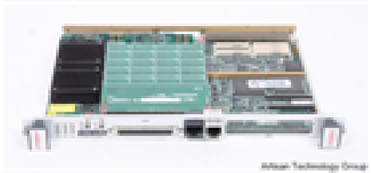


# VGM5

## *VMEbus Dual G3/G4 PowerPC Single Board Computer*

## User Guide

**Revision 5.1** **October 12, 2004**

Curtiss-Wright Controls Embedded Computing
9605 Scranton Road, Suite 700
San Diego, CA 92121-1773
(858) 452-0020 • (858) 452-0060 (FAX)
www.cwembedded.com

# VGM5 User Guide

This document contains material of a proprietary nature to Synergy Microsystems, Inc. All manufacturing, use, and sales rights pertaining to this product are expressly reserved. Distribution of this material does not convey any license or title under any patent or copyright. It is submitted in strict confidence to provide technical information for purchasers of this product or for those considering the purchase of the product. Each recipient, by accepting this document, agrees that its contents will not be disclosed in any manner or any person except to serve this purpose.

Synergy Microsystems, Inc. reserves the right to make changes to the specifications and contents in this document without prior notification. If in doubt, users are urged to consult Synergy to determine whether any such changes have been made.

Synergy products are not intended for use in life support systems or other applications where a failure of the product could result in injury or loss of life. Customers using or selling this product in systems or applications serving such a function do so at their own risk and agree to fully indemnify Synergy Microsystems, Inc. for any and all damages arising from improper use.

This product and associated manuals are sold “AS IS” without implied warranty as to their merchantability or fitness for any particular use. In no event shall Synergy Microsystems, Inc. or anyone involved in the creation, production, or delivery of this product be liable for any direct, incidental, or consequential damages, such as, but not limited to, loss of anticipated profits, benefits, use, or to data resulting from the use of this product or associated manuals or arising out of any breach of warranty. In states that do not allow the exclusion or limitation of direct, incidental, or consequential damages, this limitation may not apply.

PRINTED IN THE USA

# Table of contents

# Manual revision summary

| Revision level | Revision date | Section | Affected chapter/description |
|---|---|---|---|
| 1.0a | 11/24/98 | | 1st preliminary release |
| 1.0b | 5/20/99 | | 2nd preliminary release |
| 1.0c | 10/15/99 | | pre-GA release |
| 1.0d | 11/16/99 | | 2nd pre-GA release |
| 1.0 | 3/21/00 | | GA Release |
| 2.0 | 1/5/00 | *All* | Cleaned up various items as required. |
| | | Section 4 | **Onboard registers**/Added register for P0•PCI Interrupt. |
| | | Appendix C | Added VGM5 Rev. G to revision summary. |
| 3.0 | 4/27/01 | *All* | Cleaned up various items as required. |
| | | Section 2/App. B | Updated Power Specification. |
| | | Section 7 | Added Slave image programming caveat. |
| | | Appendix D | Deleted P0•PCI references. Added PEX3 board layout drawing. |
| 3.1 | 6/6/01 | *All* | Cleaned up various items as required. |
| | | Section 3 | Revised PCI bus discussion. |
| | | Section 4 | Revised address map information. |
| 3.2 | 10/5/01 | *All* | Cleaned up various items as required. |
| | | Section 2 | **Installing the P0 overlay**/Removed all references to passive overlay models. |
| | | Section 4 | **Non-volatile 128K x 8 SRAM**/Revised drawing to show how mark on battery and mark on chip are aligned. Pointed to board layout drawing in Appendix B for battery location. |
| | | Appendix C | Removed VGM5 Rev. G from revision summary. |
| | | Appendix D | Added clarification on PEX3 PCI interrupt scheme. Added Device Number info to PEX3 Type 0 configuration table. |
| 4.0 | 5/17/02 | *All* | Cleaned up various items as required. |
| | | Section 2 | **Installing PMC cards**/All PEX3 info moved to Appendix D. |
| | | Section 4 | **User Flash memory**/Corrected address ranges listed in User Flash bank selection table. |
| | | | **Mailbox**/Added “Programming Notes” chapter describing recommended handling of mailbox writes. |
| | | Appendix D | Revised and cleaned up. Consolidated PEX3 info in this Appendix. |

| Revision level | Revision date | Section | Affected chapter/description |
|---|---|---|---|
| 5.0 | 3/28/03 | *All* | Cleaned up various items as required. |
| | | Section 2 | **Installing a monitor PROM**/Revised J902 configuration jumper diagram to include Boot Flash Write Protect jumper function (see next item below). |
| | | Section 4 | **Boot Flash ROM/DIP EPROM**/Added info describing J902 jumper settings for Boot Flash Write Enable/Disable (applicable only to boards that incorporate ECO specifying this change). |
| | | Appendix D | Revised PCI 9080 basic setup info. Added info on VxWorks BSP PEX3 driver. Put back PSTK stacking adapter info. Revised PMC connector pin assignments. |
| 5.1 | 10/12/04 | *All* | Cleaned up various items as required. |

# 1 Overview

This section introduces the **VGM5** single board computer.

- VGM5 features
- Manual conventions

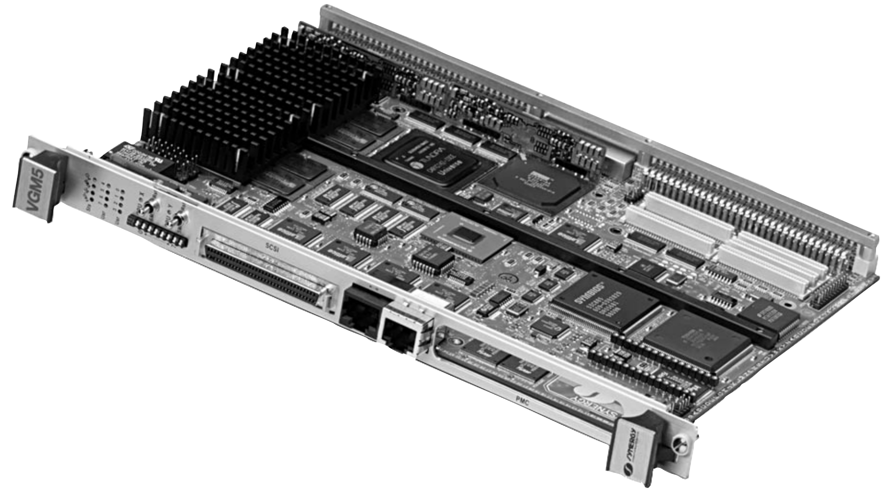
# VGM5 features

Synergy Microsystems' VGM Series is a family of 6U-sized VMEbus Single Board Computers (SBCs) designed from the ground up to run dual PowerPC microprocessors from IBM/Motorola. The VGM Series architecture is optimized for Real-time VME applications. All VGM Series SBCs are provided with a PCI bus mezzanine card slot for easy I/O expansion.

The VGM5 SBC is based on dual G3/G4 PowerPC microprocessors running at up to 400+ MHz. A performance-boosting backside L2 cache (1 MB for G3, 2 MB for G4) running at or near CPU core frequency is standard. An upgradeable DRAM module provides up to 512 MB of high performance SDRAM memory. SCSI, Ethernet, PMC slots, up to 64 MB 8-bit User Flash and 128 KB NVRAM give the VGM5 the flexibility to meet almost any requirement.

The VGM Series architecture supports VME Real-time multi-processor computing. Real-time VME support includes true VME Read-Modify-Write cycle support, a direct low latency PowerPC to VMEbus interface, hot-insertion/extraction support, and VME64x support with up to 110 pins of P2 I/O. Additional support is provided for Open Firmware and Symmetric Multiprocessor.

Multiprocessing support includes: dual CPUs with shared high bandwidth DRAM memory, a private mailbox for each CPU, and the OpenPIC™ interrupt controller.



***VGM5 single board computer***

The VMEbus may be accessed either via the PowerPC to VME bridge (direct VME interface) or via the more traditional PCI to VME bridge (Universe II interface). Support for 64-bit VME (VME64) and auto-system controller is included. Special hardware supports the translation of PowerPC semaphore instructions to/from VME Read-Modify-Write cycles. This feature fixes a basic PowerPC to VME incompatibility, and allows multiprocessing with a mix of PowerPC and 68xxx based CPU boards.

Onboard peripherals include an optional Wide Ultra SCSI (8/16-bit wide) interface, Fast Ethernet (10Base-T/100Base-TX) RJ-45 interface, two RS-232D serial ports, and a real-time clock/calendar with four digits for the year.

An industry-standard PMC slot provides optional daughterboard I/O connection to the PCI bus. The optional PEX3 PMC expansion board provides up to 3 additional PMC slots. The VGM5 also provides Synergy Microsystems' P0•PCI™ interface which is a secondary PCI bus accessed through the P0 connector. P0•PCI™ provides support for board-to-board communications and additional PCI I/O expansion.

A full line of system **monitor**, **kernel**, and **operating system** software/firmware is also available from Synergy and leading developers.

## VGM5 physical configuration

VGM5 has onboard connectors for:

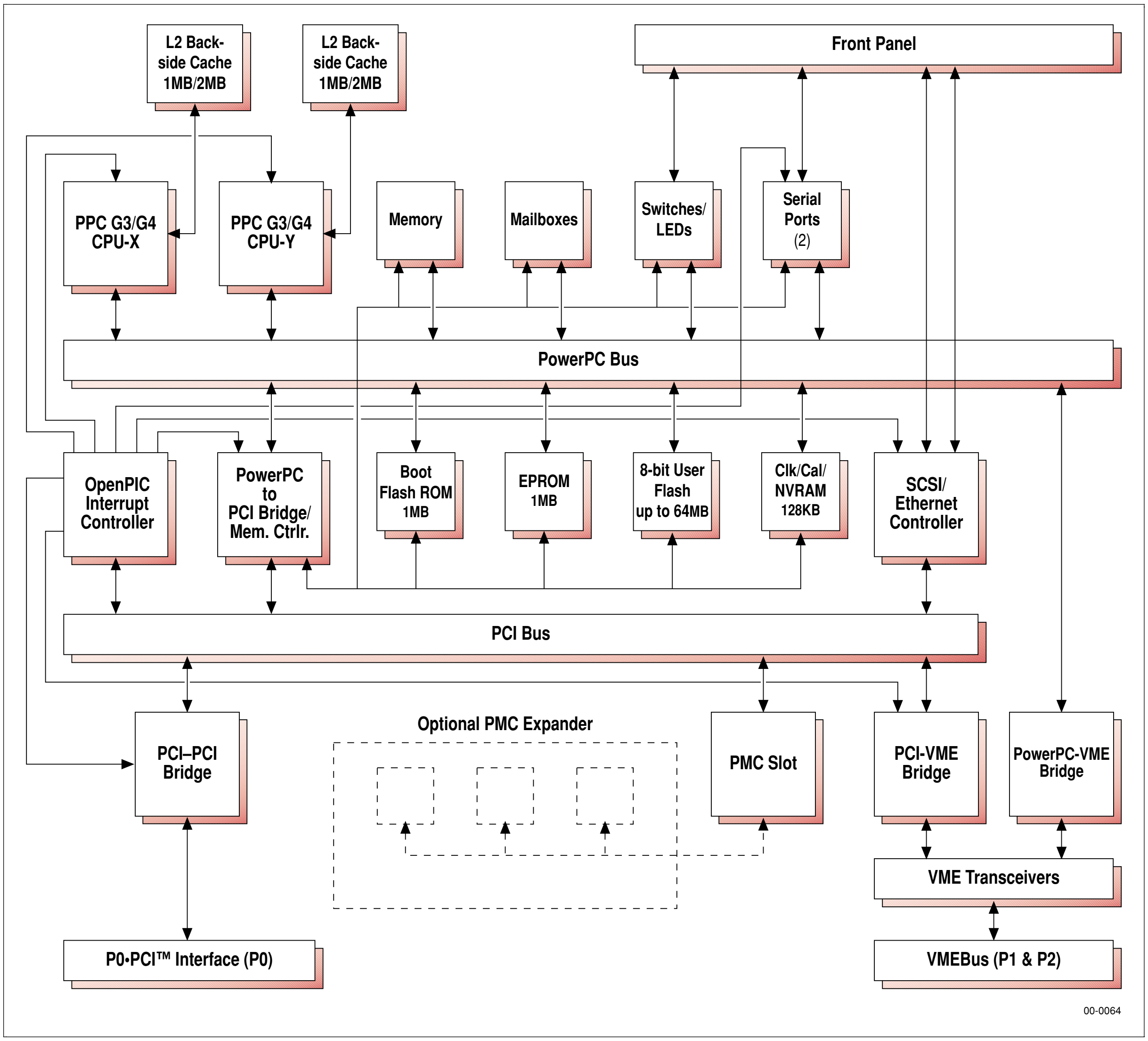
- Modular memory card
- PMC I/O card and/or PMC carrier board expansion via PEX3
- P0•PCI™ interface via P0 connector

VGM5's front panel includes:

- Modular RJ-45 jack for Fast Ethernet
- Modular RJ-69 jack for dual serial ports (A and B)
- SMI & reset switch for each processor
- 8 user LEDs
- 8 status LEDs (count includes 2 ea. dual-color LEDs)
- Software-readable 8-bit DIP switch
- Optional 68-pin Wide Ultra SCSI connector (single-ended)

## Functional block diagram

The figure below shows the functional block diagram for the VGM5 SBC.



VGM5 functional block diagram

## Feature summary

VGM5 provides the following list of powerful features and functions:

- SMP compliant
- Synergy's PowerPC to VME bridge (direct VME interface) for low latency VME access and true VME Read-Modify-Write cycle support
- PCI to VME64 bridge (with auto-system controller) rated at 50 MB/sec
- OpenPIC™ compliant — any interrupt source can be routed to any CPU at any priority
- Four 32-bit counters can be read at any time as well as generate interrupts
- Single or dual G3/G4 PowerPC processor(s)
- Backside L2 cache, 1 MB (G3) or 2 MB (G4)
- 16-512 MB of high performance SDRAM supporting parity (32-512 MB, Rev. E or higher boards)
- 1 MB of system boot Flash loadable via a 32-pin JEDEC socket
- 4/8/16/32/64 MB 8-bit wide User Flash memory
- 128 KB of NVRAM
- Real time clock/calendar (4-digit year)
- RTC/NVRAM backed up by battery or capacitor option (for high altitude applications)
- Two RS-232D serial ports up to 115.2Kbps
- Two 8-bit CPU mailboxes
- P0•PCI™ sub-bus interface goes beyond VMEbus bandwidth limitations by providing an aggregate 266 MB/sec (theoretical maximum) transfer rate for up to 8 boards (4 pair) in a system
- Fast-20 SCSI (8/16-bit wide) option
- Fast Ethernet 10Base-T/100Base-TX
- PMC compliant slot
- Optional 6U expansion board provides up to 3 more PMC slots
- Geographical addressing support
- Twelve status LEDs (8 front panel, 4 on board), eight user-programmable LEDs, an 8-bit software-readable switch, two CPU reset/interrupt switches

# Manual conventions

## Typographical conventions

This manual observes the following typographical conventions:

❶ The term **VGM Series** is used in conjunction with information that applies to **ALL** models of the board series. When differences among models exist, specific model numbers are used to describe any special features.

❷ In diagrams and descriptions in this manual, signal names followed by a backslash (\) are **active low**.

## Notes chapter

Within this manual, a **notes chapter** is provided in select sections. Some or all of the following notes chapters (or sections) may be found in this manual:

- *Programming notes* — deals with programming issues
- *Installation notes* — deals with installation issues
- *Operating notes* — deals with operating/usage issues

Refer to this special information for any notes or caveats for the device or topic under discussion.

## Bit numbering conventions

To avoid confusion, be aware that there are two bit numbering conventions.

The PowerPC architecture was invented by IBM, who number their bits with 0 on the left (most-significant bit or MSB) and 31 on the right (least-significant bit or LSB). This zero-on-the-left numbering convention is reflected in both IBM's and Motorola's PowerPC documentation.

PCI bus and VMEbus both number the bits with 0 on the right (LSB) and 31 on the left (MSB). This zero-on-the-right numbering convention is used in the bit descriptions contained in this manual.

***Bit numbering conventions***

| **Binary bits of data (example)** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **Zero-on-the-right** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Zero-on-the-left** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

## Web page address

Note that any web page address (URL) is subject to change without notice. If the listed web page address in the manual doesn't work, do a site search or use your favorite search engine to find the information you need. Over time, some information may no longer be posted online. In this case, contact the manufacturer directly using the contact information usually provided in the home page.

## Manual comments

Synergy invites your comments or corrections to this manual. Email comments/corrections to:

**doc@synergymicro.com**

# 2 Getting Started

This section provides configuration, setup and general information for the VGM5 SBC.

- Minimum system requirements
- Front panel
- Setting up the VGM5 hardware
- Installing a monitor PROM
- Installing the RGSx memory module
- Installing PMC cards
- Installing the P0 overlay
- Installation notes

# *Minimum system requirements*

The following system components are required to install and test VGM5 boards:

- **6U VMEbus-compatible card cage with VME64x-compatible P1 and P2 backplanes installed (with P0 connector)** — A card cage with forced air cooling is required:
  - minimum, 300 LFM (@sea level)
  - recommended, 400 LFM (@sea level)

Note the recommended airflow rating in linear ft./min. This is the rate of air flowing over the component side of the SBC in its chassis and not the air moved through the chassis (CFM rating). LFM can be measured using a hand-held anemometer such as the Kestrel 1000 Pocket Wind Meter by Nielsen-Kellerman (**www.nkelectronics.com**).

VGM5 boards feature state-of-the-art, high-speed, transfers across the VMEbus that in some cases may approach the maximum VME specifications for transfer speeds. To support these transfers, the underlying connectors, circuitry, and printed circuit boards used in the VME card cage must be constructed of high-quality materials that are fully compliant with VME specifications.

For example, VME card cages containing 10-layer, PCB boards are normally required to support high-speed VME transfers. Older style card cages containing 6-layer boards may have some difficulty conducting these signals without generating excessive noise.

⚠ Pin row **B** of the P2 backplane is defined by VMEbus specifications and is bussed across the entire backplane. Pin rows **A** and **C** are user configured and, if connected at all, are normally connected to adjacent slots via wirewrap or special cables.

Because the P2 and P0 pinout may vary between backplanes or even slots in the same backplane, **DO NOT INSTALL the VGM5 into a system slot whose backplane is not compatible with the VGM5's P2 and P0 pinout. Failure to observe this warning can cause the complete destruction of many on-board components and also voids the product warranty.**

The VGM5 pinout meets standard VME specifications for row ***B***, but rows ***A*** and ***C*** (and for 5-row boards, the majority of pins on row ***D*** and half the pins on row ***Z***) will vary according to the PMC card installed. Synergy PMC card pinouts are shown in the associated manual. If no PMC card is present, P2 backplane rows ***A*** and ***C*** (and ***D*** & ***Z***) are defined as ***no-connects***.

For a complete list of the VGM5 P2 assignments, see the ***P2 connector pinouts*** table in Appendix A, page 271.

- **Power supply** — VGM5 boards (with 256 MB memory module and no PMC installed) typically require the following power supply voltage levels:

| | |
|---|---|
| **Dual G3/466 MHz** | +5.0V +/-5%, 5.8A typical @ 5.00V (29W).<br>±12V ±5%, 50 mA for -12V, 150 mA for +12V |
| **Dual G4/466 MHz** | +5.0V +/-5%, 7.8A typical @ 5.00V (39W).<br>±12V ±5%, 50 mA for -12V, 150 mA for +12V |

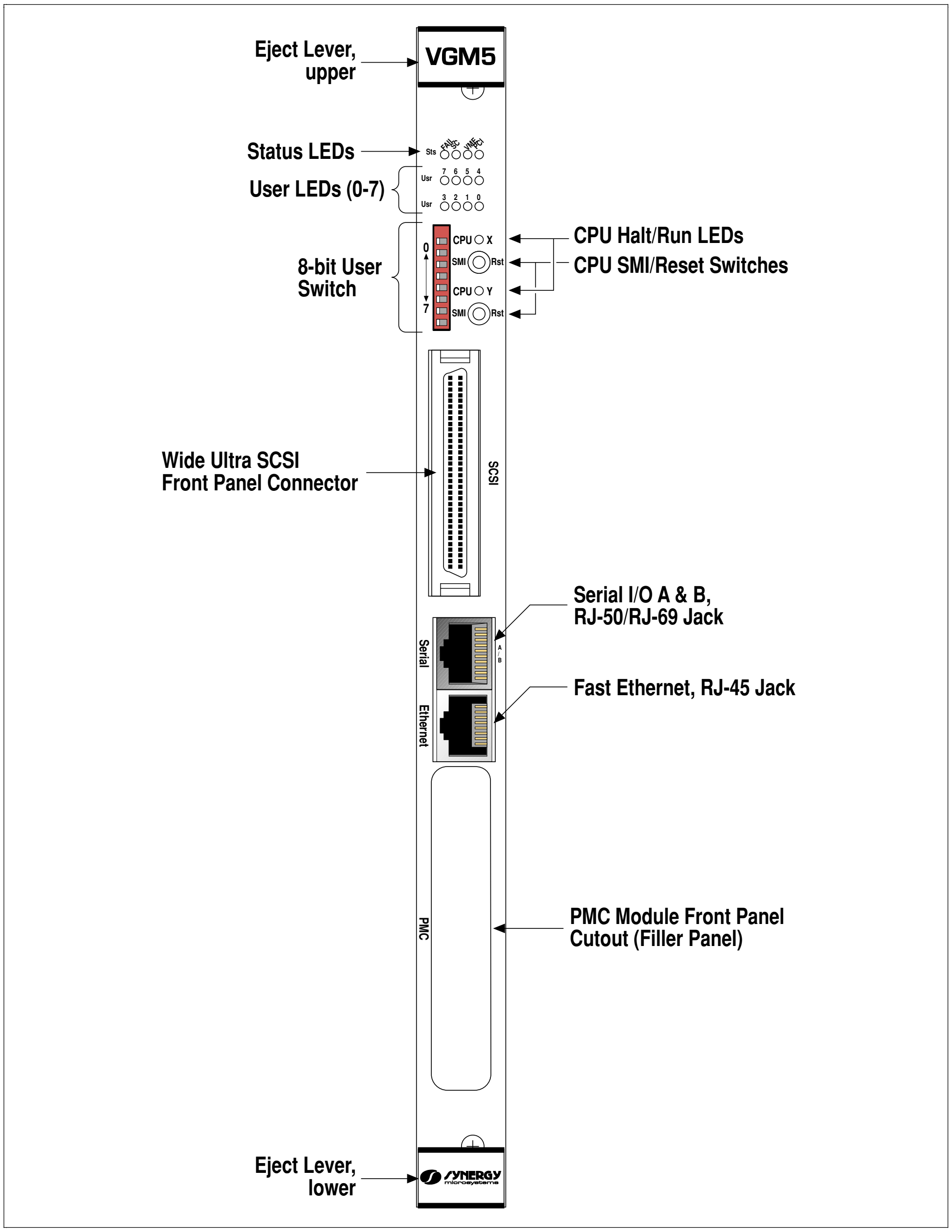
⚠ Ensure that the power supply is capable of meeting the above requirements plus the requirements of any additional boards in the system. An extra 20% margin of current capacity should be factored in for safety.

- **One modular serial I/O cable or dual-port adapter cable (for Ser. Port B) —** The serial ports on VGM5 boards share a 10-pin modular RJ-69 connector. A regular 8-pin RJ-45 cable can be used to access Serial Port A. Dual-port adapter cables, available from Synergy, allow access to Serial Port B. Refer to ***Serial I/O cabling options*** in Appendix A, page 299, for more information.

- **RS-232 compatible video display terminal or a PC with a COM port and terminal emulator software**

# Front panel

The drawing below shows the layout of the connectors, controls, and indicators on the VGM5 front panel.



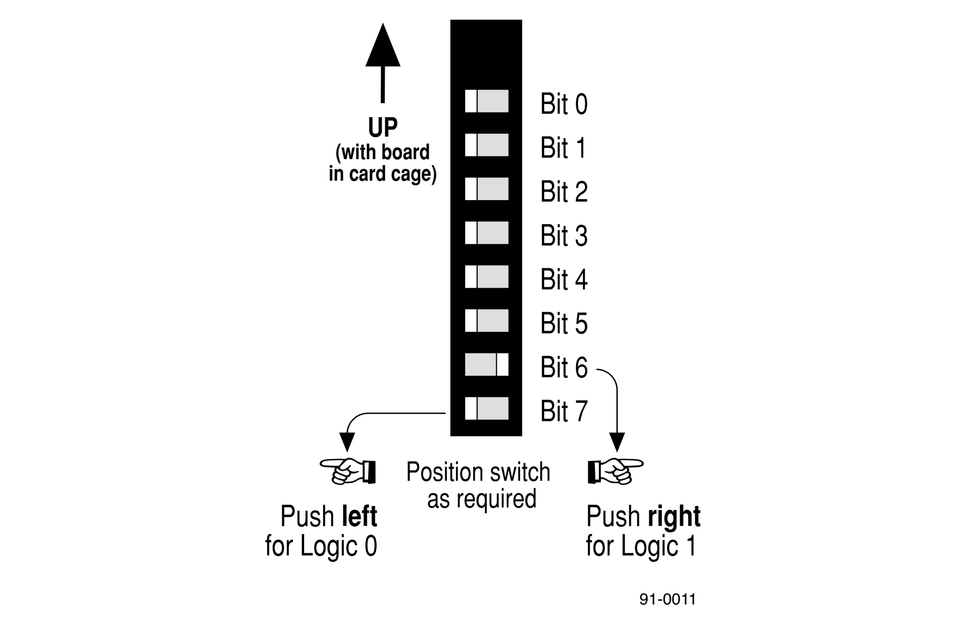
*VGM5 front panel*

## 8-bit user switch

The eight position switch on the VGM5 front panel provides an 8-bit software-readable switch.

Readable switches can be very useful in target applications where applications programs can read the switch to discover what their function should be, the nature of their peripherals, etc.

The CPU reads the switch setting by performing a byte-wide read from the 8-bit User Switch register at memory location **0xFFEF_FD00.** The figure below shows the register bits corresponding to each of the eight switch positions.



***8-bit user switch polarity***

⚠ Numbering may appear on the switch component itself that conflicts with the numbering shown above. Ignore all numbering schemes except what is shown above and on the ***Quick Reference Card***.

## Toggle switches

VGM5 boards have a RESET and SMI switch for each CPU:

***RESET***

Asserts either a **CPU** or **board-level** RESET as described in the figure and text below:

Pushing a switch to the **right** asserts a CPU-level RESET to the corresponding CPU. The **CPU-X** (top) switch asserts a reset to the CPU on single CPU models and to **CPU-X** on the dual CPU models. The **CPU-Y** switch (bottom) asserts a reset to CPU-Y which has an effect only on dual CPU models.

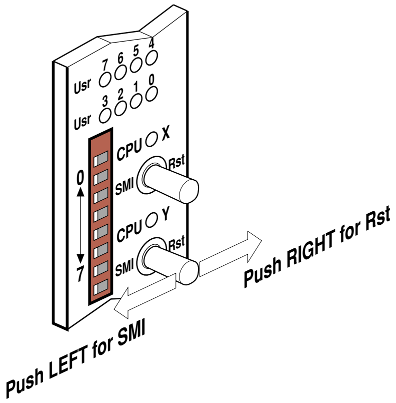
Pushing **both** switches to the right at the same time asserts a board-level reset on all VGM Series models:

- Resets the CPU(s).
- Resets all on-board components that have such a function and clears all on-board control registers.
- Asserts a VME RESET if the board is serving as the System Controller.

***SMI***

Pushing a switch to the **left** asserts an SMI interrupt to the respective  CPU.

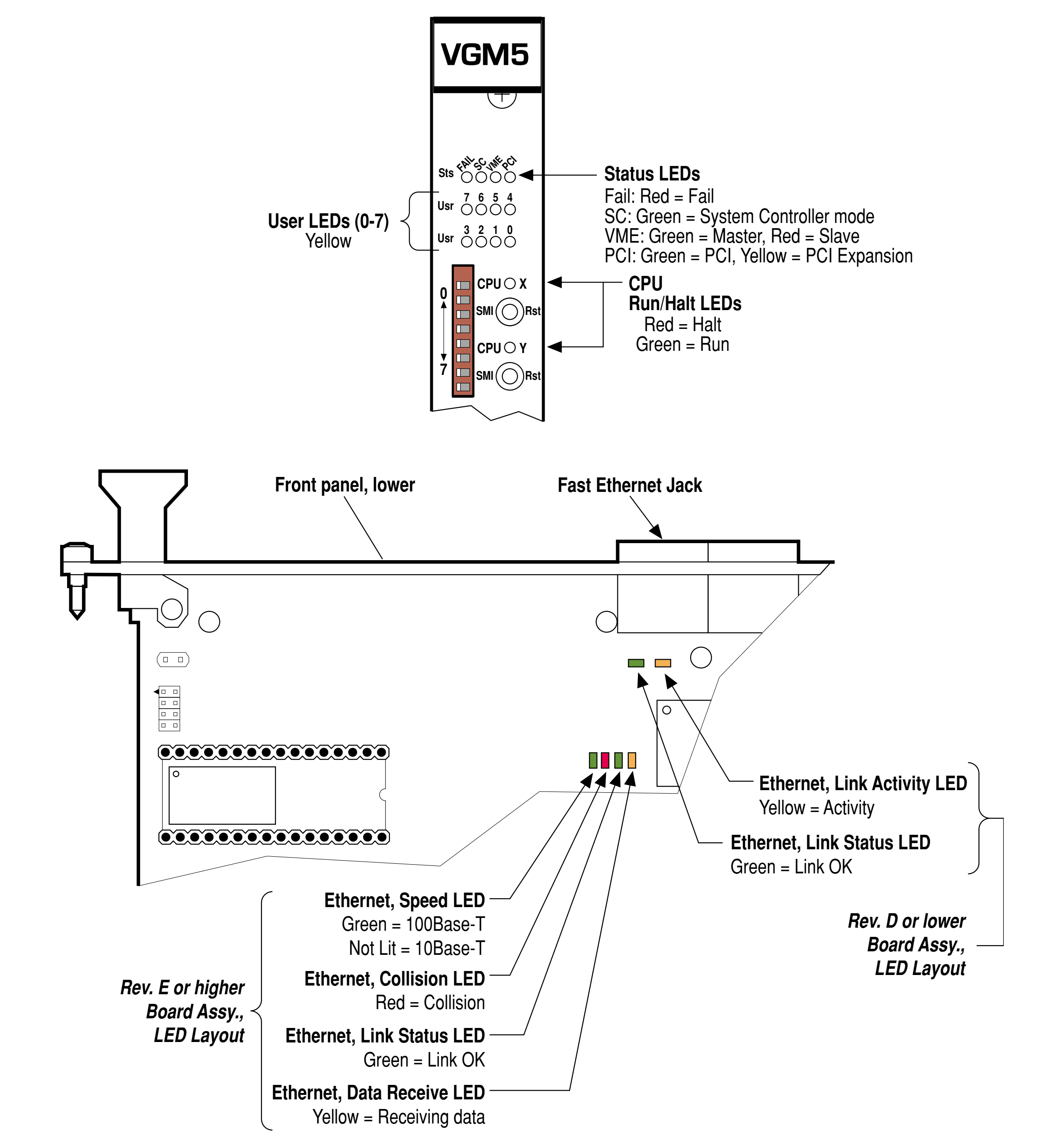
Pushing the bottom switch to the left has **no effect** on single processor boards.



***Close-up of RESET and SMI switches***

## LEDs

Shown below are the VGM5 front panel and onboard LEDs which provide a quick indication of board activity. The following describes the LED functions.



*VGM5 LEDs*

The eight **User** LEDs indicate **application events**:

***0 - 7*** Software-programmable LEDs are controlled by the User LED registers. They indicate the current **operating mode** of the board as defined by the software currently running.

For more information on the registers that control these LEDs, refer to the ***User LED registers*** discussion in Section 4, page 134.

The **Status** LEDs indicate various status items:

| LED Label | Indication | |
|---|---|---|
| **FAIL** | Red | |
| | When on, indicates a condition that caused the CPU to reset (VMEbus SysRst\ line or the front panel RESET toggle).<br>During normal operation, the system boot software clears this condition shortly after RESET. | |
| **SC** | Green | |
| | When on, indicates System Controller function assumed by board. | |
| **VME** | Dual-color LED, VMEbus Activity | |
| | Green (VME Master) | Red (VME Slave) |
| | Flickers green in response to VME Master activity. When the VMEbus interface is idle, the VME LED lights up green on the last release-on-request (ROR) VMEbus master to have used the bus. | Flickers red in response to VME Slave activity. |
| **PCI** | Dual-color LED, PCI Bus Activity | |
| | Green (PCI Bus) | Yellow (PCI Expansion) |
| | Flickers green in response to PCI bus activity. | Flickers yellow in response to PCI expansion activity. |

The **CPU** LEDs indicate the **run status** of CPU-X and CPU-Y:

| LED Label | Indication | |
|---|---|---|
| **CPU X** | Dual-color LED, CPU-X Run Status | |
| | Green (Run) | Red (Halt) |
| | Flickers green in response to PowerPC bus activity by CPU-X. If not on, indicates CPU is not executing bus cycles as it executes instructions in cache or waits for an interrupt. | Lights red when CPU has halted. |
| **CPU Y** | Dual-color LED, CPU-Y Run Status (non-functional for single CPU boards) | |
| | Green (Run) | Red (Halt) |
| | Flickers green in response to PowerPC bus activity by CPU-Y. If not on, indicates CPU is not executing bus cycles as it executes instructions in cache or waits for an interrupt. | Lights red when CPU has halted. |

The two LEDs on the VGM5 motherboard, Revision D or lower, indicate Ethernet port status as follows:

***Green*** **Link OK** — lit when 10Base-T/100Base-TX cable is properly plugged into a functioning Ethernet network and onboard software has initialized the Ethernet interface.

***Yellow*** **Link Activity** — flickers whenever data is being received or transmitted. If the VGM5 is connected to a repeater-type hub instead of a switch-type hub, this LED may still flicker even when the VGM5 is not transferring data since packets sent over the network to other nodes will also be sent to the VGM5.

For VGM5 board assembly Revision E or higher, four LEDs indicate Ethernet port status as follows:

***Green*** **Speed** — lit when operating as 100Base-TX. LED is OFF when operating as 10Base-T.

***Red*** **Collision** — lit whenever the VGM5 produces a collision (VGM5 transmits at same time as packet data is being received). When collisions occur, retries are automatically performed as part of the Ethernet protocol.

***Green*** **Link OK** — lit when 10Base-T/100Base-TX cable is properly plugged into a functioning Ethernet network and onboard software has initialized the Ethernet interface.

***Yellow*** **Receive Data** — flickers whenever data is being received. If the VGM5 is connected to a repeater-type hub instead of a switch-type hub, this LED may still flicker even when the VGM5 is not the intended destination since packets sent over the network to other nodes will also be sent to the VGM5.

## Lamp test feature

During board level reset, all LEDs are illuminated to provide a lamp test. You can confirm proper operation of the LED indicators as you do a board level reset by observing the LEDs and pushing **both** CPU toggle switches to the right. Hold switches in this position and wait 2 seconds for LED illumination.

# Setting up the VGM5 hardware

This chapter describes the general hardware configuration of VGM5 boards. This configuration is done via jumpers on jumper block **J02N**.

Additional jumpers are provided on jumper block **J902** for EPROM configuration. Refer to the next chapter on ***Installing a monitor PROM*** (page 29) for details on EPROM configuration.

## Default configuration

The table shown below lists the default hardware configuration for the VGM5 board before the installation of any jumpers on **J02N**.

***Default hardware conditions***

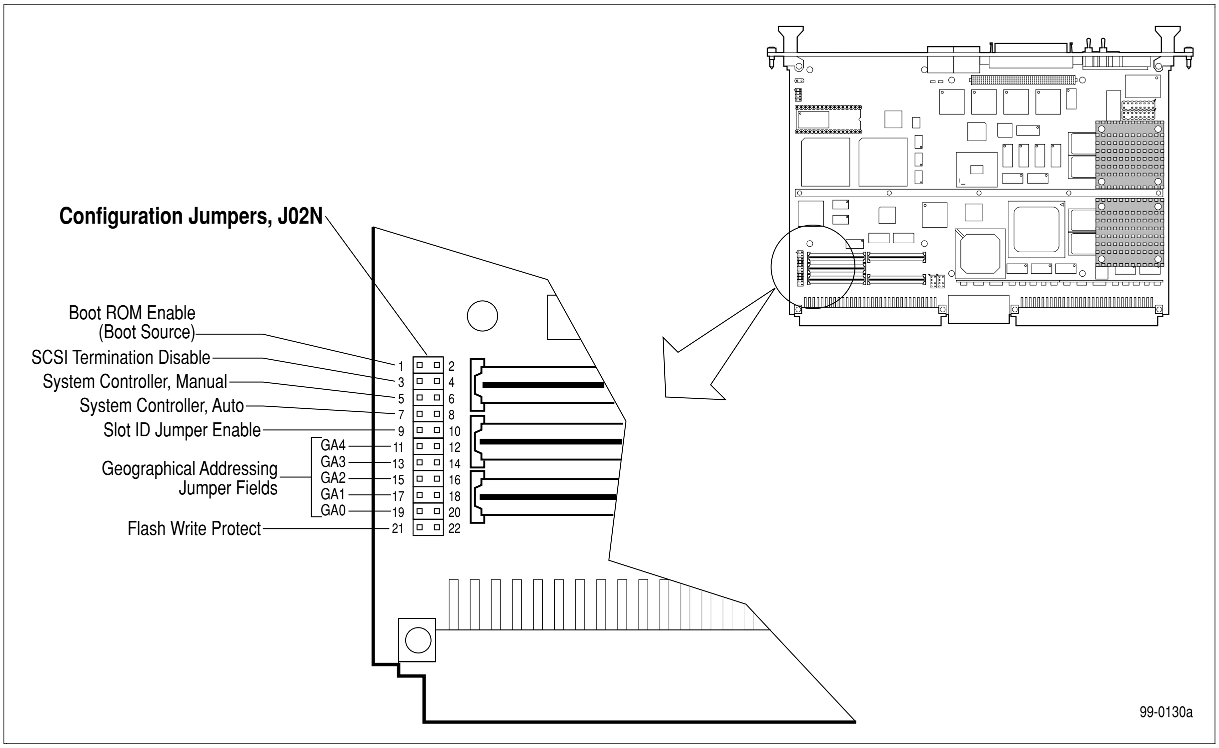
| Jumpers (presumes no jumper installed) | Default |
|---|---|
| J02N — Boot ROM Enable | Disabled |
| — SCSI Termination | Enabled |
| — Force VME System Controller | Disabled |
| — VME64 Auto System Controller | Enabled |
| — User Defined Slot Number | None |
| — Flash Write Protect | Disabled |

## Installing jumpers

No jumpers need to be set for most applications. However, the **J02N** user configuration jumpers let you change the default conditions listed in the table above if necessary. Note that the jumper shunts used are of the smaller 2 mm size and not the larger .100" size commonly found on older SBCs. The jumpers are summarized below.

- Verify that the board has a monitor EPROM and memory module installed.
- Verify/install shunt at J02N 1 & 2 to boot from DIP EPROM.
- Install a shunt at J02N 3 & 4 to unterminate the SCSI bus.
- Install a shunt at J02N 5 & 6 to force VME System Controller.
- Install shunt at J02N 7 & 8 to disable Auto System Controller function.
- Install a shunt at J02N 9–20 for manual slot number configuration.
- Install a shunt at J02N 21 & 22 for global Flash write protection.

The drawing below shows the location and pinout of jumper block **J02N**.



*Configuration jumpers, J02N*

*Jumper J02N functions*

| Jumper Pins | Function | For more info, see Section, Chapter |
|---|---|---|
| 1 & 2 | Boot ROM Enable: When installed, the board will boot from the DIP EPROM instead of Boot Flash ROM. | Section 4, Boot Flash ROM/DIP EPROM (page 159) |
| 3 & 4 | SCSI Termination Disable: When installed, the on-board SCSI termination is disabled | Section 3, SCSI bus (page74) |
| 5 & 6 | VME System Controller: When installed, the on-board VME system controller function is active regardless of the VME64 auto-system controller function or the actual VME slot in which the board is installed. | Section 3, VME64 bus (page 58) |
| 7 & 8 | VME64 Auto-System Controller Disable: When installed, the on-board auto-system controller function is disabled. | |
| 9 & 10 | User Defined Slot Number: When installed, indicates that the slot number has been set by the user installing jumpers in positions 6-10. When not installed, the slot number is determined from the geographical address pins on VME64x P1. | See text below. |
| 11 & 12 | Jumper for GA4, slot number | See Table Below<br>For Jumper Settings |
| 13 & 14 | Jumper for GA3, slot number | |
| 15 & 16 | Jumper for GA2, slot number | |
| 17 & 18 | Jumper for GA1, slot number | |
| 19 & 20 | Jumper for GA0, slot number | |
| 21 & 22 | Flash Write Protect: When installed, all Flash (Boot Flash, User Flash, DIP Flash EPROM) is protected from writes. | Section 4, Boot Flash ROM/DIP EPROM (page 159); Section 4, User Flash memory (page 165) |

## Setting the slot number manually

On a VME64x backplane (5-row connectors), a board can automatically sense which slot it is plugged into by reading special pins on P1. These geographical address pins are encoded on the backplane. Boards that can read these pins present the geographical address in the VME64 Slot register. These boards also check the geographically addressing parity and signal its validity in the same register.

In some cases the VGM5 will not be able to read these pins because the board is configured with 3-row VME connectors or it may be plugged into an old VME64 backplane (3-row connectors). For these situations, jumpers on J02N (pins 9–22) are provided to set the board's slot number manually.

To set a slot number, install jumpers over the appropriate pair of pins as shown with a bullet (●) in the table below. Pins with no jumper installed are shown with a dash (–). This is a binary encoded scheme with pins 11 & 12 MSB and pins 19 & 20 LSB.

Install the User Defined Slot Number jumper (pins 9 & 10) to let hardware/software read the user defined slot number from the VME64 Slot register. The value in this register is set by the 5 slot number jumpers.

Note that user defined slot number jumpers must not be used if both the board and backplane have 5-row connectors.

