers. For larger transfers, the more traditional DMA approach is sufficient since the initial kernel overhead is a smaller percentage of the entire transfer.

In its simplest form, a three way transfer occurs in five steps:

1. Read the data word at the start address of the block.
2. Read the data word at the end address of the block.
3. Write the destination address to a special trigger address inside PIC1.
4. Write the previously saved data to the start address of the block.
5. Write the previously saved data to the end address of the block.

Name	Address	R/W	Description	Bits
3WAYTRIGGER	0x 1f a7 00 00	R/W	3-way transfer trigger data	32
3WAYSTART	0x 1f a8 00 04	R	3-way transfer start address	32
3WAYMASK	0x 1f a8 00 08	R/W	3-way address mask	29
3WAYSUBS	0x 1f a8 00 0c	R/W	3-way address substitution register	29

Table 8-4 3-Way Transfer Registers

Bit assignments are outlined below.

3WAYTRIGGER

0 - 28 = DESTINATION (28:0)
29, 30 = X
31 = ARM

3WAYSTART

0 - 27 = STARTADDR (27:0)
28 30 = X
31 = STARTVALID

ARM	Set to zero, disarms 3-way transfers. Set to one means armed.
DESTINATION	29 bits of destination address to the graphics subsystem. This address is used by the graphics system to identify GL tokens, frame buffer addresses, color map address, etc.
STARTADDR	28 bits of physical start address.
STARTVALID	One means start address was written. Zero means start address has not been written yet.

3WAYSUBS/3WAYMASK

0 - 28 = ADDRESS/MASK (28:0)
29 - 31 = X

VMEbus Interface Registers

The V30/35/35 supports access from VME bus masters to local memory in either single or block transfer mode with either A24 addressing or A32 addressing. DIP switches on the CPU board allow you to select between A24 and A32 addressing. See Appendix D for information on the DIP switches.

In A24 mode, only the low 8 Mbytes of local memory are accessible to VME masters in non-privileged mode. There is no privileged access to local memory in A24 mode.

In A32 mode, the entire local memory is accessible to VME masters in either privileged or non- privileged mode.

PIC1 controls a bi-directional VME interface which allows CPU access to VME-resident devices and allows VME-resident bus masters access to local memory (including support for block transfers).

Serial Ports (DUARTs)

Table 8-5 outlines the serial port control registers.

Name	Address	R/W	Description
DUART0	0x 1f b8 0d 0n	R/W	DUART 0 (Zilog 8530) address where: n=3 for command to /from DUART pin B 7 for data to/from DUART pin B B for command to/from DUART pin A F for data to/from DUART pin A (for accessing an 8-bit DUART "byte")
DUART1	0x 1f b8 0 d 1n	R/W	DUART 1 (Zilog 8530) address "n" as specified in DUART0
DUART2	0x 1f b8 0d 2n	R/W	DUART 2 (Zilog 8530) address "n" as specified in DUART0
DUART3	0x 1f b8 0d 3n	R/W	DUART 3 (Zilog 8530) address "n" as specified in DUART0

Table 8-5 Serial Port DUART Registers

Note: The V30/35 uses the Zilog 85130 DUART microchip as the serial port controller. For additional information, please consult the microchip specifications.

Ethernet Controller Register

The Ethernet interface is designed to support Ethernet connection at a through-put rate close to the maximum Ethernet rate of 10 MBits/sec using minimum GIO bandwidth and CPU intervention.

Table 8-6 outlines the Ethernet control registers.

Name	Address	R/W	Description
RESERVED	0x 1f b8 00 00	W	Reserved - do not use.
RESERVED	0x 1f b8 00 04	W	Reserved - do not use.
ENET.X.COUNT	0x 1f b8 00 08	W	Reserved - for debug purpose only.
ENET.CXBP	0x 1f b8 00 0C	W	ENET Current transmit buffer pointer, EOX.
ENET.NXBDP	0x 1f b8 00 10	W	ENET Next transmit buffer descriptor pointer.
ENET.XBC	0x 1f b8 00 14	W	ENET transmit byte count register.
ENET.X.PNTR	0x 1f b8 00 18	W	Reserved - for debug purpose only.
ENET.X.FIFO	0x 1f b8 00 1C	W	Reserved - for debug purpose only.
ENET.CXBDP	0x 1f b8 00 20	W	ENET Current transmit buffer pointer, EOX.
ENET.CPFXBDP	0x 1f b8 00 24	W	ENET Current packet first transmit buffer descriptor pointer.
ENET.PPFXBDP	0x 1f b8 00 28	W	ENET Previous packet first transmit buffer descriptor pointer.
ENET.Timer	0x 1f b8 00 2C	W	Timer register.
RESERVED	0x 1f b8 00 30	W	Reserved - for debug purpose