***J02N jumper settings for slot number selection***

| Slot No. | Pins 11 &12 (GA4) | Pins 13 & 14 (GA3) | Pins 15 & 16 (GA2) | Pins 17 & 18 (GA1) | Pins 19 & 20 (GA0) |
|---|---|---|---|---|---|
| 1 | — | — | — | — | ● |
| 2 | — | — | — | ● | — |
| 3 | — | — | — | ● | ● |
| 4 | — | — | ● | — | — |
| 5 | — | — | ● | — | ● |
| 6 | — | — | ● | ● | — |
| 7 | — | — | ● | ● | ● |
| 8 | — | ● | — | — | — |
| 9 | — | ● | — | — | ● |
| 10 | — | ● | — | ● | — |
| 11 | — | ● | — | ● | ● |
| 12 | — | ● | ● | — | — |
| 13 | — | ● | ● | — | ● |
| 14 | — | ● | ● | ● | — |
| 15 | — | ● | ● | ● | ● |
| 16 | ● | — | — | — | — |
| 17 | ● | — | — | — | ● |
| 18 | ● | — | — | ● | — |
| 19 | ● | — | — | ● | ● |
| 20 | ● | — | ● | — | — |
| 21 | ● | — | ● | — | ● |

# Installing a monitor PROM

The VGM5 comes with one, 32-pin, 8-bit, monitor DIP EPROM socket that accepts any of the following devices[†]:

| | |
|---|---|
| 27C010 | 1 Mbit (128KB) DIP EPROM |
| 27C020 | 2 Mbit (256KB) DIP EPROM |
| 27C040 | 4 Mbit (512KB) DIP EPROM |
| 28F020 | 2 Mbit (256KB) Flash DIP EPROM |
| 29C040 | 4 Mbit (512KB) Flash DIP EPROM |

Some boards ship from the factory with the appropriate monitor PROM already installed. However, a new or updated PROM is easily added or changed in the field.

The paragraphs below describe a field installation of a new DIP EPROM and all of the potential configuration changes you may need to make to the VGM5 CPU board as a result.

Note If the desired monitor PROM is already present on the VGM5 board or if the monitor ROM is programmed into Boot Flash (e.g. VxWorks), proceed to the next chapter in this section.

[†] TI brand EPROMs cannot be used. Their requirement for Vcc on unused pins prevents a TI PROM from being used in a general purpose socket. EPROMs from other manufacturers such as Intel, AMD, etc. work without problem.

## Materials

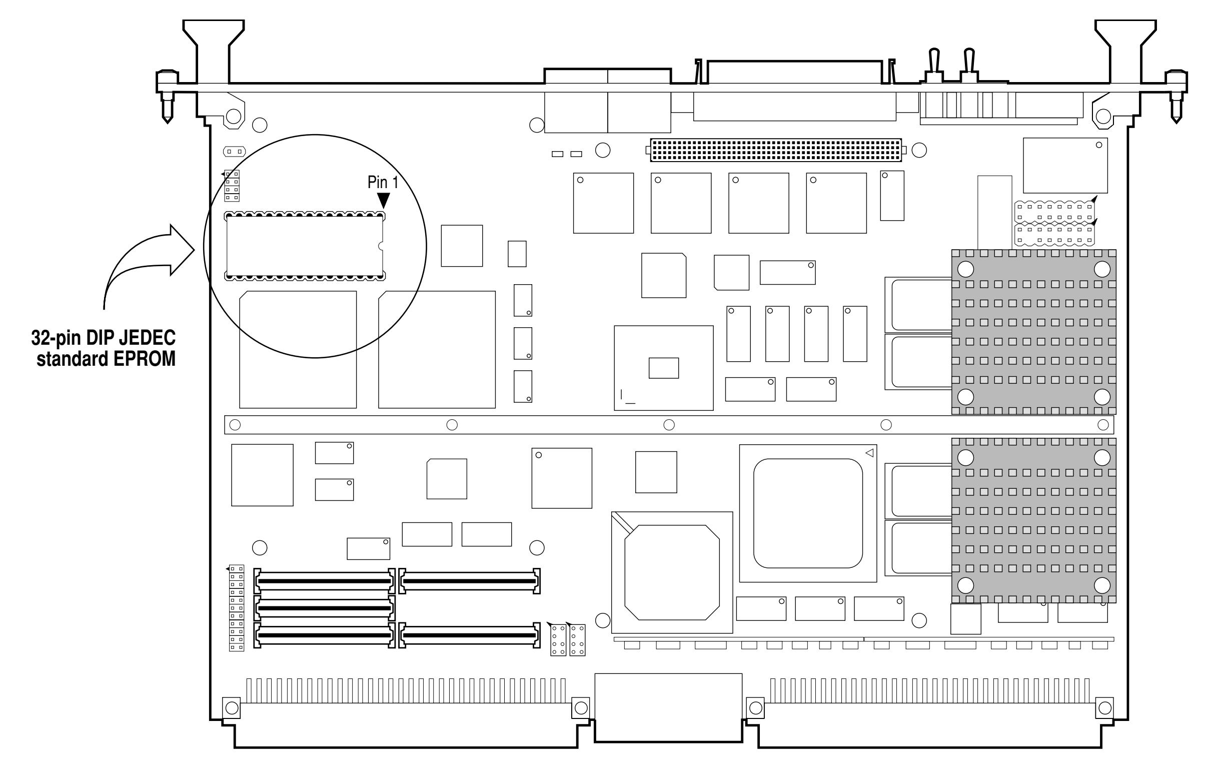
To complete this procedure, you will need the following materials:

- The desired monitor firmware PROM
- A 32-pin, 0.6" wide DIP extractor tool to remove the current PROM (if necessary)
- The manual for the software product on the new EPROM

## Procedural steps

To install a monitor EPROM, complete the following procedure:

1. **Verify proper operation of the motherboard** (if replacing an existing monitor PROM) — Before attempting to install a new EPROM on an existing board, ensure that the motherboard (and any attached mezzanine cards) are operating properly.



*VGM5 DIP EPROM socket location*

❷ **Power-down and remove the SBC from the card cage** — Power-down the system and remove the VGM5 CPU board from the card cage.

⚠ Synergy SBCs contain static-sensitive devices. Make sure you are properly grounded (by putting on a ground-strap, touching a system ground such as a metallic chassis or case, etc.) before removing and handling the board. Use an ESD-protected workstation for module removal and installation work.

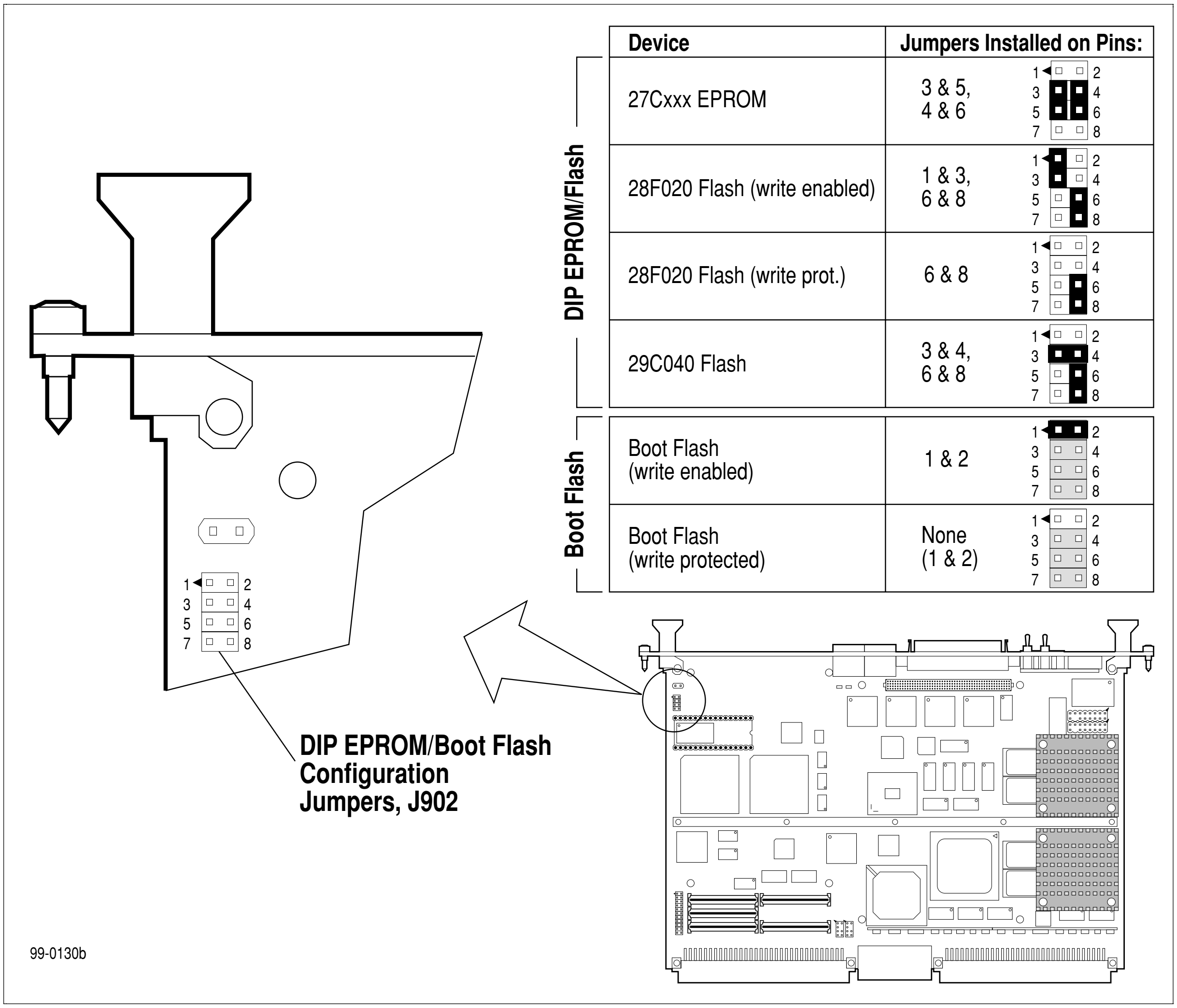
❸ **Locate the current monitor PROM or DIP socket on the CPU board and remove the current monitor PROM** (if necessary) — The figure on the previous page shows the location of the DIP EPROM socket on the VGM5 board.

⚠ **Use a chip extraction tool to remove the DIP EPROM from the socket to avoid damaging parts underneath.**

❹ **Install the DIP EPROM** — Install the DIP EPROM in the socket. The VGM5 EPROM socket accepts 32-pin DIP EPROM devices. The figure above shows the orientation of the DIP EPROM after proper installation (note orientation of notch end).

❺ **Install DIP EPROM configuration jumpers (J902)** — Place jumpers on **J902** to configure the board for the device used in the DIP EPROM socket as shown in the drawing below.

Note Some VGM5 boards include an ECO that adds a Boot Flash write protect function to the J902 jumper block, pins 1 and 2. Refer to the ***Additional write protection of Boot Flash*** discussion in Section 4 (page 164) for more information on this jumper.



| | Device | Jumpers Installed on Pins: |
|---|---|---|
| DIP EPROM/Flash | 27Cxxx EPROM | 3 & 5,<br>4 & 6 |
| | 28F020 Flash (write enabled) | 1 & 3,<br>6 & 8 |
| | 28F020 Flash (write prot.) | 6 & 8 |
| | 29C040 Flash | 3 & 4,<br>6 & 8 |
| Boot Flash | Boot Flash<br>(write enabled) | 1 & 2 |
| | Boot Flash<br>(write protected) | None<br>(1 & 2) |

*DIP EPROM/Boot Flash configuration jumpers, J902*

# Installing the RGSx memory module

VGM5 boards provide all on-board DRAM on upgradable memory modules. Modules are available with the following amounts of DRAM:

RGS1 memory module:

- 16 MB
- 32 MB
- 64 MB

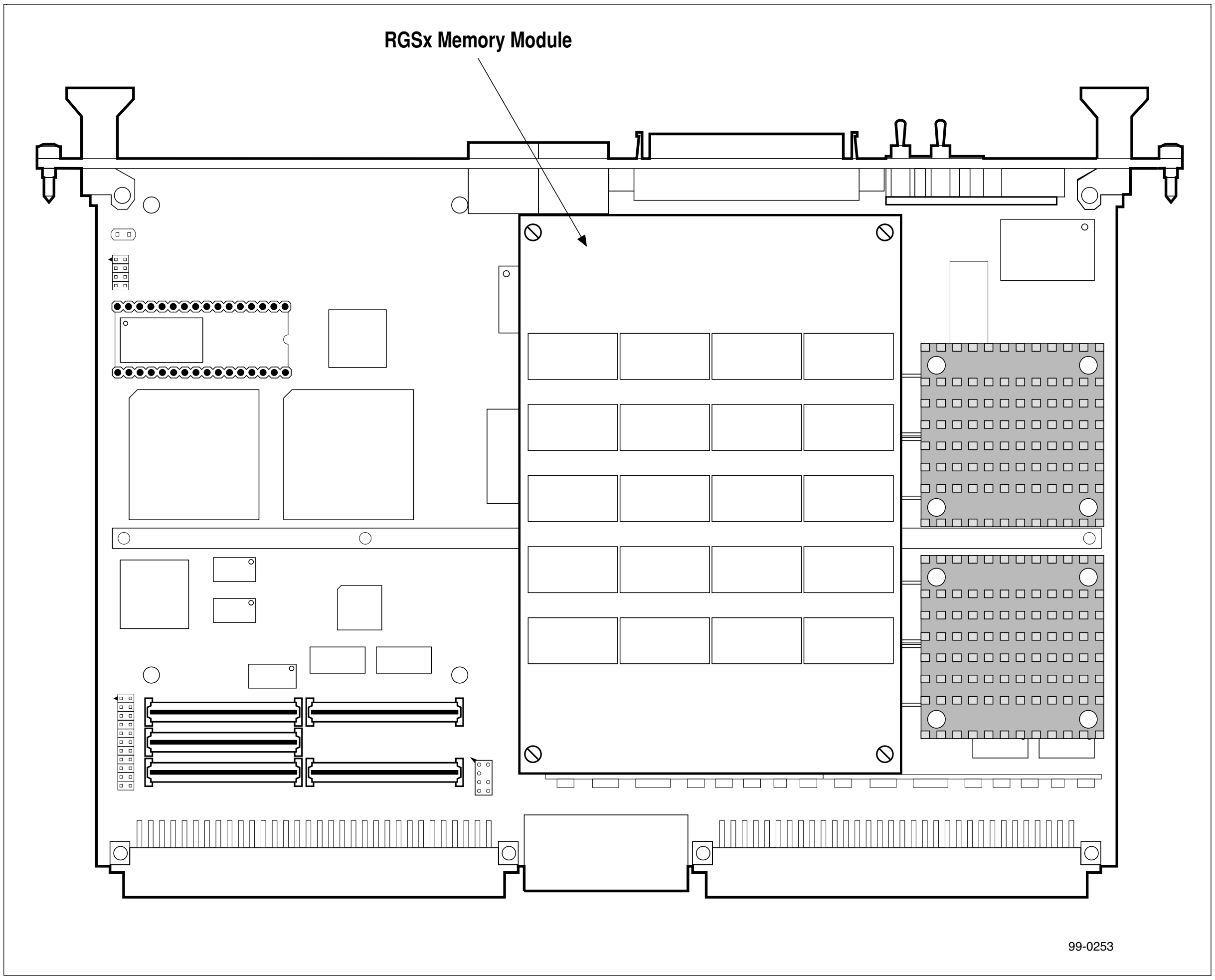
RGS2/RGS3 memory module:

- 32 MB
- 64 MB
- 128 MB
- 256 MB
- 512 MB

Note: Rev. E VGM5 boards and higher use the RGS3 memory module. Rev. D and lower VGM5 boards use the RGS1/RGS2 memory module.

Normally, all VGM5 boards ship from the factory with a memory module installed. The modular design of the VGM5 DRAM interface, however, allows for easy DRAM upgrades in the field.

The drawing below shows the location of the RGSx memory module on the motherboard.



***RGSx memory module location***

This chapter describes field installation of an RGSx memory module.

If the desired RGSx module is already present on the VGM5 board, proceed to the next chapter in this section.

## *Installing/upgrading the RGSx memory module*

Perform the following steps to install or upgrade an RGSx memory module.

❶ **Verify proper operation of motherboard** (if replacing an existing RGSx memory module) — Before attempting to install a new RGSx memory module on a working CPU motherboard, consider checking that the motherboard (and any attached PMC cards) are operating properly.

❷ **Power-down and remove SBC from card cage** — Power-down the system and remove the VGM5 CPU board from the card cage.

> ⚠ Synergy SBCs contain static-sensitive devices. Make sure you are properly grounded (by putting on a ground-strap, touching a system ground such as a metallic chassis or case, etc.) before removing and handling the board. Use an ESD-protected workstation for module removal and installation work.

❸ **Remove existing RGSx memory module from CPU motherboard** (if you are replacing — refer to *RGSx module installation* drawing below for assembly details):

a. Place VGM5/RGSx assembly face-down, that is with large circuit board (motherboard) on top, on a flat surface of an ESD-protected workstation.

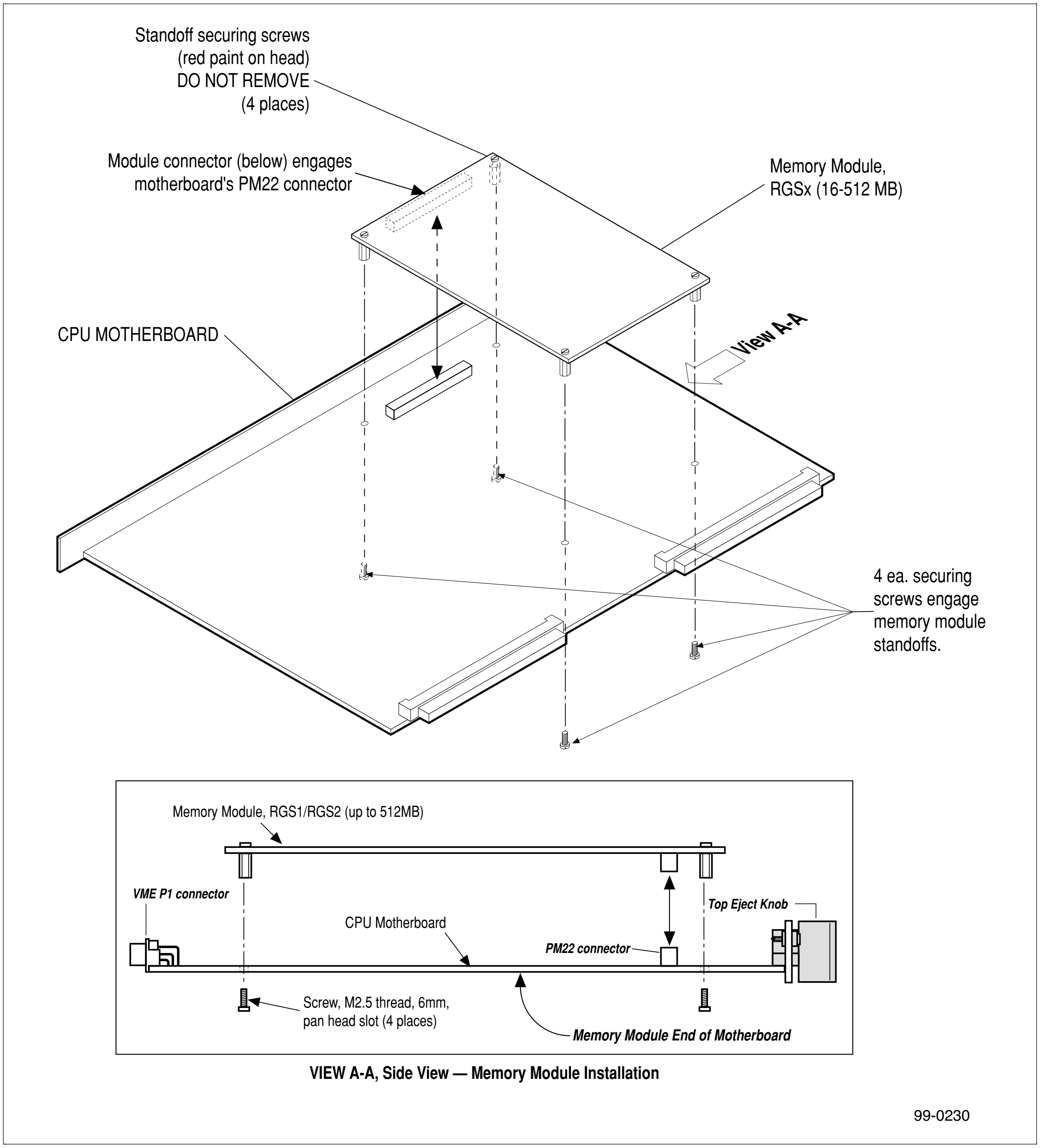
b. Remove four M2.5 slot-head screws from rear (solder) side of VGM5 motherboard. See *Location, memory module securing screws* drawing below.

c. Turn VGM5/RGSx assembly over.

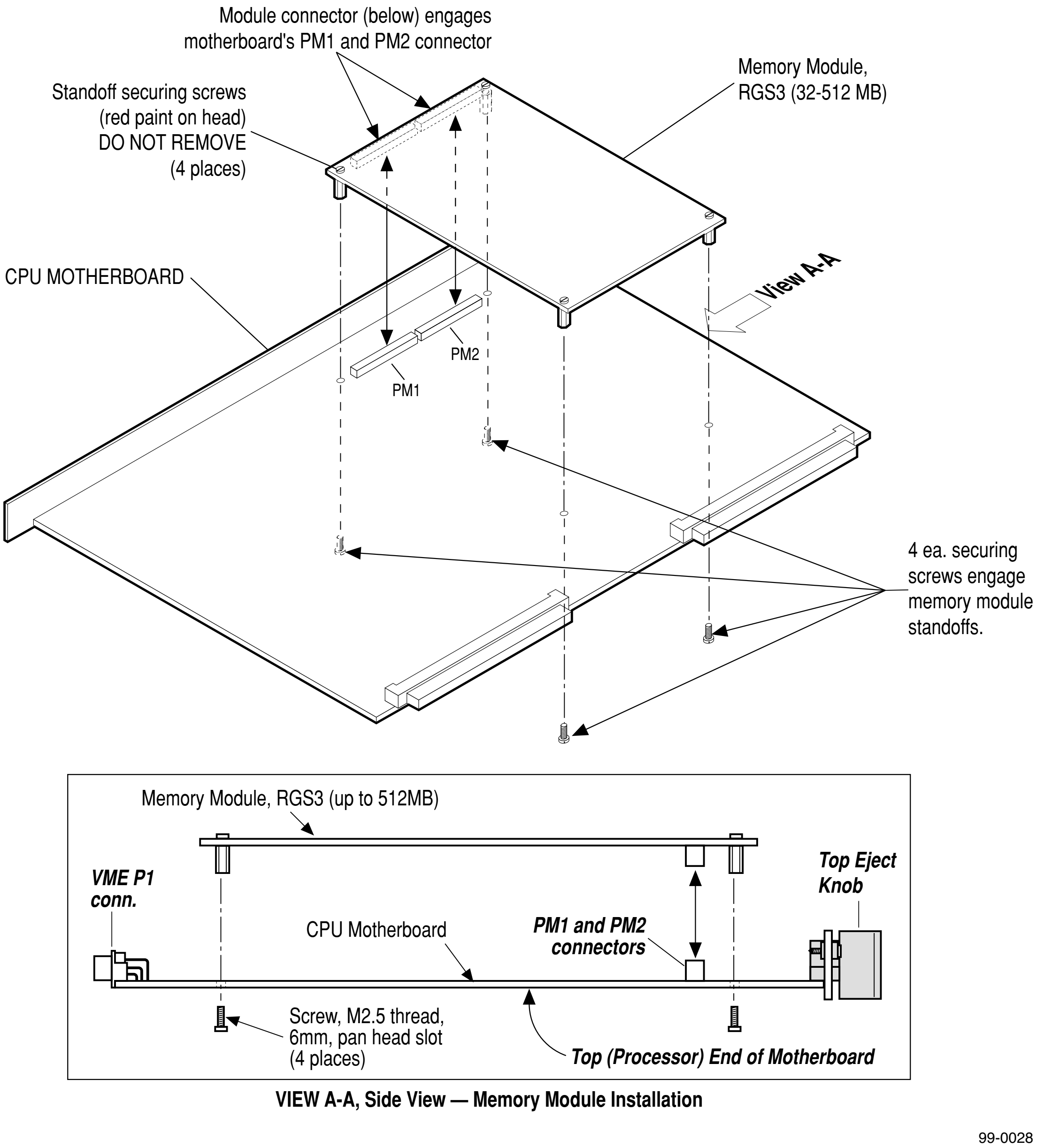
d. Grasp RGSx sides at connector end (toward SBC front panel) and gently pull up until the connector comes loose (rocking back and forth may help).

❹ **Install RGSx module on motherboard** — Installation of RGSx memory module is reverse of removal. Refer to *RGSx module installation* drawing below for assembly details.

⚠ The SBC memory module connector is fragile. To avoid connector damage, make sure that module connector is properly aligned with SBC connector before fully seating module.


Standoff securing screws
(red paint on head)
DO NOT REMOVE
(4 places)
Module connector (below) engages motherboard's PM22 connector
Memory Module, RGSx (16-512 MB)
CPU MOTHERBOARD
View A-A
4 ea. securing screws engage memory module standoffs.
Memory Module, RGS1/RGS2 (up to 512MB)
VME P1 connector
CPU Motherboard
PM22 connector
Top Eject Knob
Screw, M2.5 thread, 6mm, pan head slot (4 places)
Memory Module End of Motherboard
VIEW A-A, Side View — Memory Module Installation
99-0230

*RGS1/RGS2 module installation*

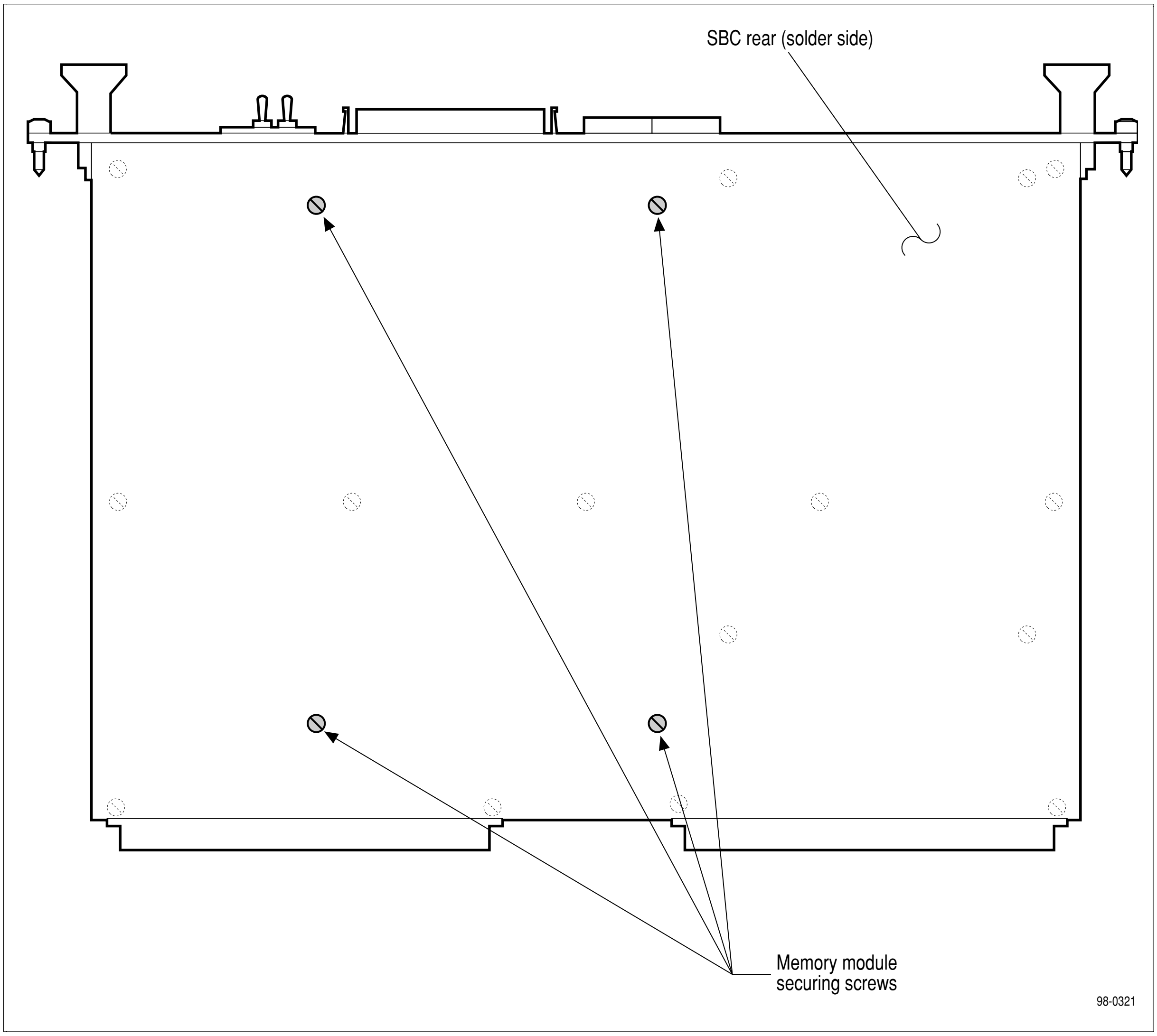


VIEW A-A, Side View — Memory Module Installation

*RGS3 module installation*

## Memory module securing screws

To aid in installation, the location of the memory module securing screws on the VGM5 is shown in the drawing below.



***Location, memory module securing screws***

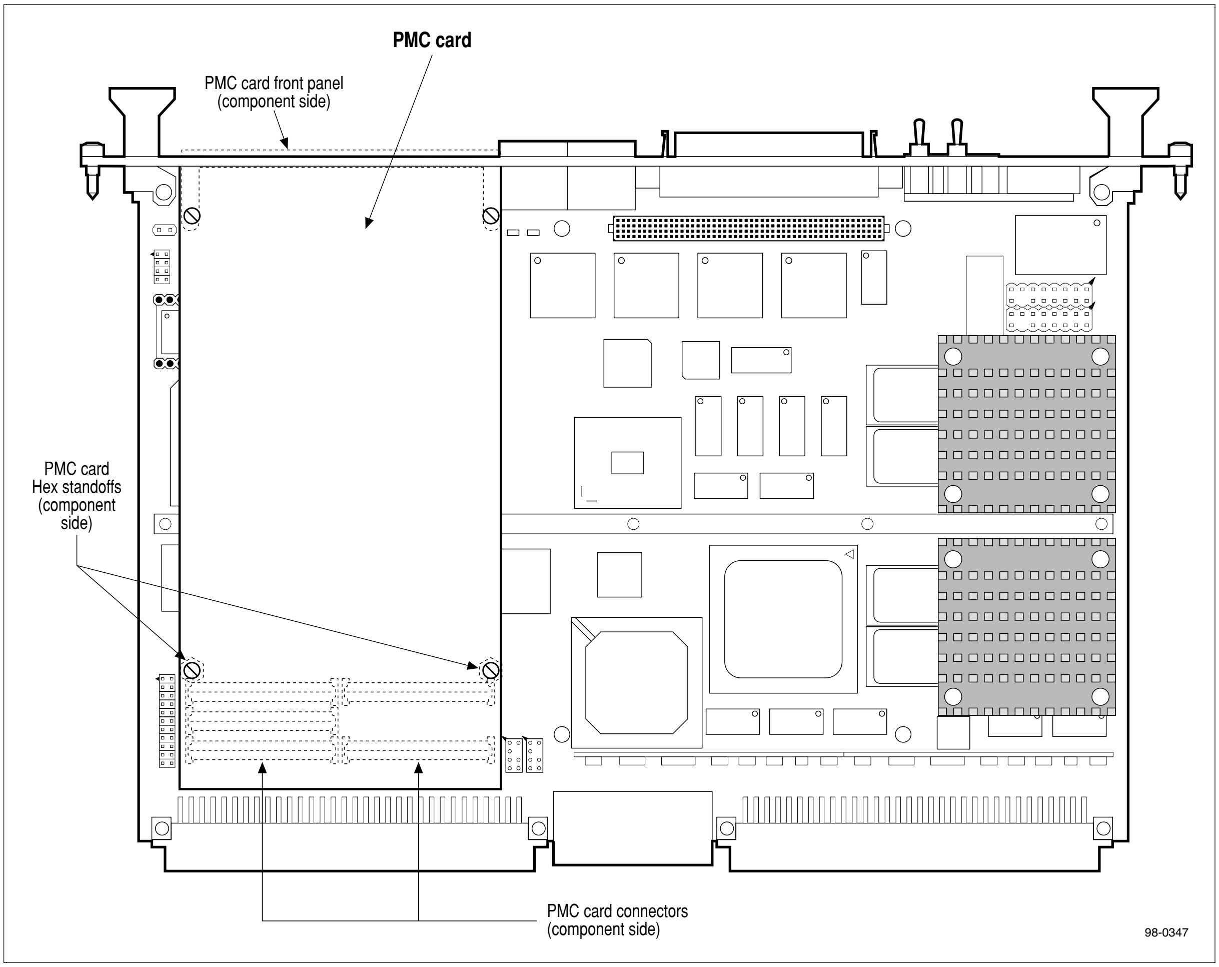
# *Installing PMC cards*

VGM5's I/O expansion is provided by PMC (PCI Mezzanine Card) cards.

This chapter describes PMC card installation.

## VGM5 PMC connectors

The VGM5 comes with PMC connectors for direct installation of one PMC card. The drawing below shows the location of a PMC card on the VGM5 board.


PMC card
PMC card front panel (component side)
PMC card Hex standoffs (component side)
PMC card connectors (component side)
98-0347

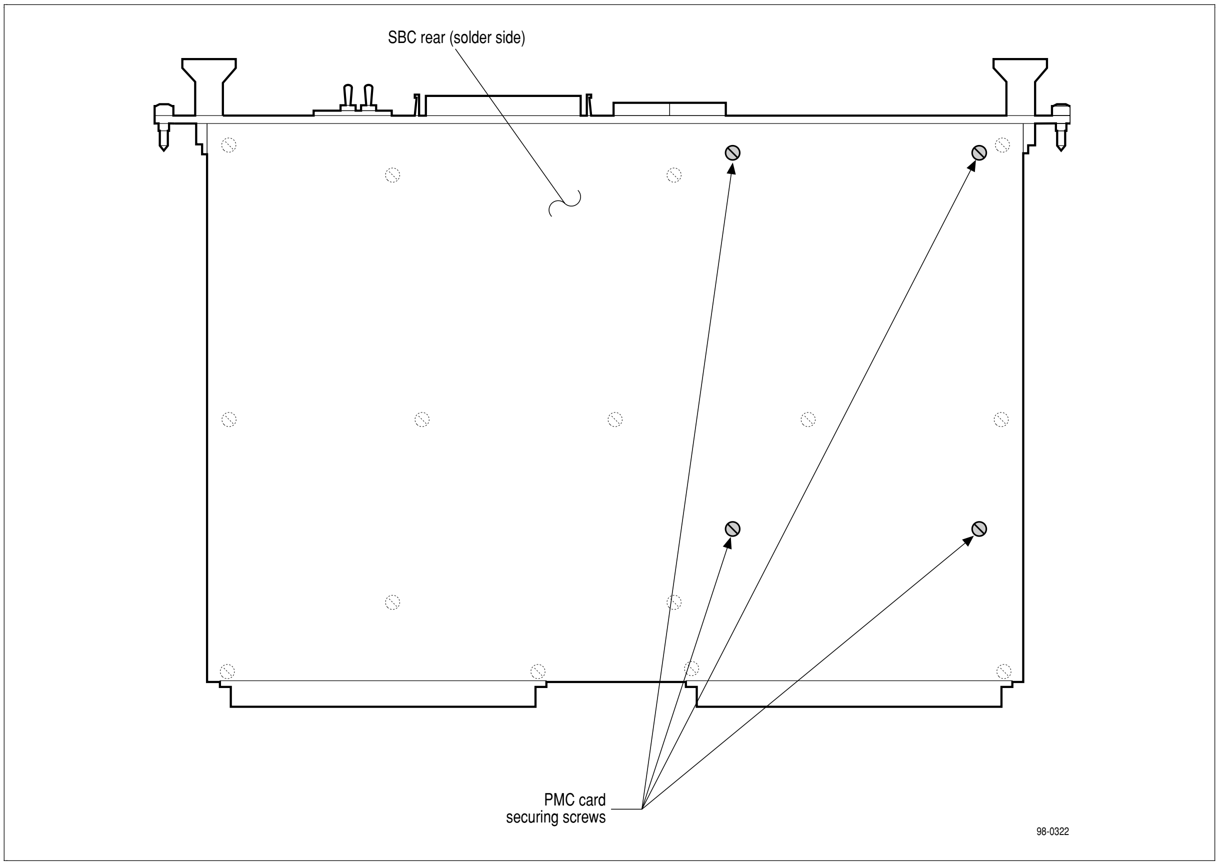
**PMC location (top view)**

### Adding additional PMC cards with the PEX3 expansion board

The PEX3 PMC expansion option provides VGM5 with up to 3 additional PMC slots plus additional SDRAM and Flash memory. Refer to Appendix D (page 313) for complete PEX3 information

### PMC card securing screws

To aid in installation, the location of the PMC card securing screws on the VGM5 is shown in the drawing below.



**Location, PMC card securing screws**

## Installing a PMC card

Perform the following steps to install a PMC card.

Note: The VGM5 PMC slot accepts 5V VI/O or 5V-tolerant PMCs only.

❶ **Power-down and remove SBC from card cage** — Power-down the system and remove the VGM5 SBC from the card cage.

⚠ Synergy SBCs contain static-sensitive devices. Make sure you are properly grounded (by putting on a ground-strap, touching a system ground such as a metallic chassis or case, etc.) before removing and handling the board. Use an ESD-protected workstation for module removal and installation work.

❷ **Remove PMC filler panel from SBC front panel** — The filler panel will be one of two types. The first type simply snaps in place — remove by pushing from the inside. The second type is an actual blank PMC front panel — remove 2 ea. 6 mm M2.5 slot head securing screws from solder side of board to remove (see PMC card securing screws location drawing above).

❸ **Install PMC card onto SBC** (refer to drawing below for assembly details):

a. Place VGM5 assembly face-up on a flat surface of an ESD-protected workstation.

b. If not already on, install PMC card's front panel O-ring gasket (included with PMC card) by slipping gasket into groove around front panel.

c. Grasp PMC at sides: with card front panel towards SBC front panel from rear, tilt PMC front panel into SBC front panel cutout and engage front panel O-ring gasket with chamfer in SBC panel cutout. With PMC front panel in place, place card over SBC connectors. Ensure both PMC and SBC connectors are aligned then press down over PMC connector area to fully engage SBC connectors.

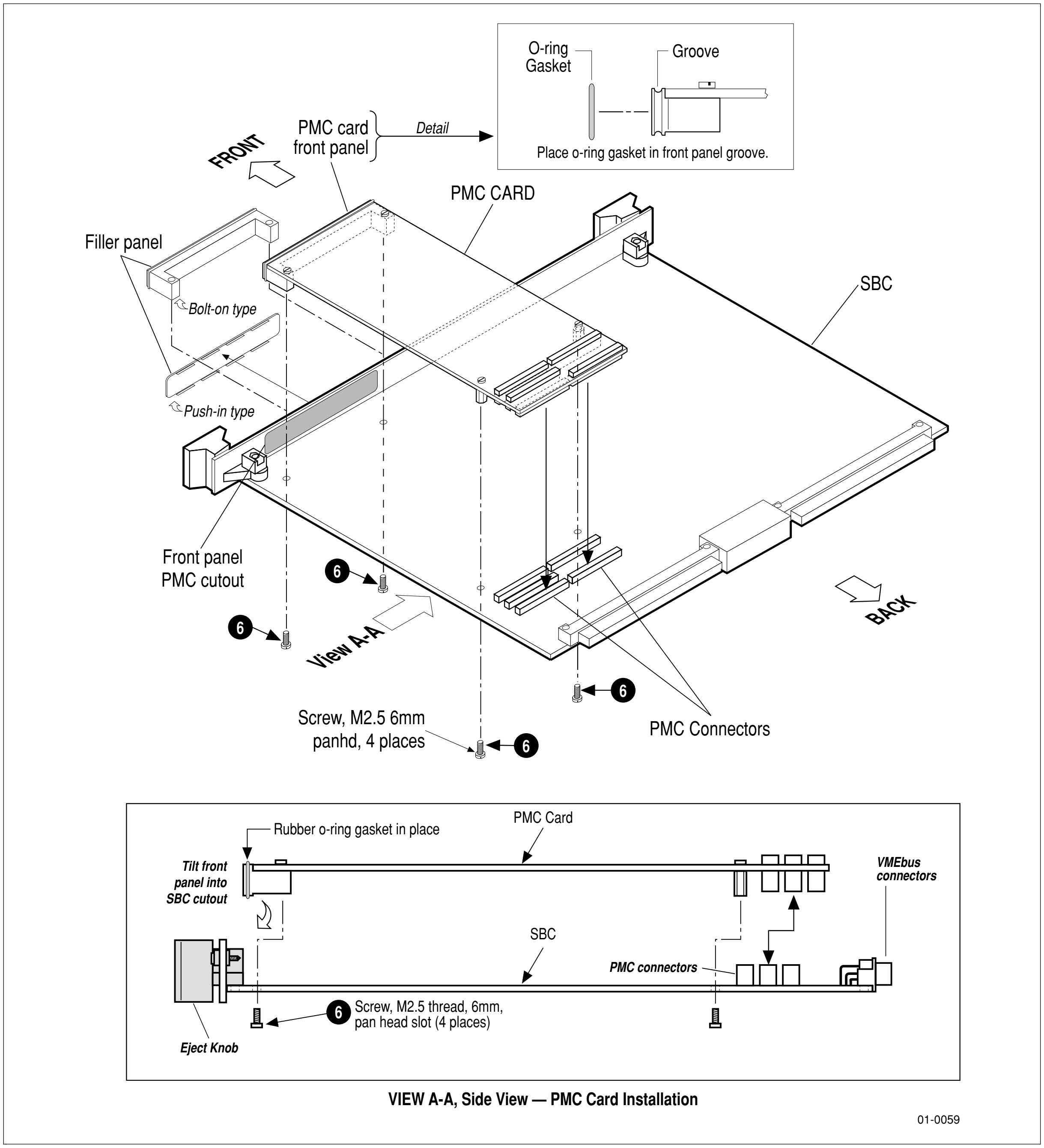
d. Turn VGM5 assembly over.

e. Install four 6 mm M2.5 slot-head screws (item 6, typically supplied with PMC) from rear (solder) side of VGM5 motherboard. Two screws engage the standoffs on the PMC card. The other two screws engage the threaded holes in the PMC card front panel. See *Location, PMC card securing screws* drawing earlier in this chapter.

Removal is reverse of installation.

### Single PMC installation — Required hardware

| Item locator | Quantity in assy | Synergy part number | Item description |
|---|---|---|---|
| 6 | 4 | Fas/SwM25FP6SS | Screw, M2.5 thread, pan head, phillips, 6 mm long, stainless steel (or use whatever screw fastener is supplied in PMC card kit) |


O-ring Gasket
Groove
Place o-ring gasket in front panel groove.
PMC card front panel
Detail
FRONT
PMC CARD
Filler panel
Bolt-on type
Push-in type
SBC
Front panel PMC cutout
6
6
View A-A
6
6
Screw, M2.5 6mm panhd, 4 places
PMC Connectors
BACK
Rubber o-ring gasket in place
PMC Card
Tilt front panel into SBC cutout
VMEbus connectors
SBC
PMC connectors
6
Screw, M2.5 thread, 6mm, pan head slot (4 places)
Eject Knob
VIEW A-A, Side View — PMC Card Installation
01-0059

PMC card installation

# Installing the P0 overlay

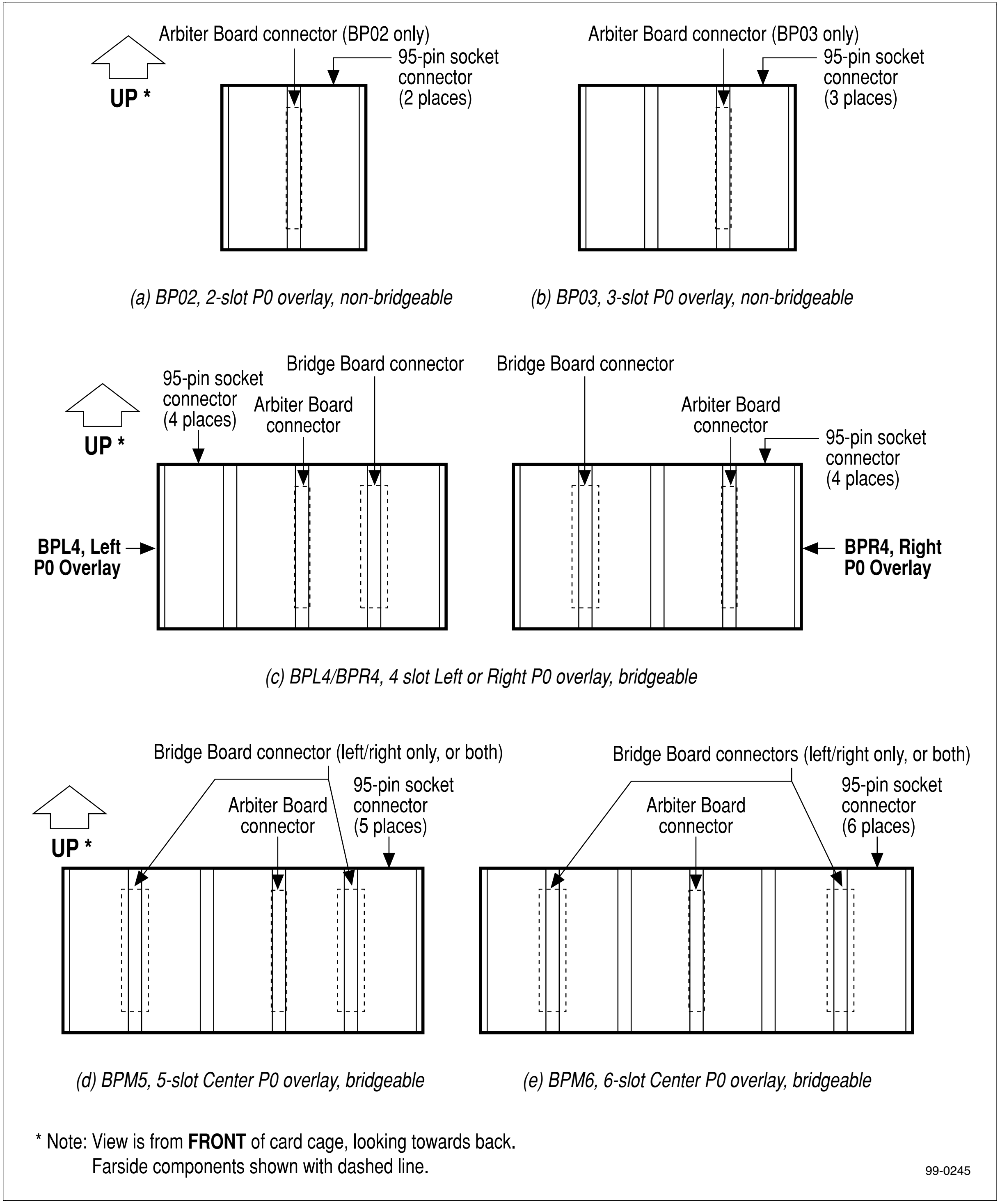
The P0 overlay is used to interconnect boards within the same cardcage via Synergy's P0•PCI™ interface. Refer to the ***PCI-PCI Bridge Interface*** chapter (page 253) for a description of the interface.

The P0 overlay board comes in left, right, and center configurations of varying slot capacities. Each overlay uses a small, plug-in arbiter board. Some overlay models allow joining with another overlay section via a bridge board. The table below lists the P0 overlay components for use with the P0•PCI™ interface.

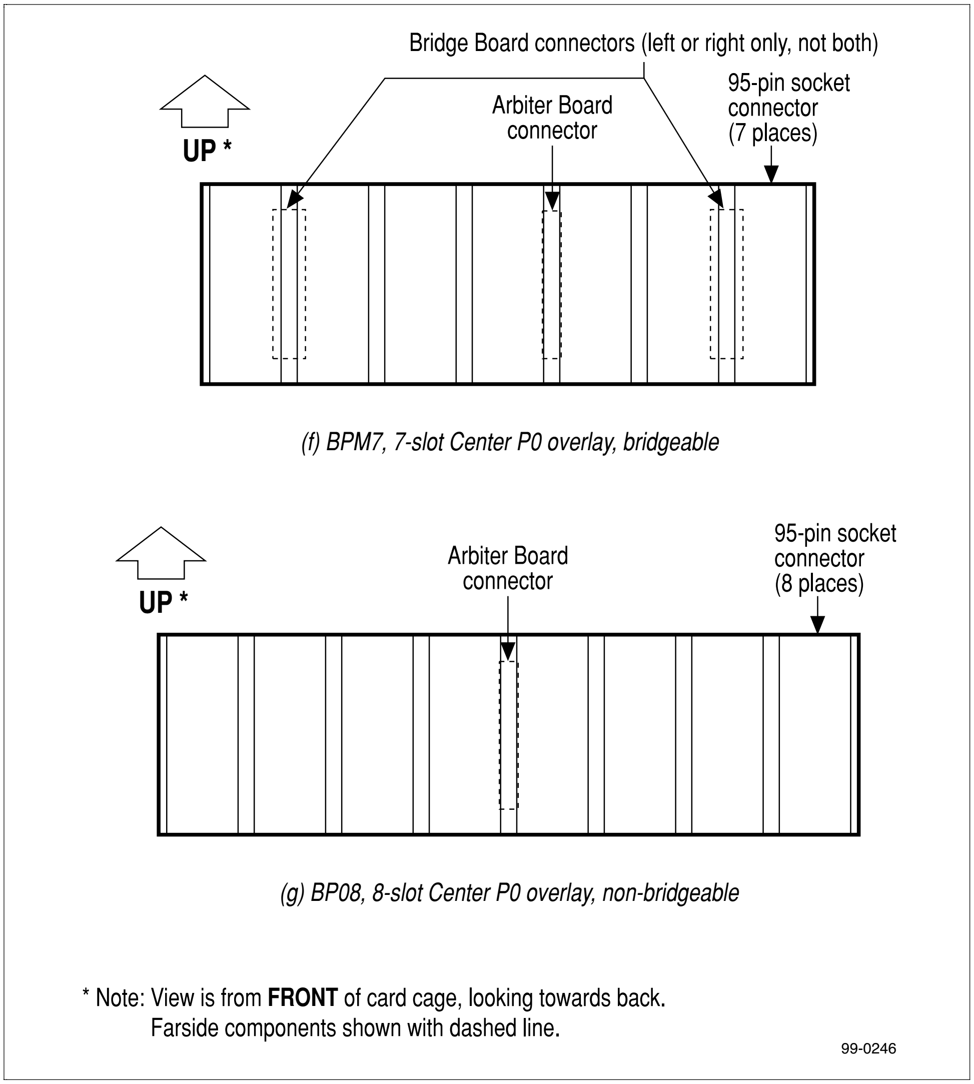
***P0 overlay components***

| Model Number | Description |
|---|---|
| BP08 | 8-slot P0 overlay, non-bridgeable |
| BPM7 | 7-slot P0 overlay, bridging to left or right |
| BPM6 | 6-slot P0 overlay, bridging to left/right/both |
| BPM5 | 5-slot P0 overlay, bridging to left/right/both |
| BPR4 | 4-slot P0 overlay, bridging to left only |
| BPL4 | 4-slot P0 overlay, bridging to right only |
| BP03 | 3-slot P0 overlay, non-bridgeable |
| BP02 | 2-slot P0 overlay, non-bridgeable |
| BBP0 | Bridge board |
| DPPA | Arbiter board, standup type |
| DPPB | Arbiter board, flat type |

The drawing below shows the available P0 overlay boards.


Arbiter Board connector (BP02 only)
95-pin socket connector (2 places)
UP *
Arbiter Board connector (BP03 only)
95-pin socket connector (3 places)
(a) BP02, 2-slot P0 overlay, non-bridgeable
(b) BP03, 3-slot P0 overlay, non-bridgeable
95-pin socket connector (4 places)
Bridge Board connector
Arbiter Board connector
Bridge Board connector
Arbiter Board connector
95-pin socket connector (4 places)
UP *
BPL4, Left P0 Overlay
BPR4, Right P0 Overlay
(c) BPL4/BPR4, 4 slot Left or Right P0 overlay, bridgeable
Bridge Board connector (left/right only, or both)
Arbiter Board connector
95-pin socket connector (5 places)
Bridge Board connectors (left/right only, or both)
Arbiter Board connector
95-pin socket connector (6 places)
UP *
(d) BPM5, 5-slot Center P0 overlay, bridgeable
(e) BPM6, 6-slot Center P0 overlay, bridgeable
* Note: View is from FRONT of card cage, looking towards back.
Farside components shown with dashed line.
99-0245

**P0 overlay board models**


Bridge Board connectors (left or right only, not both)
Arbiter Board connector
95-pin socket connector (7 places)
UP *

(f) BPM7, 7-slot Center P0 overlay, bridgeable

Arbiter Board connector
95-pin socket connector (8 places)
UP *

(g) BP08, 8-slot Center P0 overlay, non-bridgeable

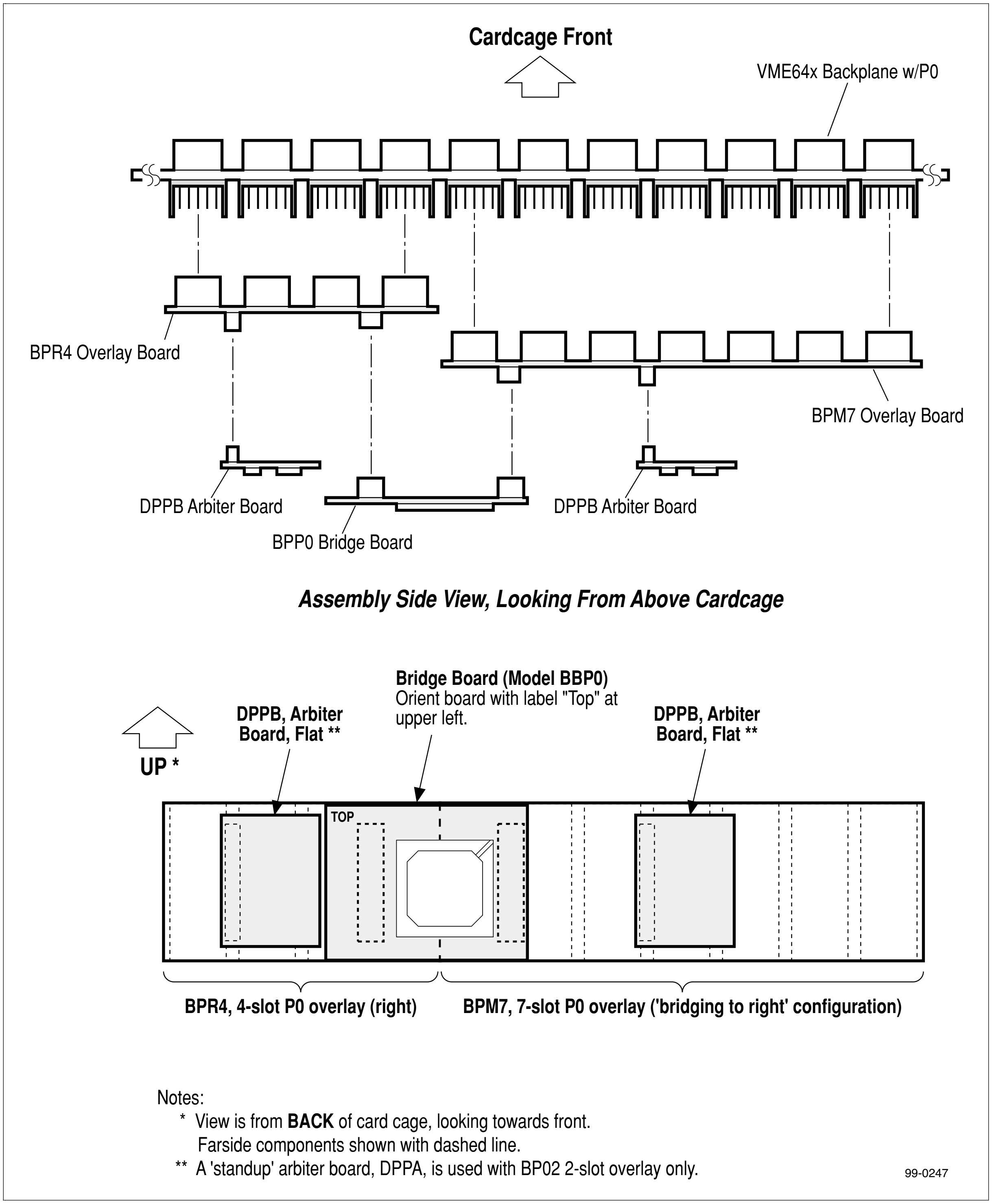
* Note: View is from **FRONT** of card cage, looking towards back.
Farside components shown with dashed line.

***P0 overlay board models (continued)***

The drawing below is provided as a guide to assembly of P0 overlay components. The minimum installation is an overlay board (with its arbiter plugged in) plugged into the VME backplane. Other installations may have one or more overlay sections joined together with a bridge board. Observe the following precautions when installing the P0 overlay boards.

⚠ First check to make sure that the P0 backplane pins are straight. With the pins verified (or made) straight, carefully align the overlay board onto the P0 backplane pins. Ensure that all backplane P0 pins properly engage the overlay board socket before fully seating the board! If board doesn't begin to seat when pressed against the pins, BACK OFF and try aligning the board again.

⚠ To avoid accidental damage to the arbiter board, **always** remove it from its P0 overlay socket before handling the P0 overlay board. This applies especially to the standup type arbiter (DPPA) used with the BP02 2-slot P0 overlay.


Cardcage Front
VME64x Backplane w/P0
BPR4 Overlay Board
BPM7 Overlay Board
DPPB Arbiter Board
DPPB Arbiter Board
BPP0 Bridge Board
Bridge Board (Model BBP0)
Orient board with label "Top" at upper left.
DPPB, Arbiter Board, Flat **
DPPB, Arbiter Board, Flat **
UP *
TOP
BPR4, 4-slot P0 overlay (right)
BPM7, 7-slot P0 overlay ('bridging to right' configuration)
99-0247

*Assembly Side View, Looking From Above Cardcage*

Notes:
* View is from **BACK** of card cage, looking towards front.
Farside components shown with dashed line.
** A 'standup' arbiter board, DPPA, is used with BP02 2-slot overlay only.

***P0 overlay boards, typical component assembly***

# Installation notes

## Bus grant signal problems with Hybricon VME64x backplanes

### Problem

Installing a Universe II-equipped SBC in slot 1 of some early Hybricon VME64x backplanes prevents the bus grant daisy chain signals from being passed on to the next slot (slot 2).

### Observation

Some early Hybricon VME64x backplanes use active logic to drive the bus grant daisy chain signals to the next slot. The logic requires that the signal be low (active) coming out of the slot AND low (active) going into the slot in order to be driven to the next slot. This logic is also included on the backplane for slot 1 even though there is no slot to the left of slot 1. The backplane includes pull-down resistors (82K ohms) for the BGnIn\s and BGnOut\s for slot 1. The Universe II VME interface chip, however, has an internal pull-up (~10K ohms) on its BGnIn\s that is stronger than those on the backplane. A board using the Universe II chip in slot 1 causes the active logic on the backplane to not pass the BGnOut\s to the next slot because the active logic sees a high (inactive) signal going into slot 1. BG3Out\ from slot 1 works because of the pull-down on BG3In\ for VME auto-system controller.

### Solution

If the VGM5 board (or any SBC using the Universe II) is to be used in slot 1 of an early Hybricon VME64x backplane, connect the backplane's slot 1 BG_In\s to ground. This can be accomplished by adding four (4) short wires as listed in the table below:

**Slot 1 wiring fix for some early Hybricon VME64x backplanes**

| From | To |
|---|---|
| P1 - Z4 (GND) | P1 - B4 (BG0In\) |
| P1 - Z6 (GND) | P1 - B6 (BG1In\) |
| P1 - Z8 (GND) | P1 - B8 (BG2In\) |
| P1 - Z10 (GND) | P1 - B10 (BG3In\) |

# 3 Basic Bus Descriptions

This section provides basic background information about the various buses/interfaces used in the VGM5.

- PowerPC bus
- VME64 bus
- PCI bus
- SCSI bus
- Fast Ethernet interface

# PowerPC bus

The system communicates with the PowerPC processor(s) through the MPC106 (Grackle) PCI Bridge/Memory controller. The Grackle connects all memory, front panel switches/LEDs, serial ports, RTC/NVRAM, User Flash, and mailboxes to the PowerPC bus. Moreover, the Grackle interfaces the PowerPC bus to the SBC's local PCI bus.

The PowerPC processor uses separate address and data buses plus various control and status signals for performing reads and writes. The address bus is 32 bits wide and the data bus is 64 bits wide. For memory accesses, the address and data buses are independent to support pipelining and split transactions.

The bus interface is synchronous, with all inputs sampled and all outputs driven from the rising edge of the bus clock. The bus runs at 66 MHz. The PowerPC chip’s internal multiplier boosts this frequency to its rated speed. The multiplier ratio is configured at assembly time by soldered-in jumpers on the board.

Access to the PowerPC bus interface is granted through an external arbitration mechanism that allows devices to compete for bus mastership. In the VGM Series, these devices include the PowerPC to PCI Bridge/Memory Controller (Grackle) chip, and the PowerPC to VME Bridge circuit.

The Grackle chip handles PCI–PowerPC bus accesses. The Power PC to VME bridge handles direct VME–PowerPC bus accesses.

For more details on PowerPC bus arbitration, refer to the Motorola/IBM User’s Manual for the appropriate processor resident on your VGM Series board.

# VME64 bus

## Overview

The VMEbus *(Versa Module Eurocard bus)* is a microcomputer architecture whose physical and electrical characteristics are defined in the IEC 821 and IEEE 1014-1987 specifications. Standard VMEbus supports separate address and data lines of up to 32 bits each. This bus uses a backplane in which VMEbus modules are interconnected using DIN-41612 connectors designated as P1/J1 and P2/J2 (module/backplane designations respectively).

Standard VMEbus modules come in two form factor types:

- Single height (3U) for single backplane using P1/J1 connectors
- Double height (6U) for two backplanes (or combined backplane) using P1/J1 and P2/J2 connectors

The original VMEbus specification has been refined through several revisions (A, B, C, C.1, IEC 821 and IEEE 1014-1987). On April 10, 1995, a new VME64 standard was approved for publication as ANSI/VITA 1-1994. The VME64 standard was based on the VME Revision C.1 specification and adds several features including 64-bit address and data transfers. However, 64-bit addressing is not supported by VGM Series boards since the PowerPC processor has only a 32-bit address bus.

The table below summarizes the VMEbus architecture and features supported by VGM Series boards.

**VGM Series VMEbus feature support**

| Standard VMEbus | VME64, add: | VME64 Extensions, add: |
|---|---|---|
| 32-bit address bus<br>address range<br>16-bit<br>24-bit<br>32-bit<br>(64-bit)<br><br>32-bit data bus<br>data path width<br>8-bit<br>16-bit<br>24-bit<br>32-bit<br>(64-bit)<br><br>7 interrupt levels<br>Master/Slave architecture<br>Functional modules<br>Master<br>Slave<br>System Controller<br>Sub-busses<br>Data Transfer Bus<br>Data Transfer Arbitration Bus<br>Priority Interrupt Bus<br>Utility Bus | – 64-bit data transfer<br>– Locked cycles<br>– Rescinding DTAck\<br>– Autoslot ID<br>– Auto Sys. Controller detection | – 160-pin P1/P2 (wide P1/P2)<br>– User defined P0 conn.[1]<br>– Slot geographical addressing<br>– Mate first, break last precharge pins on P1/P2 for hot-swapping applications<br>– ETL bus transceivers<br>– EMI front panel |

Notes: 1. Supported in select PowerPC Series models.

## VGM5 VMEbus implementation

The VGM5 VMEbus functionality is provided by two separate but interrelated interfaces:

- Direct VME interface
- Universe II PCI to VME64 bridge

The Universe II chip provides the primary VME interface. It can do most all VME functions, including system controller, master and slave single and block transfers, and interrupt generation and handling. It cannot do read-modify-write (RMW) cycles and efficient (fast) single transfers.

The direct VME interface is designed to fill in the gaps in the Universe II's VMEbus performance. It provides a fast path for single transfers in both master and slave modes, and is capable of running master and slave RMW cycles.

It is possible to program both the Universe II and the direct slave interfaces to respond to the same address space, but for different transfer types. Specifically, the Universe II may be set to respond to block transfers and the direct interface to respond to single transfers in the same address range. This gives the best performance for all types of transfers.

⚠ Due to the possibility of encountering a deadlock condition, it is recommended that the direct slave interface not be used in conjunction with the Universe II single-transfer master feature. The symptom of this deadlock is a bus error on the direct slave transfer.

The VGM5 has three different functional modules involved with bus ownership; the system controller, the Universe requester, and the direct VME requester.

The system controller resides in the Universe II chip. It is enabled when the VGM5 is installed in Slot 1. The system controller performs bus arbitration and system reset tasks. It is not directly related to the VGM5's bus requesters. All bus requesters in the system use the arbiter in this system controller.

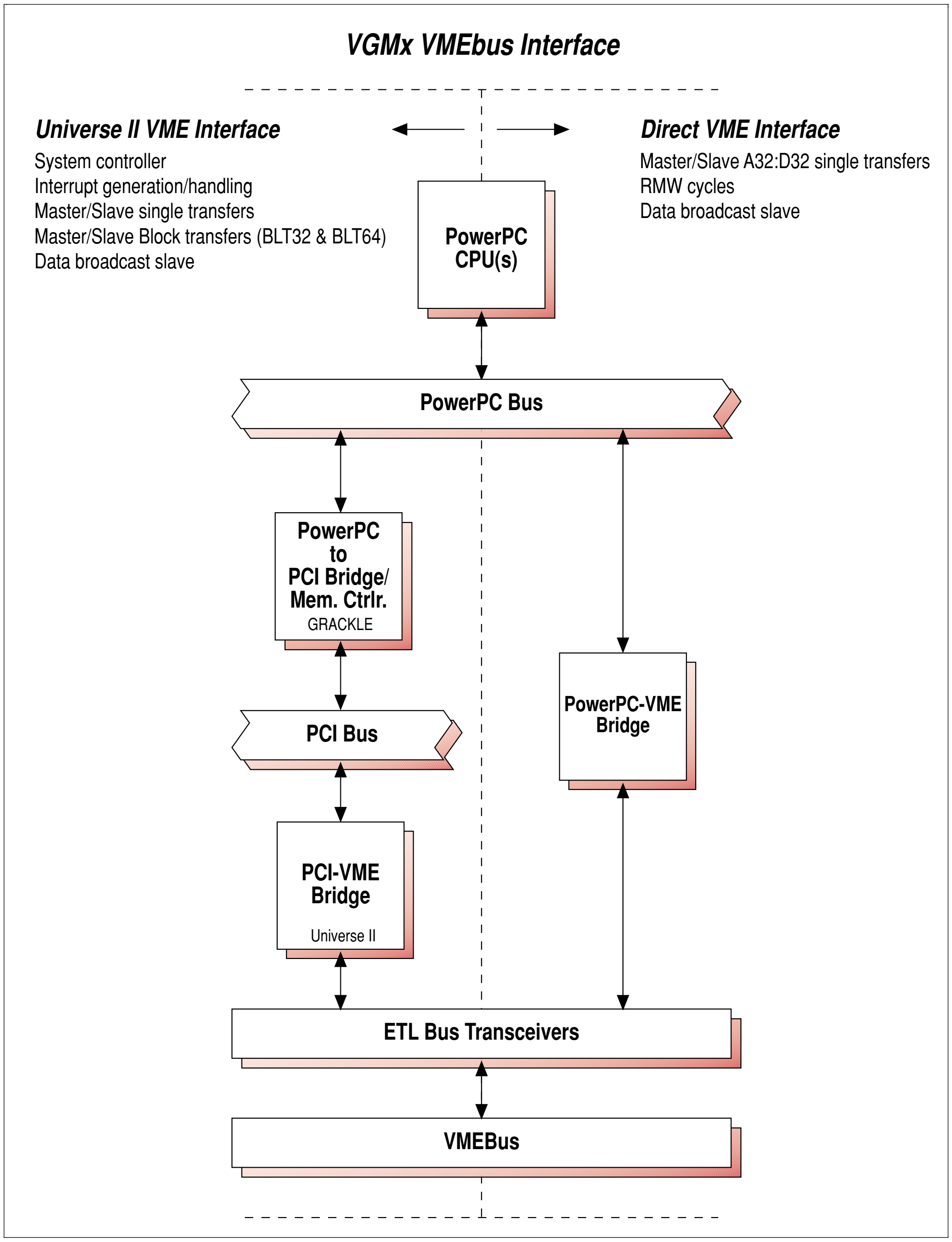
The Universe requester is used when the Universe VME master function is used; i.e. when the CPU accesses the VMEbus through a Universe VME master window.

The direct VME requester is used when the CPU accesses the VMEbus through the Direct VME address space of 0x4000_0000 to 0x7FFF_FFFF.

The Universe and direct VME requesters each have their own selectable bus request level, release mode, fair mode, etc. which may be set to different values.

The broadcast slave interface is an appendage to both the Universe II and direct slave interfaces, providing a means of writing data to several boards at once. Since it can work with both Universe II and direct slave interfaces, care must be taken to insure that a particular broadcast slave cycle is handled only by one slave interface (Universe II or direct) and not both at the same time, since there is only one broadcast slave address window.

The block diagram below shows the functional blocks that make up the VGM5 VMEbus interface.



*Block diagram, VGM5 VMEbus interface*

# PCI bus

## Introduction

PCI, or Peripheral Component Interconnect, is a computer industry specification for interconnecting peripherals with both the system memory and the CPU. Though often referred to as a “local bus” since it accesses the CPU and system memory directly, PCI is actually a separate bus isolated from the CPU. At the early stages of PCI bus use, this processor independence was typically provided by a PCI bridge chip with a 32-bit PCI bus running at 33 MHz for a maximum data transfer rate of 132 MB/sec. With the PCI 2.1 specification, both data width and clock speed doubled to 64-bits and 66 MHz respectively for a maximum data transfer rate of 528 MB/sec (in the real world, actual rate will be lower due to bus latency times). Various clarifications and enhancements to PCI 2.1 were subsequently included in the PCI 2.2 specification. The PCI 2.3 specification migrated the PCI bus from the original 5V signaling to a 3.3V signaling bus and included some changes in the system board keyed connector support. PCI performance received yet another boost with the PCI-X specification, which provided a path for ever increasing bus speeds starting at 133 MHz (or 1 GB/s). The PCI bus’ processor independence has caused the interface to gain in popularity as a solution to providing cost-effective, high performance peripheral interconnections regardless of the processor or platform used.

Key features of PCI:

- Multiple busmasters on the same bus — with bus mastering, a device can take control of the bus and provide main memory I/O without CPU intervention.

- Auto-configuring — all components plugged into the PCI bus can be automatically detected and configured for use during the system startup routine.
- Interrupt (IRQ) sharing — the PCI bus is able to share a single interrupt between cards.
- High bus bandwidth — 132 MB/sec and 264 MB/sec for 32-bit and 64-bit PCI respectively (@33 MHz).

The VGM5 PCI interface is provided by the MPC106 (Grackle chip). It provides a 33 MHz PCI bus interface that is compliant with the PCI 2.1 specification which is backwards compatible with PCI 2.0. The MPC106, however, supports only 32-bit PCI connections. Refer to the Grackle chip discussion in Section 4, page 95, for more information.

## PMC cards

The PCI Mezzanine Card (PMC) is an industry-standard design that allows PCI based I/O cards to be used in VMEbus and CompactPCI motherboard designs. The IEEE P1386 CMC (Common Mezzanine Card) standard defined the available PMC card sizes. The table below lists the PMC sizes typically used for VMEbus and CompactPCI systems.

***Typical PMC card size designations and dimensions***

| Designation | Width in mm | Depth in mm |
|---|---|---|
| Single | 74.0 | 149.0 |
| Double | 149.0 | 149.0 |

PMC cards come in 32- and 64-bit designs. With 64-bit designs, an additional connector is required over the standard 32-bit design. The VGM5 boards come with this extra connector, allowing 64-bit PMC cards to be used.

## PCI implementation details

The following is a nuts and bolts description of how PCI is implemented in a system from a software standpoint.

### PCI address spaces

PCI devices are accessed by the CPU from three address spaces, PCI I/O, PCI Memory and PCI Configuration space. The PCI I/O and PCI

Memory address spaces are used by the device drivers. The PCI Configuration space is used by the PCI initialization code being run at bootup.

All of these spaces are for PCI chips or devices. Exactly what registers are used and their location depend on the assigned IDSel number and/or its slot location (if PMC) and specific programming as set during PCI configuration.

## PCI configuration

Every PCI device in the system, including PCI-PCI bridges, has a data structure located in PCI configuration address space called the PCI Configuration header. This structure, which has a maximum length of 256 bytes (0x00-0xFF), allows the system to identify and control the device during configuration.

| 31 | 16 | 15 | 0 | |
|---|---|---|---|---|
| Device ID | | Vendor ID | | 0x00 |
| Status | | Command 0x0000 | | 0x04 |
| Class Code = 0x020000 | | | Revision ID= 0x00 | 0x08 |
| BIST | Header Type | Latency Timer | Cache Line Size | 0x0C |
| Base Address Register(s) (BARn) | | | | 0x10-0x24 |
| Subsystem ID | | Subsystem Vendor ID | | 0x2C |
| Expansion ROM Base Address | | | | 0x30 |
| Reserved | | | Cap_Ptr | 0x34 |
| Reserved | | | | 0x38 |
| Max_Lat | Max_Gnt | Interrupt Pin | Interrupt Line | 0x3C |
| Extended Configuration Space (Capability Registers) | | | | 0x40-0xFF |

Shown above is a typical layout of a PCI configuration header. It contains the following fields:

- **Vendor Identification** — A unique number describing the originator of the PCI device. The PCI-SIG (see page 66) keeps a registry of PCI vendors and their corresponding 2-byte hex ID.
- **Device Identification** — A unique number describing the device itself. The vendor identifies a specific device in its lineup with this ID.
- **Status** — This field gives the status of the device with the meaning of the bits of this field set by the PCI Local Bus Specification

- **Command** — By writing to this field the system controls the device, for example allowing the device to access PCI I/O memory.
- **Class Code** — This identifies the device category. There are standard classes for every sort of device; video, SCSI and so on. The generic example shows a class code for an Ethernet controller (0x020000). Class codes are discussed in Appendix D of the PCI 2.2 specification.
- **Base Address Registers** — These registers are used to determine and allocate the type, amount and location of PCI I/O and PCI memory space that the device can use.
- **Interrupt Pin** — Four of the physical pins on the PCI card carry interrupts from the card to the PCI bus. The standard labels these as A, B, C and D. The Interrupt Pin field describes which of these pins this PCI device uses. Generally it is hardwired for a particular device. That is, every time the system boots, the device uses the same interrupt pin. This information lets the interrupt handling subsystem manage interrupts from this device.
- **Interrupt Line** — The Interrupt Line field of the device's PCI Configuration header is used to pass an interrupt handle between the PCI initialization code, the device's driver and OS's interrupt handling subsystem. The number written there is meaningless to the device driver but it allows the interrupt handler to correctly route an interrupt from the PCI device to the correct device driver's interrupt handling code within the operating system.
- **Capability Pointer (and Capability ID)** — This field points to a linked list of one or more special capability registers (for power management, vital product data, etc.) within PCI config space. When the Cap_Ptr points elsewhere in PCI config space to a register, the first byte of that register identifies a particular capability (for example, 0x01 for power management, 0x03 for vital product data, etc.) The Capability IDs are summarized in Appendix H of the PCI 2.2 specification. The next byte after the Capability ID (Next Item Pointer) points to another special capability register (if any). Bytes subsequent to the Next Item Pointer are used for the particular capability register indicated by the Capability ID.

During system boot time, PCI devices are detected and configured automatically via a software process called PCI Discovery. Other names for this process include PCI Enumeration and PCI Auto Configuration.

The exact mechanism for PCI Discovery is system-specific. For Synergy PowerPC SBCs, all PCI devices or slots in the system are hardwired with an address line that functions as the device's IDSEL number. IDSEL is essentially a 'chip select' for a device during PCI configuration.

Devices are detected by reading the Vendor and Device IDs in all possible device locations via IDSEL (reads from PCI Configuration Data space). If a location is empty, the read returns all 1s (0xFFFFFFFF) and the system goes on to read the next location. A valid Vendor and Device ID results in the system narrowing down the capabilities by reading additional device configuration data. If a device indicates a multifunction device (e.g., a device with 2 or more controllers), a read of all locations every 0x100 is done to tally up all the sub-functions. After all functions are identified, the device base address registers (BARs) and other miscellaneous configuration registers (if any) are set up per the programming for that type device (writes to PCI Configuration Address space).

To find out just how much of each address space a given BAR is requesting, all 1s are written to the register and the result read back. The device will return zeros in the don't care address bits, effectively specifying the address space required. This design implies that all address spaces used are a power of two and are naturally aligned.

For example when you initialize the PCI device in the above example, it tells you that it needs 0x100 bytes of space of either PCI I/O or PCI Memory. The initialization code allocates it space. The moment that it allocates space, the Fast Ethernet device's control and status registers can be seen at those addresses.

The above process repeats until all locations (maximum of 21 PCI devices) are read.

### Endian issues, byte swapping

The PCI bus is inherently little-endian where byte 0 is the LSB. The PowerPC is big-endian where byte 0 is the MSB. This difference in endianness requires byte swapping code for accesses between PCI and PowerPC. The PowerPC instruction set includes a class of load and store instructions that perform byte swapping based on the size of data being transferred. The example in-line routines below show how this is done for word and half-word data.

```
#define ASM  volatile asm

// Read a longword from adr, little-endian
extern inline int lwbrx(void *adr)
{
    int data;
    ASM("lwbrx %0,0,%1": "=r"(data):"r"(adr));
    return data;
}

// Store longword data to adr, little-endian
extern inline void stwbrx(int data, void *adr)
{
    ASM("stwbrx %0,0,%1": :"r"(data), "r"(adr));
}

// Read a 16-bit word from adr, little-endian
extern inline int lhbrx(void *adr)
{
    int data;

    ASM("lhbrx %0,0,%1": "=r"(data):"r"(adr));
    return data;
}

// Store 16-bit word data to adr, little-endian
extern inline void sthbrx(int data, void *adr)
{
    ASM("sthbrx %0,0,%1": :"r"(data), "r"(adr));
}
```

## PCI standards organization

PCI architecture specifications are maintained by the PCI Special Interest Group (PCI-SIG), an industry standards organization formed in 1992 to develop and manage the PCI standard. PCI specification documents are available for purchase from:

The PCI-SIG:

PCI SIG Specification Distribution
5440 SW Westgate Drive, Suite 217
Portland, OR 97221 USA
Phone: 503-291-2569
Email: administration@pcisig.com
Web: **http://www.pcisig.com/**

Global Engineering Documents:

Global Engineering Documents
15 Inverness Way East
Englewood, CO, 80112
Phone:+1-800-854-7179 FAX:+1-303-397-2740
EMail:global@ihs.com
Web: **http://www.global.ihs.com/**

# SCSI bus

## Overview

The Small Computer Systems Interface (SCSI) is a parallel I/O bus that lets a host computer access various peripheral devices without the need for specialized hardware and software commands for each device. The host's SCSI interface acts as a translator between the host and a particular type of peripheral which provides the host with device independence. For example, one vendor's SCSI disk drive could be replaced with another vendor's SCSI disk drive with no changes to the existing driver code. In addition, because SCSI is a general-purpose interface, tape drives, hard disks, CD-ROMs, and a variety of other peripherals can quickly be added to the SCSI bus since they all speak the same high level language when communicating with the host.

## History

The SCSI interface started life as SASI (Shugart Associates Systems Interface), a joint development between Shugart Associates and NCR. In late 1981, SASI was submitted to the ANSI X3T9 standards committee as a proposed interface standard. The standards committee renamed the interface SCSI and in June 1986, SCSI was finally made an ANSI standard with the publication of specification X3.131-1986. This particular 8-bit SCSI was later referred to as SCSI-1.

A fast, wide SCSI interface, referred to as SCSI-2, was then designed with increased throughput of up to 20 MB/S. The SCSI-2 interface was finally approved by ANSI on January 31, 1994 and designated as specification X3.131-1994.

Even before the release of the SCSI-2 standard, work on SCSI-3 began in 1993. It became apparent to the standards committee that a variety of technology (i.e., new serial interfaces for desktop and high performance environments) was vying for inclusion in the SCSI spec. As a result, the SCSI-3 standard was turned into layers (similar to a networking standard) so that parts which were fast changing could be isolated and standardized on different schedules. Since the creation of SCSI-3, the confusion surrounding SCSI standards has increased. SCSI-3 is defined in a collection of about 30 different standards.

> Note VGM5’s Wide Ultra (Fast-20) SCSI interface is defined in ANSI standard X3.277-1996 which is an addendum to the SCSI-3 Parallel Interface (SPI) standard.

Two organizations exist to maintain and promote the SCSI standard:

- SCSI Trade Association (STA) — This industry trade organization communicates the benefits of SCSI. For more information on this organization, refer to the STA website:

  **http://www.scsita.org/**
- T10, National Committee on Information Technology Standards (NCITS) Technical Committee — This is a standards committee that promulgates low-level interface standards. This group works with industry members to gain consensus on the low-level interface rules. These rules start out as draft standards which eventually become ANSI standards. For more information on the T10 committee, refer to the T10 home page on the web:

  **http://www.t10.org/**

To keep pace with improvements in computer technology, SCSI continues to evolve with wider data paths and increased transfer speeds. The table below lists the current varieties of the SCSI interface as given by the SCSI Trade Association.

***List of SCSI types (SCSI Trade Organization)***

| SCSI Type (per STA) | Bus Speed, MBytes/Sec, Maximum | Bus Width, Bits | Device Support (including Host) |
|---|---|---|---|
| SCSI-1† | 5 | 8 | 8 |
| Fast SCSI† | 10 | 8 | 8 |
| Fast Wide SCSI | 20 | 16 | 16 |
| Ultra SCSI† | 20 | 8 | 8 |
| Wide Ultra SCSI | 40 | 16 | 16 |
| Ultra2 SCSI † | 40 | 8 | 8 |
| Wide Ultra2 SCSI | 80 | 16 | 16 |
| Ultra3 SCSI or Ultra160 SCSI | 160 | 16 | 16 |
| Ultra320 SCSI | 320 | 16 | 16 |
| Ultra640 SCSI | 640 | 16 | 16 |

†Use of the word "narrow", preceding SCSI, Ultra SCSI, or Ultra2 SCSI is optional.