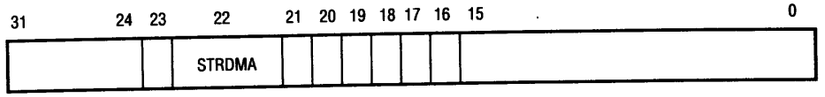
Table 8-6 Ethernet Control Registers

Name	Address	R/W	Description
ENET.TRSTAT	0x 1f b8 00 34	W	ENET Transmitter status. DATA BITS 23 to 16.
ENET.RCVSTAT	0x 1f b8 00 38	W	ENET Receiver Status. DATA BITS 15 to 8.
ENET.RESET	0x 1f b8 00 3C	W	ENET RESET (bit 0), CLRINT (bit 1), LPBK (bit 2), and RxBuf Overflow (bit 3). DATA BITS 7 to 0.
RESERVED	0x 1f b8 00 40	W	Reserved - do not use.ENET.R.COUNT0x 1f b8 00 44WReserved - for debug purpose only.
ENET.R.BC	0x 1f b8 00 48	W	Receive byte count.
ENET.R.CBP	0x 1f b8 00 4C	W	Receiver current buffer pointer & 'EOR'.
ENET.R.NRBDP	0x 1f b8 00 50	W	Next receive buffer descriptor pointer.
ENET.R.CRBDP	0x 1f b8 00 54	W	Current receive buffer descriptor pointer.
ENET.R.PNTR	0x 1f b8 00 58	W	Reserved - for debug purpose only.
ENET.R.FIFO	0x 1f b8 00 5C	W	Reserved - for debug purpose only.
RESERVED	0x 1f b8 00 60	W	Reserved - do not use. to 7F
ENET.EXTREG	0x 1f b8 01 nn	W	ENET External registers, where nn= 00, 04, 08, 0C, 10, 14, 18, 1C. GIO DATA BITS 7 to 0.

Table 8-6 (continued) Ethernet Control Registers

Note: The V30/35 uses a SEEQ 8003 microchip as the Ethernet controller. For additional information, please consult the microchip specifications.

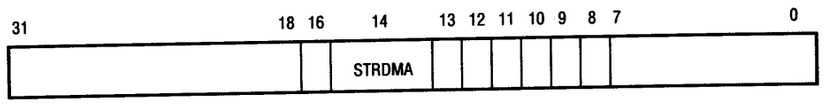
Transmit Status (ENET.TRSTAT)



STTRDMA — Bit 22 is start the transmit DMA. Writing a '1' starts the transmission section of the HPC. Once HPC reads the descriptor with EOT, it will reset this bit. If there is transmit underflow or 16 collisions for a packet then also, this bit will be reset to 0.

TRSTAT — Bits 16 to 23 are the transmit status bits.

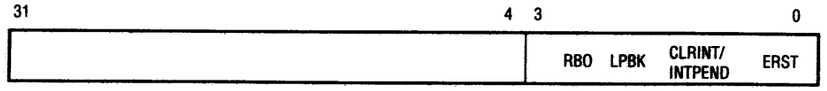
Receive Status (ENET.RCVSTAT)



STRCVDMA — Bit 14 is start the receiver DMA. Writing a '1' starts the receiver section of the HPC. Once HPC reads the descriptor with EOR, it will reset this bit. If the end of packet is not received within RBC receive byte count, then also this bit will reset to 0. If receiver buffer overflow is true, HPC clears this bit.

RCVSTAT — Bits 8 to 15 are the receive status bits.

Reset Status (ENET.RESET)



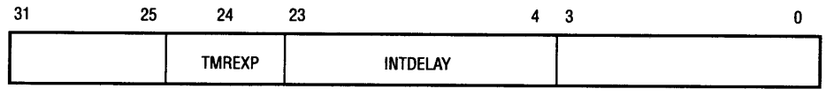
ERST — Bit 0 is Ethernet channel reset. Resets the 8003 as well as Ethernet interface in the HPC.

INTPEND/CLRINT — Bit 1 is Interrupt pending/clear. Whenever host has an interrupt pending, this will be a 1. Writing a 1 clears the interrupt.

LPBK_ — Bit 2 is loopback. This bit directly controls the LPBK_ pin on the 8020. A '0' in this bit puts the Ethernet in loopback mode. Should be a '1' for normal operation.

RBO — Bit 3 is receive byte overflow. If a packet has more than 'RBC' bytes, this will be set to 1. The host clears by writing a 1; writing a 0 has no effect.

Timer Status (ENET.TIMER)



INTDELAYT — Bits 4 to 23 are the interrupt delay count. This value is counted before some interrupts are generated.

TMREXP — Bit 24 is delay count expired. When the delay count is reached, this bit is set to one.

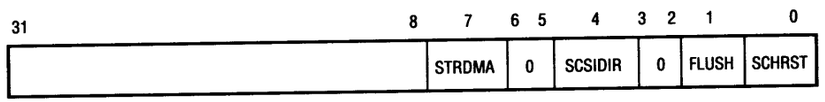
SCSI Controller Registers

Table 8-7 outlines the SCSI control registers.