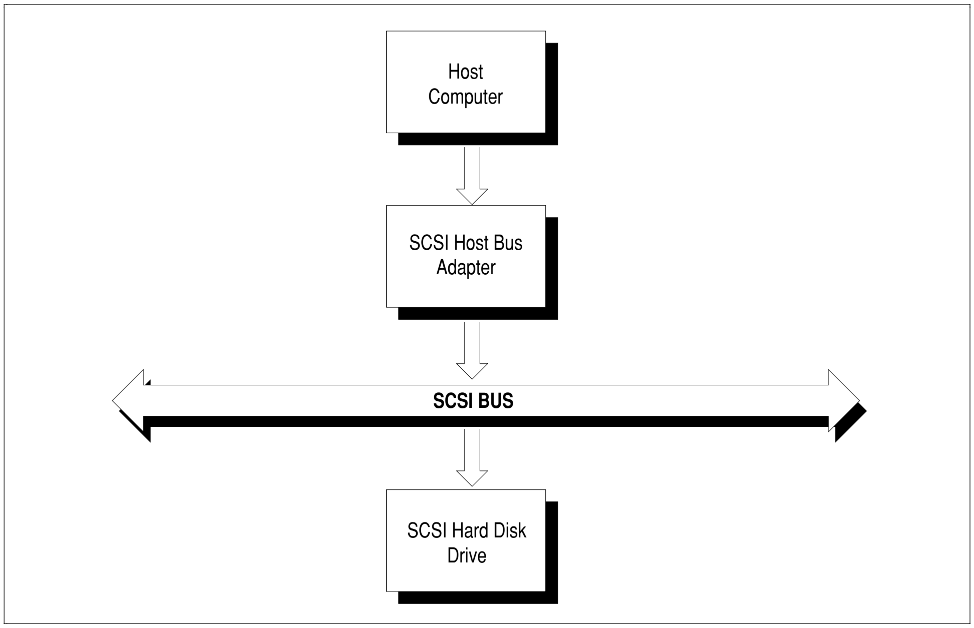
## SCSI specifications and publications

For a complete list of SCSI-related specifications (draft and approved) and other publications, refer to the T10 publications list on the web:

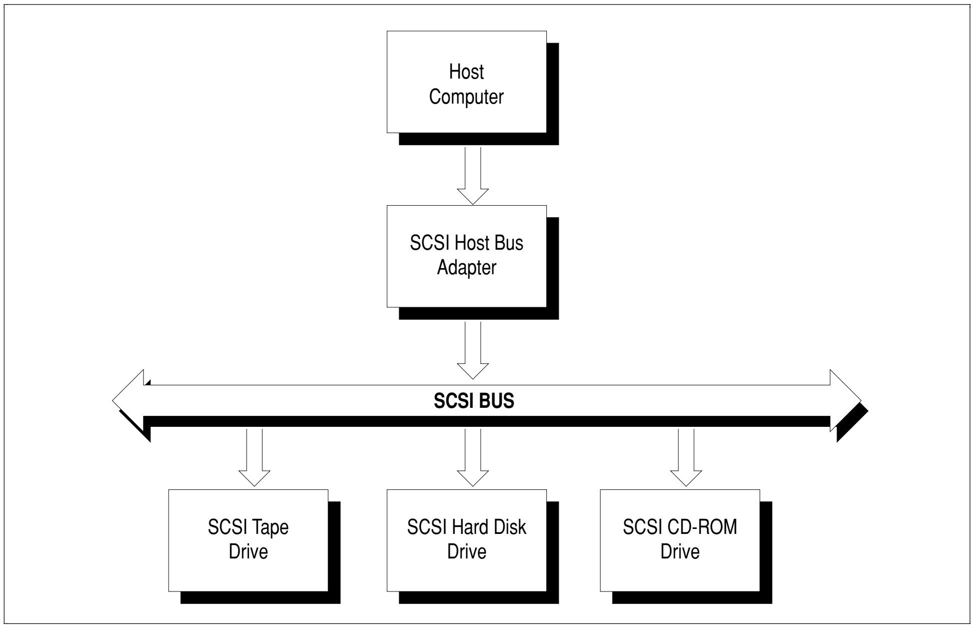
**http://www.t10.org/pubs.htm**

## Device connections

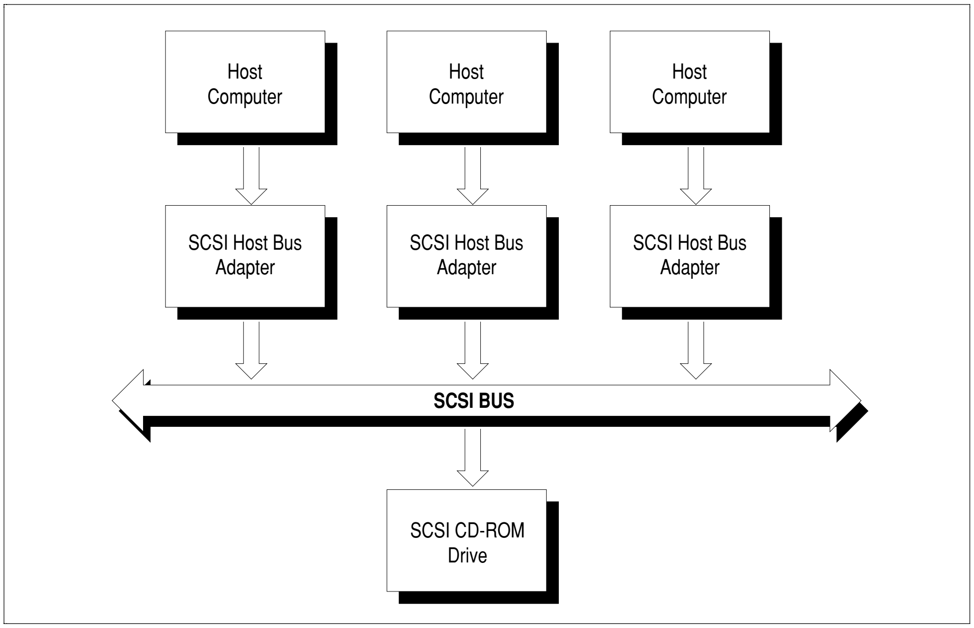
As shown in the table above, the SCSI bus supports 8 or 16 devices including the host(s). A SCSI device is either an initiator (host) or target (device that responds to the requests of an initiator to perform an operation). The bus protocol accommodates four types of SCSI device configurations as shown in the following figures:



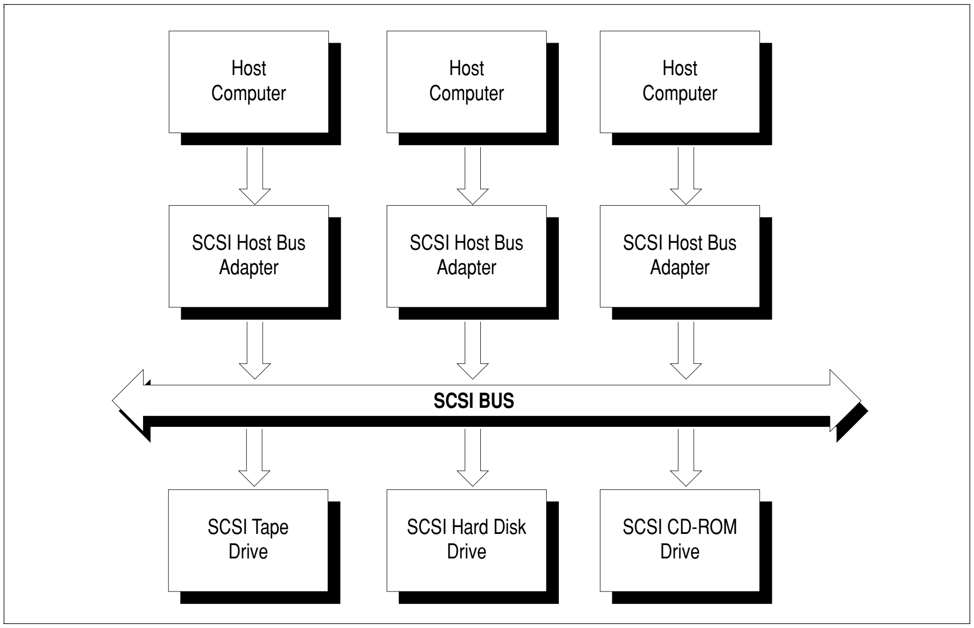
*Single Host (initiator)/Single Controller (target)*



*Single Host (initiator)/Multiple Controllers (targets)*



*Multiple Hosts (initiators)/Single Controller (target)*



*Multiple Hosts (initiators)/Multiple Controllers (targets)*

## Electrical connections

Electrical connections on the SCSI bus are either single-ended or differential. Single-ended SCSI connections use TTL level signals to drive a cumulative cable length of up to 6 meters (20 feet) while differential connections use either EIA-485 (high voltage differential or HVD) or EIA-644 (low voltage differential signaling or LVD) to drive a maximum cumulative cable length of up to 25 meters (82 feet).

For older SCSI interfaces, device connections to a particular SCSI bus must be all single-ended or differential — they cannot be combined on the same bus. To protect the differential driver circuits, the DIFFSENS signal is provided. This signal is a single-ended signal that is used as an active high enable for the differential drivers. If a single-ended device or terminator is inadvertently connected, DIFFSENS is grounded, which disables the differential drivers by placing them in high impedance state (tristate).

Newer SCSI interfaces that use low voltage differential (LVD) signaling can operate in "multimode" which allows a mix of low voltage differential and single-ended devices on the bus. In multimode, the DIFFSENS line is used to differentiate between SE (DIFFSENS ≤ 0.5 V), and LVD (DIFFSENS = 0.7 V–1.9 V). When a single-ended device is connected to the bus, the DIFFSENS line senses the voltage which causes all other attached devices to automatically configure themselves for single-ended operation (all "+" signal pins grounded). Since only SE or LVD buses can accommodate multimode operation, HVD devices are not allowed to be connected to an SE or LVD bus.

The VGM5 SCSI interface is fixed at single-ended. For connector information, refer to Appendix A, ***Wide Ultra SCSI connector*** (page 293).

The SCSI signal lines are divided into two basic groups, data lines (DB0–DB15, DBP0 and DBP1) and control signals (SEL, BSY, C/D, I/O, MSG, REQ, ACK, ATN and RST). Altogether, the SCSI-2 interface uses 68 lines.

The following table summarizes the ANSI standard SCSI-2/SCSI-3 bus signals supported by VGM5.

***SCSI-2/SCSI-3 bus signals***

| Pin No. | Mnemonic (see Note) | Signal | Driven By |
|---|---|---|---|
| 40 | ***DB0*** | Data Bus Line 0 | Initiator/Target |
| 41 | ***DB1*** | Data Bus Line 1 | Initiator/Target |
| 42 | ***DB2*** | Data Bus Line 2 | Initiator/Target |
| 43 | ***DB3*** | Data Bus Line 3 | Initiator/Target |
| 44 | ***DB4*** | Data Bus Line 4 | Initiator/Target |
| 45 | ***DB5*** | Data Bus Line 5 | Initiator/Target |
| 46 | ***DB6*** | Data Bus Line 6 | Initiator/Target |
| 47 | ***DB7*** | Data Bus Line 7 | Initiator/Target |
| 65 | ***DB8*** | Data Bus Line 8 | Initiator/Target |
| 66 | ***DB9*** | Data Bus Line 9 | Initiator/Target |
| 67 | ***DB10*** | Data Bus Line 10 | Initiator/Target |
| 68 | ***DB11*** | Data Bus Line 11 | Initiator/Target |
| 35 | ***DB12*** | Data Bus Line 12 | Initiator/Target |
| 36 | ***DB13*** | Data Bus Line 13 | Initiator/Target |
| 37 | ***DB14*** | Data Bus Line 14 | Initiator/Target |
| 38 | ***DB15*** | Data Bus Line 15 | Initiator/Target |
| 48 | ***DBP0*** | Data Bus Parity1 | Initiator/Target |
| 39 | ***DBP1*** | Data Bus Parity2 | Initiator/Target |
| — | DIFFSENS | Differential Sense | Any device |
| 17, 18, 51, 52 | TERMPWR | Terminator Power | Any device |
| 55 | ***ATN*** | Attention | Initiator |
| 57 | ***BSY*** | Busy | Initiator/Target |
| 58 | ***ACK*** | Acknowledge | Initiator |
| 59 | ***RST*** | Reset | Any device |
| 60 | ***MSG*** | Message | Target |
| 61 | ***SEL*** | Select | Initiator/Target |
| 62 | ***C/D*** | Control/Data | Target |
| 63 | ***REQ*** | Request | Target |
| 64 | ***I/O*** | Input/Output | Target |
| 19, 53 | No Connection | — | — |

Note: Signals in ***bold italic*** are provided as plus and minus signal pairs for differential SCSI. For single-ended SCSI, each of these signals is provided as one negative polarity line and the DIFFSENSE signal is unused.

## Physical topology

SCSI devices are connected one after the other in daisy-chain fashion. Up to seven devices can make up this chain. See ***Electrical connections*** above for the maximum cumulative length of the chain.

### Bus terminations

All SCSI signals must be terminated at each end of the SCSI chain to ensure clean signals and proper timing of bus operations. This is achieved either by a voltage divider resistor network powered by the TERMPWR pin on one of the SCSI device connectors or by active circuitry that provides the same function.

The VGM5 SCSI interface has active termination circuitry that is enabled/disabled with onboard jumper J02N (pins 3 & 4). See Section 2 ***Setting up the VGM5 hardware*** (page 25) for more information on the SCSI termination jumper.

## Bus communication control

The SCSI interface uses the following eight phases or bus states to control communication over the bus:

- ❏ Bus Free — indicates that no SCSI device is actively using the bus and that it is free.
- ❏ Arbitration — an optional phase in which SCSI devices arbitrate for use of the bus.
- ❏ Selection — lets an initiator select a target to perform a function such as a Read or Write command.
- ❏ Reselection — an optional phase in which a target reconnects to an initiator to continue an operation that was previously started but was suspended by the target.
- ❏ Command — lets a target request command information from the initiator.
- ❏ Data — allows data transfer from target to initiator or from initiator to target.
- ❏ Status — allows status information to be sent from target to initiator.
- ❏ Message — allows sending of single or multi-byte messages from target to initiator or from initiator to target.

The last four phases listed above are called information phases since they transfer command, data, status or message information.

## Data transfer options

Asynchronous and synchronous protocols are used in the SCSI bus. The asynchronous protocol requires a handshake for every byte transferred. The synchronous protocol transfers a series of bytes before the handshake occurs. This means a higher data transfer rate for synchronous mode versus that for asynchronous mode.

The asynchronous data transfer mode is the default (normal) mode since this mode does not need to be selected. All commands, messages, and status are always transferred asynchronously.

A synchronous target, however, can ask the initiator for synchronous transfer of data. The initiator responds to this request by either maintaining asynchronous data transfers or establishing synchronous data transfers by an exchange of messages containing the minimum transfer period and maximum REQ/ACK offset for each device. When synchronous data transfer is established, it is done using the greater of the two minimum transfer periods and the lesser of the two maximum REQ/ACK offsets.

The VGM5 SCSI interface supports 8-bit Ultra SCSI synchronous transfers of up to 20 MB/s and 16-bit Wide Ultra SCSI synchronous transfers of up to 40 MB/s.

# Fast Ethernet interface

Ethernet is a LAN (local area network) architecture that provides the means for computers and other peripherals located in a moderately sized geographical area to communicate with each other at high speed.

The Ethernet prototype was developed by Xerox Corporation in 1975 and grew to a standard LAN specification 10 years later (IEEE 802.3-1985) with the collaborative efforts of Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. From the standard Ethernet specification came the 10Base-T Ethernet standard which used inexpensive unshielded twisted pair cable terminated in modular plugs. The popularity of 10Base-T fueled the development of Fast Ethernet 100Base-TX which incorporated new signaling schemes to provide a 100 Mbps data rate over a range of twisted pair (100Base-TX, 100Base-T4) and fiber cabling (100Base-FX) types. 100Base-TX provides an easy migration path to higher performance since it can use existing 10Base-T cables and equipment for interim 10Base-T operation. Changing over to Category 5 cable and Fast Ethernet hubs automatically switches the network to Fast Ethernet operation.

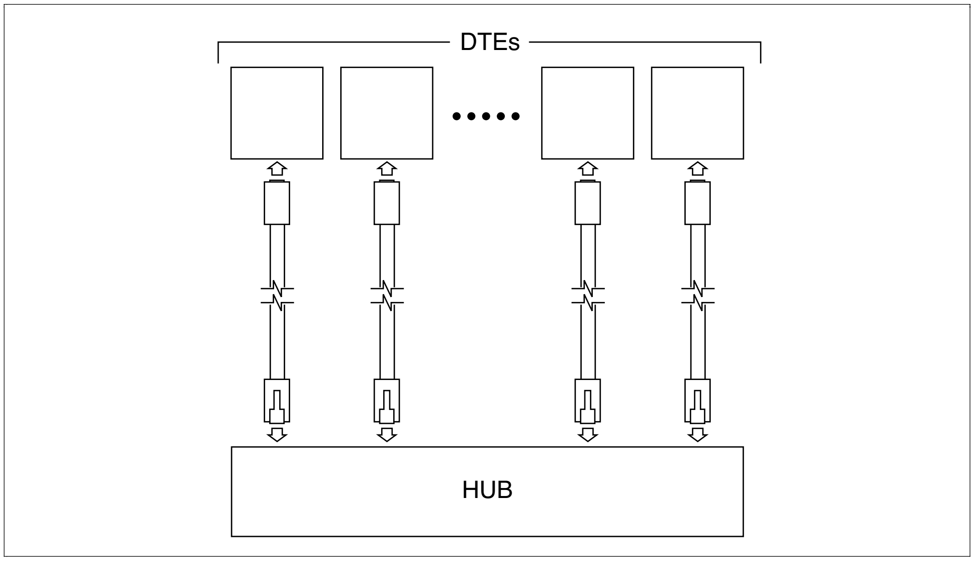
Ethernet provides what is called a “link level” facility since it deals with the lowest two layers of network architecture as defined by the ISO Model for Open Systems Interconnection: the Physical Level and the Data Link Layer.

With Ethernet, the type of data it transmits is immaterial since it does not concern itself with data protocol and interpretation. As such, Ethernet LANs are used for various types of computing platforms such as mainframe computers, Macs, PCs, UNIX/Linux systems, etc.

## Ethernet network connections

The VGM5 provides a 10Base/100Base-TX Ethernet (Fast Ethernet) port at the front panel. This type of Ethernet uses a star topology in which each DTE (data terminal equipment) is connected to a shared hub through a single, 4-pair unshielded twisted pair (UTP) cable. The UTP cable is similar to modular telephone cable. For network use, however, a higher grade (or category) of cable is typically used. For 10Base-T, Category 3 is the minimum, but Category 4 or 5 is more often recommended. For 100Base-TX, no less than Category 5 cable is recommended. Cable connections are made to an 8-pin RJ-45 modular jack. The maximum distance between DTE and hub is 100 m (328 ft.) for both 10Base-T and 100Base-TX.

The figure below shows a typical 10Base/100Base-TX Ethernet single hub network.

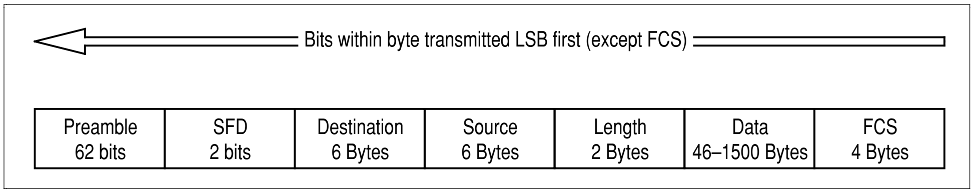


**10Base/100Base-TX Ethernet single hub network**

## Data transmission

Both clock and NRZ data information is Manchester-encoded in bit-serial form and encapsulated in a basic unit called a frame packet.

The frame packet is made up of seven fields in which the data field is bracketed by several bytes of information. The figure below shows the format of an Ethernet frame.



***Ethernet Frame Packet Format***

The packet fields are summarized below.

**Preamble** — is a series of alternating 1's and 0's that serve to synchronize the clock and other circuitry on all the receivers and repeaters on the network.

**Start of Frame Delimiter (SFD)** — consists of two consecutive 1's to signal the start of a frame.

**Destination** — six bytes to indicate the destination of the packet on the network.

**Source** — six bytes to indicate the node that sent the packet.

**Length** — two bytes to indicate the number of bytes contained in the data field.

**Data** — 46–1500 data bytes. Stations that need to send less than 46 bytes of data must pad the data to reach the minimum requirement. Stations that need to send in excess of 1500 bytes of data must send multiple frame packets.

**Frame Check Sequence** — CRC value of packet (not including preamble and SFD fields) for error detection. Receiver rejects the frame if the calculated CRC value of the received data does not match the transmitted CRC value.

### Ethernet ID or physical address

An Ethernet board is typically designed with a unique Ethernet ID (also called physical address) in ROM; by default any Ethernet packet sent to this ID will be received by the board and passed to the host. Packets addressed to other Ethernet IDs will be seen by the board, but ignored (by default).

The Ethernet ID is a 12-digit number. This number is made up of three bytes of manufacturer's ID followed by another three bytes of a unique identifier number. The Ethernet ID is what's contained in the Destination and Source fields of the Ethernet packet.

For Synergy boards, Synergy's 3-byte manufacturer's ID (00:80:F6) is compiled into the Ethernet driver code as a macro. The second half of the Ethernet ID is made up of the 7-digit SBC serial number which is stored as 3 bytes of BCD (leading '1' in board serial number ignored) in these NVRAM locations:

- NVRAM address 0xFFE9_E778: single processor, CPU-X
- NVRAM address 0xFFE9_E774: dual processor, CPU-Y

Synergy's 3-byte manufacturer's ID is combined with the board serial number to produce the Ethernet ID of the board's Ethernet interface. For example, for a board serial number of '1123456', the Ethernet ID is "00:80:F6:12:34:56".

For more information on the VGM5 non-volatile SRAM, refer to the ***Non-volatile 128K x 8 SRAM*** chapter in Section 4, page 155.

### Avoiding bus contention — CSMA/CD

To avoid contention from two or more stations trying to talk at the same time on the network, Ethernet uses a media access method called CSMA/CD (Carrier Sense Multiple Access with Collision Detection). With CSMA/CD, a station transmits a frame only when the network is not busy. If a collision does occur after a transmission, the station resolves it by retransmitting the frame.

❑ To avoid contention, stations monitor a carrier signal (an encoded clock signal integrated with the data) that indicates whether or not another station is transmitting. If a station has data of its own to transmit and the network is not busy, it is sent immediately. Otherwise, if the network is busy, the station waits until it senses no activity plus an extra delay time padding for channel recovery before transmitting its own data.

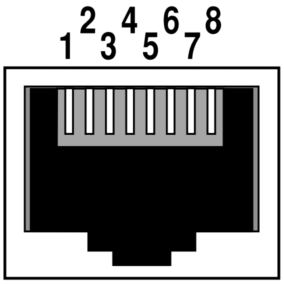
❏ When a collision does occur, all stations are notified of the occurrence by a signal applied to their Collision Detect input. Any station that is currently transmitting must stop and wait a certain amount of time before retransmitting the frame. The station's location on the network is factored into the time delay to ensure that no overlap occurs with other stations that may also be retransmitting their data. A packet less than the minimum size (512 bits) is considered a collision remnant and is ignored by the receiving station.

## Interchange signals

Ethernet uses differential driver circuits for its interchange signals. For the onboard 10Base/100Base-TX interface, the transmit data and receive data signals are transformer coupled internally on the SBC and routed to the VGM5 front panel RJ-45 jack. The table below lists the interchange signals and their pin assignments on the RJ-45 jack.

*Ethernet interchange signals, RJ-45 pin assignments*

| RJ-45 Pin | IEEE 802.3 Name | Function | Signal from: |
|---|---|---|---|
| 1<br>2 | DO+ (Data Out +)<br>DO– (Data Out –) | Transmit Pair | VGM5 |
| 3<br>6 | DI+ (Data In +)<br>DI– (Data In –) | Receive Pair | External Device |



*Ethernet 10Base/100Base-TX connector pin numbering*

## LED indicators

Rev. D and lower boards: Two onboard LED indicators provide port status indication.

- **Link OK** — lights up green when cable is connected to a functioning 10Base-T/100Base-TX network.
- **Link Activity** — flickers yellow whenever data is being received or transmitted. If the VGM5 is connected to a repeater-type hub instead of a switch-type hub, this LED may still flicker even when the VGM5 is not transferring data since packets sent over the network to other nodes will also be sent to the VGM5.

Rev. E and higher boards: four onboard LED indicators provide port status indication.

- **Link OK** — lights up green when cable is connected to a functioning 10Base-T/100Base-TX network.
- **Receive Data** — flickers yellow whenever data is being received. If the VGM5 is connected to a repeater-type hub instead of a switch-type hub, this LED may still flicker even when the VGM5 is not the intended destination since packets sent over the network to other nodes will also be sent to the VGM5.
- **Speed** — lights up green if cable is connected to a 100Base-TX hub. LED is OFF if cable is connected to a 10Base-T hub.
- **Collision** — lights up red whenever the VGM5 produces a collision (VGM5 transmits at same time as packet data is being received).

Refer to the LED discussion in Section 2 (page 20) for more information.

# 4 Programming the PowerPC

This section provides programming and operation information for the PowerPC processor and for other devices under its direct control.

- PowerPC architecture
- MPC106 PCI bridge/memory controller
- Programming notes, MPC106
- Address map
- Onboard registers
- Backside L2 cache controller
- Mailboxes
- Asynchronous serial interface
- Clock calendar
- Non-volatile 128K x 8 SRAM
- Boot Flash ROM/EPROM
- User Flash memory

Note: The bit numbering of registers in this section follows the zero-on-the-right convention as opposed to the zero-on-the-left bit numbering convention used by Motorola and IBM in their PowerPC documentation.

# PowerPC architecture

## Introduction

The PowerPC processor is a RISC (Reduced Instruction Set Computer) design of which development can be traced to IBM's introduction of the POWER (Performance Optimization With Enhanced RISC) architecture of the RISC System/6000 in early 1990. The multi-chip approach used by the microprocessor in this system led to discussions among IBM, Apple, and Motorola (now Freescale Semiconductor) to collaborate on the design and production of a more economical single chip solution. Thus was born the PowerPC ("PC" stands for Performance Computing) family of RISC processors starting with the 601 chip. The PowerPC architecture is scalable so that it can take advantage of new technological breakthroughs.

The PowerPC architecture defines the following features:

- Separate registers for integer and floating point operations. Integer data uses the general purpose registers (GPR) while floating point data uses the floating point registers (FPR).
- Instructions for moving integer and floating point data between the registers (GPR and FPR) and memory.
- Multiple execution units for parallel processing.
- Uniform length instructions for easy instruction pipelining and parallel processing.
- Liberal use of registers (up to four) during arithmetic operations.
- An exception handling mechanism.
- IEEE-754 floating point support.
- Single and double precision floating point operations.

- Separate L1 instruction and data caches.
- Instructions for controlling L1 data cache coherency at the user level.
- Support for both big and little endian addressing.

### Further information

For further PowerPC processor information, refer to the following Freescale Semiconductor/IBM documentation:

- The appropriate processor model User's Manual
- PowerPC Microprocessor Family: The Programmer's Reference Guide (Document No. MPCPRG/D)

For these and other literature, contact:

LDC for Freescale Semiconductor
19521 East 32nd Parkway
Aurora, CO 80011-8141
USA
**PHONE:** (800) 441-2447 or (303) 675-2140
**FAX:** (303) 675-2150
**EMAIL:** LDCForFreescaleSemiconductor@hibbertgroup.com

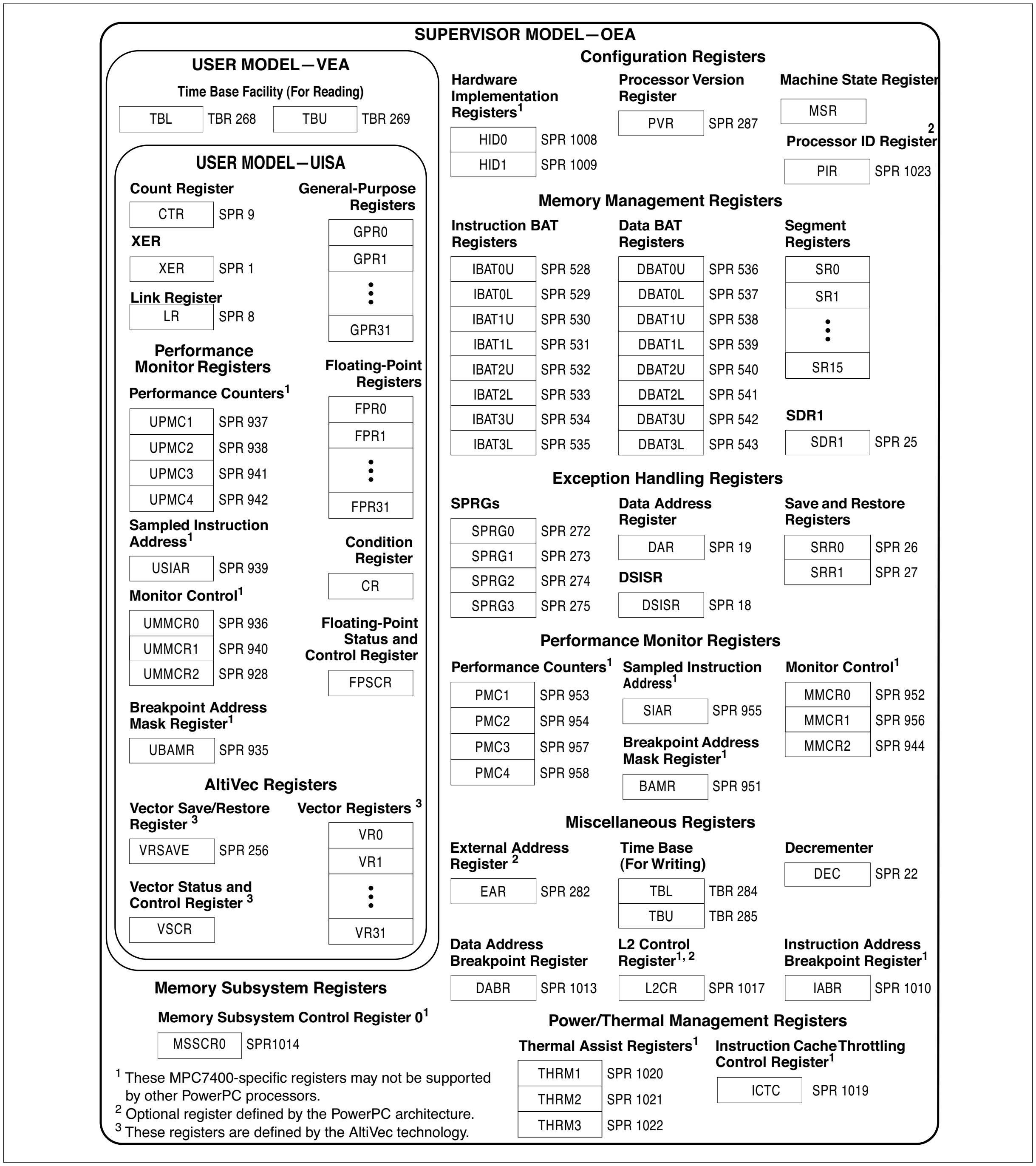
PowerPC literature in PDF form is also available at Freescale Semiconductor's website:

**http://www.freescale.com/**

## Architecture models

There are three models within the PowerPC architecture:

1. User model containing the user instruction set architecture (UISA) registers. This model contains the GPR, FPR, condition, floating point status and control, XER, link, and count registers. These register are accessible by all software all the time.
2. User model containing the virtual environment architecture (VEA) registers. This model includes the UISA model and time base facility registers. The time base facility registers are read only in this model.
3. Supervisor model containing the operating environment architecture registers. This model includes all the registers. In this model the time base facility registers are read/write.



[1] These MPC7400-specific registers may not be supported by other PowerPC processors.

[2] Optional register defined by the PowerPC architecture.

[3] These registers are defined by the AltiVec technology.

*PowerPC programming model (7400)*

## Register set

The PowerPC architecture defines register-to-register operations for most computational instructions. For example, there are no instructions that modify storage directly. For a storage operand to be used in a computation that modifies the same or another location, the content of storage must be loaded into a register, modified, and then stored back to the target location.

The PowerPC programming model includes 32 general purpose registers (GPRs), 32 floating point registers (FPRs), special purpose registers (SPRs), and several miscellaneous registers. A PowerPC processor also includes several processor-specific registers that are excluded from the PowerPC programming model. These registers provide functions unique to the processor and thus may not be supported by other PowerPC processors.

The following paragraphs give a brief description of the PowerPC register set. For more detailed register set information on a particular processor, refer to that processor's user's manual.

General Purpose Registers (GPRs) — 32 user-level, general purpose registers are defined in the PowerPC architecture. These registers are either 32- or 64-bits wide in 32- and 64-bit wide PowerPC processors. GPRs serve as the data source or destination for all integer instructions.

Floating Point Registers (FPRs) — 32 user-level, 64-bit wide floating point registers are defined in the PowerPC architecture. These registers serve as the data source or destination for floating-point instructions. FPRs can contain either single- or double-precision floating-point data.

Condition Register (CR) — The CR is a 32-bit user-level register that is used to show the results of certain operations such as move, integer and floating-point compare, and provide a mechanism for testing and branching.

Floating-point Status and Control Register (FPSCR) — The FPSCR provides compliance to the IEEE 754 standard by containing all exception signal bits, exception summary bits, exception enable bits, and rounding control bits.

Machine State Register (MSR) — The MSR is a supervisor level register that reflects the state of the processor. The contents of this register is saved when an exception is taken and restored when the exception handling routine completes. This is a 32- or 64-bit register depending on the processor.

Segment Registers (SRs) — SRs (16 ea. for 32-bit processors) are provided for memory management.

Special Purpose Registers (SPRs) — Special purpose registers serve a variety of functions. Some of these functions include control, status indication, processor configuration, and other special operations. The SPRs in PowerPC processors are 32-bits wide. A program accesses SPRs according to its privilege level (user or supervisor).

## Instruction set overview

All PowerPC instructions are encoded as single-word (32-bit) opcodes. The PowerPC instructions are divided into the following categories:

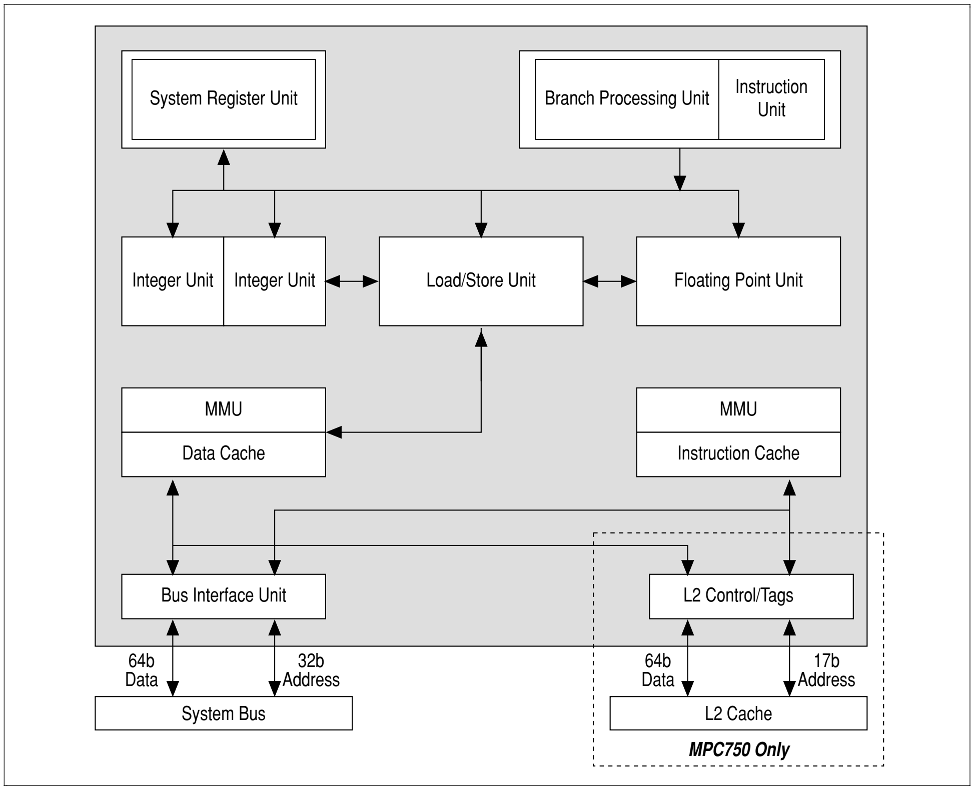
- Integer instructions — These include computational and logical instructions.
  - Integer arithmetic instructions
  - Integer compare instructions
  - Integer logical instructions
  - Integer rotate and shift instructions
- Floating-point instructions
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load/store instructions
  - Integer load and store instructions
  - Integer load and store multiple instructions
  - Floating point load and store
  - Integer load and store with byte reversal instructions
- Flow control instructions
  - Branch and trap instructions
  - Condition register logical instructions
- Processor control instructions
  - Move to/from SPR instructions
  - Move to/from MSR
  - Synchronize
  - Instruction synchronize
  - Order loads and saves.

- Memory control instructions
  - Supervisor-level cache management instructions
  - User-level cache instructions
  - Segment register manipulation instructions
  - Translation lookaside buffer management instructions

Detailed information on the PowerPC architecture can be found in the "PowerPC™ Microprocessor Family: The Programming Environments" manual available from IBM or Motorola.

## PowerPC 750 G3 processor

The PowerPC 750 is a third generation PowerPC processor often referred to as the 'G3' processor. This processor is a low-power 32-bit implementation of the PowerPC RISC architecture. A functional block diagram of the 750 is shown below.



**Block diagram, PowerPC 750**

The 740 processor is also a G3 processor though it is not used in the VGM5. The only difference between the 750 and 740 is that the 750 includes a performance-boosting 1 MB backside L2 cache interface. The following is a brief description of the 750.

### Superscalar microprocessor

The 750 processor is superscalar, capable of issuing three instructions per clock cycle into six independent execution units:

- Two integer units
- Floating-point unit
- Branch processing unit
- Load/store unit
- System register unit

Additionally, the 750 has the ability to fetch four instructions per cycle from the instruction cache.

### Power management

To conserve power, the 750 features low-power 2.6V or 1.9V core voltage design with three user programmable power-saving modes: doze, nap, and sleep. To further manage power efficiently, these processors use:

- Dynamic power management — selectively activates areas of the processor only as required by the executing code.
- Thermal management — controls heat dissipation without the need for conventional power/space-robbing cooling solutions.

### Cache and MMU support

The 750 is provided with on-chip level-one cache consisting of separate data and instruction caches of 32 KBytes each. The cache is 8-way set-associative, which means that there are 8 chances for data to be stored in each of 128 sets, which are selected by address bits 20-26. (For more information, read the PowerPC 750 User Manual, Chapter 3.)

The 750 is provided with an backside L2 cache interface. This is an onboard L2 cache controller with a dedicated L2 cache bus and on-chip L2 tags. This scheme bypasses the limitations normally imposed on transactions between the processor and Level 2 cache. The backside L2 cache has these key features:

- Operates at or near the CPU core frequency.

- Uses its own bus which avoids the effects of slowdowns and stalls typically encountered in system bus traffic.

The backside cache provides a measureable performance gain which is beneficial to resource intensive tasks such as imaging and real-time (data intensive) mathematical computations.

The 750 also has separate MMUs for instruction and data. The MMUs support 4 Petabytes ($2^{52}$) of virtual memory and 4 Gigabytes ($2^{32}$) of physical memory.

### Bus interface

The 750 has a 64-bit data bus and a 32-bit address bus. Pipelined, split, and burst transactions are supported. A snooping mechanism is provided for data cache coherency.

### MPC750-specific registers

The PowerPC architecture allows implementation-specific SPRs for various model processors. The SPRs specific to the VGM5's 750 processor are summarized below (all are supervisor level registers):

Instruction address breakpoint register (IABR) — can be used to cause a breakpoint exception if a specified instruction address is encountered.

Hardware implementation-dependent register 0 (HID0) — controls various functions such as enabling checkstop conditions, and locking, enabling and invalidating the instruction and data caches.

Hardware implementation-dependent register 1 (HID1) — reflects state of PLL_CFG[0-3] clock signals.

L2 cache control register (L2CR) — configures and operates the L2 cache. This register includes bits for parity checking enable, setting the L2-to-processor clock ratio, and identifying the type RAM used for the L2 cache (not supported by 740 processor).

Performance monitor registers — several registers are provided to define and count events for use by the performance monitor.

Instruction cache throttling control register (IATC) — enables the instruction cache throttling feature and controls the interval at which instructions are forwarded to the instruction buffer in the fetch unit.

Thermal management registers (THRM1, THRM2, THRM3) — enables and sets thresholds for the processor's thermal management facility.

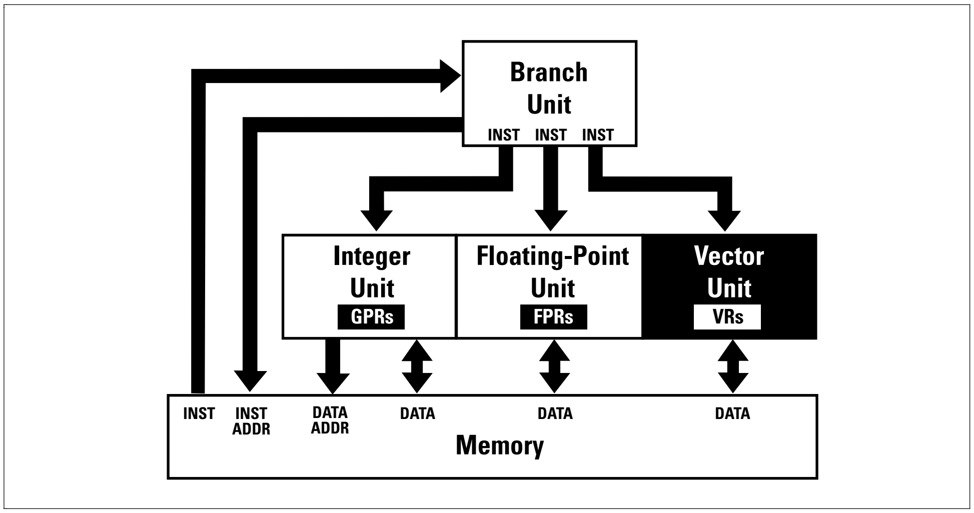
## PowerPC G4 processor

The PowerPC G4 (74xx) is a 4th generation PowerPC processor. This processor is similar to the PowerPC G3 (750) with exception of the G4's 128-bit vector unit which operates concurrently with the 32-bit integer and floating point units. The addition of the vector execution unit is the basis for Motorola's AltiVec™ technology.

With its AltiVec™ technology, the G4 provides for highly parallel operations with the ability to execute up to 16 operations in a single clock cycle. The G4 performs a type of parallel processing called SIMD (single instruction, multiple data) which speeds high bandwidth applications such as 3-D imaging/video processing, scientific array processing, speech processing, etc.

The vector unit has 32 128-bit registers. Depending on data size, each register can hold sixteen 8-bit elements, eight 16-bit elements, or four 32-bit elements. The vector unit's ALU can operate on three source vectors and produce a single result vector on each instruction (there are 162 AltiVec™ specific instructions). Hence, the smaller the data size, the more data that can be processed in a single clock cycle.

Below is a block diagram of PowerPC with AltiVec™ technology.