Name	Address	R/W	Description
RESERVED	0x1f b800 80	W	Reserved - do not use.
SCSI.COUNT	0x1f b800 84	W	Reserved - debug purpose only.
SCSI.BC	0x1f b800 88	W	Byte count.
SCSI.CBP	0x1f b800 8C	W	SCSI Current buffer pointer & 1 EOX.
SCSI.NBDP	0x1f b800 90	W	SCSI Next buffer descriptor pointer.
SCSI.CNTL	0x1f b800 94	W	SCSI control register.
SCSI.PNTR	0x1f b800 98	W	Reserved - debug purpose only.
SCSI.FIFO	0x1f b800 9C	W	Reserved - debug purpose only.
SCSI.EXTREG	0x1f b801 2n	W	SCSI External registers, where n=0, 4. GIO DATA BITS 15 to 8.

Table 8-7 SCSI control Registers

The HPC has a SCSI control register which manages the DMA. It has four bits.



SCHRST — SCSI channel reset. Resets the SCSI bus as well as SCSI interface internal to HPC.

SCSIDIR — SCSI direction bit controls direction of transfer.
1 = SCSI to memory
0 = memory to SCSI.

STRTDMA — 0 to 1 starts the DMA. It also resets all FIFO pointers. When HPC sees EOX=1 it clears this bit after the last data is read from or written to the memory. If flush is set to '1', HPC flushes the bytes to memory and clears this 'bit'.

FLUSH — When set to 1, flushes the remaining words to memory and HPC clears STRTDMA bit once all the bytes have been flushed to memory.

Note: The V30/35 uses a Western Digital 33C93A microchip as the SCSI controller. For additional information, please consult the microchip specifications.

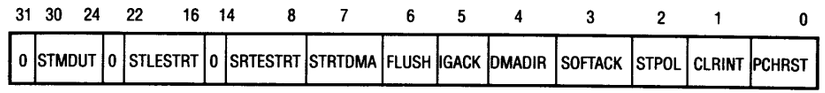
Centronics Controller Registers

Table 8-8 outlines the Centronics control registers.

Name	Address	R/W	Description
RESERVED	0x1f b800 a0	W	Reserved - do no use.
PARALLEL.COUNT	0x1f b800 a4	W	Reserved - debug purpose only.
PARALLEL.BC	0x1f b800 a8	W	Byte count.
PARALLEL.CBP	0x1fb 800 ac	W	Current buffer pointer & "EOX".
PARALLEL.NBDP	0x1f b800 b0	W	Next buffer descriptor pointer.
PARALLEL.CNTL	0x1f b800 b4	W	Parallel control register.
PARALLEL.PNTR	0x1f b800 b8	W	Reserved - debug purpose only.
PARALLEL.FIFO	0x1f b800 bc	W	Reserved - debug purpose only.
PARALLEL.EXTREG	0x1f b801 34	W	Parallel external reads from status. Writes to remote. GIO DATA BITS 23 to 16.

Table 8-8 Centronics Control Registers

The HPC PARALLEL-PORT control register fields are illustrated below.



PCHRST — PARALLEL-PORT channel reset. Resets the PARALLEL-PORT interface internal to Parallel port. when this is active all the enables will be inactive. 1=reset; 0=ready.

CLRINT — Writing a '1' clears a interrupt if it is pending. Reading this bit tells whether a interrupt is pending or not.

STOBE POLARITY
"0" active low. "1" active high. Applies only when the direction is from memory to HPC.

SOFTACK — Normally should be a 1. A negative pulse acts as soft ack. This bit should be toggled only when DMA is stopprd.

DMA DIRECTION
"1" = Parallel Port to memory. '0' = memory to parallel-port. This bit should be changed only when DMA is stopprd.

IGNORE ACK

FLUSH — 1 flushes the remaining words to memory and HPC clears

STRTDMA — Bit when all bytes are flushed to memory. once the flush is done the dma has to be restarted in order to receive more data. This bit should be cleared before making STRTDMA bit a '1' again. STRTDMA - Writing a '1' starts the DMA. When EOX=1, clears this bit after the DMA is done. When FLUSH=1, clears this bit after the DMA is done

Appendix A

Operating Conditions

The V30/35 is designed to operate in accordance with the standards for any VME-based equipment. Operating temperatures must fall between 0°C to 55°C; for safe storage, between –40°C to 70°C. Relative humidity during operation must fall within 10% and 95%, noncondensing.

The CPU subsystem requires a maximum of 13.5 amps at 5 volts, plus or minus 5 amps at 12 volts; typical values will be less.

Appendix B

Routine Maintenance

Replacing the Clock Battery

The clock battery should be replaced every 3 years. It is a 3V lithium button cell (Duracell® model number DL2450), approximately 1 inch in diameter, located on the right side of the CPU subsystem's I/O board. The battery is held in place with a clip, which also acts as a terminal for the battery connection. To replace the battery:

1. Power down the system and remove the CPU subsystem from the card cage.
2. Grasp the battery firmly and lift the end not blocked by the terminal clip up and away from the board, being careful not to damage the clip. See Figure B-1(A).
3. Slide the battery out from under the clip. See Figure B-1(B).
4. Slide the new battery under the clip and into the housing.

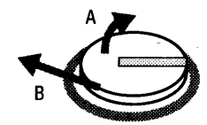


Figure B-1 Removing the Clock Battery

Appendix C

Pinouts for Peripheral Connectors

Table C-1 lists pinout data for the various peripheral connectors; all connectors are female. Table C-2 shows the pinouts for rows A and C of the P2 VMEbus connector on the I/O board.

Connector	Pin	Signal	Pin	Signal
J1 & J2: Serial Ports	1	NC	2	TXD -
General Purpose	3	RXD -	4	RTS -
9-pin	5	CTS -	6	NC
D-Subminiature	7	GND	8	DCD -
	9	DTR -		
J3 & J4: Serial Ports	1	HSKO	2	HSKI
Apple Type	3	TXD -	4	GND
8-pin D-Sub	5	RXD -	6	TXD +
	7	GPI	8	RXD +
J5: MISC I/O	1	1_HSKO	2	1_HSKI
26-pin D Mini-sub	3	1_TXD -	4	1_GND
with latchblocks	5	1_RXD -	6	1_TXD +
	7	1_GPI	8	1_RXD +
	9	LEDCOM	10	LED1 (Power)
	11	LED2 (Fault)	12	LED3 (Cache error)
	13	LED4 (Heartbeat)	14	2_HSKO
	15	2_HSKI	16	2_TXD -
	17	2_GND	18	2_RXD -
	19	2_TXD +	20	2_GPI
	21	2_RXD +	22	GND
	23	RESETSW	24	GND
	25	GND	26	GND

Table C-1 Pinouts for Peripheral Connectors

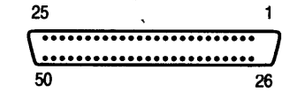


Figure C-1 SCSI Connector Pin Numbering (front view)

Connector	Pin	Signal	Pin	Signal
J6: Centronics/Parallel	1	STB -	2	DATA0
25-Pin High Density	3	DATA1	4	DATA2
D-Sub	5	DATA3	6	DATA4
(IBM-compatible)	7	DATA5	8	DATA6
	9	DATA7	10	ACK -
	11	BUSY -	12	NOPAPER
	13	ONLINE	14	PRT
	15	FAULT -	16	RESET -
	17	NOINK	18-25	GND
J7: SCSI	1-11	GND	12	reserved (nc)
50-Pin	13	open (nc)	14	reserved (nc)
D Mini-sub	15-25	GND	26	DB(0)
with latchblocks	27	DB(1)	28	DB(2)
	29	DB(3)	30	DB(4)
	31	DB(5)	32	DB(6)
	33	DB(7)	34	DB(P)
	35	GND	36	GND
	37	reserved (nc)	38	TERMPWR
	39	reserved (nc)	40	GND
	41	ATN	42	GND
	43	BSY	44	ACK
	45	RST	46	MSG
	47	SEL	48	C/D
	49	REQ	50	I/O
J8: Ethernet	1	SHIELD GROUND	2	COLLISION P
15-Pin D-Sub	3	TRANSMIT P	4	GND
	5	RECEIVE P	6	GND
	7	NC	8	GND
	9	COLLISION M	10	TRANSMIT M
	11	GND	12	RECEIVE M
	13	+12V (fused)	14	GND
	15	NC		
J9: Keyboard/Mouse	1	NC	2	KBD.RXD -
9-Pin D-Sub	3	NC	4	- 8VM
	5	MOUSE.RXD -	6	GND
	7	+ 8 VP	8	KBD.TXD -
	9	GND		

Table C-1 (continued) Pinouts for Peripheral Connectors

Pin #	Row A	Row B	Row C
1	GND		GND
2	GR1.CLK		GR1.PRECLK
3	GND		GND
4	GFX.DMASYNC -		GR1.OENSTB
5	GFX.STB -		GFX.READ
6	GND		GND
7	GFX.GR1BURST -		GFX.VERT.INT -
8	GND		GFX.VERT.STAT -
9	GFX.MASDLY		GFX.INT -
10	GFX.GFXDLY		GFX.FIFO.FULL -
11	GND		GND
12	GFX.DATA0		GFX.DATA16
13	GFX.DATA1		GFX.DATA17
14	GFX.DATA2		GFX.DATA18
15	GFX.DATA3		GFX.DATA19
16	GND		GND
17	GFX.DATA4		GFX.DATA20
18	GFX.DATA5		GFX.DATA21
19	GFX.DATA6		GFX.DATA22
20	GFX.DATA7		GFX.DATA23
21	GND		GND
22	GFX.DATA8		GFX.DATA24
23	GFX.DATA9		GFX.DATA25
24	GFX.DATA10		GFX.DATA26
25	GFX.DATA11		GFX.DATA27
26	GND		GND
27	GFX.DATA12		GFX.DATA28
28	GFX.DATA13		GFX.DATA29
29	GFX.DATA14		GFX.DATA30
30	GFX.DATA15		GFX.DATA31
31			
32	GND		GFX.GFXRST -

Table C-2 I/O Board P2 VMEbus Connector

Appendix D

Switches, Jumpers, and Memory Maps

The CPU subsystem must be configured for the graphics subsystem you are using. The configuration of the subsystem is determined by a jumper on the I/O board and both a jumper and an oscillator on the CPU board. This appendix outlines the appropriate oscillator and jumper positions for the TG - V and Elan/XS - V graphics subsystems.