*Overview, PowerPC with AltiVec™ technology*

Though there are numerous differences between G3 and G4 CPUs (G4, for example, supports a 2 MB L2), the G4 is still compatible with the industry standard PowerPC architecture. For more detailed information on the PowerPC and AltiVec™, refer to Motorola's website:

**http://www.mot.com/SPS/PowerPC/**

### Summary of differences, 750 vs. 7400 Processor

- Completion queue: 750 has 6-entry completion queue vs. 7400's 8-entry completion queue. G4's extra completion queue entries reduce the opportunity for bottlenecks from the G4's additional execution units.
- Floating point: 750 has 4-cycle latency for double-precision floating-point multiply and 3-cycle latency for all other floating point add and multiply. 7400 has 3-cycle latency for all floating-point add and multiply. As a result, the 7400 has equal latency for double-precision and single-precision operations.
- AltiVec™: 7400 has special vector execution units to implement the AltiVec™ instruction set which speeds high bandwidth applications via parallel processing (SIMD).
- Memory subsystem: 7400 improves data flow with increased queue sizes and queue additions.
- L1 cache block allocation policy: 750 has allocate-on-miss policy. 7400 has allocate-on-reload policy. 7400's block allocation occurs in parallel with reload which which uses the cache more efficiently (5 cycles for 7400 vs. 6 cycles for 750).
- L2 cache: 750 supports 1 MB max. L2 cache. 7400 supports 2 MB max. L2 cache. 7400 has fewer sectors per tag than the 750 which allows for more efficient caching. Moreover, the L2 cache reload policy was changed in the 7400 to improve performance.
- Processor bus: In addition to the 60x bus, the 7400 supports a higher performance processor bus called the MPX bus (not supported by VGM5).

### 7410 G4 Processor

The VGM5 provides as an option the 7410 — a 2nd generation G4 PowerPC processor. In comparison to the 7400, the 7410 has lower power consumption and higher processor speeds. Other major differences between the two G4 processors:

- Private memory — can use L2 SRAM as direct-mapped private memory.
- L2 data bus width — can use either a 32- or 64-bit L2 data bus.
- Processor version register — PVR data for 7410: **0x800C_1xxx**. PVR data for 7400: **0x000C_0xxx**.

# MPC106 PCI bridge/memory controller

## General description

The MPC106 PCI bridge/memory controller (aka Grackle) is an integrated high-bandwidth, high-performance interface for:

- PowerPC processor(s)
- Secondary (L2) cache
- Memory (EDO DRAM/SDRAM/ROM)
- PCI bus

In addition to interface support, the MPC106 provides hardware support for power management functions via register programming.

In Sept. 2003, Tundra Semiconductor purchased the MPC106 (and MPC107) product line from Motorola. These chips are now known as Tsi106 and Tsi107 respectively. Though your SBC may be installed with a Motorola part, any technical info on the MPC106 is now obtained through Tundra under the Tsi106 name. Refer to the Tundra web site for info: **www.tundra.com**

## PowerPC processor interface

The MPC106 provides an interface to a variety of PowerPC processors (up to 4) using a 32-bit address bus and a 64-bit data bus. The address and data bus are decoupled for pipelining of 60x accesses. The MPC106 processor interface supports full memory coherency and an optional local bus slave.

## Secondary (L2) cache interface

The MPC106 supports various combinations of L2 cache/60x processors. For the VGM5, however, individual backside L2 cache is provided for each processor, so the MPC106's cache controller is not used. This frees the MPC106 to support full arbitration and interface functions for multiprocessor operation.

## Memory interface

The MPC106 memory interface controls processor accesses to/from main memory using a 64-bit data path. The memory configuration/size and error checking scheme (normal parity, RMW parity, ECC) is programmable.

ROM/Flash interfacing is also provided by the MPC106.

PCI access to/from main memory is provided by the PCI bus interface (see next).

## PCI bus interface

The MPC106 provides the PCI interface which connects to the processor and memory buses. The MPC106 PCI interface is compliant with PCI Local Bus Specification, Revision 2.1. The PCI bus is 32-bits wide and runs at 33 MHz. Refer to the PCI bus description in Section 3 for more information.

As a PCI interface, the MPC106 functions as both a master and target device. As a PCI bus master, the MPC106 configures all PCI devices using PCI configuration cycles in addition to supporting read/write operations to PCI memory space and PCI I/O space. As a PCI target, the MPC106 supports read/write operations to system memory.

### Power management functions

The table below lists the power management functions supported by the MPC106.

**MPC106 power management functions**

| Mode | Description |
|---|---|
| Full-On | This is the normal operating mode. |
| Doze | All functions disabled except; PCI address decoding, RAM refresh, CPU bus requests, and NMI monitoring. The CPUs can continue to operate normally. |
| Nap | All functions disabled except; PCI address decoding, RAM refresh, CPU bus requests, and NMI monitoring. The CPUs are also in Nap mode. (603s will not snoop, but 604s will. Flush 603 L1 caches before entering this mode.) |
| Sleep | All functions disabled except RAM refresh, CPU bus requests, and NMI monitoring. CPUs are also in sleep (or nap) mode. |
| Suspend | All functions disabled except RAM refresh. (This mode not supported by VGM Series boards.) |

## Programming the MPC106

The Grackle must be programmed in order to:

- Access RAM
- Access PCI
- Write to the serial ports
- Write to the LEDs
- Write to the board configurations registers
- Enable CPU-Y
- Enable and diagnose certain memory and PCI error conditions

# MPC106 registers

The following is an overview of the MPC106 registers. For in-depth register programming information, refer to the MPC106 User's Manual by Motorola.

## Address maps

The MPC106 supports three address mapping configurations designated address map A, address map B, and emulation mode address map. Address map A conforms to the now obsolete PowerPC Reference Platform Specification (PREP). Address map B conforms to the Common Hardware Reference Platform Architecture (CHRP). The emulation mode address map, which is not used for VGM Series boards, supports software emulation of x86 hardware. On reset, onboard hardware selects address map B by default. After reset, the address map can be changed by programming bit 16 in the MPC106's Processor Interface Configuration register 1 (PICR1). VGM Series boards default to address map B for all models. Refer to the next chapter (***Address map***) for more information on the address map structure.

## Configuration registers

Using CHRP address map B, the base address of the Grackle chip is:

- 0xFEC0_0000, Address Register
- 0xFEE0_0000, Data Register

The OS initialization software sets up the MPC106 in the PCI configuration space header. The PCI configuration space header format is shown below. Note that Grackle operates in PCI memory space only (write '2' [0x0010] to command register to enable PCI memory access).

Access Grackle's configuration registers 0x00–0xFC by writing 0x8000_00XX to 0xFEC0_0000 then reading or writing **32 bits only** to/from address 0xFEE0_0000. In both cases the data must be byte-reversed as the PCI bus uses little-endian bit format whereas the PowerPC bus uses big-endian bit format.

| 31 | 16 | 15 | 0 | |
|---|---|---|---|---|
| Device ID<br>0x0002 | | Vendor ID<br>0x1057 | | 0x00 |
| Status | | Command 0x0000* | | 0x04 |
| Class Code | Subclass Code | Standard Programming | Revision ID | 0x08 |
| BIST Control | Header Type | Latency Timer | Cache Line Size | 0x0C |
| MAX GNT | MIN GNT | Interrupt Pin | Interrupt Line | 0x3C |
| — | Disconnect Cntr | Subordinate Bus No. | Bus Number | 0x40 |
| — | | Special Cycle Address | | 0x44 |
| — | PMCR2 | Power Management Configuration | | 0x70 |
| Memory Starting Address | | | | 0x80 |
| Memory Starting Address | | | | 0x84 |
| Extended Memory Starting Address | | | | 0x88 |
| Extended Memory Starting Address | | | | 0x8C |
| Memory Ending Address | | | | 0x90 |
| Memory Ending Address | | | | 0x94 |
| Extended Memory Ending Address | | | | 0x98 |
| Extended Memory Ending Address | | | | 0x9C |
| Pg. Mode Cntr/Timer | //// //// | | Memory Enable | 0xA0 |
| //////// | | | | 0xA4 |
| Processor Interface Configuration 1 | | | | 0xA8 |
| Processor Interface Configuration 2 | | | | 0xAC |
| Alternate OS Visible Params 2 | Alternate OS Visible Params 1 | ECC Single Bit Trigger | ECC Single Bit Counter | 0xB8 |
| 60x Bus Error Status | //////// | Error Detection 1 | Error Enabling 1 | 0xC0 |
| PCI Bus Error Status | //////// | Error Detection 2 | Error Enabling 2 | 0xC4 |
| 60x/PCI Error Address | | | | 0xC8 |
| Emulation Support Configuration 1 | | | | 0xE0 |
| Modified Memory Status (No Clear) | | | | 0xE4 |
| Emulation Support Configuration 2 | | | | 0xE8 |
| Modified Memory Status (Clear) | | | | 0xEC |
| Memory Control Configuration 1 | | | | 0xF0 |
| Memory Control Configuration 2 | | | | 0xF4 |
| Memory Control Configuration 3 | | | | 0xF8 |
| Memory Control Configuration 4 | | | | 0xFC |

*Note: Bit 1 of the PCI command register, when set to a 1, enables MPC106 to respond to accesses to the PCI Memory Address space.

## Power management configuration registers

Power management functions of the MPC106 are controlled by the power management configuration registers (PMCRs).

| Register Name | Size | Address Offset |
|---|---|---|
| PMCR1 | half-word (16 bits) | 0x70 |
| PMCR2 | byte | 0x72 |

## Error handling registers

The error handling registers control the MPC106's error handling and reporting.

| Register Name | Size | Address Offset |
|---|---|---|
| ECC single-bit error counter register | byte | 0xB8 |
| ECC single-bit error trigger register | byte | 0xB9 |
| ErrEnR1 | byte | 0xC0 |
| ErrDR1 | byte | 0xC1 |
| 60x Bus error status register | byte | 0xC3 |
| ErrEnR2 | byte | 0xC4 |
| ErrDR2 | byte | 0xC5 |
| PCI bus error status register | byte | 0xC7 |
| 60x/PCI error address register | byte | 0xC8 |

## Memory interface registers

Memory boundaries (starting and ending addresses), memory bank enables, memory timing, and external memory buffers are all controlled by the memory interface configuration registers (MICRs).

| Register Name | Size | Address Offset |
|---|---|---|
| Memory starting address register 1 | word | 0x80 |
| Memory starting address register 2 | word | 0x84 |
| Ext. memory starting address register 1 | word | 0x88 |
| Ext. memory starting address register 2 | word | 0x8C |
| Memory ending address register 1 | word | 0x90 |
| Memory ending address register 2 | word | 0x94 |
| Ext. memory ending address register 1 | word | 0x98 |
| Ext. memory ending address register 2 | word | 0x9C |
| Memory bank enable register | byte | 0xA0 |
| Memory page mode register | byte | 0xA3 |
| MCCR1 | word | 0xF0 |
| MCCR2 | word | 0xF4 |
| MCCR3 | word | 0xF8 |
| MCCR4 | word | 0xFC |

### Processor interface configuration registers

Programmable parameters of the PowerPC bus and L2 cache interface are controlled by the processor interface configuration registers (PICRs).

| Register Name | Size | Address Offset |
|---|---|---|
| PICR1 | longword | 0xA8 |
| PICR2 | longword | 0xAC |

### Alternate OS-Visible parameters registers

Operating systems have an alternate means to access some of the bits of the PICR1 using the alternate OS-visible parameters registers.

| Register Name | Size | Address Offset |
|---|---|---|
| Alternate OS-visble parameters reg. 1 | byte | 0xBA |
| Alternate OS-visble parameters reg. 2 | byte | 0xBB |

### Emulation support configuration registers

The emulation support configuration registers controls MPC106 operation in emulation mode. (Not used for VGM Series boards.)

### External configuration registers

The external configuration registers allow access to certain configuration bits when using Address map A (PREP).

| Register Name | Size | Address Offset |
|---|---|---|
| External configuration register 1 | byte | Port 0x092 |
| External configuration register 2 | byte | Port 0x81C |
| External configuration register 3 | byte | Port 0x850 |

# Programming notes, MPC106

## Setting PCI device base address

Each PCI device has a standard set of configuration registers, acessed with PCI configuration cycles using the Grackle's CFG_ADDR @ 0xFEC0_0000 and CFG_DATA @ 0xFEE0_0000 registers. Refer to the 'type 0' and 'type 1' configuration register tables from the PCI spec. The standard VGM or VSS Series configuration addresses for PCI devices are listed in the VxWorks **svgm1.h** header file.

Most OS's contain PCI configuration access routines. VxWorks, for example, has `readMPC` and `writeMPC`:

```
   readMPC(0x80006800);
```

returns a 32 bit value which is the VendorID and DeviceID registers of the first PMC daughterboard, or 0xFFFF_FFFF if none exists. And...

```
   writeMPC(0x80006810, 0xe00000000);
```

...writes the address 0xE000_0000 to a 'type 0' device's configuration register 0x10 (BAR0).

Each device defines some of its Base Address Registers to be the address of a particular bank within it, usually additional registers or dual-ported memory. Bit 0 of the register specifies which PCI space it is to be placed in: 0 for PCI Memory and 1 for PCI I/O space. By first writing an 0xFFFF_FFFF to the register and reading it back, it is possible to tell which bits are writable — those that aren't won't change. This tells the size of the bank and whether it is fixed in a particular space.

On the PowerPC Series SBC, the allowed ranges for the PCI spaces are:

Memory base = 0xDFFF_0000 – 0xEFFF_FFFF

I/O base = 0xFEB0_0000 - 0xFEBF_FFFF

You also need to enable the PCI device's response to PCI Memory space and/or PCI I/O space by setting the corresponding bits in the device's Command register; typically, bits 1 and 0 respectively.

The MPIC interrupt vector assigned to PCI Interrupt A is 13, and for B it is 12.

## Write posting to ROM Space

### Problem

If you have a program that writes very frequently to that address region which the Grackle considers to be "ROM space", the Grackle services those writes at the expense of other requests for memory access that might happen at the same time. The ROM space includes everything at high address range (any address that is 0xFFxx_xxxx). This includes the ROMs which are meaningless to write to, but it also includes all the CPU control registers, the LEDs being among them. So it is normal for the CPU to make writes to this region even though it's considered ROM space by the Grackle. This poses a problem as the Grackle gives priority to write-posts writes to ROM space at the expense of other processes waiting for Grackle's services.

### Observation

A program had a tight loop which was reading 8 bytes of data from a file and then writing a number to the LEDs which caused 8 write operations to ROM space. At the same time that was going on there was lots of contention for the Grackle's services: another CPU card tried to make many accesses to the CPUs RAM through the VME interface and onboard DMA transfers were occurring from the Ethernet interface. The Grackle in effect gave priority to the writes to ROM space (which it performed rather slowly, too) and serviced the DMA and VMEbus requests in whatever time was left over. Occasionally, the Grackle would make the Universe II chip wait longer than 16µs for the memory access it requested. Since the VME timeout is set to the standard value of 16µs, this longer than 16µs wait caused the VMEbus system controller to generate a bus timeout — a bus error.

### Solution

The problem was solved by altering the LED writing routine. After each write to an LED, a read of that LED is performed. The LED read data itself is ignored. The act of reading the LED register is what causes the Grackle to resume paying attention to other pending Grackle accesses.

Bottom line: If frequent writes to ROM space are needed, intersperse ROM space writes with ROM space reads whether or not you need the data being read. It makes the Grackle give fairer access to the onboard memory to the various competing sources of memory access requests.

# Address map

This chapter provides address map information:

- Overall CHRP address map, processor view and PCI master (memory and I/O) view.
- VGM5 address map as viewed by the PowerPC processor(s)
- VGM5 address map as viewed by PCI devices

Additional information on the board's memory spaces follows the board address map listing.

## CHRP address map

By default, the VGM5 uses a Common Hardware Reference Platform (CHRP) compliant address map designated Address Map B. Alternatively, the board can use Address Map A (PREP) by programming the MPC106 (see page 95).

The tables below list Address Map B as viewed by the processor and as viewed by PCI memory and I/O Masters. The figure following the tables shows a graphical view of Address Map B. These tables are listed in Motorola's MPC106 User's Manual with additional notes. Refer to the MPC106 user's manual for more information about Address Map B and Address Map A (PREP).

### *Address map B — Processor view*

| PowerPC Processor Address Range | | | | PCI | |
|---|---|---|---|---|---|
| Hex | | Decimal | | Address Range | Definition |
| 0x0000_0000 | 0x0009_FFFF | 0 | 640K–1 | No PCI cycle | System memory space |
| 0x000A_0000 | 0x000B_FFFF | 640K | 768K–1 | 0x000A_0000–0x000B_FFFF | Compatibility hole[4] |
| 0x000C_0000 | 0x3FFF_FFFF | 768K | 1G– | No PCI cycle | System memory space |
| 0x4000_0000 | 0x7FFF_FFFF | 1G | 2G–1 | No PCI cycle | Reserved5 |
| 0x8000_0000 | 0xFCFF_FFFF | 2G | 4G–48M–1 | 0x8000_000–0xFCFF_FFFF | PCI memory space |
| 0xFD00_0000 | 0xFDFF_FFFF | 4G–48M | 4G–32M–1 | 0x0000_0000–0x00FF_FFFF | PCI/ISA memory space[3] |
| 0xFE00_0000 | 0xFE7F_FFFF | 4G–32M | 4G–24M–1 | 0x0000_0000–0x0000_FFFF | PCI/ISA I/O space (64Kbytes or 8 Mbytes) |
| 0xFE80_0000 | 0xFEBF_FFFF | 4G–24M | 4G–20M–1 | 0x0080_0000–0x00BF_FFFF | PCI I/O space |
| 0xFEC0_0000 | 0xFEDF_FFFF | 4G–20M | 4G–18M–1 | CONFIG_ADDR | PCI configuration address register[1] |
| 0xFEE0_0000 | 0xFEEF_FFFF | 4G–18M | 4G–17M–1 | CONFIG_DATA | PCI configuration data register[1] |
| 0xFEF0_0000 | 0xFEFF_FFFF | 4G–17M | 4G–16M–1 | 0xFEFF_0000–0xFEFF_FFFF | PCI interrupt acknowledge |
| 0xFF00_0000 | 0xFF7F_FFFF | 4G–16M | 4G–8M–1 | 0xFF00_0000–0xFF7F_FFFF | 64-bit system ROM space[2] |
| 0xFF80_0000 | 0xFFFF_FFFF | 4G–8M | 4G–1 | 0xFF80_0000–0xFFFF_FFFF | 8- or 64-bit system ROM space[3] |

### *Address map B — PCI memory master view*

| PCI Memory Transaction Address Range | | | | PowerPC Processor | |
|---|---|---|---|---|---|
| Hex | | Decimal | | Address Range | Definition |
| 0x0000_0000 | 0x0009_FFFF | 0 | 640K–1 | 0x0000_0000–0x0009_FFFF | System memory space |
| 0x000A_0000 | 0x000F_FFFF | 640K | 1M–1 | 0x000A_0000–0x000F_FFFF | Compatibility hole[4] |
| 0x0010_0000 | 0x3FFF_FFFF | 1M | 1G–1 | 0x0010_0000–0x3FFF_FFFF | System memory space |
| 0x4000_0000 | 0x7FFF_FFFF | 1G | 2G–1 | 0x4000_0000–0x7FFF_FFFF | Reserved |
| 0x8000_0000 | 0xFCFF_FFFF | 2G | 4G–48M–1 | No system memory cycle | PCI memory space |
| 0xFD00_0000 | 0xFDFF_FFFF | 4G–48M | 4G–32M–1 | 0x0000_0000–0x00FF_FFFF | System memory space |
| 0xFE00_0000 | 0xFEFF_FFFF | 4G–32M | 4G–16M–1 | No system memory cycle | Reserved |
| 0xFF00_0000 | 0xFF7F_FFFF | 4G–16M | 4G–8M–1 | 0xFF00_0000–0xFF7F_FFFF | 64-bit system ROM space[2] |
| 0xFF80_0000 | 0xFFFF_FFFF | 4G–8M | 4G–1 | 0xFF80_0000–0xFFFF_FFFF | 8- or 64-bit system ROM space[3] |

Notes: 1. Used for PCI configuration cycles.

2. Maps to unused space in VGM5.

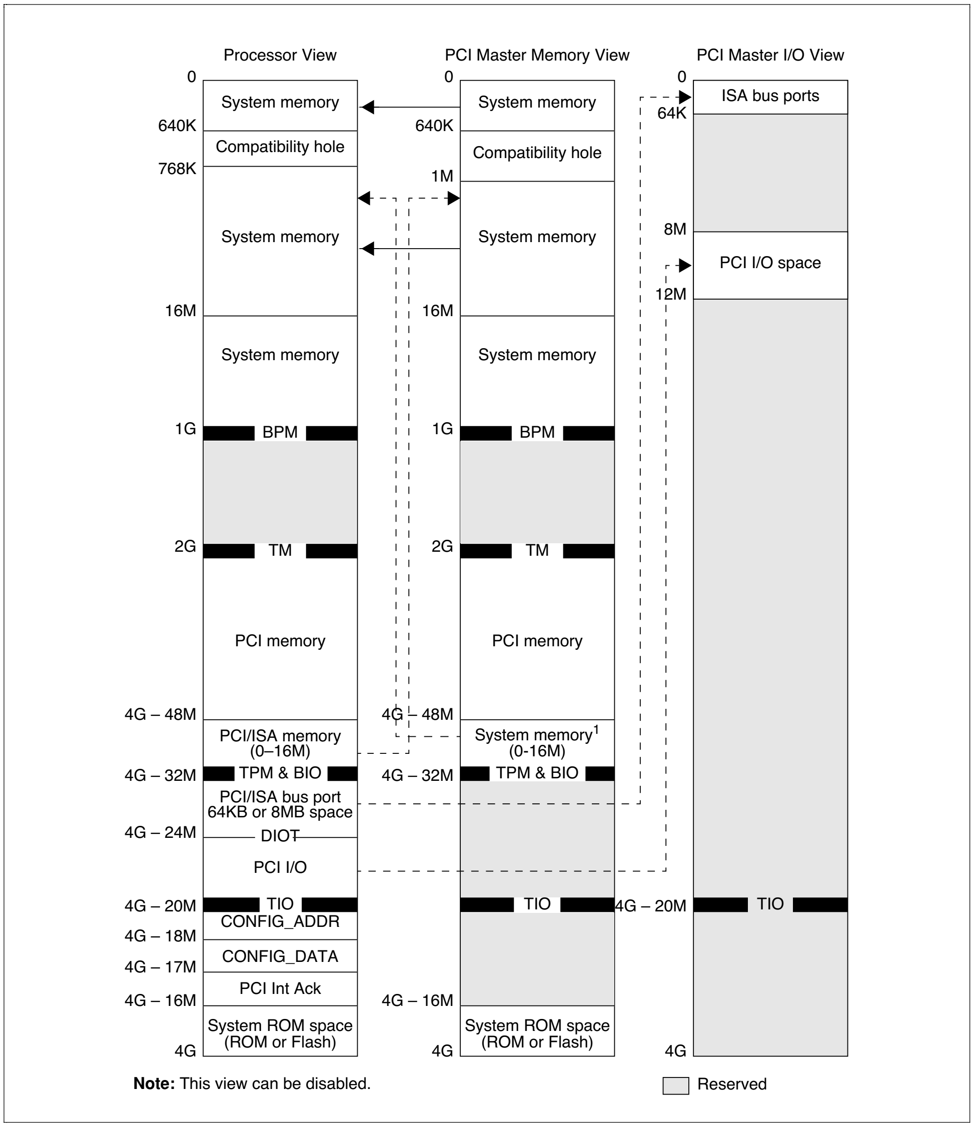
3. Maps to VGM5's onboard registers and all ROM/NVRAM.

4. Synergy does not use the compatibility hole which needs to be explicitly enabled to be present. For the VGM5, this space is part of the 1GB system memory (RAM) space allocation permitted by the CHRP address map.

5. Maps to direct VME Master space in VGM5.

## Address map B — PCI I/O master view

| PCI I/O Transaction Address Range | | | | PowerPC Processor | |
|---|---|---|---|---|---|
| Hex | | Decimal | | Address Range | Definition |
| 0x0000_0000 | 0x0000_FFFF | 0 | 64K–1 | No system memory cycle | PCI/ISA I/O space |
| 0x0001_0000 | 0x007F_FFFF | 64K | 8M–1 | No system memory cycle | Reserved |
| 0x0080_0000 | 0x00BF_FFFF | 8M | 12G–1 | No system memory cycle | PCI I/O space |
| 0x00C0_0000 | 0xFFFF_FFFF | 12M | 4G–1 | No system memory cycle | Reserved |



**Note:** This view can be disabled.

*Address map B*

## PCI configuration and address

To help in programming VGM5's PCI devices, the table below lists each PCI device with its Type 0 PCI configuration data and address.

*Type 0 configuration for devices on VGM5*

| ID Sel | Vendor | Device ID | Manufacturer | Part No./Description | PCI Config. Address |
|---|---|---|---|---|---|
| 0 | 0x1057 | 0x0002 | Motorola | MPC106/PCI bridge, mem. ctrlr. | 0x8000_0000 |
| 11 | 0x1014 | 0x0046 | IBM | MPIC/multiproc. int. controller | 0x8000_5800 |
| 12 | 0x1000 | 0x000D | Symbios | SYM53C885/SCSI controller | 0x8000_6000 |
| 12.1 | 0x1000 | 0x0701 | Symbios | SYM53C885/Ethernet controller | 0x8000_6100 |
| 13 | (Note 4) | (Note 4) | (Note 4) | PMC Slot1 | 0x8000_6800 |
| 14 | (Note 4) | (Note 4) | (Note 4) | PMC Slot2 [1] | 0x8000_7000 |
| 15 | 0x1011 | 0x0026 | DEC | 21154/PCI-PCI bridge [2] | 0x8000_7800 |
| 17 | 0x10E3 | 0x0000 | Tundra Semiconductor | Universe II/PCI–VME64 bridge | 0x8000_8800 |
| 18 | 0x1011 | 0x0046 | DEC | 21554/PCI-PCI bridge [3] | 0x8000_9000 |

Notes:
1. PMC Slot2 present on Model VGMD & VCMD SBCs only.
2. This PCI-PCI bridge (21154) on PEX3 only.
3. This PCI-PCI bridge (21554) on SBC models VGM5 and VSS4 only.
4. Vendor and Device IDs set by PMC manufacturer.

# VGM5 address map

The VGM5 address map as viewed by the PowerPC processor(s) and PCI devices is shown in the tables below.

### PowerPC Address Map

| Address | Device/address space description | Access |
|---|---|---|
| 0000_0000 - 00FF_FFFF | RAM (16 MB) | D8–D64 (RW) |
| 0000_0000 - 01FF_FFFF | RAM (32 MB) | D8–D64 (RW) |
| 0000_0000 - 03FF_FFFF | RAM (64 MB) | D8–D64 (RW) |
| 0000_0000 - 07FF_FFFF | RAM (128 MB) | D8–D64 (RW) |
| 0000_0000 - 0FFF_FFFF | RAM (256 MB) | D8–D64 (RW) |
| 0000_0000 - 1FFF_FFFF | RAM (512 MB) | D8–D64 (RW) |
| 0000_0000 - 3FFF_FFFF | RAM (1 GB) | D8–D64 (RW) |
| 0000_0080 | Mailbox 0 Write | D8 (WO) Note 2 |
| 0000_00A0 | Mailbox 1 Write | D8 (WO) Note 2 |
| 4000_0000 - 77FF_FFFF | Direct VME Access (A32 space, 896 MB) | D8, D16, D32 (RW) |
| 7800_0000 - 7AFF_FFFF | Reserved | — |
| 7B00_0000 - 7BFF_FFFF | Direct VME Access (A24 space, 16 MB) | D8, D16, D32 (RW) |
| 7C00_0000 - 7FFE_FFFF | Reserved | — |
| 7FFF_0000 - 7FFF_FFFF | Direct VME Access (A16 space, 64 KB) | D8, D16, D32 (RW) |
| 8000_0000 - FCFF_FFFF | PCI Memory Space (2GB) | D8, D16, D32 (RW) |
| FD00_0000 - FDFF_FFFF | Reserved | — |
| FE00_0000 - FE7F_FFFF | PCI I/O Space (8 MB), 0-based | D8, D32 (RW) |
| FE80_0000 - FEBF_FFFF | PCI I/O Space (4 MB), 0-based | D8, D16, D32 (RW) |
| FEC0_0000 - FEDF_FFFF | PCI Configuration Address Reg | D8, D16, D32 (RW) |
| FEE0_0000 - FEEF_FFFF | PCI Configuration Data Reg | D8, D16, D32 (RW) |
| FEF0_0000 - FEFF_FFFF | PCI Interrupt Acknowledge | D8, D16, D32 (RO) |
| FF00_0000 - FFDF_FFFF | Reserved (14 MB) | — |
| FFE0_0000 - FFE7_FFFF | Boot Flash ROM (lower 512 KB, ROMBoot) | D8 (RW), D64 (RO) Note 1 |
| FFE0_0000 - FFE7_FFFF | EPROM (512 KB), FlashBoot | D8 (RO), D64 (RO) Note 1 |
| FFE8_0000 - FFE9_FFEF | NVRAM (128 KB-16 bytes) | D8 (RW), D64 (RO) |
| FFE9_FFF0 - FFE9_FFFF | Real Time Clock/Calendar | D8 (RW), D64 (RO) |
| FFEA_0000 - FFEF_FAFF | Reserved | — |
| FFEF_FB00 - FFEF_FB07 | Serial Port B (8B) | D8 (RW) |
| FFEF_FB08 - FFEF_FB0F | Serial Port A (8B) | D8 (RW) |
| FFEF_FB10 - FFEF_FBFF | Reserved | — |
| FFEF_FC00 | Mailbox 0 Read | D8 (RO) |
| FFEF_FC08 | Mailbox 1 Read | D8 (RO) |
| FFEF_FD00 | 8-bit User Switch Register | D8 (RO) |
| FFEF_FE00 | Board Type and Revision Reg | D8 (RO) |
| FFEF_FE08 | Special Mod and ECO Level Reg | D8 (RO) |
| FFEF_FE10 | Feature Register | D8 (RO) |
| FFEF_FE18 | Board Status Register | D8 (RO) |
| FFEF_FE20 | CPU Status Register | D8 (RO) |
| FFEF_FE28 | CPU Timebase Register | D8 (RW) |
| FFEF_FE30 | L2 Cache Register | D8 (RO) |

### PowerPC Address Map (continued)

| Address | Device/address space description | Access |
|---|---|---|
| FFEF_FE28 | CPU Timebase Register | D8 (RW) |
| FFEF_FE30 | L2 Cache Register | D8 (RO) |
| FFEF_FE38 | Memory Register | D8 (RO) |
| FFEF_FE40 | Flash ROM Register | D8 (RW) |
| FFEF_FE48 | P0 PCI Slot Register | D8 (RO) |
| FFEF_FE50 | Flash Window Register | D8 (WO) |
| FFEF_FE68 | P0•PCI™ Interrupt Assert/Pending Register | D8 (RW) |
| FFEF_FE80 | User LED 0 Register | D8 (RW) |
| FFEF_FE88 | User LED 1 Register | D8 (RW) |
| FFEF_FE90 | User LED 2 Register | D8 (RW) |
| FFEF_FE98 | User LED 3 Register | D8 (RW) |
| FFEF_FEA0 | User LED 4 Register | D8 (RW) |
| FFEF_FEA8 | User LED 5 Register | D8 (RW) |
| FFEF_FEB0 | User LED 6 Register | D8 (RW) |
| FFEF_FEB8 | User LED 7 Register | D8 (RW) |
| FFEF_FF00 | PPC-VME64 Slave Mask Register | D8 (RW) |
| FFEF_FF08 | PPC-VME64 Slave Address Register | D8 (RW) |
| FFEF_FF10 | VME64 Broadcast Address Register | D8 (RW) |
| FFEF_FF18 | PPC-VME64 Master Address Register | D8 (RW) |
| FFEF_FF20 | PPC-VME64 Mode Register | D8 (RW) |
| FFEF_FF28 | PPC-VME64 RMW Register | D8 (RW) |
| FFEF_FF30 | VME64 Slot Register | D8 (RO) |
| FFEF_FF38 | VME64 SysReset Register | D8 (RW) |
| FFEF_FF40 - FFEF_FFFF | Reserved | — |
| FFF0_0000 - FFF7_FFFF | EPROM (512 KB), ROMBoot | D8 (RW), D64 (RO) Note 1 |
| FFF0_0000 - FFFF_FFFF | Boot Flash ROM (1 MB, FlashBoot) | D8 (RW), D64 (RO) Note 1 |
| FFF8_0000 - FFFF_FFFF | Boot Flash ROM (upper 512 KB, ROMBoot) | D8 (RW), D64 (RO) Note 1 |
| FFF8_0000 - FFFF_FFFF | User Flash Bank (512 KB) | D8 (RW), D64 (RO) Note 3 |

### PCI Memory Space Address Map

| Address | Device/address space description | Access |
|---|---|---|
| 0000_0000 - 00FF_FFFF | RAM (16 MB) | D8–D64 (RW) |
| 0000_0000 - 01FF_FFFF | RAM (32 MB) | D8–D64 (RW) |
| 0000_0000 - 03FF_FFFF | RAM (64 MB) | D8–D64 (RW) |
| 0000_0000 - 07FF_FFFF | RAM (128 MB) | D8–D64 (RW) |
| 0000_0000 - 0FFF_FFFF | RAM (256 MB) | D8–D64 (RW) |
| 0000_0000 - 1FFF_FFFF | RAM (512 MB) | D8–D64 (RW) |
| 0000_0000 - 3FFF_FFFF | RAM (1 GB) | D8–D64 (RW) |
| 0000_0080 | Mailbox 0 Write | D8 (WO) Note 2 |
| 0000_00A0 | Mailbox 1 Write | D8 (WO) Note 2 |
| 4000_0000 - 7FFF_FFFF | Reserved (1 GB) | — |
| 8000_0000 - FCFF_FFFF | PCI Memory Space (2 GB-48 MB) | D8, D16, D32 (RW) |
| FD00_0000 - FEFF_FFFF | Reserved | — |
| FF00_0000 - FFDF_FFFF | Reserved | — |
| FFE0_0000 - FFE7_FFFF | Boot Flash ROM (lower 512 KB), ROMBoot | D8 (RO), D64 (RO) Note 1 |
| FFE0_0000 - FFE7_FFFF | EPROM (512 KB), FlashBoot | D8 (RO), D64 (RO) Note 1 |
| FFE8_0000 - FFEF_FFFF | Reserved | — |
| FFF0_0000 - FFF7_FFFF | EPROM (512 KB), ROMBoot | D8 (RO), D64 (RO) Note 1 |
| FFF0_0000 - FFFF_FFFF | Boot Flash ROM (1 MB), FlashBoot | D8 (RO), D64 (RO) Note 1 |
| FFF8_0000 - FFFF_FFFF | Boot Flash ROM (upper 512 KB), ROMBoot | D8 (RO), D64 (RO) Note 1 |
| FFF8_0000 - FFFF_FFFF | User Flash Bank (512 KB) | D8 (RO), D64 (RO) Note 3 |

### Address Map Table Notes:

Note 1: The Boot Flash ROM and EPROM devices have specific locations and accesses depending on whether the board is configured to boot from Flash (FlashBoot) or DIP EPROM (ROMBoot). This is summarized below:

***FlashBoot mode*** (Boot ROM Enable jumper not installed): The system boots from Boot Flash ROM at location FFF0_0100. The entire device is addressed at FFF0_0000 - FFFF_FFFF and is D8 read/write if Flash Write Protect jumper (J02N pins 21 and 22) not installed and if FlashWP bit is not on; otherwise, device is read-only. EPROM is addressed at FFE0_0000 - FFE7_FFFF. The EPROM/Flash ROM address map for FlashBoot is shown below:

**Address Map, FlashBoot**

| Address | Device/address space description | Access |
|---|---|---|
| FFE0_0000 - FFE7_FFFF | EPROM (512 KB) | D8 (RW), D64 (RO) |
| FFF0_0000 - FFFF_FFFF | Boot Flash ROM (1 MB) | D8 (RW), D64 (RO) |

***ROMBoot mode*** (Boot ROM Enable jumper installed): The system boots from EPROM at location FFF0_0100. EPROM is addressed at FFF0_0000 - FFF7_FFFF. Boot Flash ROM is read/write and is split-addressed at FFE0_0000 - FFE7_FFFF and FFF8_0000 - FFFF_FFFF. Flash ROM reading/programming code must deal with the address discontinuity to properly access the device. The EPROM/Flash ROM address map for ROMBoot is shown below:

**Address Map, ROMBoot**

| Address | Device/address space description | Access |
|---|---|---|
| FFE0_0000 - FFE7_FFFF | Boot Flash ROM (lower 512 KB) | D8 (RW), D64 (RO) |
| FFF0_0000 - FFF7_FFFF | EPROM (512 KB) | D8 (RW), D64 (RO) |
| FFF8_0000 - FFFF_FFFF | Boot Flash ROM (upper 512 KB) | D8 (RW), D64 (RO) |

Note 2: Mailbox writes are done by writing an 8-bit value to memory. To ensure proper operation, disable cache for page 0 of memory.

Note 3: User Flash is selected by setting bit 7 of the Flash Window Register (0xFFEF_FE50). This register also addresses a particular 512 KB bank of User Flash. The number of available User Flash banks depend on the amount of User Flash installed in the system. User Flash and the upper 512 KB of Boot Flash cannot be accessed at the same time. Access routines in application programs must be mutually exclusive when both Boot Flash and User Flash are used.

## Address space descriptions

The following discussion provides additional information on the VGM5's memory spaces from a software perspective. Addresses shown here reflect use of the default CHRP address map, Address Map B.

Acronyms found in this discussion are described in the table below.

*Address space abbreviations*

| Term | Description | Access function(s) (if applicable) |
|---|---|---|
| **PCA** | PCI configuration address | int readMPC(int address);<br>void writeMPC(int address, int data); |
| **MA** | Processor Memory Address | int readNormal(int address);<br>int readNormal8(int address);<br>void writeNormal(int address, int data);<br>void writeNormal8(int address, int data); |
| **PMA** | PCI Memory Space Address | int readPCI(int address);<br>int readPCI16(int address);<br>int readPCI8(int address);<br>void writePCI(int address, int data);<br>void writePCI16(int address, int data);<br>void writePCI8(int address, int data); |
| **PIA** | PCI I/O Space Address | int readPCI(int address);<br>int readPCI16(int address);<br>int readPCI8(int address);<br>void writePCI(int address, int data);<br>void writePCI16(int address, int data);<br>void writePCI8(int address, int data); |
| **VA** | VME Address | n/a |
| **VA32** | VME Address (A32) | n/a |
| **VRW** | VME Read/Write signal | n/a |
| **RA** | Read-ahead | n/a |
| **WP** | Write Posting | n/a |

### Processor Memory Space

The VGM Series processor memory space tree is:

**MA = DRAM space**

MA = PMA = 0x0000_0000 to 0x3FFF_FFFF

This space can be read as D64 or D32 or D16 or D8.

This space can be written as D64 or D32 or D16 or D8.

This space is reserved by Synergy hardware for DRAM.

**MA = Direct VME space**

MA = PMA = 0x4000_0000 to 0x7FFF_FFFF

VA = 0x0000_0000 to 0x3FFF_FFFF or

VA = 0x4000_0000 to 0x7FFF_FFFF or

VA = 0x8000_0000 to 0xBFFF_FFFF or

VA = 0xC000_0000 to 0xFFFF_FFFF

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

AM Code = A32: 0x0D, A24: 0x3D, A16: 0x2D

This space is reserved by Synergy hardware for the direct VME interface.

The Direct VME interface maps the onboard address region from 0x4000_0000 to 0x77FF_FFFF onto the VMEbus as A32 access, and performs an address translation at the same time. The (almost) 1 GB of onboard address space is mapped onto any one of the four 1-GB segments of VMEbus address space. The mapping is determined by the contents of the Master Address Register at 0xFFEF_FF18. It allows one to specify a 2-bit number which controls which 1 GB of VME address space is being accessed. So, setting those 2 bits to 0, for instance, would cause an A32 read from 0x4000_0000 to appear on the VMEbus as a read to address 0x0000_0000. Setting the 2 bits to 01 would cause a read from 0x4000_0000 to appear on the VMEbus as a read from 0x4000_0000, and so on.

The 16 MB Address range from 0x7B00_0000 to 0x7BFF_FFFF maps out onto the VMEbus as A24 space. So, a read/write to 0x7B00_0000 causes a read/write cycle on the VMEbus from address 0 in A24 space (supervisor access). Only one such window is needed since it covers the entire A24 address space.

The 64 Kb address range from 0x7FFF_0000 to 0x7FFF_FFFF is mapped into VME A16 space starting at VME address 0. So, a read/write from 0x7FFF_0000 causes a read/write cycle on the VMEbus at address 0 in

A16 space. Only one such window is needed since it covers the entire A16 address space.

Other regions above 0x7800_0000 not mentioned above are reserved for future use and should not be accessed.

Note that if the PPC's MMU is enabled, then it is the physical (not virtual) address which is presented to the direct VME interface for translation into a VME address. In that case, the application program's access to a virtual address undergoes 2 translations by the time it appears on the VMEbus. The first is done inside the PPC by its internal MMU. The second is done by the Direct VME interface converting the onboard physical address into a VME address.

**MA = Universe II PCI/VME space**

MA = PMA = 0x8000_0000 to 0xBFFF_FFFF

VA is programmable.

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

> This space is reserved by Synergy convention for PCI to VME transfers.
>
> VA is programmable in the Universe II.

The onboard address range from 0x8000_0000 to 0xBFFF_FFFF is used to access the VMEbus via the Universe II PCI/VME bridge chip. The Universe has 8 sets of master window registers which provide for 8 different possible master VME windows. Access to any of those windows performs a translation of the onboard address onto some other address in VME space. The Universe master window registers require the specification of 1) a beginning onboard address in this address range (0x8000_0000–0xBFFF_FFFF), 2) an ending onboard address and 3) an offset to be added to those addresses in order to convert them to their corresponding VMEbus addresses. See Universe II documentation for reference to “PCI Target Image” registers.

For example, one could program the Universe II to map the 0x9000_0000 to 0x9FFF_FFFF range of onboard addresses into VME address space starting at 0x2000_0000. So, a read from 0x9000_0000 would appear on the VMEbus as a read from 0x2000_0000. A read from 0xA000_0100 in this case would generate a machine check exception, since it would be outside of one of the defined windows. Only regions described by Universe II PCI Target Image registers are accessible. You may use the Universe II to describe all or none of this address space as your requirements dictate.

Note that if the PPC's MMU is enabled, it is the physical (not virtual) address which is presented to the Universe II chip for translation into a VME address. In the MMU case the address of a VME access goes through 2 translations by the time it shows up on the VMEbus.

**MA = Misc. PCI space**

MA = PMA = 0xC000_0000 to 0xFCFF_FFFF

VA is programmable.

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

> This space is assigned by programming PCI base addresses in a PCI chip.
>
> PMA and VA are programmable in the Universe II.

**MA = PCI ISA Memory space**

MA = 0xFD00_0000 to 0xFDFF_FFFF

Don't read.

Don't write.