In addition, the CPU subsystem provides two banks of DIP switches that enable or disable hardware options and specify memory maps for the VMEbus address space. This appendix also describes these switches and their associated options.

Note: Information in this Appendix applies only to a specific version of the CPU subsystem. The version ID numbers can be located on the CPU and I/O boards by referring to Appendix E, "V30/35 Component Layout." The I/O board version ID number should be 030-5024-002. The CPU board version ID number should be 030-5023-002. If any other version ID numbers appear on either board, please contact your support representative or sales representative for further information.

Jumpers

The CPU and I/O boards each have a jumper that is used by the V30/35 to configure the graphics subsystem.

Table D-1 outlines the jumper positions for TG - V graphics, Elan/XS - V graphics, and no installed graphics. Figures D-1 and D-2 illustrate the location of the jumpers and their default positions.

If you are installing an Elan/XS - V subsystem, verify that the jumper is in the correct default position.

If you are installing a TG - V graphics subsystem, you must:

1. Move the CPU board jumper to pins 2 and 3.
2. Add a jumper to the I/O board. The jumper comes packaged with the TG - V subsystem.

Warning: **Only the jumper positions outlined in Table D-3 and illustrated in Figures D-2 and D-3 are recommended. Do not place the jumpers in any other positions.**

CPU Board	I/O Board	Configuration
1 - 2	2 - 3	no installed graphics
1 - 2	none	Elan/XS - V
2 - 3	1 - 2	TG - V or no installed graphics

Table D-1 Jumper positions.

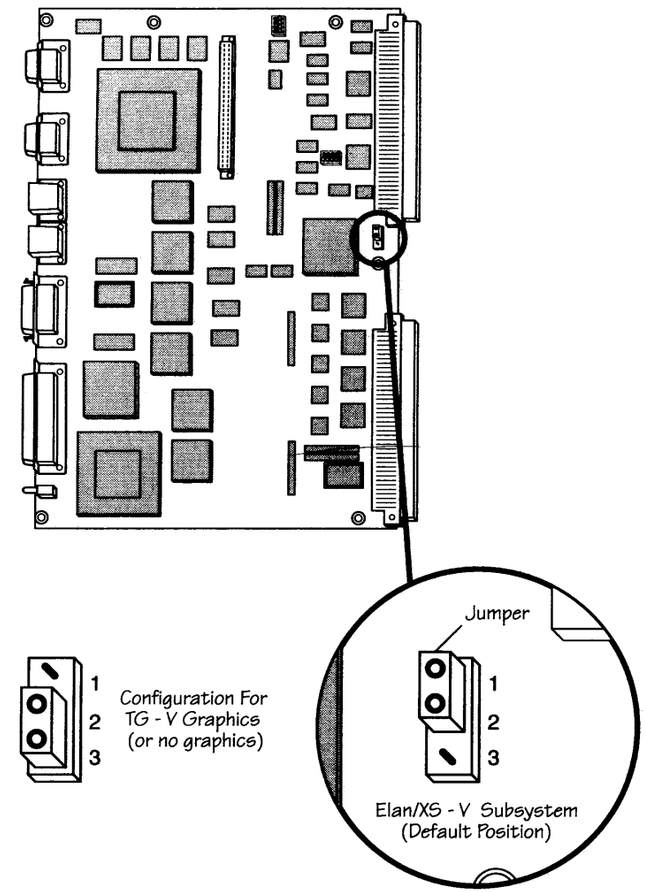


Figure D-1 CPU Board Jumpers

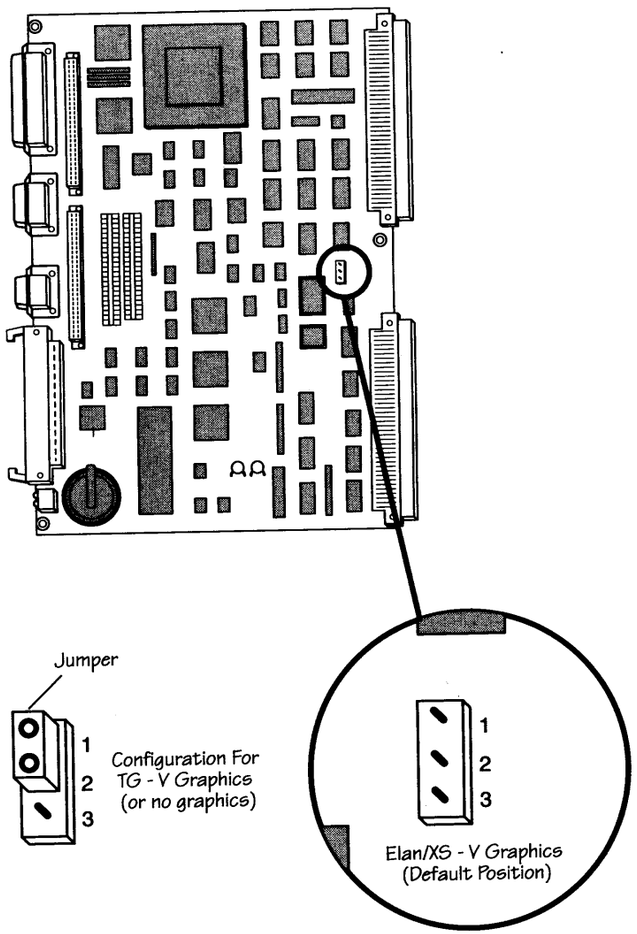


Figure D-2 I/O Board Jumpers

Oscillator

If you are installing a Elan/XS - V graphics subsystem with the V30/35 CPU subsystem, you should use the 50 MHz oscillator that is pre-installed on the CPU board.

If you are installing a TG - V graphics subsystem, you must replace the oscillator with a 60 MHz version that was packaged with the TG - V subsystem.

Figure D-3 illustrates the location of the oscillator on the CPU board.

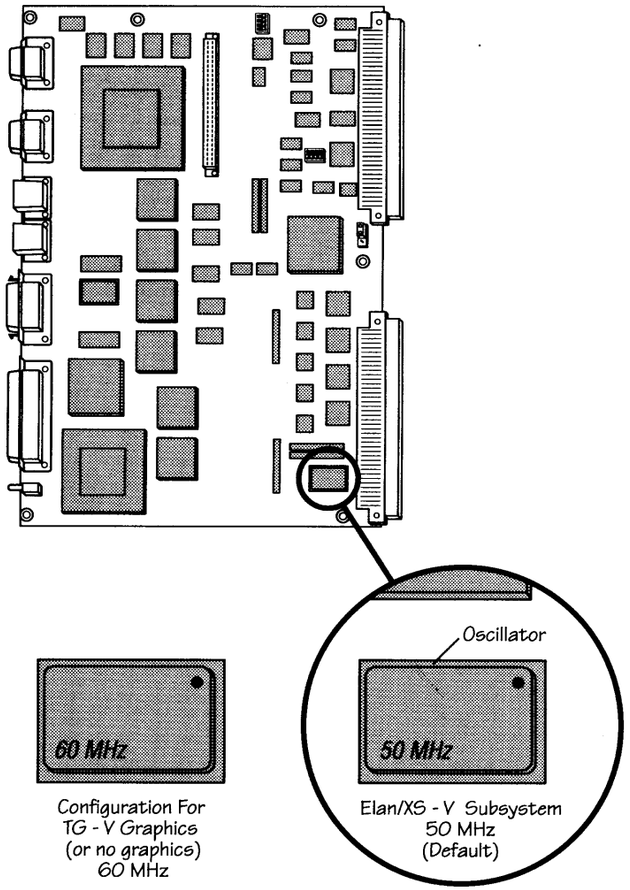


Figure D-3 Oscillator for Graphics Subsystem

CPU Switches

The CPU switch banks, S1 and S2, are located on the component side of the top board in the CPU subsystem as shown in Figure D-4. The two switch banks each contain four positions. Their various settings and corresponding options are listed in Tables D-2 and D-3.

Switch	Position	Option	Default Position
S1	1	A24 ID1	On
	2	A24 ID0	On
	3	A24 EN1	On
	4	A24 EN0	On
S2	1	Slot 1 enable	On
	2	A30 (Off = 1, On = 0)	On
	3	A29 (Off = 1, On = 0)	On
	4	A28 (Off = 1, On = 0)	On

Table D-2 DIP Switch Positions

Caution: Use of BR[2 : 0] is not supported.

Switch bank S2 is for A32 slave access address comparison and enabling slot 1. The V30/35 only responds to a slave access if VME-ADR[31]=0 and VME-ADR[30 : 28]=A[30:28].

Switch bank S1 functionality is outlined in Table D-3.

A24 EN1	A24 EN0	Slave	
0	0	no A24 access	pass A[23:22]
0	1	non-privileged programmable 4Mb windows	A[23:22] = 00 (map 4Mb window into lower 4Mb window)
1	0	non-privileged lower 8Mb windows	pass A[23:22]
1	1	privileged and non-privileged lower 8Mb windows	pass A[23:22]