**MA = PCI ISA I/O space**

MA = 0xFE00_0000 to 0xFE7F_FFFF

Don't read.

Don't write.

**MA = PCI I/O space**

MA = 0xFE80_0000 to 0xFEBF_FFFF

PIA = 0x00BF_FFFF to 0x0080_0000

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

> Note The Universe II can be set up to write from VME address space into this PCI I/O space.

**MA = PCI Configuration Address Indirect space**

MA = 0xFEC0_0000

This space can be read as D32.

This space can be written as D32.

**MA = PCI Configuration Data Indirect space**

MA = 0xFEE0_0000

This space can be read as D32.

This space can be written as D32.

**MA = PCI Interrupt Acknowledge space**

MA = 0xFEF0_0000 to 0xFEFF_FFFF

This space can be read as D32 or D8.

This space cannot be written.

**MA = Control space**

MA = PMA = 0xFFE0_0000 to 0xFFEF_FFFF

This space can be read as D64 or D8.

This space can be written as D8.

It is important to note that a read occurs as D64. When you read the serial port registers as D32, you read 64 bits at a time. These are eight 8-bit registers. Note that reading some of the serial port registers causes side effects. Therefore, you can get really weird results if you read this memory space using D32. Use D8 instead.

**MA = ROM space**

MA = PMA = 0xFFF0_0000 to 0xFFFF_FFFF

This space can be read as D64 or D8.

This space can be written as D8.

## PCI Memory space

Note The Universe II can be set up to to do VME slave accesses (read or write) anywhere in PCI memory space.

The PCI memory space tree is:

**PMA = DRAM space**

PMA = 0x0000_0000 to 0x3FFF_FFFF

MA = 0x0000_0000 to 0x3FFF_FFFF

This space can be read as D64 or D32 or D16 or D8.

This space can be written as D64 or D32 or D16 or D8.

This space is reserved by Synergy hardware for DRAM.

**PMA = Direct VME space**

PMA = 0x4000_0000 to 0x7FFF_FFFF

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

This space is reserved by Synergy hardware for the direct processor to VME interface.

VA = 0x0000_0000 to 0x3FFF_FFFF or

VA = 0x4000_0000 to 0x7FFF_FFFF or

VA = 0x8000_0000 to 0xBFFF_FFFF or

VA = 0xC000_0000 to 0xFFFF_FFFF

AM Code = 0x0D

**PMA = Universe II PCI/VME space**

PMA = 0x8000_0000 to 0xBFFF_FFFF

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

This space is reserved by Synergy hardware for PCI to VME transfers.

PMA and VA are programmable in the Universe II.

**PMA = Misc. PCI space**

PMA = MA = 0xC000_0000 to 0xFCFF_FFFF

VA is programmable.

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8.

This space is assigned by programming PCI base addresses in a PCI chip.

PMA and VA are programmable in the Universe II.

**PMA = PCI Configuration Address Indirect space**

PMA = 0xFEC0_0000

This space can be read as D32.

This space can be written as D32.

**PMA = PCI Configuration Data Indirect space**

PMA = 0xFEE0_0000

This space can be read as D32.

This space can be written as D32.

**PMA = Control space**

PMA = MA = 0xFFE0_0000 to 0xFFEF_FFFF

This space can be read as D64 or D8.

This space cannot be written.

It is important to note that a read occurs as D64. When you read the serial port registers as D32, you read 64 bits at a time. These are eight 8-bit registers. Note that reading some of the serial port registers causes side effects. Therefore, you can get really weird results if you read this memory space using D32. Use D8 instead.

**PMA = ROM space**

PMA = MA = 0xFFF0_0000 to 0xFFFF_FFFF

This space can be read as D64 or D8.

This space cannot be written.

## PCI I/O space

The Universe II can be set up to do VME slave accesses (read or write) anywhere in PCI I/O space.

The PCI I/O space tree is:

**PIA = PCI I/O space**

PIA = 0x0080_0000 to 0x00BF_FFFF

MA = PMA = 0xFE80_0000 to 0xFEBF_FFFF

This space can be read as D32 or D16 or D8.

This space can be written as D32 or D16 or D8

## VME master space

The VME master space tree is programmable, through the Universe II and the direct interface.

Each VME address can have only 1 AM Code associated with it for Universe II accesses (at any given time). There are 2 Universe II master windows of 4 KB to 1 GB in size. There are 6 Universe II master windows of 64 KB to 1 GB in size.

When programming the Universe II master windows, make sure you understand the ramifications of write posting before enabling this feature.

The direct master has AM Code=0x0D. The direct master space is 1 GB in size. The direct master space can be different for the X and Y processors.

It is suggested that VGM5 users make up their own VME master space tree for reference purposes.

### VME slave space

The VME slave space tree is programmable, via the Universe II, and the direct interface.

There are 2 Universe II slave windows of 4 KB to 1 GB size, and 6 Universe II slave windows of 64 KB to 1 GB size. Each Universe II slave window responds to only one AM Code. The windows can overlap (if the AM Codes are different). Each slave window can only generate reads and writes of one type of PCI access (i.e., PCI memory access, PCI I/O access, or PCI configuration access).

When programming the Universe II slave windows, make sure you understand the ramifications of read-ahead and write posting before enabling either of these features.

The direct VME slave window can be placed on any 16 MB boundary, with size ranging from 16 MB to 256 MB. The direct slave only responds to AM Code=0x0D. The direct slave generates memory accesses only. The base MA of the direct VME slave window is always 0.

Make sure that no two slave windows (of either Universe II or direct) overlap in VME space, if the windows respond to the same AM Code.

It is suggested that VGM5 users make up their own VME slave space tree for reference purposes.

# Onboard registers

This chapter describes the contents and use of the onboard registers which monitor and control the operation of various features and functions on VGM5 boards. All registers described in this chapter are 8-bits wide and limited to PowerPC bus accesses only. The onboard registers fall into one of three categories:

- **Board information registers** — are *read-only* registers that provide the system with ID and configuration information.
- **Status registers** — are *read-only* registers that indicate the *status or condition* of on-board devices or processes. Using these registers involves reading the register and interpreting the bit pattern found there.
- **Control/Mode registers** — are *read/write* registers that *set up the board* to perform a given operation or function. Using a Control/Mode register involves writing a particular hex value to the register's address location. A read of the control/mode address location gives the current value of the register.

The following register descriptions include address location, access mode (read/write [RW], write-only [WO] or read-only [RO]), bit description, and a brief summary of what it does.

The register bit description uses the notation listed below in each bit position to show the register's value after a board reset (i.e., power cycling or system reset).

***Register bit description notations for reset value***

| Notation | What it means |
|---|---|
| *x* | Unused bit; set to 0 for future compatibility |
| *—* | Read-only bit |
| *1* | Set to 1 upon reset |
| *0* | Set to 0 upon reset |

## Board information registers

VGM5 board information registers include:

- Board type and revision register
- Special mod and ECO level register
- Board family and feature register
- L2 cache register
- Memory register
- Secondary PCI slot register
- VME64 slot register

### Board type and revision register, 0xFFEF_FE00 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *1* | *0* | *1* | *—* | *—* | *—* | *—* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b4 | Board Type | 0x1 = VGM1<br>0x2 = VGM2<br>0x4 = VSS4<br>0x5 = VGM5<br>0xC = VGMC<br>0xD = VGMD |
| b3–b0 | Revision Level | 0x0 = a<br>0x1 = b<br>0x2 = c<br>↓ :<br>0xF = p |

A byte read of this register reveals the board type (higher order nibble) and board revision level (lower order nibble).

## Special mod and ECO level register, 0xFFEF_FE08 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b4 | Special Mod | 0x0 = none<br>0x1 = ________<br>0x2 = ________<br>0x4 = ________<br>0x5 = ________<br>0x6 = ________ |
| b3–b0 | ECO Level | 0x0 = none<br>0x1 = 1<br>0x2 = 2<br>↓ :<br>0xF = 15 |

A byte read of this register reveals the special modification code (higher order nibble) and ECO level of the board (lower order nibble). Space is provided above to write in the special mod to the code that applies to your board.

## Board family and feature register, 0xFFEF_FE10 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | — | — | — | — | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b4 | Board Family | 0 = VGM Series<br>1 = VSS Series<br>2 = KGM Series<br>3 = VGR Series |
| b3–b0 | Reserved | — |

A byte read of this register reveals the board's special features, if any, and the board family to which it belongs.

## L2 cache register, OxFFEF_FE30 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b2 | Reserved | — |
| b1-b0 | L2 Clock Ratio (CPU core to L2 frequency divider) | 0 = 1:1.0<br>1 = 1:1.5<br>2 = 1:2.0<br>3 = Reserved |

This read-only register returns the CPU's L2 clock ratio based on the relative clock speed of the processor and L2 SRAM parts that are installed on the board. This value must then be programmed into the CPU's L2 cache control registers for the L2 cache to function properly.

## Memory register, OxFFEF_FE38 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b6 | Type of Memory | 0 = SDRAM, 15nS, CL=2, Flow-Thru<br>1 = SDRAM, 15nS, CL=2, Registered<br>2 = reserved<br>3 = reserved |
| b5–b3 | Memory per Bank | 0 = 8 MB<br>1 = 16 MB<br>2 = 32 MB<br>3 = 64 MB<br>4 = 128 MB |
| b2–b0 | Number of Banks | 0 = no memory<br>1 = 1 bank<br>2 = 2 banks<br>3 = 4 banks<br>4 = 8 banks |

A byte read of this register provides information on the board's installed memory. Information provided in this register includes number of banks, capacity per bank, and type of memory used (SDRAM type, etc.).

## Secondary PCI Slot register, OxFFEF_FE48 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | x | x | x | x | x | x | x | — | Reset value |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b1 | Reserved | — |
| b0 | Sec. PCI Bus Slot controller Indicator | 0 = Not System Controller<br>1 = System Controller |

A byte read of this register informs which CPU board is responsible for configuring the P0•PCI™ bus map. If b0 = 0, the SBC configures the P0•PCI™ bus map. If b0 = 1, a CPU board on the P0 side configures the P0•PCI™ bus map.

## VME64 Slot register, OxFFEF_FF30 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | x | — | — | — | — | — | — | — | Reset value |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Reserved | — |
| b6 | Slot Number Source | 0 = Manual, Slot Jumper Field (J02N)<br>1 = Auto, VME64x Geographical Address |
| b5 | Valid/Invalid Geographical Address | 0 = invalid<br>1 = valid |
| b4–b0 | Binary Encoded Slot Number | 1 = slot 1<br>2 = slot 2<br>↓ :<br>21 = slot 21 |

A byte read of this register returns information on the board's VME64x slot number assignment. The 5 lower order bits make up the binary encoded slot number. Additional bits show whether or not the VME64x geographical address is valid, and the source of the slot number assignment (board's jumper field [J02N] or board's geographical address pins).

## Status registers

VGM5 status registers include:

- Eight-bit user switch register
- Board status register
- CPU status register

### Eight-bit user switch register, 0xFFEF_FD00 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Switch 7 | 0 = Off<br>1 = On |
| b6 | Switch 6 | 0 = Off<br>1 = On |
| b5 | Switch 5 | 0 = Off<br>1 = On |
| b4 | Switch 4 | 0 = Off<br>1 = On |
| b3 | Switch 3 | 0 = Off<br>1 = On |
| b2 | Switch 2 | 0 = Off<br>1 = On |
| b1 | Switch 1 | 0 = Off<br>1 = On |
| b0 | Switch 0 | 0 = Off<br>1 = On |

A read of this register shows the setting of the 8-position front panel switch.

## Board status register, 0xFFEF_FE18 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | x | x | x | x | — | — | — | — | Reset value |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b4 | Reserved | — |
| b3-b2 | PowerPC Bus Speed | 0 = 66 MHz<br>1 = 83 MHz<br>2 = 100 MHz<br>3 = undefined |
| b1 | Board Ejector Handle Switches* | 0 = Both eject handle switches are closed<br>1 = one/both eject handles switch(es) open |
| b0 | Fail LED | 0 = OFF<br>1 = ON |

NOTE: This bit valid only for boards with locking ejector handle option.

A read of this register shows the status of the board's FAIL LED (on/off), ejector handle close/open state, and the PowerPC bus speed.

## CPU status register, 0xFFEF_FE20 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | Reset value |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | CPU-W status | 0 = Halted<br>1 = Running |
| b6 | CPU-Z status | 0 = Halted<br>1 = Running |
| b5 | CPU-Y status | 0 = Halted<br>1 = Running |
| b4 | CPU-X status | 0 = Halted<br>1 = Running |
| b3-b2 | Number of CPUs | 0 = 4<br>1 = 1<br>2 = 2<br>3 = 3 |
| b1-b0 | CPU ID* | 0 = W<br>1 = Y<br>2 = Z<br>3 = W |

NOTE: This is a processor-dependent register. All CPUs access the same address, but the information in b1-b0 identifies which CPU is performing the read.

A read of this register returns the status of the CPU(s) on the board. Information in this register includes the CPU ID of the processor doing the read, the total number of CPUs on the board, and the halt/run status for all onboard CPUs.

Note that the number of CPUs will always be either 1 or 2 on a VGM5, with CPU-Z and CPU-W always “0”. Your application software should be written to assume the existence of up to four CPUs for upward compatibility.

## Control/Mode registers

VGM5 control/mode registers include:

- CPU timebase register
- Flash ROM register
- Flash Window register
- P0•PCI™ Interrupt assert/pending register
- User LED registers
- PowerPC–VME64 slave mask register
- PowerPC–VME64 slave address register
- PowerPC–VME64 slave offset register
- PowerPC–VME64 master address register
- Power–PC VME64 mode register
- VME64 SysReset register
- PowerPC–VME64 Read-Modify-Write register

Note that the symbol ↗ indicates that the register is used for the Direct VME Interface. Information on the Direct VME interface is found in Section 8 (page 217).

## Using Control/Mode register functions

Activating a Control/Mode register function involves writing the appropriate hexadecimal data value to the appropriate register.

For example, turning on LED 0 would require writing **0x01** to the User LED register at **0xFFEF_FE80** using the following PowerPC instructions:

```
lis   3, 0xFFEF
ori   3, 3, 0xFE80
li    4, 1
stb   4, 0 (3)
```

## CPU Timebase register, 0xFFEF_FE28 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *x* | *x* | *x* | *x* | *1* | *1* | *1* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b4 | Reserved | — |
| b3 | CPU-W Timebase Enable | 0 = Disabled<br>1 = Enabled |
| b2 | CPU-Z Timebase Enable | 0 = Disabled<br>1 = Enabled |
| b1 | CPU-Y Timebase Enable | 0 = Disabled<br>1 = Enabled |
| b0 | CPU-X Timebase Enable | 0 = Disabled<br>1 = Enabled |

A write to this register enables (1) or disables (0) the CPU's (X or Y) internal timebase generator. This internal timebase generator is a free running counter that runs at the CPU's core frequency. Software can use this free running counter to determine the speed at which the processor is running, or other timing functions as required. A read of this register returns the status of the timebase enables/disables.

Bits 2 and 3 are reserved for CPU-Z and CPU-W respectively. Though these CPUs aren't present on the VGM5, the associated bits are provided for upward compatibility.

## Flash ROM register, 0xFFEF_FE40 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | *x* | *x* | *x* | *x* | *x* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Boot ROM Enable jumper, J02N 1 & 2 (RO) | 0 = Jumper OFF (Boot Source is Boot Flash)<br>1 = Jumper ON (Boot Source is DIP EPROM) |
| b6 | Flash ROM Write Protect Jumper, J02N, 21 & 22 (RO) | 0 = Jumper OFF (no write protect)<br>1 = Jumper ON (write protect) |
| b5-b1 | Reserved | — |
| b0 | Software Flash ROM Write Protect (RW) * | 0 = Flash ROM not write-protected (jumper ON** overrides this)<br>1 = Flash ROM write-protected (set bit overrides jumper OFF) |

NOTE: * When Flash ROM is not write-protected, only CPU-X has access to it.
** Jumper refers to J02N Flash Write Protect jumper.

This register addresses the board's Flash ROM operation.

Bit 0 of this register is a software Flash write-protect bit. Setting this bit protects Flash from writes even if the Flash write protect jumper (J02N) is removed. Clearing this bit allows Flash to be written provided that the J02N Flash write protect jumper is also removed and that the MPC106 ROM write protect bit has not been set since the last board reset.

A read of bit 6 indicates the state of the Flash write protect jumper (J02N pins 21 & 22). When 1, the Flash write protect jumper is ON for Flash write protection. When 0, the Flash write protect jumper is OFF to enable Flash writes. However, this is true only if the write-protect bit (b0) of this register is cleared and if the MPC106 ROM write-protect bit has not been set since the last board reset.

Flash write protection indicated or set by this register applies to Boot Flash, User Flash, and Flash EPROM (if installed) in the DIP EPROM socket.

A read of bit 7 indicates the state of the Boot ROM enable jumper (J02N pins 1 & 2) which selects between two possible boot sources. When 0, the jumper is OFF and boot is from Boot Flash. When 1, the jumper is ON and boot is from DIP EPROM.

Refer to Section 2, ***Setting up the VGM5 hardware*** chapter (page 25) for more information on the Flash write protect jumper and the Boot ROM enable jumper.

## Flash Window register, 0xFFEF_FE50 (WO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | User Flash Select | 0 = Boot Flash<br>1 = User Flash |
| b6-b0 | User Flash 512K Bank Select | 4 MB = 0x00–0x87 (8 banks)<br>8 MB = 0x00–0x8F (16 banks)<br>16 MB = 0x00–0x9F (32 banks)<br>32 MB = 0x00–0xBF (64 banks)<br>64 MB = 0x00–0xEF (128 banks) |

This write-only register selects which Flash memory appears at address range 0xFFF8_0000–0xFFFF_FFFF (512 KB). When User Flash is selected (b7 = 1) a particular **512 KB** bank or window of that memory is addressed by bits b6-b0. As listed above, the total available 512 KB banks of Flash memory depend on the amount of User Flash installed on the board.

With User Flash selected, only the lower 512 KB of Boot Flash is accessible. With Boot Flash selected, the entire 1 MB of Boot Flash is available for use.

This register returns an undefined value when read.

Refer to this section's ***User Flash memory*** chapter (page 165) for more information.

## P0 • PCI™ interrupt assert/pending register, 0xFFEF_FE68 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | *x* | *x* | *x* | — | — | — | — | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | P0•PCI™ Bridge Interrupt (read only) | 0 = not asserted<br>1 = asserted |
| b6-b4 | Reserved | — |
| b3 | P0•PCI™ Interrupt D (PciIntD\) | 0 = not asserted<br>1 = asserted |
| b2 | P0•PCI™ Interrupt C (PciIntC\) | 0 = not asserted<br>1 = asserted |
| b1 | P0•PCI™ Interrupt B (PciIntB\) | 0 = not asserted<br>1 = asserted |
| b0 | P0•PCI™ Interrupt A (PciIntA\) | 0 = not asserted<br>1 = asserted |

NOTE: All P0•PCI™ Interrupts/Errors come over MPIC line 6.

Setting b3–b0 of the P0 • PCI™ interrupt assert/pending register asserts a PCI interrupt (Int A–D) on the P0 • PCI™ bus. Clearing b3–b0 deasserts the corresponding interrupt. A read of this register returns the current state of the P0 • PCI™ bus interrupt signals.

## User LED registers, 0xFFEF_FE80, 0xFFEF_FE88, 0xFFEF_FE90, 0xFFEF_FE98, 0xFFEF_FEA0, 0xFFEF_FEA8, 0xFFEF_FEB0, 0xFFEF_FEB8 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *x* | *x* | *x* | *x* | *x* | *x* | *x* | *1* | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b1 | Reserved | — |
| b0 | User LED *n*<br>(n = 0, 1, 2, 3, 4, 5, 6, 7) | 0 = OFF<br>1 = ON |

There are eight User LED registers, one for each user LED. A write to one of these registers turns a user LED ON or OFF. A read of one of these registers returns the ON/OFF status of the appropriate user LED.

## PowerPC-VME64 Slave Mask register, 0xFFEF_FF00 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | PowerPC-VME64 Slave Enable | 0 = Slave disabled<br>1 = Slave enabled |
| b6 | VME64 Broadcast Enable | 0 = Broadcasting slave support disabled<br>1 = Broadcasting slave support enabled |
| b5 | VME64 Broadcast, DTAck Driver Enable[1] | 0 = DTAck disabled<br>1 = DTAck enabled |
| b4 | VME64 Broadcast, Wait Line Selection | 0 = VME SerA<br>1 = VME SerB |
| b3-b0 | Slave Address Mask bits 29-24[2]<br>(Sets the size of the PowerPC-VME64 slave window. Must be less than or equal to DRAM size.) | 0 = 16 MB window size<br>1 = 32 MB window size<br>3 = 64 MB window size<br>7 = 128 MB window size<br>F = 256 MB window size |

NOTE S: 1. Set one board in a broadcast group for DTAck drive enabled; set all others disabled.

2. All other values for b3-b0 (window size) are reserved and should not be used.

A write to this register enables/disables VME64 slave operation, sets the size of the slave window, and sets up/enables VME64 broadcasting support. A read of this register returns the enable/disable status of VME64 slave operation, the size of the slave window, and VME64 broadcast configuration. This register is used in the Direct VME interface. Refer to Section 8 (page 217) for more information. In addition, this register and the VME64 Broadcast Slave Address register (0xFFEF_FF10) controls the broadcast feature (contact factory for availability of this feature).

## PowerPC-VME64 Slave Address register, 0xFFEF_FF08 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b0 | VME64 Slave Addr bits 31-24 | 00 = 0000_0000 – 00FF_FFFF<br>01 = 0100_0000 – 01FF_FFFF<br>↓ :<br>FF = FF00_0000 – FFFF_FFFF |

A write to this register sets the VME64 slave address bits 31–24. A read of this register returns the value of those bits. This register is used in the Direct VME interface. Refer to Section 8 (page 217) for more information.

### VME64 Broadcast Slave Address register, 0xFFEF_FF10 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b0 | VME64 Broadcast Slave Addr bits 31-24 | 00 = 0000_0000 – 00FF_FFFF<br>01 = 0100_0000 – 01FF_FFFF<br>↓ :<br>FF = FF00_0000 – FFFF_FFFF |

The VME64 broadcast slave address register contains the upper 8-bits of the VME 32-bit address used as the broadcasting window. This register and the PowerPC–VME64 Slave Mask register (0xFFEF_FF00) controls the broadcast feature (contact factory for availability of this feature).

### PowerPC–VME64 Master Address register, 0xFFEF_FF18 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *x* | *x* | *x* | *x* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b6 | Master Address bits 31-30 | Top two bits of CPU-X VME Master address have this value. |
| b5-b4 | Master Address bits 31-30 | Top two bits of CPU-Y VME Master address have this value. |
| b3-b0 | Reserved | — |

A write to this register sets the top two bits (bits 31–30) of the CPU-X and CPU-Y VME64 master address. A read of this register returns the value of the top two bits of the CPU-X and CPU-Y VME64 master address. This register is used in the Direct VME interface. Refer to Section 8 (page 217) for more information.

## PowerPC-VME64 Mode register, 0xFFEF_FF20 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *x* | *x* | *0* | *1* | *0* | *1* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Broadcast Slave Interface Select | 0 = Universe PCI-VME64 Bridge<br>1 = Direct VME interface |
| b6-b5 | Reserved | — |
| b4 | BLT Speedup, PCI-VME64 Bridge | 0 = slow BLT (default)<br>1 = fast BLT |
| b3 | Bus Release Mode | 0 = RWD (Release When Done)<br>1 = ROR (Release On Request) |
| b2 | FAIR Requestor | 0 = normal, non-FAIR mode.<br>1 = FAIR requestor mode, wait for BRQ false before re-asserting bus request. |
| b1-b0 | Bus Request Level | 0 = Level 0<br>1 = Level 1<br>2 = Level 2<br>3 = Level 3 |

A write to this register sets the VMEbus request level (b0–b1), FAIR or non-FAIR bus requestor (b2), and ROR or RWD bus release mode (b3). It also selects the interface for broadcast slave (direct VME or Universe II, b7) and sets the Universe II PCI-VME64 bridge for slow (normal) or fast BLT (b4). A read of this register returns the value of the selected modes. This register is used in the PCI-VME64 Bridge and Direct VME interfaces. Refer to Sections 7 and 8 (pages 199 and 217, respectively) for more information. Contact the factory for availability of the broadcast slave feature.

When set, the BLT speedup bit can improve BLT performance in some systems by disabling noise filtering from the DS0\ and DS1\ VMEbus signals. The setting of this bit is recommended only for systems with just a few boards. Refer to the ***Improving BLT performance*** chapter in Section 7 (page 211) for more information on this and other ways to improve BLT throughput.

## PowerPC-VME64 RMW register, OxFFEF_FF28 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *x* | *x* | *x* | *x* | *x* | *x* | *x* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b1 | Reserved | — |
| b0 | VME Master Read-Modify-Write | Read:<br>0 = no RMW cycle in progress<br>1 = RMW cycle in progress<br>Write:<br>0 = stop RMW cycle in progress<br>1 = start RMW cycle |

A write to this register starts or stops a RMW (Read-Modify-Write) cycle. A read of this register indicates whether or not a RMW cycle is in progress. This register is used in the Direct VME interface. Refer to Section 8 (page 217) for more information.

## VME64 SysReset register, OxFFEF_FF38 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *x* | *x* | *x* | *x* | *x* | *x* | *x* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b1 | Reserved | — |
| b0 | VME64 SysReset | 0 = Board does not respond to its own SysReset<br>1 = Board does respond to its own SysReset |

A write to this register sets whether or not the board responds to its own SysReset. A read of this register returns the value of the bits setting this register.

# Backside L2 cache controller

The 750/7400 processor has an onboard L2 cache controller with a dedicated port to the external synchronous SRAMs (1 MB for Rev. D or lower boards, 1 MB or 2 MB for Rev. E or higher boards). The backside L2 cache maintains cache coherency through snooping and is normally configured for copyback mode.

For the 750 (G3) processor, the L2 cache is a two-way set associative tag memory with 4096 tags per way. With 1 MB of SRAM, the L2 tags are configured for four sectors (128 bytes) per L2 cache block.

For the 7400 (G4) processor, the L2 cache is a two-way set associative tag memory with 8K tags per way. The tags are sectored depending on L2 cache size. A 1 MB L2 supports 2 cache line blocks per tag entry (2 sectors, 64 bytes). A 2 MB L2 supports 4 cache lines blocks per tag entry (4 sectors, 128 bytes).

Because the cache runs at or near the CPU core frequency and has its own bus to the cache SRAMs, the 750/7400's performance is noticeably improved over similar processors running at the same speed.

When a CPU read memory access is detected:

1. It looks in its L1 cache to service the CPU request. If not in L1, it looks in L2.
2. If what the CPU wants is in either L1 or L2, it's a "hit" and the CPU fetches the data without using a memory access cycle.
3. If the requested data is not in either L1 or L2 cache, it's a "miss" and the CPU accesses the main memory for the data using the Grackle.

For a memory write, the backside L2 cache will "copy-back" the write:

1. The CPU, due to an L1 cache castout, attempts to write data to memory.

2. The backside L2 cache captures the write and stores the data.

3. If the L2 cache write caused an L2 castout because that cache line was already used, then the L2 cache writes (copies back) the previous data to memory.

## How to use the backside L2 cache

The configuration of the backside L2 cache including the enable/disable status is set by the processor's L2 cache control register (L2CR). The VGM5 is provided with an onboard register at 0xFFEF_FE30 to discover the proper L2 clock ratio. Information on this register is found in this section on page 126. Refer to the Motorola 750/7400 User's Manual for more information about the backside L2 cache interface operation.

# Mailboxes

The VGM5 provides one or two 8-bit wide mailboxes for interprocessor communication. A single CPU board has only mailbox 0, while a dual CPU board has both mailboxes 0 and 1. Each mailbox generates an interrupt while it contains any data.

The mailbox write addresses are in RAM space, which is accessible by all memory owners including either CPU, all PCI masters including PMC cards, the direct VME slave and the Universe VME slave. The mailbox read registers are located in ROM space, and are readable only by the CPUs.

Mailbox 0 is typically used to interrupt CPU-X, and mailbox 1 is typically used to interrupt CPU-Y. This interrupt steering is programmed by the user into the MPIC interrupt controller, and may be changed if needed. Refer to Section 6, ***MPIC Interrupt Controller*** (page 177) for more information on the MPIC.

Each mailbox contains a FIFO containing 256 storage locations which allows up to 256 pending messages in each mailbox. This provides buffering for multiple near-simultaneous messages from many different processes. Messages written to the mailbox will be read by the CPU in the order they were received.

Note

For proper operation of the mailbox write function, set memory page 0x0000_0000 to **non-cacheable** mode in the MMU.

Any processor, but no other device (e.g., a PMC card), is allowed to read a mailbox. This restriction is in place to prevent CPU interrupts from being mishandled. Reading the mailbox will return the least recently written value and will cause that entry to be removed from the FIFO. When all pending mailbox data has been read, the interrupt will be cleared. The ISR does not need to read all data from the FIFO; in fact, it's best to have the interrupt routine read one entry from the mailbox and return to program execution. If additional data is in the mailbox, the interrupt will still be active and the ISR will be entered again automatically.

Mailbox addresses are listed below.

***Mailbox read/write addresses***

| Mailbox | Write | Read |
|---|---|---|
| Mailbox A | 0x0000_0080 | 0xFFEF_FC00 |
| Mailbox B | 0x0000_00A0 | 0xFFEF_FC08 |

The mailbox write may be any data size, but only the byte of data at exactly the specified address (i.e., the most significant byte) will be written to the FIFO.

# Programming notes, Mailbox

## Handling mailbox writes

### Problem

If the mailbox address is written to many times in a row, it is possible for the SBC to miss mailbox interrupts due to the Grackle's store gathering behavior.

### Solution

One must assume that the mailbox FIFO is only one element deep.

To ensure that no mailbox interrupts are lost, it is recommended that your application's mailbox routine include the following:

- A mechanism, such as a separate counter variable, that determines if more than one mailbox write occurred.
- Data protection during loads and stores — typically done by using atomic Read-Modify-Write operations.

Example code for this can be found in:

- Shared Memory Library, VxWorks Board Support Package (BSP)

Refer to the `sysMailboxWrite` and `sysMailboxInt` functions in the Shared Memory Library of this BSP. Contact Customer Service for ordering information for this and other BSPs offered by Synergy Microsystems.

# *Asynchronous serial interface*

The VGM5 boards provide two asynchronous channels (**A** & **B**) via National Semiconductor's PC16552D chip, which is essentially a dual 16550 UART. The two serial channels are brought out to a single RJ-69 jack on the front panel.

For more information about programming the PC16552D, refer to the National Semiconductor PC16552D datasheet, no. RRD-B30M75. This can be obtained by contacting:

National Semiconductor, Corp.
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1 (800) 272-9959

The datasheet is also available on National Semiconductor's website:

**http://www.national.com/pf/PC/PC16552D.html#Datasheet**

The PC16552D contains two identical register sets, one for each channel. The registers per channel are listed below.

***Asynchronous serial interface registers***

| Address | DLAB | Register | |
|---|---|---|---|
| FFEF_FB00 | 0 | Receive Data (read)/Transmit Data (write) | PORT B |
| FFEF_FB00 | 1 | Divisor Latch (LSB) | |
| FFEF_FB01 | 0 | Interrupt Enable | |
| FFEF_FB01 | 1 | Divisor Latch (MSB) | |
| FFEF_FB02 | 0 | Interrupt ID (read)/FIFO Control (write) | |
| FFEF_FB02 | 1 | Alternate Function | |
| FFEF_FB03 | X | Line Control | |
| FFEF_FB04 | X | Modem Control | |
| FFEF_FB05 | X | Line Status | |
| FFEF_FB06 | X | Modem Status | |
| FFEF_FB07 | X | Scratch | |
| **Address** | **DLAB** | **Register** | |
| FFEF_FB08 | 0 | Receive Data (read)/Transmit Data (write) | PORT A |
| FFEF_FB08 | 1 | Divisor Latch (LSB) | |
| FFEF_FB09 | 0 | Interrupt Enable | |
| FFEF_FB09 | 1 | Divisor Latch (MSB) | |
| FFEF_FB0A | 0 | Interrupt ID (read)/FIFO Control (write) | |
| FFEF_FB0A | 1 | Alternate Function | |
| FFEF_FB0B | X | Line Control | |
| FFEF_FB0C | X | Modem Control | |
| FFEF_FB0D | X | Line Status | |
| FFEF_FB0E | X | Modem Status | |
| FFEF_FB0F | X | Scratch | |

For further explanation of these registers including detailed bit descriptions, refer to the PC16552D datasheet.

## Serial interface interrupts

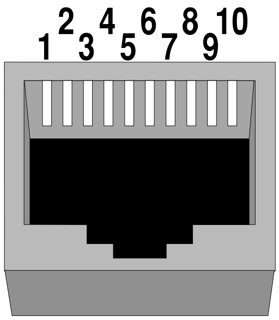
The two serial interfaces each provide an interrupt which is given to the MPIC chip. The priority and destination of these interrupts are programmable. Refer to Section 6, ***MPIC interrupt controller*** (page 177) for more information.

### Enabling/disabling the serial ports as interrupt sources

The serial ports A and B are enabled/disabled via the PC16552D Interrupt Enable Register. Refer to the PC16552D datasheet for programming details.

## Serial interface connector

The figure and table below identify the pinout numbers and signals for the VGM5 front panel serial port RJ-69 connector (P347 for Rev. D or lower boards; P345 for Rev. E or higher boards).



*Asynchronous serial connector pin numbering*

*Serial Ports A & B (P347/P345) pin assignments*

| Pin | Function |
|---|---|
| 1 | Port B, Request to Send (RTS) |
| 2 | Port A, Request to Send (RTS) |
| 3 | Port B, Transmit Data (TxD) |
| 4 | Port A, Transmit Data (TxD) |
| 5 | Ground (Gnd) |
| 6 | Ground (Gnd) |
| 7 | Port A, Receive Data (RxD) |
| 8 | Port B, Receive Data (RxD) |
| 9 | Port A, Clear to Send (CTS) |
| 10 | Port B, Clear to Send (CTS) |

Various serial connection options are available:

- Serial port A connection only via modular cable with a modular-to D adapter.
- Serial ports A and B connections via optional cable adapter assembly with dual RJ-45 jacks.
- Serial ports A and B connections via optional cable adapter assembly with DB-25 and RJ-45 connectors.

Refer to the serial port connector discussion in Appendix A (page 297) for more information on VGM5 serial port connectivity.

# Clock calendar

The VGM5 provides clock/calendar data including the year (in 4 digits), month, date, day, hour, minutes, and seconds data in 24-hour BCD format from a SGS-Thomson M48T201Y Timekeeper SRAM controller chip. The clock calendar is backed-up by a user-replaceable lithium battery that should last for 5–10 years.

For high-altitude applications, a capacitor backup option is available to back up the clock calendar in lieu of the regular lithium battery which is not manufacturer-certified for use in a low pressure (high-altitude) environment. The capacitor backup option provides 12 days (typical) of backup and raises the board's maximum operating altitude to approximately 50,000 ft.

Note: For more information about this device, see the *M48T201Y Timekeeper Controller* datasheet which is available as a PDF file from the SGS-Thomson website:

**http://www.st.com/stonline/books/index.htm**

## Clock address locations

The M48T201Y Clock/SRAM is an 8-bit peripheral. Each M48T201Y memory location must be accessed on successive byte boundaries as illustrated in the table below.

**Clock/calendar registers**

| Register address | Data bits 0-7 | | | | | | | | Range | |
|---|---|---|---|---|---|---|---|---|---|---|
| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | | |
| FFE9_FFFF | 10 Years | | | | Year | | | | Year | 00-99 |
| FFE9_FFFE | 0 | 0 | 0 | 10M | Month | | | | Month | 01-12 |
| FFE9_FFFD | 0 | 0 | 10 Date | | Day of Month | | | | Day of Month | 01-31 |
| FFE9_FFFC | 0 | FT | 0 | 0 | 0 | Day of the Week | | | Day of Week | 01-07 |
| FFE9_FFFB | 0 | 0 | 10 Hours | | Hours (24-hr Format) | | | | Hour | 00-23 |
| FFE9_FFFA | 0 | 10 Minutes | | | Minutes | | | | Minutes | 00-59 |
| FFE9_FFF9 | ST | 10 Seconds | | | Seconds | | | | Seconds | 00-59 |
| FFE9_FFF8 | W | R | S | Calibration | | | | | Control | |
| FFE9_FFF7 | WDS | BMB 4 | BMB 3 | BMB 2 | BMB 1 | BMB 0 | RB1 | RB0 | Watchdog | |
| FFE9_FFF6 | AFE | SQWE | ABE | Al. 10M | Alarm Month | | | | Al. Month | 01-12 |
| FFE9_FFF5 | RPT4 | RPT5 | Al. 10 Date | | Alarm Date | | | | Al. Date | 01-31 |
| FFE9_FFF4 | RPT3 | 0 | Al. 10 Hrs. | | Alarm Hours | | | | Al. Hours | 00-23 |
| FFE9_FFF3 | RPT2 | Alarm 10 Minutes | | | Alarm Minutes | | | | Al. Minutes | 00-59 |
| FFE9_FFF2 | RPT1 | Alarm 10 Seconds | | | Alarm Seconds | | | | Al. Seconds | 00-59 |
| FFE9_FFF1 | 1000 Years | | | | 100 Years | | | | Century | 00-99 |
| FFE9_FFF0 | WDF | AF | 0 | BL | RS3 | RS2 | RS1 | RS0 | Flags | |

Key:

S= Sign bit
FT= Frequency Test Bit
R = Read Bit
W = Write Bit
ST = Stop Bit
0 = Must be set to ‘0’
Z = ‘0’ and are read only

WDS = Watchdog Steering Bit
AF = Alarm Flag
BL = Battery Low Flag
SQWE = Square Wave Enable Bit
BMB0-BMB4 = Watchdog Multiplier Bits
RB0-RB1 = Watchdog Resolution Bits
AFE = Alarm Flag Enable
ABE = Alarm in Battery Back-up Mode Enable
RPT1-RPT5 = Alarm Repeat Mode Bits
WDF = Watchdog Flag
RS0-RS3 = SQW Frequency

## Accessing clock data

Access to the clock is as simple as conventional byte-wide RAM access because the RAM and the clock are combined on the same die. The Timekeeper registers are located in the upper 16 locations of the RAM as listed in the table above.

These registers contain, beginning from the top: year, month, day of month, day of week (Sunday = 1), hour, minutes, and seconds data in 24-hour BCD format. Corrections for leap year and the number of days in the month are made automatically. These registers are not the actual clock counters, but BiPort read/write static RAM memory locations. The M48T201Y includes a clock control circuit that, once a second, dumps the counters into the BiPort RAM.

## Clock operations

Updates to the Timekeeper registers should be temporarily suspended before clock data is read to prevent reading of data in transition. Because the BiPort Timekeeper cells in the RAM array are only data registers and not the actual counters, updating the registers can be suspended without disturbing the clock itself.

Updating the data registers is suspended when a 1 is written into the Read bit, the seventh most significant bit in the Control register. As long as a 1 remains in that position, data register updates are suspended. After the Read bit is set, the registers reflect the count, i.e., the day, date, and time that were current at the moment the Read command was issued. All of the Timekeeper registers are updated simultaneously. The Read command will not interrupt an update in progress. Registers are again updated in a normal fashion within a second after the Read bit is reset to a 0.

**Setting the clock** — The eighth bit of the Control register is the Write bit. Setting the Write bit to a 1, like the Read bit, suspends updates to the Timekeeper registers. The user can then load them with the correct day, date, and time data in 24-hour BCD format.

Resetting the Write bit to a 0 transfers those values into the actual Timekeeper counters and allows normal operation to resume. The FT bit, as well as the bits marked with zeros in the above table, must be written with zeros to allow normal Timekeeper and RAM operation.

**Stopping and starting the oscillator** — The oscillator may be stopped at any time. If the CPU board is going to spend a significant amount of time on the shelf, the oscillator can be turned off to minimize current drain from the battery. The STOP bit is the MSB (b7) of the Seconds register. Set this bit to “1” to stop the oscillator.

To start the oscillator, perform these steps.

➊ Set the Write bit to "1".
➋ Reset the Stop bit to "0".
➌ Reset the Write bit to "0".
➍ Wait two seconds.
➎ Set the Write bit to "1".
➏ Set the correct time and date.
➐ Reset the Write bit to "0".

**Calibrating the clock speed** — The low-order 5 bits of the control register ('Calibration' in the table above) represent any value between 0 and 31 in binary form. The sixth bit is a **sign** bit (the **s** bit in the table above) where:

- **S=1** indicates a positive calibration and speeds up the oscillator.
- **S=0** indicates a negative calibration and slows down the oscillator.

Calibration corrections are applied within a 64 minute cycle. The first 62 minutes in each 64 minute cycle may, once per minute, have one second either shortened or lengthened by:

**128/32768 seconds (3.906 ms)**

If a binary 1 is loaded into the ccccc bits, only the first 2 minutes in the 64 minute cycle will be modified; if a binary 6 is loaded , the first 12 minutes of the 64 minute cycle will be affected, and so on. If the oscillator is running precisely at its nominal frequency (32768 Hz), each of the 31 increments in the calibration bits represents 5.35 seconds per (average) month, or, more precisely, 175.78 ms per day. This affords a total calibration range of about 5.4 seconds per day.

The simplest and most accurate method to calibrate the clock is as follows:

❶ **Synchronize** the clock to an accurate timing source such as a GPS receiver or WWV radio transmissions from the National Bureau of Standards in Fort Collins, Colorado (available at 5,000 kHz, 10,000 kHz, and 15,000 kHz on the AM band).

❷ **Accumulate** an error for a few weeks or months if necessary.

❸ **Compare** the clock to the original source.

This procedure yields an accurate correction. Even a manual comparison, which has an error of a second or more, is sufficient to adjust the clock to within a single count of the calibration register.

# Non-volatile 128K x 8 SRAM

VGM5 boards provide 128K bytes of general use non-volatile SRAM. The contents of this non-volatile SRAM are backed-up by a user-replaceable lithium battery that should last for 5–10 years.

For high-altitude applications, a capacitor backup option is available to back up the NVRAM in lieu of the regular lithium battery which is not manufacturer-certified for use in a low pressure (high-altitude) environment. The capacitor backup option provides 12 days (typical) of backup and raises the board's maximum operating altitude to approximately 50,000 ft.

Note

Configuration data in NVRAM may be lost in a capacitor backup-equipped board that is stored unused or plugged into an unpowered system in excess of 12 days. A reprogramming of the NVRAM is required if this occurs.