Table D-3 Switch Bank S1 Functionality

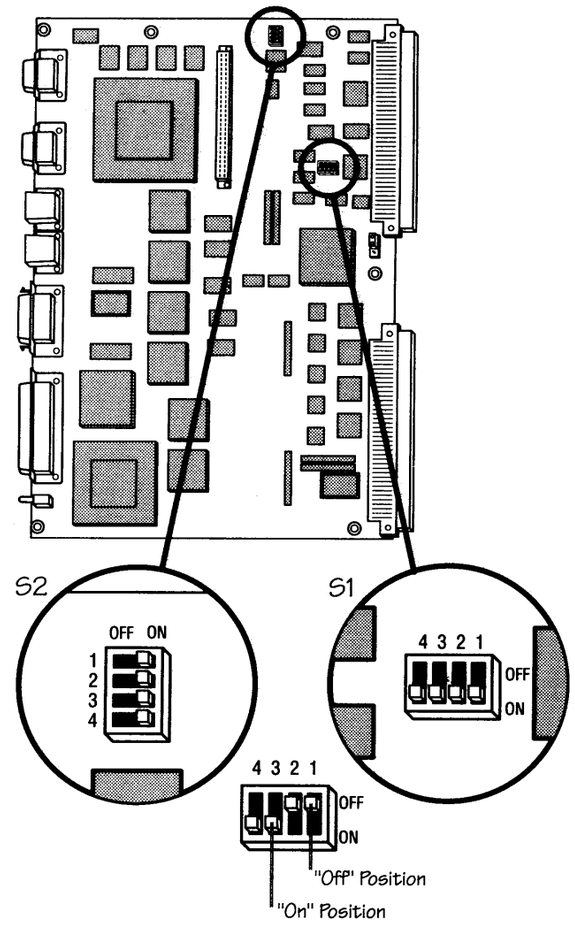


Figure D-4 CPU DIP Switches

Appendix E

V30/35 Component Layout

This appendix illustrates the front panel and circuit boards used in the V30/35 subsystem. See Figures E-1 through E-4.

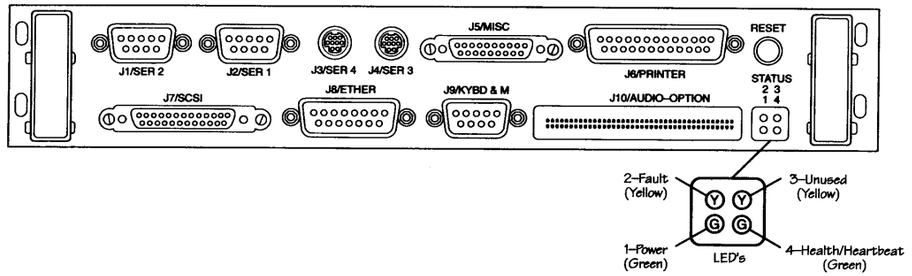


Figure E-1 V30/35 Front Panel

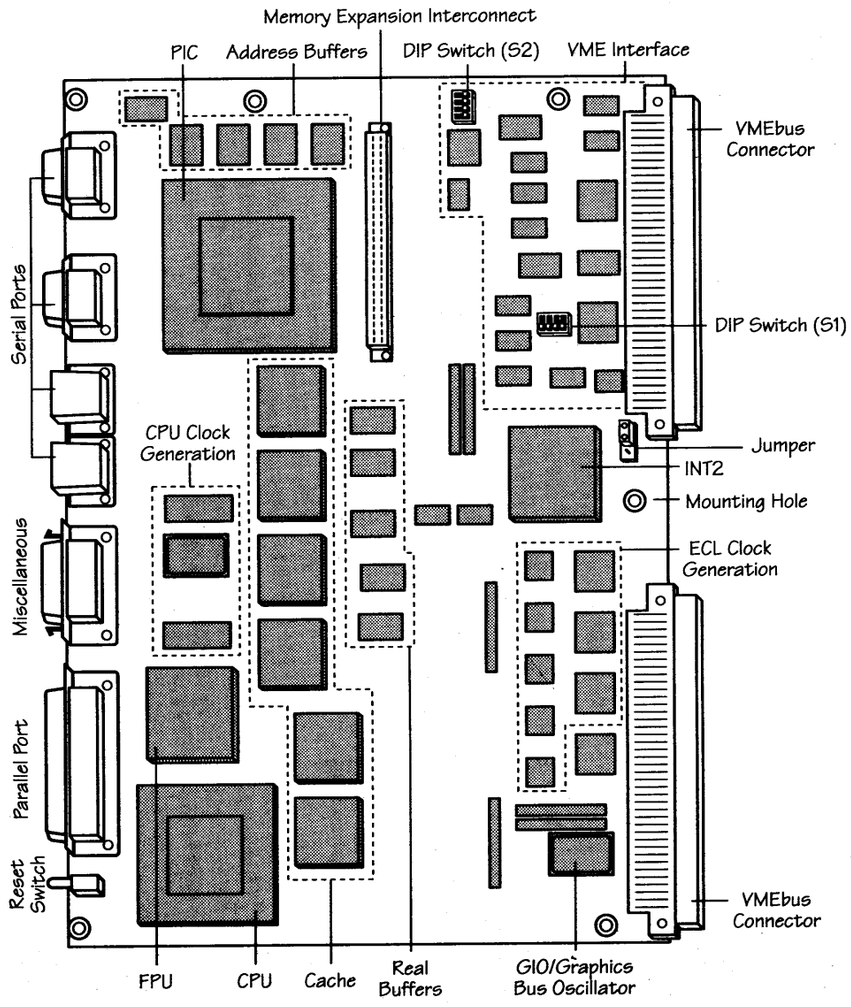


Figure E-2 Topside of CPU Board.