Consult the factory if your capacitor backup-equipped board requires storage or inactivity prior to being placed in service.

Each SRAM location must be accessed on successive byte-aligned boundaries in the address range shown in the table below:

***Non-volatile SRAM address location***

| Address | Data width | Description |
|---|---|---|
| 0xFFE8_0000 – 0xFFE9_FFF0 | D8* | 128K bytes of battery-backed SRAM |

*Note: SRAM can be read with D8, D16, or D32 accesses. For write accesses, however, only byte-wise writes are allowed. A D32 write to non-volatile SRAM results in a bus error.

## NVRAM space allocation

The table below is a guide to the VGM5's NVRAM space allocation for various operating systems and factory test functions. Observe these space allocations as required by your application. Note that you have the option to use any unassigned or unused OS spaces in the listed NVRAM allocations.

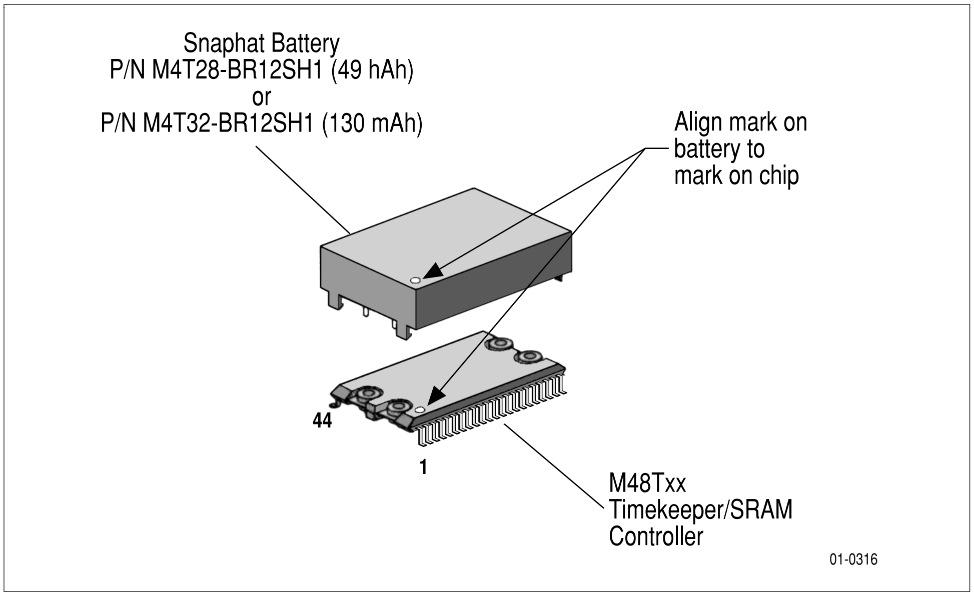
***Non-volatile SRAM space allocations***

| Address | Size | Description |
|---|---|---|
| 0xFFE8_0000 – 0xFFE9_CFFF | 116KB | ***Free User Space*** |
| 0xFFE9_D000 – 0xFFE9_D4FF | 1.25KB | Reserved (factory testing) |
| 0xFFE9_D500 – 0xFFE9_D5FF | 256B | Reserved (Boot up write/verify Scratch Space) |
| 0xFFE9_D600 – 0xFFE9_DAFF | 1.25KB | ***OS9*** |
| 0xFFE9_DB00 – 0xFFE9_DBFF | 256B | unassigned |
| 0xFFE9_DC00 – 0xFFE9_E0FF | 1.25KB | ***LynxOS*** |
| 0xFFE9_E100 – 0xFFE9_E2FF | 512B | unassigned |
| 0xFFE9_E300 – 0xFFE9_E6FF | 1.0KB | ***VxWorks*** |
| 0xFFE9_E700 – 0xFFE9_E8FF | 512B | ***OS common*** (Boot config/Bd. Serial No.[1]) |
| 0xFFE9_E900 – 0xFFE9_EDFF | 1.25KB | ***Linux*** |
| 0xFFE9_EE00 – 0xFFE9_EFFF | 512B | unassigned |
| 0xFFE9_F000 – 0xFFE9_F4FF | 1.25KB | ***pSOS*** |
| 0xFFE9_F500 – 0xFFE9_F7FF | 768B | unassigned |
| 0xFFE9_F800 – 0xFFE9_FCFF | 1.25KB | ***SMon*** |
| 0xFFE9_FD00 – 0xFFE9_FFCF | 720B | unassigned |
| 0xFFE9_FFD0 – 0xFFE9_FFEF | 32B | Reserved (factory testing) |

Note: 1. Board's 7-digit serial no. is encoded as a three-byte value (leading '1' in board serial number ignored) in 0xFFE9_E778 (single processor or CPU-X) and 0xFFE9_E774 (CPU-Y). These three bytes are part of the 6-byte (12-digit) Ethernet ID (also called 'Physical Address') that uniquely identifies the board's Ethernet node(s). Refer to the ***Ethernet ID*** discussion in Section 3 (page 80) for more information about the Ethernet ID.

## Replacing the battery

The replaceable battery (p/n M4T28-BR12SH1) is a Snaphat type with two latches on each end that secure it onto the M48T201Y chip. Refer to the board layout drawing in Appendix B (see page 306) for the location of the NVRAM/clock/calendar battery on your particular board. To remove, use a small screwdriver to pry the battery off the chip. Snap in the replacement battery matching the orientation of the chip pin 1 to the dot marking on the battery. The battery has a key tab to ensure proper mating to the chip. See drawing below.


Snaphat Battery
P/N M4T28-BR12SH1 (49 hAh)
or
P/N M4T32-BR12SH1 (130 mAh)
Align mark on battery to mark on chip
44
1
M48Txx Timekeeper/SRAM Controller
01-0316

***Timekeeper Snaphat battery removal/installation***

For your convenience, one source for the Snaphat battery is listed below:

Mouser Electronics
Phone: 1-800-34MOUSE
**http://www.mouser.com/**
Stock No.: 511-M4T28BR12SH1
Unit price (as of March 2003): $3.10 (USD)
Stock No.: 511-M4T32BR12SH1
Unit price (as of March 2003): $4.43 (USD)

# Boot Flash ROM/DIP EPROM

The VGM5 board can boot from DIP EPROM or Boot Flash ROM.

The VGM5 comes with one 32-pin socket (.6 inches wide) at UE10. This socket accepts one of the following types of JEDEC-standard byte-wide DIP EPROM/Flash memories[†] :

| | |
|---|---|
| 27C010 | 1 Mbit EPROM (128 KB) |
| 27C020 | 2 Mbit EPROM (256 KB) |
| 27C040 | 4 Mbit EPROM (512 KB) |
| 28F020 | 2 Mbit Flash EPROM (256 KB) |
| 29C040 | 4 Mbit Flash EPROM (512 KB) |

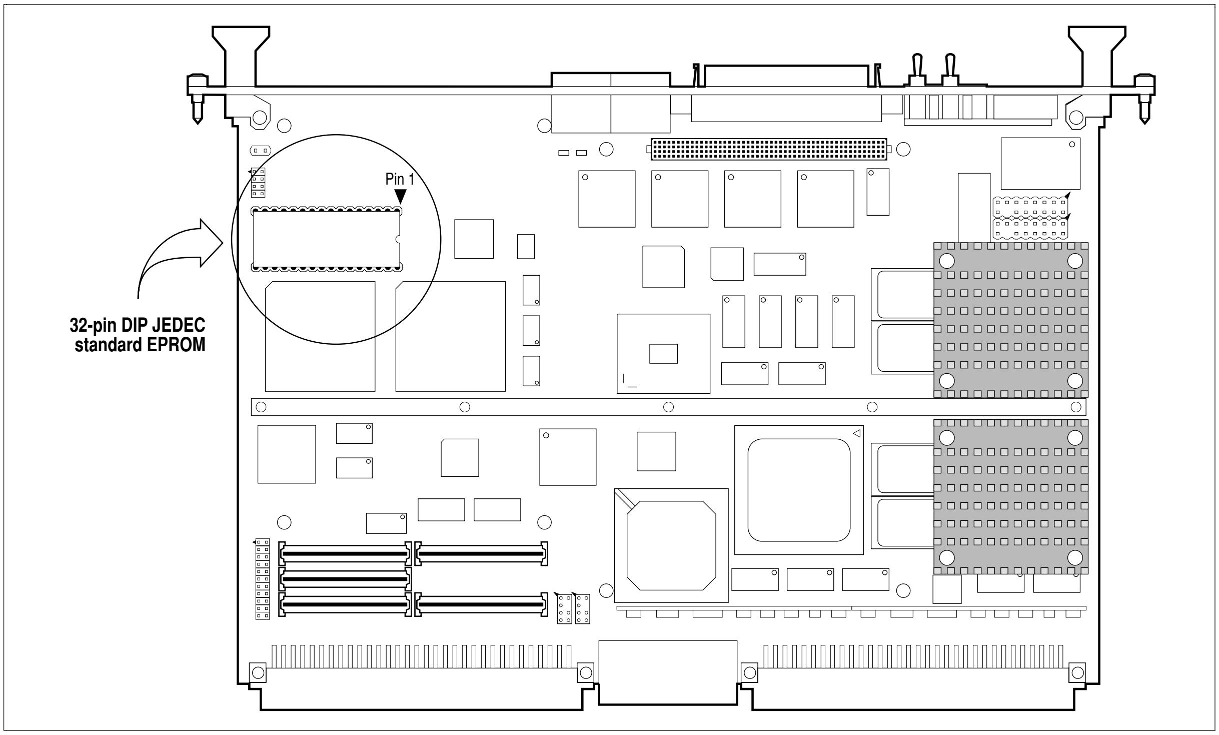
VGM5 boards also come with **1 MB** of onboard boot Flash memory.

The figure below shows the location of the DIP EPROM socket on VGM5 boards. The Boot Flash ROM itself is under this socket.

DIP EPROM is limited to a maximum size of 512 KB. If a larger ROM space is desired, use the Boot Flash (1 MB).

Use a DIP extractor tool (OK Industries model EX-2 or equivalent) to remove the EPROM from its socket. This will avoid damaging the parts underneath and voiding the warranty.

[†] TI brand EPROMs cannot be used. Their requirement for Vcc on unused pins prevents a TI PROM from being used in a general purpose socket. EPROMs from other manufacturers such as Intel, AMD, etc. work without problem.



*VGM5 DIP EPROM socket location*

## Boot options

The Boot EPROM Enable jumper at J02N pins 1 & 2 selects the boot device as follows:

- Jumper ON = boot from **DIP EPROM**
- Jumper OFF = boot from **Boot Flash**

Refer to the configuration discussion in Section 2 ***Getting Started***, page 25, for a jumper diagram and detailed configuration information.

To boot from onboard Flash, boot from DIP EPROM and program the Flash memory with the reset vector and boot code. Once this is done, remove the Boot ROM Enable jumper from J02N pins 1 & 2. With this jumper removed, a power cycle or local reset will cause the CPU to look for its reset vector at the base of Boot Flash (0xFFF0_0000) instead of the EPROM.

## DIP EPROM use

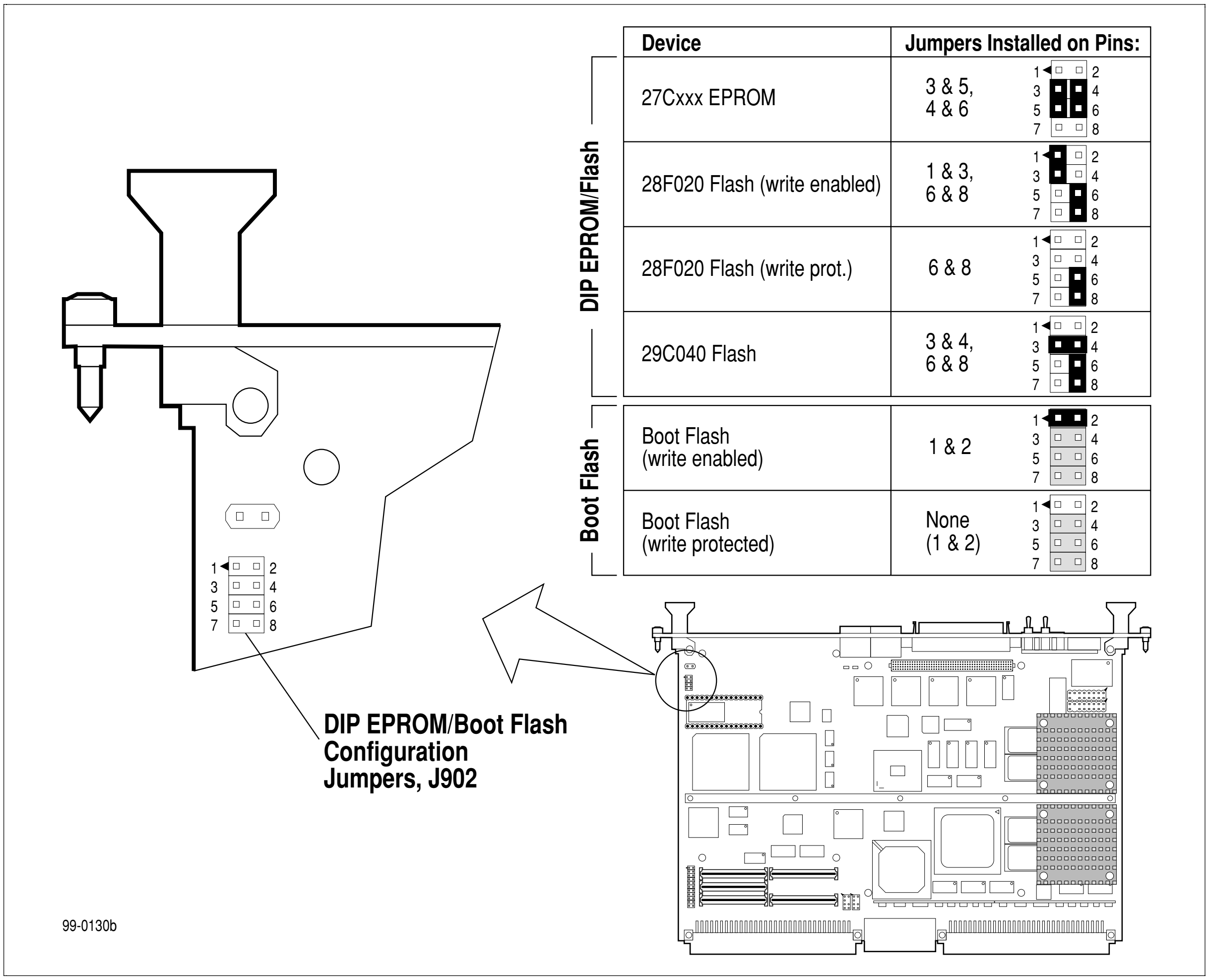
The PowerPC processor fetches its reset vector from default address 0xFFF0_0100. It is possible to use either the DIP EPROM or the soldered-down Boot Flash to contain the reset code. Since the reset address is fixed, the address mapping of the two boot devices must be changed to change the boot device. Consequently, the DIP EPROM appears in one of two address space locations depending on whether the board is configured to boot from DIP EPROM or Boot Flash:

- If the board is configured to boot from EPROM, the EPROM is accessed in the range 0xFFF0_0000 – 0xFFF7_FFFF.
- If the board is configured to boot from Boot Flash, the EPROM is accessed in the range 0xFFE0_0000 – 0xFFE7_FFFF.

Refer to the ***Address Map*** chapter in this section.

### EPROM type configuration

Jumper J902 is used to configure the type device used in the DIP EPROM socket. Set the jumpers as required for your application as shown in the drawing below. See also ***Installing a monitor PROM*** in Section 2, page 29.



| | Device | Jumpers Installed on Pins: |
|---|---|---|
| DIP EPROM/Flash | 27Cxxx EPROM | 3 & 5,<br>4 & 6 |
| | 28F020 Flash (write enabled) | 1 & 3,<br>6 & 8 |
| | 28F020 Flash (write prot.) | 6 & 8 |
| | 29C040 Flash | 3 & 4,<br>6 & 8 |
| Boot Flash | Boot Flash<br>(write enabled) | 1 & 2 |
| | Boot Flash<br>(write protected) | None<br>(1 & 2) |

*DIP EPROM/Boot Flash configuration jumpers, J902*

## Boot Flash use

Boot Flash memory is made up of 1 ea. 8-bit Flash memory chip with 1 MB total space.

***Boot Flash device***

| Onboard Flash size | Manufacturer | Part Number | Organization |
|---|---|---|---|
| 1 MB | Intel | 28F008SA | 1 MB x 8 |

The 1 MB space provided by this device is split in half in two different memory locations if the board is configured to boot from DIP EPROM. If the board is configured to boot from Flash, the 1 MB space is contiguous. See table below.

***Boot Flash memory address location***

| Address | Data width | Description |
|---|---|---|
| 0xFFE0_0000 – 0xFFE7_FFFF | D8/D64 | Boot Flash (lower 512 KB, ROMBoot) |
| 0xFFF8_0000 – 0xFFFF_FFFF | D8/D64 | Boot Flash (upper 512 KB, ROMBoot) |
| 0xFFF0_0000 – 0xFFFF_FFFF | D8/D64 | Boot Flash (Flash Boot) |

## *Block organization*

For convenience in programming or erasing, the block information for the Boot Flash memory is listed in the table below.

***Boot Flash memory block information***

| Flash Memory Size | Total Blocks | Block Size | Block Numbers |
|---|---|---|---|
| 1 MB | 16 | 64 KB | 0-15 |

Note that full Flash support is supplied in Synergy's SMon Application Developer and Debugger package. Example Flash driver code is also available from Synergy. Contact Customer Service for details and ordering information.

## *Writing and erasing*

Write protection of all Flash (Boot Flash, User Flash, and DIP Flash EPROM [if installed]) is set in either or both of two ways:

**Jumper Flash Write Protect (J02N pins 21 & 22)** — write protect if jumper ON; no write protect if jumper OFF. Refer to the ***Setting up the VGM5 hardware*** chapter in Section 2, page 25, for more information on the configuration jumpers.

**Software Flash Write Protect (Flash ROM register at 0xFFEF_FE40)** — bit 0 controls protect (1, default) and no protect (0) status. Refer to the *Flash ROM register* discussion in Section 4, page 132, for more information on the Flash ROM register.

To enable writing to Flash memory, remove the Flash Write Protect jumper (J02N pins 21 & 22) ***and*** clear bit 0 of the Flash ROM register at 0xFFEF_FE40.

To protect Flash from writes, install the Flash Write Protect jumper (J02N pins 21 & 22) or set bit 0 of the Flash ROM register at 0xFFEF_FE40, or take both actions if desired.

### Additional write protection of Boot Flash

VGM5 Revision F with ECO 'b' (or higher) incorporates a retrofit of a Boot Flash Write Protect jumper on J902 pins 1 and 2. This jumper provides Boot Flash with additional write protection for data security during board startup. Refer to the J902 jumper diagram on page 162.

- To **write protect** Boot Flash, remove jumper from J902 pins 1 & 2. With this jumper removed, Boot Flash has unconditional write protection regardless of the J02N jumper and/or Flash ROM register (0xFFEF_FE40) Flash write enable configuration.
- To **write enable** Boot Flash, install jumper J902 pins 1 & 2 ***and*** remove jumper (if installed) from J02N pins 21 & 22.

> Note
>
> On other VGM5 revisions, this jumper has no function other than to serve as a place to store a jumper when the 28F020 DIP Flash is write protected. However, future ECOs may be issued to retrofit other VGM5 board revisions for this jumper function. Contact Customer Service for assistance in determining whether or not your board has this added jumper function.

### Additional Flash memory information

The Flash memory chips have embedded byte write and block/sector erase algorithms. For more information on the chip itself and on the software aspects of writing/erasing Flash memory, refer to the Intel Flash memory databook.

- Intel Flash Memory Databook (Order no. 210830)

  For ordering information, contact:

  **Intel Literature Sales**
  **P.O. Box 7641**
  **Mt. Prospect, IL 60056-7641**
  ***In U.S. and Canada, call toll free: (800) 548-4725***

  For general technical information via the Web:

  **Intel's Developer site:**

  **http://developer.intel.com/**

- Datasheets for the Flash parts are available in PDF (Adobe Acrobat) from the Intel's Developer site:

  **http://developer.intel.com/design/flash/datashts/index.htm**

# User Flash memory

## Introduction

Rev. E or lower VGM5 boards provide **2/4/8 MB** of onboard user Flash memory as an option. Rev. F or higher VGM5 boards provide **4/8/16/32/64 MB** of onboard user Flash memory as an option.

The memory chips have a byte write and block erase architecture with data storage similar to that of a sectored hard disk. A typical use of User Flash includes operation as a RAM disk for loading an operating system kernel or accessing other files as needed by the system.

**User Flash memory address location**

| Address | Data width | Description |
|---|---|---|
| 0xFFF8_0000 – 0xFFFF_FFFF | D8 | User Flash, 512 KB bank |

User Flash memory is made up of 1, 2 or 4 ea. Flash memory chips used in byte-wide mode. These devices are connected so that they appear as 8-bit devices. The table below lists the devices used for the VGM5's User Flash memory.

**User Flash device**

| Onboard Flash size | Manufacturer | Part Number | Organization |
|---|---|---|---|
| 2/4 MB[1] | Intel | 28F016SA/28F160S5[2] | 2 MB x 8 |
| 4/8 MB | Intel | 28F320J5 | 4 MB x 8 |
| 16/32/64 MB | Intel | 28F128J3A | 16 MB x 8 |

* Note: 1. Rev. E or lower boards only.

2. The 28F160S5 Flash device is an alternate part for the 28F016SA device which is being phased out by the manufacturer.

## Block organization

For convenience in programming or erasing, the block information for the onboard Flash memory is listed in the table below.

***User Flash memory block information***

| Flash Memory Size | Total Blocks | Block Size | Block Numbers |
|---|---|---|---|
| 2/4 MB | 32 | 64 KB | 0-31 |
| 8 MB | 64 | 64 KB | 0-63 |
| 16 MB | 128 | 64 KB | 0-127 |
| 32 MB | 256 | 64 KB | 0-255 |
| 64 MB | 512 | 64 KB | 0-511 |

Note that full Flash support is supplied in Synergy's SMon Application Developer and Debugger package. Example Flash driver code is also available from Synergy. Contact Customer Service for details and ordering information.

## Bank selection

The MPC106 provides only 2 Megabytes of address space for all 8-bit ROM devices. Because of this limitation and the need to incorporate three different ROM devices plus onboard control/status registers into this space, the User Flash ROM must be accessed piecemeal.

User Flash is accessed in multiple 512 KB banks using the onboard Flash Window register at 0xFFEF_FE50. Refer to the ***Onboard registers*** chapter in this section. The total number of banks depends on the amount of User Flash installed on the board. See table below.

With Boot Flash selected (reg. 0xFFEF_FE50, bit 7 = 0), User Flash is inactive. With User Flash selected (reg. 0xFFEF_FE50, bit 7 = 1), User Flash is active and a 512 KB window of User Flash space can be selected. Note that accesses to User Flash and the upper 512 KB of Boot Flash are not allowed to occur at the same time.

Only 512 KB of User Flash is visible at any time in the User Flash window, and this 512 KB window is shared with the upper 512 KB of the Boot Flash. The result of this limit is that code stored in the User Flash must be read into RAM and executed from RAM instead of being directly executed from Flash ROM.

### User Flash bank selection

| Flash device | Address within device | Register Value | User Flash Amount | | | |
|---|---|---|---|---|---|---|
| Boot | 0x0008_0000 – 0x000F_FFFF | 0x00 | | | | |
| User | 0x0000_0000 – 0x0007_FFFF | 0x80 | ▨ | ▨ | ▨ | ▨ |
| User | 0x0008_0000 – 0x000F_FFFF | 0x81 | ▨ | ▨ | ▨ | ▨ |
| User | 0x0010_0000 – 0x0017_FFFF | 0x82 | **2 MB** | ▨ | ▨ | ▨ |
| User | 0x0018_0000 – 0x001F_FFFF | 0x83 | ▨ | ▨ | ▨ | ▨ |
| User | 0x0020_0000 – 0x0027_FFFF | 0x84 | | **4 MB** | ▨ | ▨ |
| User | 0x0028_0000 – 0x002F_FFFF | 0x85 | | ▨ | ▨ | ▨ |
| User | 0x0030_0000 – 0x0037_FFFF | 0x86 | | ▨ | ▨ | ▨ |
| User | 0x0038_0000 – 0x003F_FFFF | 0x87 | | ▨ | ▨ | ▨ |
| User | 0x0040_0000 – 0x0047_FFFF | 0x88 | | | **8 MB** | ▨ |
| User | 0x0048_0000 – 0x004F_FFFF | 0x89 | | | ▨ | ▨ |
| User | 0x0050_0000 – 0x0057_FFFF | 0x8A | | | ▨ | ▨ |
| User | 0x0058_0000 – 0x005F_FFFF | 0x8B | | | ▨ | ▨ |
| User | 0x0060_0000 – 0x0067_FFFF | 0x8C | | | ▨ | ▨ |
| User | 0x0068_0000 – 0x006F_FFFF | 0x8D | | | ▨ | ▨ |
| User | 0x0070_0000 – 0x0077_FFFF | 0x8E | | | ▨ | ▨ |
| User | 0x0078_0000 – 0x007F_FFFF | 0x8F | | | ▨ | ▨ |
| User | 0x0080_0000 – 0x0087_FFFF | 0x90 | | | | **16 MB** |
| User | 0x0088_0000 – 0x008F_FFFF | 0x91 | | | | ▨ |
| User | 0x0090_0000 – 0x0097_FFFF | 0x92 | | | | ▨ |
| User | 0x0098_0000 – 0x009F_FFFF | 0x93 | | | | ▨ |
| User | 0x00A0_0000 – 0x00A7_FFFF | 0x94 | | | | ▨ |
| User | 0x00A8_0000 – 0x00AF_FFFF | 0x95 | | | | ▨ |
| User | 0x00B0_0000 – 0x00B7_FFFF | 0x96 | | | | ▨ |
| User | 0x00B8_0000 – 0x00BF_FFFF | 0x97 | | | | ▨ |
| User | 0x00C0_0000 – 0x00C7_FFFF | 0x98 | | | | ▨ |
| User | 0x00C8_0000 – 0x00CF_FFFF | 0x99 | | | | ▨ |
| User | 0x00D0_0000 – 0x00D7_FFFF | 0x9A | | | | ▨ |
| User | 0x00D8_0000 – 0x00DF_FFFF | 0x9B | | | | ▨ |
| User | 0x00E0_0000 – 0x00E7_FFFF | 0x9C | | | | ▨ |
| User | 0x00E8_0000 – 0x00EF_FFFF | 0x9D | | | | ▨ |
| User | 0x00F0_0000 – 0x00F7_FFFF | 0x9E | | | | ▨ |
| User | 0x00F8_0000 – 0x00FF_FFFF | 0x9F | | | | ▨ |
| *...31 additional 512 KB windows....* | | | | | | |
| User | 0x01F8_0000 – 0x01FF_FFFF | 0xBF | End of 32 MB User Flash | | | |
| *...63 additional 512 KB windows....* | | | | | | |
| User | 0x07F8_0000 – 0x07FF_FFFF | 0xEF | End of 64 MB User Flash | | | |

## Writing and erasing

Write protection of all Flash (Boot Flash, User Flash, and DIP Flash EPROM [if installed]) is set in either or both of two ways:

**Jumper Flash Write Protect (J02N pins 21 & 22)** — write protect if jumper ON; no write protect if jumper OFF. Refer to the ***Setting up the VGM5 hardware*** chapter in Section 2, page 25, for more information on the configuration jumpers.

**Software Flash Write Protect (Flash ROM register at 0xFFEF_FE40)** — bit 0 sets protect (1, default) and no protect (0) status. Refer to the *Flash ROM register* discussion in Section 4, page 132, for more information on the Flash ROM register.

To enable writing to Flash memory, **remove** the Flash Write Protect jumper (J02N pins 21 & 22) ***and*** clear bit 0 of the Flash ROM register at 0xFFEF_FE40.

To protect Flash from writes, **install** the Flash Write Protect jumper (J02N pins 21 & 22) or set bit 0 of the Flash ROM register at 0xFFEF_FE40, or take both actions if desired.

## Additional information

The Flash memory chips have embedded byte write and block/sector erase algorithms. For more information on the chip itself and on the software aspects of writing/erasing Flash memory, refer to the Intel Flash memory databook.

- Intel Flash Memory Databook (Order no. 210830)

  For ordering information, contact:

  **Intel Literature Sales**
  **P.O. Box 7641**
  **Mt. Prospect, IL 60056-7641**

  **In U.S. and Canada, call toll free: (800) 548-4725**

  For general Flash information via the Web:

  **Intel Flash memory page**

  **http://www.intel.com/support/flash/memory/**

- Datasheets for the Flash parts are available in PDF format (Adobe Acrobat) from the Intel Corp. website:

  **http://developer.intel.com/design/flash/datashts/index.htm**

# 5 Reset

This section provides VGM5 reset information.

- General description
- PCI reset
- Hard reset sources
- Soft reset

# Reset information

## General description

A reset of the VGM5 causes the board to reset all processors, registers and onboard peripheral devices such as I/O controllers and bridges. If the board is the system controller, the VME SysReset is also asserted to force a system-wide reset.

Once reset, CPU-Y is disabled since the MP_ENABLE bit in the MPC106 is cleared by hardware reset.

CPU-X enters the boot state in which it will execute the code at address 0xFFF0_0100 in either DIP EPROM (512 KB) or Boot Flash (1 MB) depending on the Boot ROM Enable jumper configuration (see Section 2, ***Setting up the VGM5 hardware***, page 25). The code at this address is typically a jump to the cold-start routine, which performs the initialization tasks that ready the board facilities for use.

The above describes a hard reset or cold-start.

The VGM5 is also capable of a warm start in which only the CPU(s) is/are reset using the MPIC Processor Init register. Refer to the ***Soft reset*** discussion below.

## PCI reset

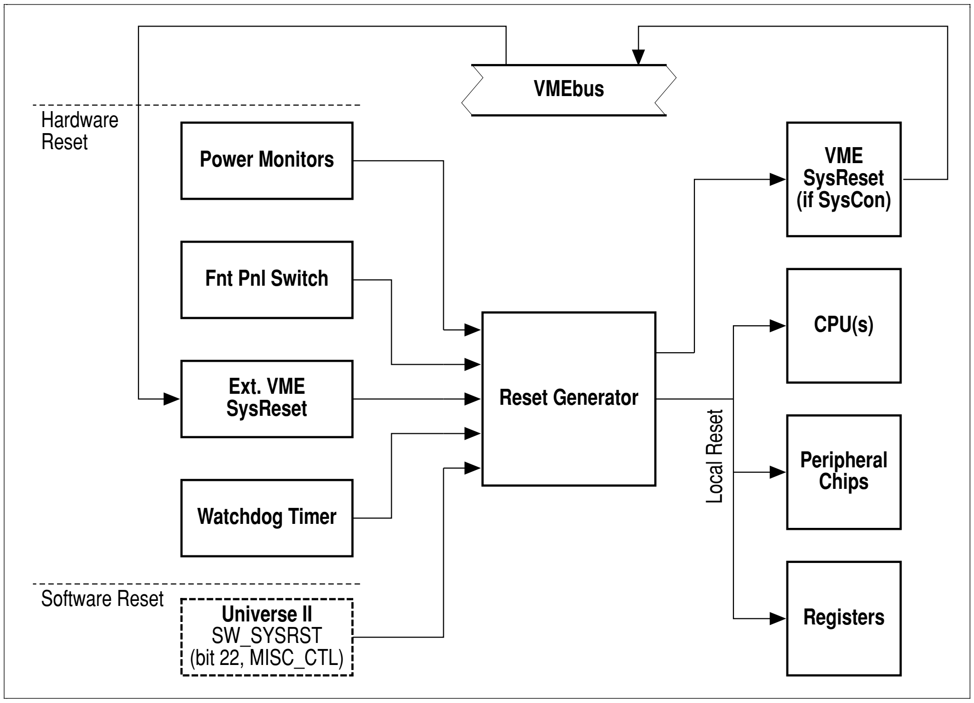
The PCI reset (RST\) line is driven only by a board-level reset. It is not allowed to be driven by any onboard PCI device. On the VGM5, the Universe II PCI reset facility (SW_LRST bit in MISC_CTL register) is not usable to reset the PCI bus.

## Hard reset sources

There are five reset sources, each of which produces a hard reset:

- Power monitors
- Front panel switch
- External VME SysReset signal
- Watchdog timer
- Universe II software SysReset

The functional block diagram below shows the VGM5 reset sources.



*VGM5 hard reset sources*

## Power monitor

The VGM5 power monitor is based on the LTC1536 power monitor and reset generator. The following bus voltages are monitored:

- +5V Vcc supply from backplane, threshold is +4.75V
- 3.3V DC-DC converter output, threshold is +3.15V
- 2.0V CPU core DC-DC converter output, threshold value is dependent on core voltage used (set at factory).

The power monitor ensures that all bus voltages are at a valid level for reliable operation. When powering up, the power monitor holds all devices at reset until all voltages rise past the threshold. If, during the course of normal operation, a voltage level should dip below the threshold, the power monitor triggers a reset. This action avoids the unpredictable nature of operating in a twilight zone.

## Front panel reset switch

The front panel reset switch (see ***Front panel*** in Section 2, page 17) provides the user with a means to manually reset the board or system (if board is system controller). Push the front panel reset switch handle to the right to reset the board/system.

## External VME SysReset

Assertion of the VMEbus SysReset\ signal (P1 pin C12) causes a board/system reset.

Note that there is a bit called SysReset Enable that allows the board to prevent itself from being reset by its own SysReset. This bit is in the VME64 SysReset onboard register at address 0xFFEF_FF38 (see page 138). Its default value is 1 (board responds to its own SysReset). Use this register to program the board to reset only the ***other*** VMEbus boards in the system without resetting itself. To reset the VMEbus without resetting the VGM5:

1. Temporarily disable board's VMEbus reset from resetting CPU by writing 0 to 0xFFEF_FF38, b0.
2. Start VMEbus reset by setting Universe SysReset bit (write 1).
3. Stop VMEbus reset by clearing Universe SysReset bit (write 0).
4. Re-enable board's VMEbus reset to also reset the CPU by writing 1 to 0xFFEF_FF38, b0. This last step is needed to put the system

back into normal operating mode in case it is desired to have other VME boards reset this CPU.

Allow time for the reset signal to settle — add a 0.1 second delay between each of the above operations.

The VME64 SysReset register is readable, which lets the system check its status at anytime by a simple read of the register.

### Watchdog timer

For Rev. E or higher boards, a watchdog timer based on the MPIC Processor 3 registers provides a means for the hardware to automatically reset the board when it is no longer executing code properly.

For more information on setting up the MPIC for watchdog, refer to the ***Watchdog timer*** chapter in Section 6 (page 195).

### Universe II software reset

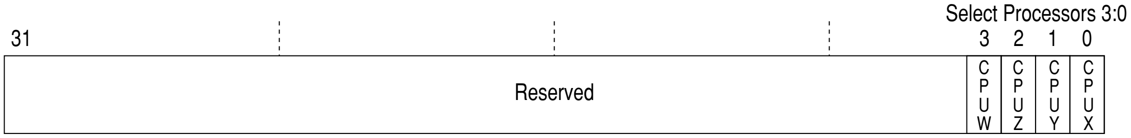
Sotware can initiate a board reset by setting bit 22 of the Universe II PCI-VME64 bridge Miscellaneous Control register (MISC_CTL). If the board is the system controller, the software reset will also reset the system. Some OS board support packages from Synergy include a facility for software reset via the Universe II. The VxWorks BSP, for example, includes the `sysReset()` function.

## Soft reset

The CPUs may be independently reset by means of the Soft Reset feature. The MPIC interrupt controller generates a signal called Init\ to each CPU on the board. The two signals Init0\ and Init1\ are wired to the SReset\ inputs of CPUs X and Y respectively.

The SReset\ function in the CPU is edge-triggered. This means that when a CPU's Init bit in the MPIC is changed from 0 to 1, the CPU will be reset. The Init bit must be reset to 0 before another soft reset may be performed.

The Processor Init register is a 32-bit read/write register of which the least 4 significant bits are defined. Bit 0 (on right) is CPU-X Init, bit 1 is CPU-Y, etc.



To soft reset a particular CPU the appropriate bit(s) in the Processor Init register is/are cleared to 0 then set to 1. This is done by masking the desired bit(s) off and writing the data word back to the register [bit(s) cleared], then ORing the bit(s) with the data word and writing that back to the register [bit(s) set]. This initiates a soft reset of the corresponding processor(s) while the register bits of unselected processors remain unchanged. A simplified form of this procedure in ‘C’ is shown below:

```
reg & = ~bit;        // Clear bit
reg | = bit;         // Set bit
```

Where `reg` is the MPIC Processor Init register and `bit` is the value selecting one or more CPUs (1, 2, 4, 8 corresponds to CPU-X, CPU-Y, CPU-Z & CPU-W). Though CPU-Z and CPU-W aren’t present on the VGM5, the associated bits are provided for upward compatibility.

Refer to Section 6 ***MPIC Interrupt Controller*** (page 177) for more on the MPIC.

# 6 MPIC Interrupt Controller

This section provides information about the PowerPC multiprocessor interrupt controller (MPIC).

- General description
- MPIC registers
- Watchdog timer

Note: The bit numbering of registers in this section follows the zero-on-the-right convention as opposed to the zero-on-the-left bit numbering convention used by Motorola and IBM in their PowerPC documentation.

# General description

Interrupt control on the VGM5 board is provided by IBM's MPIC (multiprocessor interrupt controller) chip which is specifically designed for PowerPC systems. The chip provides interrupt management for board devices and the processors themselves.

The MPIC chip supports:

- 16 I/O device interrupts
- Up to 4 processors (0–3)
- 4 interprocessor interrupts
- 4 global timers

# MPIC registers

## MPIC base address

The operating system PCI discovery routines dynamically set up the MPIC base address by writing to the appropriate registers in PCI configuration address space. MPIC's PCI configuration header is shown below.

| 31 16 | 15 0 | |
|---|---|---|
| Device ID<br>0x0046 | Vendor ID<br>0x1014 | 0x00 |
| Status 0x0200 | Command 0x0000* | 0x04 |
| Reserved | | 0x08 |
| Reserved | | 0x0C |
| Base Address Register 0x0000_0000 | | 0x10 |

*Note: Set PCI command register bit 1 to 1 to enable PCI Memory Address space access. MPIC does not support I/O space access.

The MPIC chip operates only in PCI Memory Space (operation in PCI I/O Space not supported).

For more information on PCI configuration, refer to the ***PCI implementation details*** discussion in the **PCI bus** chapter in Section 3, page 62, and the ***Setting PCI device base address*** discussion in Section 4, page 103. Also refer to the ***Type 0 configuration table*** on page 110.

*Overall Address map, MPIC*

| Address Offset* | Register |
|---|---|
| 0x0_0000 | Reserved |
| 0x0_1000 | Global Registers |
| 0x1_0000 | Interrupt Source Configuration Registers |
| 0x2_0000 | Processor 0 Per Processor Registers |
| 0x2_1000 | Processor 1 Per Processor Registers |
| 0x2_2000 | Processor 2 Per Processor Registers |
| 0x2_3300 | Processor 3 Per Processor Registers |
| 0x3_F000 | Reserved |

* Note: Base address automatically set by PCI Enumeration.

The following paragraphs describe the three major MPIC register groups:

- Global registers
- Interrupt Source Configuration registers
- Per Processor registers

## Global registers

The table below lists the address map locations of the global registers.

*Address map, global registers*

| Address Offset* | Register | Access |
|---|---|---|
| 0x0_1000 | Feature Reporting register | RO |
| 0x0_1020 | Global Configuration register | R/W |
| 0x0_1080 | Vendor Identification register | RO |
| 0x0_1090 | Processor Init register | R/W |
| 0x0_1000–0x0_10A0 | IPI Vector/Priority registers | R/W |
| 0x0_10E0 | Spurious Vector register | R/W |
| 0x0_10F0–0x0_11F0 | Global Timer registers | — |

* Note: Base address automatically set by PCI Enumeration.

### Feature reporting register

| 31 27 | 26 16 | 15 13 | 12 8 | 7 0 |
|---|---|---|---|---|
| Reserved | Num IRQ Sources | Reserved | Num CPU | Version ID |

**Version ID**: Version ID for this interrupt controller. This value reports what level of the OpenPIC specification is supported by this implementation, **1** = Spec. Revision 1.0; **2** = Spec. Revision 1.2

**Num CPU**: The number of the highest physical CPU supported. For a 4-processor MPIC chip, this value is **3**; for a 2-processor MPIC chip, this value is **1**; for a 1-processor EPIC chip, this value is **0**.

**Num IRQ**: The number of the highest IRQ source supported. For example, in a system with 16 I/O interrupt sources, this value is 15.

### Global configuration registers

| 31 | 30 | 29 | 28 20 | 19 0 |
|---|---|---|---|---|
| R |  | M | Reserved | Base (not used in PowerPC-based systems) |

**Base**: Base Address Relocation field. This field is not used in Power PC-based systems.

**M**: Cascade mode. Set this bit to **1** to enable the MPIC. This bit is provided to support an 8259 interrupt controller, which is not used on the VGM5.

When set to **0** (reset default), the MPIC passes interrupt input 0 directly through to CPU-X, which disables all other interrupt encoding and steering operations. This effectively disables the MPIC.

When set to 1, the MPIC processes all interrupt inputs normally.

**R**: Reset controller. Writing a one to this bit forces the controller logic to be reset. This bit is cleared automatically when the reset sequence is complete. While this bit is set, the values of all other registers are undefined.

## Vendor identification register

| 31 24 | 23 16 | 15 8 | 7 0 |
|---|---|---|---|
| Reserved | Stepping | Device ID | Vendor ID |

**Vendor ID**: Specifies the manufacturer of this part. For this part, the value is **0x14**.

**Device ID**: Vendor specified indentifier for this device. Value for MPIC-2A part is **0x46**.

**Stepping**: Stepping (silicon revision) for this device. Initially **0**.

## Processor init register

|  | Select Processors 3:0 | | | |
|---|---|---|---|---|
| 31 | 3 | 2 | 1 | 0 |
| Reserved | CPUW | CPUZ | CPUY | CPUX |

Writing to this register (b3–b0) causes the INIT lines(s) to one or more processors to be activated. Writing a one to a bit activates the corresponding INIT line. Writing a zero to a bit deactivates the corresponding INIT line. (The INIT lines(s) are connected to the Soft Reset pin(s) on PowerPC processors. The Soft Reset input on a PowerPC processor is normally edge triggered.)