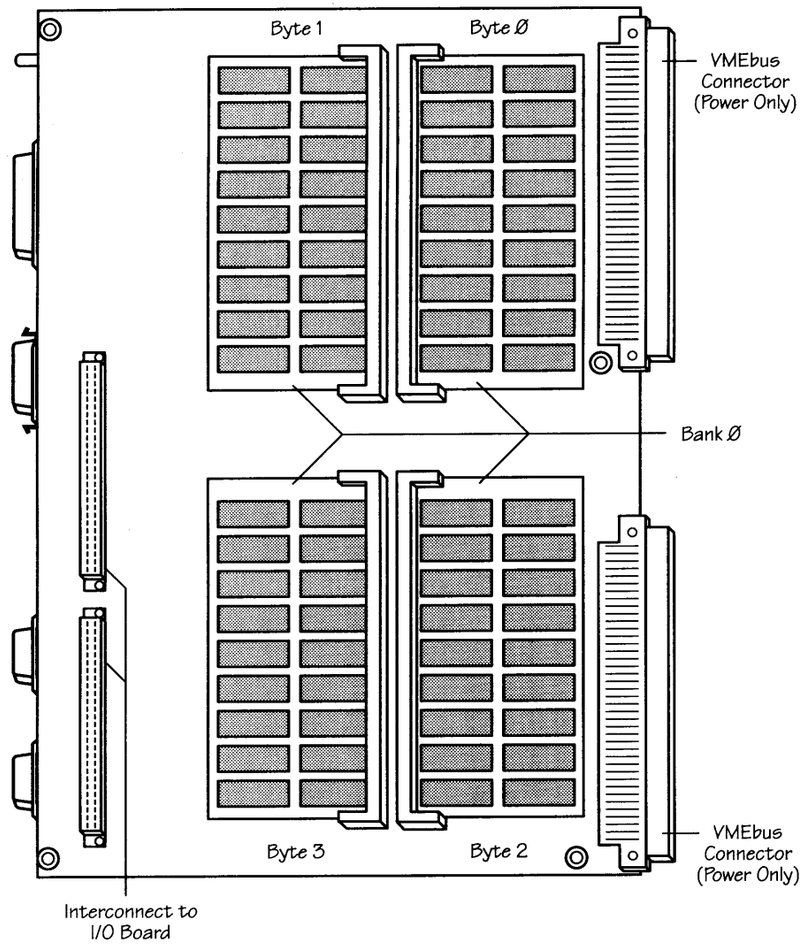


Figure E-3 Bottom of CPU Board.

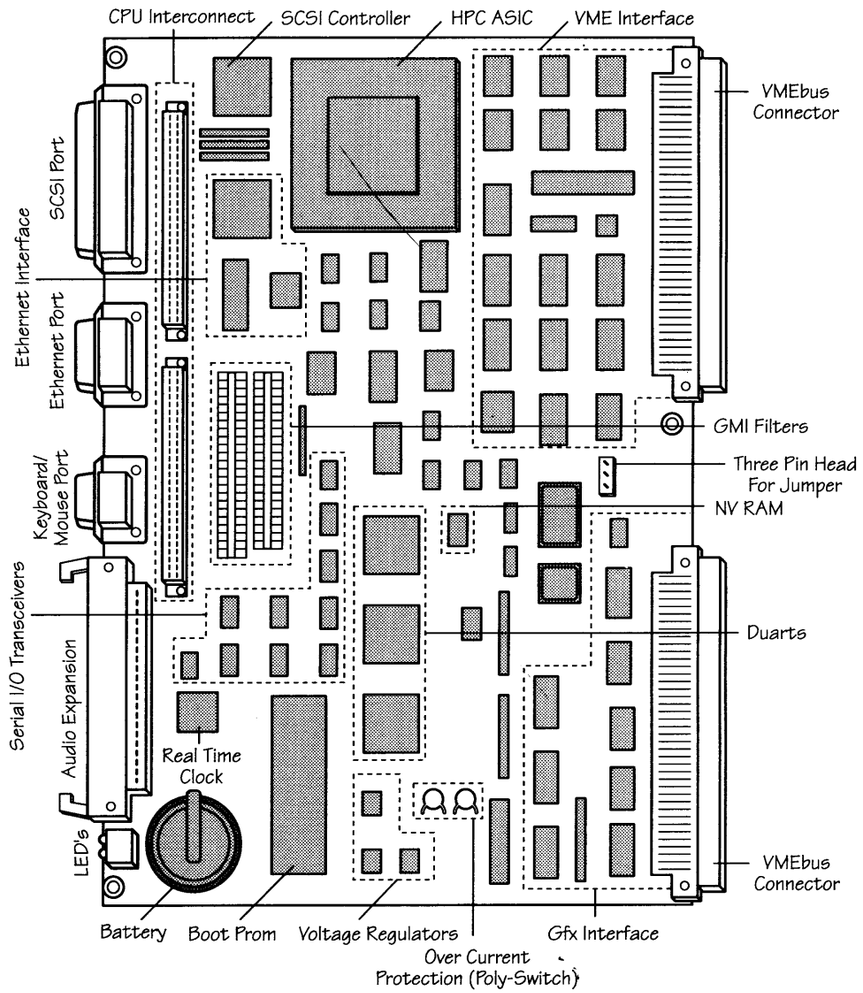


Figure E-4 Topside of I/O Board

Index

F

G

H

I

J

K

M

N

O

P

R

S

T

U

V

W

X