The Processor Init Register may be used to perform a soft reset of any or all CPUs on the VGM5. To issue a soft reset to a CPU, write a 0 to the bit corresponding to the CPU to be reset, then write a 1 to that bit. An eieio instruction between the two writes is recommended to enforce proper sequencing of the hardware write cycles.

Beware that a soft reset performed inside an interrupt service routine will leave the MPIC's internal interrupt-under-service bit set for that interrupt, which will prevent future interrupts from that source from being serviced. This interrupt-under-service bit may be cleared by writing a zero to the End Of Interrupt register for the processor servicing the interrupt.

For more information on VGM5 reset, refer to Section 5, ***Reset*** (page 169).

## IPI vector/priority registers

There are 4 IPI vector/priority registers, one for each IPI dispatch register. The IPI vector/priority register bit assigments are described below:

**Address map, IPI vector/priority registers**

| Offset* | Register | Access |
|---|---|---|
| 0x010A0 | IPI 0 Vector/Priority register | R/W |
| 0x010B0 | IPI 1 Vector/Priority register | R/W |
| 0x010C0 | IPI 2 Vector/Priority register | R/W |
| 0x010D0 | IPI 3 Vector/Priority register | R/W |

* Note: Base address automatically set by PCI Enumeration.

| 31 | 30 | 29 20 | 19 16 | 15 8 | 7 0 |
|---|---|---|---|---|---|
| M<br>S<br>K | A<br>C<br>T | Reserved | Priority | Reserved for Vector Expansion | Vector |

**Vector**: Interrupt vector. The vector value in this field is returned when the Interrupt Acknowledge register is examined and the interrupt associated with this vector is requested.

**Priority**: Interrupt priority. This field sets the interrupt priority. The lowest priority is 0 and the highest is 15. Setting the priority level to 0 disables interrupts.

**ACT**: Activity bit, read only. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to 1 when its associated bit in the Interrupt Pending or In-Service register is set. Note that this bit is read only. The Vector, Priority, and Sense values should not be changed while the ACT bit is 1.

**MSK**: Mask bit. Setting this bit to 1 disables any further interrupts from this source. If the mask bit is cleared while the bit associated with this interrupt is set in the IPR, the interrupt request will be generated. This bit is always set to 1 following a reset.

## Spurious vector register

| 31 — 16 | 15 — 8 | 7 — 0 |
|---|---|---|
| Reserved | Reserved for<br>Vector Expansion | Vector |

The MPIC responds to a spurious interrupt by presenting the CPU with the vector stored in this register when it is unable to determine the vector for the interrupt under service. This will occur if the interrupt request is negated by the source before the vector read cycle is performed by that CPU.

Only one spurious interrupt register exists, so it will be used only for interrupts directed to the CPU to which it is steered.

Reset sets this register to 0x0000_00FF.

## Global Timer registers

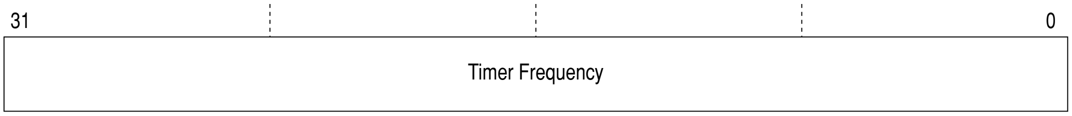
MPIC contains four global timers (0–3) suitable for system timing and periodic interrupt generation. The four timers share a Timer Frequency Reporting register. Each timer has a set of 4 registers for configuration and control and each is readable on-the-fly.

**Address map, global timer registers**

| Address Offset* | Register | Access |
|---|---|---|
| 0x010F0 | Timer Frequency Reporting register | R/W |
| 0x01100 | Global Timer 0 Current Count | RO |
| 0x01110 | Global Timer 0 Base Count | R/W |
| 0x01120 | Global Timer 0 Vector/Priority register | R/W |
| 0x01130 | Global Timer 0 Destination register | R/W |
| 0x01140 | Global Timer 1 Current Count | RO |
| 0x01150 | Global Timer 1 Base Count | R/W |
| 0x01160 | Global Timer 1 Vector/Priority register | R/W |
| 0x01170 | Global Timer 1 Destination register | R/W |
| 0x01180 | Global Timer 2 Current Count | RO |
| 0x01190 | Global Timer 2 Base Count | R/W |
| 0x011A0 | Global Timer 2 Vector/Priority register | R/W |
| 0x011B0 | Global Timer 2 Destination register | R/W |
| 0x011C0 | Global Timer 3 Current Count | RO |
| 0x011D0 | Global Timer 3 Base Count | R/W |
| 0x011E0 | Global Timer 3 Vector/Priority register | R/W |
| 0x011F0 | Global Timer 3 Destination register | R/W |

* Note: Base address automatically set by PCI Enumeration.

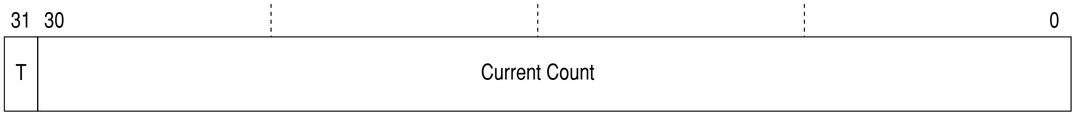
### Timer frequency reporting register



**Timer Frequency**: This register is used to report the frequency (in Hz) of the clock source for the global timers. The global timers' clock source is the MPIC input clock frequency divided by 8.

This register contains zero after a Reset. System initialization code should initialize this register to one-eighth the MPIC clock frequency once the MPIC clock frequency (VGM5's 33 MHz PCI clock) has been determined. A typical value for this register would be 0x003F940B indicating each timer is updated once every 240nS.

### Global Timer Current Count register

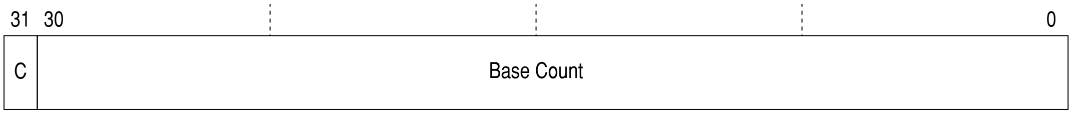


**Current Count**: The current count is loaded with the base count, and the toggle bit is cleared, whenever the Base Count register is written and the Count Inhibit bit in the Base Count register transitions from 1 to 0. The timer decrements while the Count Inhibit bit in the Base Count register is zero. When the time counts down to zero, an interrupt is generated, the toggle bit is inverted and the current count is reloaded from the base count.

Following Reset, counting is disabled and the current count register contains zero.

**T**: Toggle. This bit toggles whenever the current count decrements to zero.

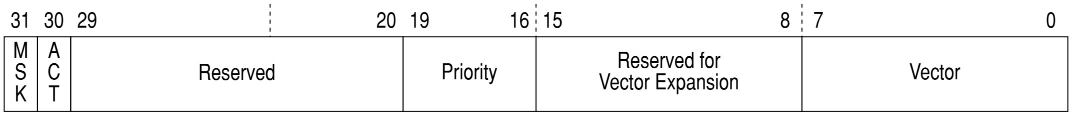
### Global Timer Base Count register



**Base Count**: This field contains the 31-bit base count for this timer. The current count is loaded with the base count, and the toggle bit in the Current Count register is cleared, whenever the Base Count register is written and the Count Inhibit bit transitions from 1 to a 0. The timer decrements while the Count Inhibit bit is zero. When the timer counts down to zero, an interrupt is generated, the toggle bit in the Current Count register is inverted and the current count is reloaded from the base count.

**C**: Count Inhibit bit; 1 = inhibit counting for this timer, 0 = proceed with counting. Reset sets this bit to one (inhibit counting).

### *Global Timer Vector/Priority register*



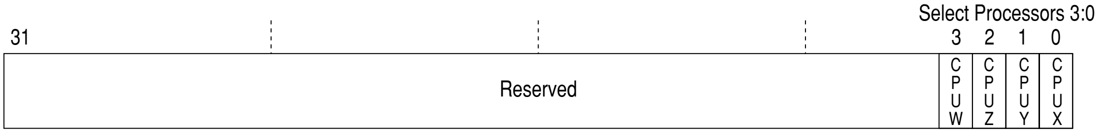
**Vector**: Interrupt Vector. The vector value in this field is returned when the Interrupt Acknowledge register is examined and the interrupt associated with this vector is requested.

**Priority**: Interrupt Priority. This field sets the interrupt priority. The lowest priority is 0 and the highest is 15. Setting the priority level to 0 disables interrupts.

**ACT**: Activity bit, read only. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to 1 when its associated bit in the Interrupt Pending or In-Service register is set. Note that this bit is READ ONLY. The vector and priority values should not be changed while the ACT bit is 1.

**MSK**: Mask bit. Setting this bit disables any further interrupts from this source. Reset sets this bit to one.

### *Global Timer Destination register*



This register indicates the destination(s) for this timer's interrupts. Timer interrupts operate only in Directed Delivery mode. This register may specify multiple destinations (multicast delivery).

**CPU_**: Setting the appropriate bit(s) (b3-b0) directs the timer interrupt to the corresponding processor(s).

## Interrupt source configuration registers

The table below lists the address map locations of the interrupt source configuration registers.

**Address map, interrupt source configuration registers**

| Address Offset* | Register | Access |
|---|---|---|
| 0x1_0000 | Interrupt Source 0 Vector/Priority register | R/W |
| 0x1_0010 | Interrupt Source 0 Destination register | R/W |
| \| | \| | \| |
| 0x1_01E0 | Interrupt Source 15 Vector/Priority register | R/W |
| 0x1_01F0 | Interrupt Source 15 Destination register | R/W |

* Note: Base address automatically set by PCI Enumeration.

Each interrupt source has an associated vector/priority register and a destination register. There are 16 sets of these. The vector/priority register sets up how an interrupt source is detected, its priority, and its vector address. The destination register routes (or steers) the interrupt source to one or more onboard processors.

The table below lists the VGM5 interrupt sources (active low trigger mode used for all).

**VGM5 interrupt sources**

| Source | Owner | Notes |
|---|---|---|
| 0 | PCI IntD (Universe II Int 3) | Note[1] |
| 1 | PCI–VME Bridge (Universe II) Int 4 | — |
| 2 | PCI IntB (Universe II Int 1) | Note[2] |
| 3 | PCI IntA (Universe Int 0) | Note[3] |
| 4 | PCI PErr | — |
| 5 | PCI SErr | — |
| 6 | PCI-PCI bridge (Po•PCI™), VGM5 | — |
| 7 | unused | — |
| 8 | PCI–VME Bridge (Universe II) Int 5 | — |
| 9 | PCI IntC (Universe II Int 2) | Note[4] |
| 10 | Ethernet | — |
| 11 | SCSI | — |
| 12 | Mailbox 1 (CPU-Y) | — |
| 13 | Mailbox 0 (CPU-X) | — |
| 14 | Serial Port B | — |
| 15 | Serial Port A | — |

**Notes:** 1. PCI Int D can be driven by any of the following:
a. Universe Int 3
b. PMC Int D
c. PEX3 Int D
2. PCI Int B can be driven by any of the following:
a. Universe Int 1
b. PMC Int B
c. PEX3 Int B
3. PCI Int A can be driven by any of the following:
a. Universe Int 0
b. PMC Int A
c. PEX3 Int A
4. PCI Int C can be driven by any of the following:
a. Universe Int 2
b. PMC Int C
c. PEX3 Int C

## *Interrupt Source Vector/Priority registers*

The vector/priority register bit assignments are described below.

| 31 | 30 | 29 … 24 | 23 | 22 | 21 | 20 | 19 … 16 | 15 … 8 | 7 … 0 |
|---|---|---|---|---|---|---|---|---|---|
| MSK | ACT | Reserved | POL | S | Rsv | Rsv | Priority | Reserved for Vector Expansion | Vector |

**Vector**: Interrupt Vector. The vector value in this field is returned when the Interrupt Acknowledge register is examined and the interrupt associated with this vector is requested.

**Priority**: Interrupt Priority. This field sets the interrupt priority. The lowest priority is 0 and the highest is 15. Setting the priority level to 0 disables interrupts.

**S**: Sense. This bit sets the sense for external interrupts. Setting this bit to 0 enables edge sensitive interrupts. Setting this bit to 1 enables level sensitive interrupts.

**POL**: Polarity. This bit sets the polarity for external interrupts. Setting this bit to 0 enables active low or negative edge. Setting this bit to 1 enables active high or positive edge.

For Synergy PowerPC series boards, all I/O interrupt sources (0–15) are set up for Active Low polarity and Level Triggered sense.

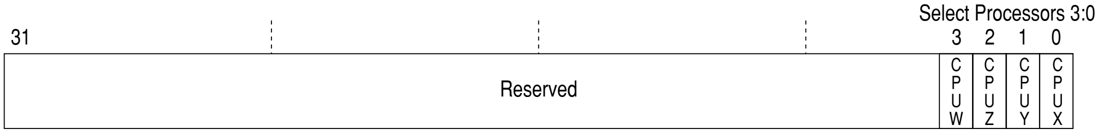
**ACT**: Activity bit, read only. The activity bit indicates that an interrupt has been requested or that it is in-service. The ACT bit is set to 1 when its associated bit in the Interrupt Pending or In-Service register is set. Note that this bit is READ ONLY. The vector and priority values should not be changed while the ACT bit is 1.

**MSK**: Mask bit. Setting this bit disables any further interrupts from this source. Reset sets this bit to one.

Unless otherwise specified, reset sets all bits in all program accessible registers to zero.

### Interrupt Source Destination register

The destination register bit assignments are described below.



This register indicates the destination processors(s) for this interrupt source.

**CPU_**: Setting the appropriate bit(s) (b3-b0) directs the interrupt source to the corresponding processor(s).

If a single destination processor is selected (directed delivery mode) then interrupts from this source are directed to that processor. If multiple destination processors are selected (distributed delivery mode) then interrupts from this source are distributed among the selected destination processors using a fair, implementation specific algorithm.

## Per processor registers

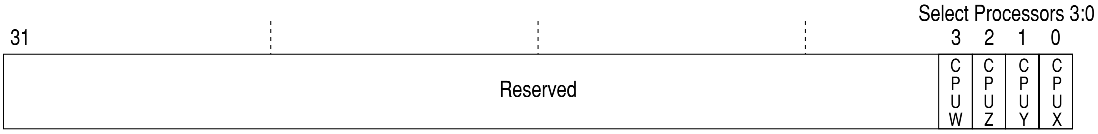
For each processor supported, MPIC provides the following registers:

- Interprocessor interrupt dispatch register
- Current task priority register
- Interrupt request register (implementation-specific, non-program accessible)
- Interrupt acknowledge register
- In-service registers (implementation-specific, non-program accessible)
- End-of-interrupt register

The following describes the bit assignments of the Per Processor registers. Not included are the implementation-specific, non-program accessible registers: interrupt request and in-service registers. For these registers, refer to the MPIC data manual for more information.

### Interprocessor Interrupt Dispatch registers

There are 4 interprocessor interrupt (IPI) dispatch registers 0-3 per processor. Writing to an IPI dispatch register causes an interprocessor interrupt request to be sent to one or more processors. A processor is interrupted if the bit in the IPI dispatch register corresponding to that processor is set during the write. Reading these registers returns zeros.



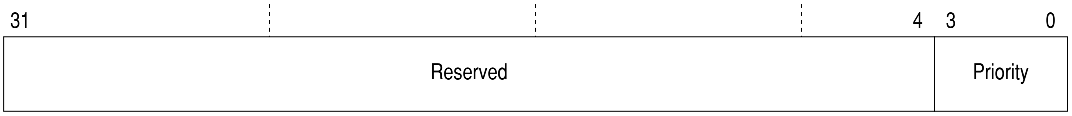
**Address map, interprocessor interrupt dispatch registers**

| Address Offset* | Register | Access |
|---|---|---|
| 0x2_0040 | IPI 0 dispatch register, processor 0 | R/W |
| 0x2_0050 | IPI 1 dispatch register, processor 0 | R/W |
| 0x2_0060 | IPI 2 dispatch register, processor 0 | R/W |
| 0x2_0070 | IPI 3 dispatch register, processor 0 | R/W |
| 0x2_1040 | IPI 0 dispatch register, processor 1 | R/W |
| 0x2_1050 | IPI 1 dispatch register, processor 1 | R/W |
| 0x2_1060 | IPI 2 dispatch register, processor 1 | R/W |
| 0x2_1070 | IPI 3 dispatch register, processor 1 | R/W |
| 0x2_2040 | IPI 0 dispatch register, processor 2 | R/W |
| 0x2_2050 | IPI 1 dispatch register, processor 2 | R/W |
| 0x2_2060 | IPI 2 dispatch register, processor 2 | R/W |
| 0x2_2070 | IPI 3 dispatch register, processor 2 | R/W |
| 0x2_3040 | IPI 0 dispatch register, processor 3 | R/W |
| 0x2_3050 | IPI 1 dispatch register, processor 3 | R/W |
| 0x2_3060 | IPI 2 dispatch register, processor 3 | R/W |
| 0x2_3070 | IPI 3 dispatch register, processor 3 | R/W |

* Note: Base address automatically set by PCI Enumeration.

## Current Task Priority register

Each processor has a Current Task Priority register.



**Priority**: Task Priority. Set from 0 (lowest) to 15 (highest). Setting the Task Priority register to 15 masks all interrupts to this processor. At reset, hardware sets the Task Priority registers to 0xF.

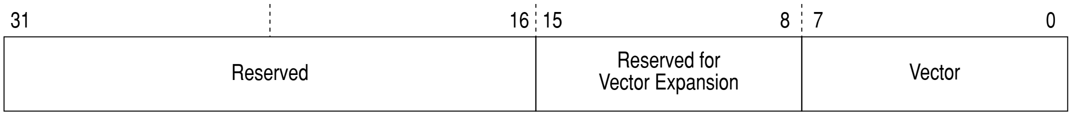
Do not use the Task Priority register to temporarily disable interrupts to a processor. Doing so may result in a spurious interrupt being generated if an interrupt is requested just as the Task Priority register is set to disable interrupts.

***Address map, current task priority registers***

| Address Offset* | Register | Access |
|---|---|---|
| 0x2_0080 | Task Priority register, processor 0 | R/W |
| 0x2_1080 | Task Priority register, processor 1 | R/W |
| 0x2_2080 | Task Priority register, processor 2 | R/W |
| 0x2_3080 | Task Priority register, processor 3 | R/W |

* Note: Base address automatically set by PCI Enumeration.

## Interrupt Acknowledge registers



On PowerPC based systems, Interrupt Acknowledge is implemented as a read request to a memory-mapped Interrupt Acknowledge register. There is one Interrupt Acknowledge register per processor. Interrupt Acknowledge:

- returns the interrupt vector corresponding to the highest priority pending interrupt in that processor's Interrupt Request Register.
- transfers the highest priority pending interrupt from that processor's IRR to that processor's In-Service register.
- clears the bit in the Interrupt Pending Register corresponding to the highest priority pending interrupt in that processor's IRR. Note: This is effective only for edge triggered interrupts. Level

triggered interrupts normally cause the bit in the IPR to be set to 1 every cycle until the device driver’s interrupt service routine has cleared the interrupt at the source.

In some implementations, Interrupt Acknowledge also flushes data buffers between the device and system memory.

***Address map, interrupt acknowledge registers***

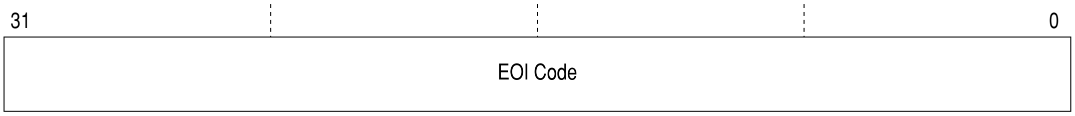
| Address Offset* | Register | Access |
|---|---|---|
| 0x2_00A0 | Int. Acknowledge register, processor 0 | RO |
| 0x2_10A0 | Int. Acknowledge register, processor 1 | RO |
| 0x2_20A0 | Int. Acknowledge register, processor 2 | RO |
| 0x2_30A0 | Int. Acknowledge register, processor 3 | RO |

* Note: Base address automatically set by PCI Enumeration.

## End-of-interrupt registers

There is one End-of-interrupt (EOI) register per processor. Writing a zero to this register signals the end of processing for the highest priority interrupt currently in service by the associated processor. EOI Code values other than 0 are currently undefined and should not be used. Reading this register returns the last value written.

MPIC Implementation Note: When the EOI register is written, the highest priority interrupt in the In-Service Priority register is reset along with the corresponding bit in the Interrupt Source In-Service register. MPIC-2A requires the EOI code be written as zero to signal the end of processing for the highest priority interrupt currently in service by the associated processor; EOI code values other than zero are ignored. The MPIC-2A returns zero on reads.



***Address map, end-of-interrupt registers***

| Address Offset* | Register | Access |
|---|---|---|
| 0x2_00B0 | End-of-interrupt register, processor 0 | RO |
| 0x2_10B0 | End-of-interrupt register, processor 1 | RO |
| 0x2_20B0 | End-of-interrupt  register, processor 2 | RO |
| 0x2_30B0 | End-of-interrupt register, processor 3 | RO |

* Note: Base address automatically set by PCI Enumeration.

# Watchdog timer

Revision E or higher VGM5 boards use MPIC's Processor 3 registers to implement a watchdog timer.

Note: The Processor 3 registers are dedicated to providing the watchdog timer function which prevents them from being used for other purposes.

The watchdog timer resets the board when improper execution of code is detected. Refer to Section 5 ***Reset*** (page 169) for more information on VGM5 reset.

The example code below shows how the MPIC is set up for the watchdog timer. Watchdog timer operation is summarized in the comments that precede the code. As shown in the code, the following Processor 3 registers are programmed:

- Timer frequency reporting register
- Current task priority register
- Destination register
- Interrupt source vector/priority register

For further programming details including the complete listing of the referenced include file(s), contact Customer Service.

# Example watchdog timer code

## Main routine, mpicWatchDog.c

```
/*------------------------------------------------------------------------
mpicWatchDog - set up mpic as a watchdog timer
--------------------------------------------------------------------------
Copyright 1998 Synergy Microsystems
--------------------------------------------------------------------------
C USAGE:
   unsigned long uses = 1000000;   // 1000000 us = 1 second

   mpicWatchDog(usecs);

ARGUMENT DESCRIPTION:

   usecs     - Time in microseconds before reset occurs
        - 0 to disable watchdog resets

DESCRIPTION:
   The calling board will be reset after the specified number of
microseconds elapses. The routine may be called repeatedly within the
number of microseconds specified to avoid the reset. The largest count
allows over 500 seconds to elapse between calls.

RESTRICTIONS & SPECIAL CONDITIONS:
Processor 3's interrupt is used to activate a reset to the onboard
reset logic. For that reason, this code will not operate on boards with
more than 3 processors (VSS4). A maximum count of 515401229 usec is allowed.
------------------------------------------------------------------------*/

#include "MpicRegs.h"

void  mpicWatchDog(long usecs)
{
    long count;
    MpicPciRegs P_MPIC_CFG = (MpicPciRegs)MPIC__CFG_BASE;
    MpicRegs P_MPIC = (MpicRegs)MPIC__REG_BASE;

//      Set MPIC PCI address for register accesses in memory space & enable
//      When complete, PCI reads and writes can be used for further
//      MPIC configuration.

    GrakWrL (&P_MPIC_CFG->PCI_BS0, MPIC__REG_BASE);
    GrakModL (&P_MPIC_CFG->PCI_CSR, MPIC__PCI_CSR__IOS, MPIC__PCI_CSR__IOS);

// Timer Frequency must be initialized to 33.3333Mhz/8

   P_MPIC->TimerFrequency = ByteSwap32(MPIC__TimerFrequency__33MHz);
//      Set Current Task Priority to disable (15) or enable (0) interrupt

   if(usecs == 0)
     P_MPIC->PPR_CTP_3 = ByteSwap32(0xf); // Disable WD interrupt
   else
     P_MPIC->PPR_CTP_3 = ByteSwap32(0);   // Enable WD interrupt

// Set the interrupt destination to processor 3

   P_MPIC->Timer3_Destination_Reg = ByteSwap32(8);

// Set Watchdog priority to 15 (0xf) (vector = 7 but is unused)

   P_MPIC->Timer3_VectorPriority_Reg = ByteSwap32(0x000f0007);

// Compute accurate count using integer arithmetic and reset counter

   count = usecs*33/8 + usecs*3/80 + usecs*3/800 + usecs*3/8000;
   P_MPIC->Timer3_BaseCount_Reg = ByteSwap32(0x80000000);
   P_MPIC->Timer3_BaseCount_Reg = ByteSwap32(count);
}
```

# 7 PCI-VME64 Bridge (Universe II)

This section provides general information about the Universe II PCI-VME64 bridge interface. This interface along with the direct PowerPC-to-VME64 interface (see Section 8) forms the VGM5 VMEbus interface.

- Introduction to Universe II
- Universe II register reference
- Improving BLT performance
- Programming notes, Universe II

Note The bit numbering of registers in this section follows the zero-on-the-right convention as opposed to the zero-on-the-left bit numbering convention used by Motorola and IBM in their PowerPC documentation.

# Introduction to Universe II

The Universe II PCI–VME64 bridge chip from Tundra Semiconductor is a part of the VGM5 VMEbus interface. Universe II's address translation provides VME access to PCI and PCI access to VME. The VME functions provided by this chip include:

- System controller
- Block transfers, master and slave
- Single data transfers, master and slave
- Interrupt generation and handling

For general information about the VGM5 VMEbus interface, refer to the ***VME64 bus*** chapter in Section 3 (page 57).

The PCI interface side of the Universe II provides the following functions:

- **PCI Target** — PCI masters address the Universe II: read transactions are coupled; write transactions are either coupled or posted depending on the PCI bus target image. PCI masters can also perform RMW and ADOH cycles via the Universe II's Special Cycle generator. For details on the mechanisms of these transfers, refer to the Universe II User Manual.

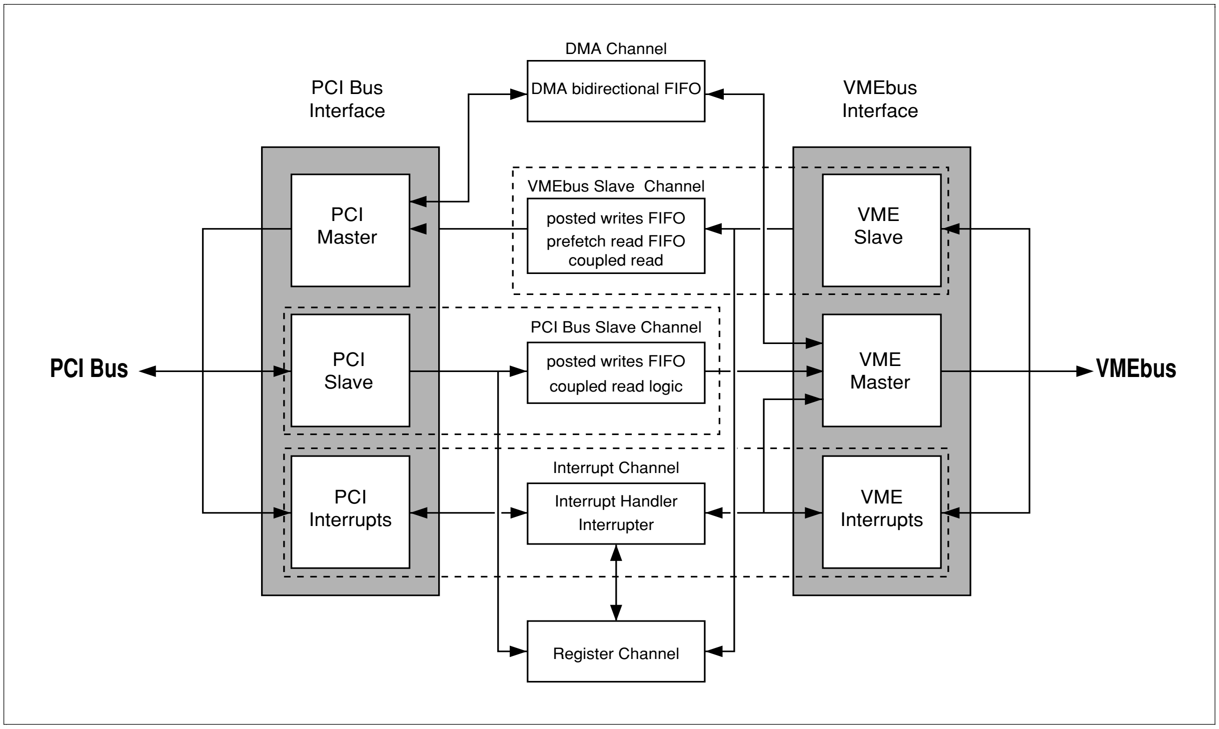
- **PCI Master** — An internal request of the Universe II's PCI Master interface by the VMEbus Slave channel or DMA channel causes the Universe II to operate as a PCI master. The user can set the relative priority of the VMEbus Slave channel and the DMA channel. For details on how this is set up, refer to the Universe II User Manual.

⚠ Due to the possibility of encountering a deadlock condition, it is recommended that the Universe II single-transfer master feature not be used in conjunction with the direct slave interface (described in Section 8). The symptom of this deadlock is a bus error on the Universe II master transfer.

As an Interrupter and Interrupt Handler, the Universe II provides flexible mapping of interrupts to the PCI bus or VMEbus interface. PCI interrupts can be routed and processed and VMEbus interrupts can be input to generate a VMEbus IAck cycle and to generate the specified interrupt signal. Software interrupts are ROAK while hardware and internal interrupts are RORA.

Universe II's DMA controller provides high performance data transfer between the PCI and VMEbus. Universe II is provided with a set of DMA registers to set up DMA transfer parameters.

The diagram below shows the architecture of the Universe II chip.



*Universe II architecture*

A register overview is given in the next chapter of this section (***Universe II register reference***).

For detailed information on the Universe II chip, refer to the Universe II User Manual and other supporting documentation available at Tundra Semiconductor Corporation's website:

**http://www.tundra.com**

# Universe II register reference

## Overview

The Universe II registers are collectively known as the Universe II Control and Status Registers (UCSR). These registers occupy 4 KB of internal memory. This space is logically divided into three groups:

- PCI Configuration Space (PCICS)
- Universe II Device Specific Registers (UDSR)
- VMEbus Control and Status Registers (VCSR)

The register access mechanism depends on whether the register space is accessed from the PCI bus or the VMEbus.

This chapter provides a reference overview of the Universe II registers. Refer to Tundra Semiconductor's Universe II User Manual and/or other supporting documentation for detailed information.

## Register access

Below is a brief description of Universe II register access.

### Universe II base address

The operating system PCI discovery routines dynamically set up the Universe II base address by writing to the appropriate registers in PCI configuration address space.

For more information on PCI configuration, refer to the ***PCI implementation details*** discussion in the **PCI bus** chapter in Section 3, page 62, and the ***Setting PCI device base address*** discussion in Section 4, page 103. Also refer to the ***Type 0 configuration table*** on page 110.

### Register access from PCI

There are two PCI access mechanisms for the Universe II registers:

- PCI Configuration space — Only the lower 256 bytes of the UCSR can be accessed as Configuration space. These bytes make up the Universe II's PCI configuration header.
- PCI Memory or I/O space — As specified in the Space bit of the PCI_BSx registers, the Universe II registers are accessed in either the PCI Memory or I/O space.

### Register access from VMEbus

There are two VMEbus access mechanisms for the Universe II registers. This mode is typically not used on the VGM5, since the VGM5 has onboard intelligence. However, it may be useful in certain applications.

- VMEbus Register Access Image (VRAI) — this mechanism allows the user to map the Universe II registers in A16, A24, or A32 address space.

- CS/CSR Space — this mechanism uses the VME64 scheme in which each slot in the VMEbus system is assigned 512 KB of CS/CSR space.

## Register map

The table below lists the Universe II registers by offset address.

For CS/CSR access, add 508 KBytes (0x7_F000) to offsets listed below.

***Universe II register map***

| Offset (Hex) | Register | Name |
|---|---|---|
| 000 | PCI Configuration Space ID register | PCI_ID |
| 004 | PCI Configuration Space Ctl & Status reg. | PCI_CSR |
| 008 | PCI Configuration Class register | PCI_Class |
| 00C | PCI Configuration Misc. 0 register | PCI_MISC0 |
| 010 | PCI Configuration Base Address register | PCI_BS0 |
| 014 | PCI Configuration Base Address 1 register | PCI_BS1 |
| 018-024 | PCI Unimplemented | |
| 028 | PCI Reserved | |
| 02C | PCI Reserved | |
| 030 | PCI Unimplemented | |
| 034 | PCI Reserved | |
| 038 | PCI Reserved | |
| 03C | PCI Configuration Misc. 1 register | PCI_MISC1 |
| 040-0FF | PCI Unimplemented | |
| 100 | PCI Target Image 0 Control register | LSI0_CTL |
| 104 | PCI Target Image 0 Base Address register | LSI0_BS |
| 108 | PCI Target Image 0 Bound Address register | LSI0_BD |
| 10C | PCI Target Image 0 Translation Offset register | LSI0_TO |
| 110 | Reserved | |
| 114 | PCI Target Image 1 Control register | LSI1_CTL |
| 118 | PCI Target Image 1 Base Address register | LSI1_BS |
| 11C | PCI Target Image 1 Bound Address register | LSI1_BD |
| 120 | PCI Target Image 1 Translation Offset register | LSI1_TO |
| 124 | Reserved | |
| 128 | PCI Target Image 2 Control register | LSI2_CTL |
| 12C | PCI Target Image 2 Base Address register | LSI2_BS |
| 130 | PCI Target Image 2 Bound Address register | LSI2_BD |
| 134 | PCI Target Image 2 Translation Offset register | LSI2_TO |
| 138 | Reserved | |
| 13C | PCI Target Image 3 Control register | LSI3_CTL |
| 140 | PCI Target Image 3 Base Address register | LSI3_BS |
| 144 | PCI Target Image 3 Bound Address register | LSI3_BD |
| 148 | PCI Target Image 3 Translation Offset register | LSI3_TO |
| 14C-16C | Reserved | |
| 170 | Special Cycle Control register | SCYC_CTL |
| 174 | Special Cycle PCI Bus Address register | SCYC_ADDR |

### Universe II register map (continued)

| Offset (Hex) | Register | Name |
|---|---|---|
| 178 | Special Cycle Swap/Compare Enable register | SCYC_EN |
| 17C | Special Cycle Compare Data register | SCYC_CMP |
| 180 | Special Cycle Swap Data register | SCYC_SWP |
| 184 | PCI Misc. register | LMISC |
| 188 | Special PCI Target Image register | SLSI |
| 18C | PCI Command Error Log register | L_CMDERR |
| 190 | PCI Address Error Log register | LAERR |
| 194-19C | Reserved | |
| 1A0 | PCI Target Image 4 Control register | LSI4_CTL |
| 1A4 | PCI Target Image 4 Base Address register | LSI4_BS |
| 1A8 | PCI Target Image 4 Bound Address register | LSI4_BD |
| 1AC | PCI Target Image 4 Translation Offset register | LSI4_TO |
| 1B0 | Reserved | |
| 1B4 | PCI Target Image 5 Control register | LSI5_CTL |
| 1B8 | PCI Target Image 5 Base Address register | LSI5_BS |
| 1BC | PCI Target Image 5 Bound Address register | LSI5_BD |
| 1C0 | PCI Target Image 5 Translation Offset register | LSI5_TO |
| 1C4 | Reserved | |
| 1C8 | PCI Target Image 6 Control register | LSI6_CTL |
| 1CC | PCI Target Image 6 Base Address register | LSI6_BS |
| 1D0 | PCI Target Image 6 Bound Address register | LSI6_BD |
| 1D4 | PCI Target Image 6 Translation Offset register | LSI6_TO |
| 1D8 | Reserved | |
| 1DC | PCI Target Image 7 Control register | LSI7_CTL |
| 1E0 | PCI Target Image 7 Base Address register | LSI7_BS |
| 1E4 | PCI Target Image 7 Bound Address register | LSI7_BD |
| 1E8 | PCI Target Image 7 Translation Offset register | LSI7_TO |
| 1EC-1FC | Reserved | |
| 200 | DMA Transfer Control register | DCTL |
| 204 | DMA Transfer Byte Count register | DTBC |
| 208 | DMA PCI Bus Address register | DLA |
| 20C | Reserved | |
| 210 | DMA VMEbus Address register | DVA |
| 214 | Reserved | |
| 218 | DMA Command Packet Pointer register | DCPP |
| 21C | Reserved | |
| 220 | DMA General Control and Status register | DGCS |
| 224 | DMA Linked List Update Enable register | D_LLUE |
| 228-2FC | Reserved | |
| 300 | PCI Interrupt Enable register | LINT_EN |
| 304 | PCI Interrupt Status register | LINT_STAT |

### Universe II register map (continued)

| Offset (Hex) | Register | Name |
|---|---|---|
| 308 | PCI Interrupt Map 0 register | LINT_MAP0 |
| 30C | PCI Interrupt Map 1 register | LINT_MAP1 |
| 310 | VMEbus Interrupt Enable register | VINT_EN |
| 314 | VMEbus Interrupt Status register | VINT_STAT |
| 318 | VMEbus Interrupt Map 0 register | VINT_MAP0 |
| 31C | VMEbus Interrupt Map 1 register | VINT_MAP1 |
| 320 | Interrupt Status/ID Out register | STATID |
| 324 | VIRQ1 Status/ID register | V1_STATID |
| 328 | VIRQ2 Status/ID register | V2_STATID |
| 32C | VIRQ3 Status/ID register | V3_STATID |
| 330 | VIRQ4 Status/ID register | V4_STATID |
| 334 | VIRQ5 Status/ID register | V5_STATID |
| 338 | VIRQ6 Status/ID register | V6_STATID |
| 33C | VIRQ7 Status/ID register | V7_STATID |
| 340 | PCI Interrupt Map 2 register | LINT_MAP2 |
| 344 | VME Interrupt Map 1 register | VINT_MAP2 |
| 348 | Mailbox 0 register | MBOX0 |
| 34C | Mailbox 1 register | MBOX1 |
| 350 | Mailbox 2 register | MBOX2 |
| 354 | Mailbox 3 register | MBOX3 |
| 358 | Semaphore 0 register | SEMA0 |
| 35C | Semaphore 1 register | SEMA1 |
| 360-3FC | Reserved | |
| 400 | Master Control register | MAST_CTL |
| 404 | Misc. Control register | MISC_CTL |
| 408 | Misc. Status register | MISC_STAT |
| 40C | User AM Codes register | USER_AM |
| 410-EFC | Reserved | |
| F00 | VMEbus Slave Image 0 Control register[1] | VSIO_CTL |
| F04 | VMEbus Slave Image 0 Base Address register[1] | VSIO_BS |
| F08 | VMEbus Slave Image 0 Bound Address reg. [1] | VSIO_BD |
| F0C | VMEbus Slave Image 0 Translation Offset reg. [1] | VSIO_TO |
| F10 | Reserved | |
| F14 | VMEbus Slave Image 1 Control register[1] | VSI1_CTL |
| F18 | VMEbus Slave Image 1 Base Address register[1] | VSI1_BS |
| F1C | VMEbus Slave Image 1 Bound Address reg. [1] | VSI1_BD |
| F20 | VMEbus Slave Image 1 Translation Offset reg. [1] | VSI1_TO |
| F24 | Reserved | |
| F28 | VMEbus Slave Image 2 Control register[1] | VSI2_CTL |
| F2C | VMEbus Slave Image 2 Base Address register[1] | VSI2_BS |
| F30 | VMEbus Slave Image 2 Bound Address reg. [1] | VSI2_BD |
| F34 | VMEbus Slave Image 2 Translation Offset reg. [1] | VSI2_TO |

Note: 1. Avoid updating slave image registers while VME traffic is using the slave window. Doing so results in VME data errors. See ***Programming notes, Universe II***on page 215 for more information.

### *Universe II register map (continued)*

| Offset (Hex) | Register | Name |
|---|---|---|
| F38 | Reserved | |
| F3C | VMEbus Slave Image 3 Control register[1] | VSI3_CTL |
| F40 | VMEbus Slave Image 3 Base Address register[1] | VSI3_BS |
| F44 | VMEbus Slave Image 3 Bound Address reg. [1] | VSI3_BD |
| F48 | VMEbus Slave Image 3 Translation Offset reg. [1] | VSI3_TO |
| F4C-F60 | Reserved | |
| F64 | Location Monitor Control register | LM_CTL |
| F68 | Location Monitor Base Address register | LM_BS |
| F6C | Reserved | |
| F70 | VMEbus Register Access Image Control reg. | VRAI_CTL |
| F74 | VMEbus Register Access Image Base Addr. reg. | VRAI_BS |
| F78 | Reserved | |
| F7C | Reserved | |
| F80 | VMEbus CSR Control register | VCSR_CTL |
| F84 | VMEbus CSR Translation Offset register | VCSR_TO |
| F88 | VMEbus AM Code Error Log register | V_AMERR |
| F8C | VMEbus Address Error Log register | VAERR |
| F90 | VMEbus Slave Image 4 Control register | VSI4_CTL |
| F94 | VMEbus Slave Image 4 Base Address register | VSI4_BS |
| F98 | VMEbus Slave Image 4 Bound Address reg. | VSI4_BD |
| F9C | VMEbus Slave Image 4 Translation Offset reg. | VSI4_TO |
| FA0 | Reserved | |
| FA4 | VMEbus Slave Image 5 Control register | VSI5_CTL |
| FA8 | VMEbus Slave Image 5 Base Address register | VSI5_BS |
| FAC | VMEbus Slave Image 5 Bound Address reg. | VSI5_BD |
| FB0 | VMEbus Slave Image 5 Translation Offset reg. | VSI5_TO |
| FB4 | Reserved | |
| FB8 | VMEbus Slave Image 6 Control register | VSI6_CTL |
| FBC | VMEbus Slave Image 6 Base Address register | VSI6_BS |
| FC0 | VMEbus Slave Image 6 Bound Address reg. | VSI6_BD |
| FC4 | VMEbus Slave Image 6 Translation Offset reg. | VSI6_TO |
| FC8 | Reserved | |
| FCC | VMEbus Slave Image 7 Control register | VSI7_CTL |
| FD0 | VMEbus Slave Image 7 Base Address register | VSI7_BS |
| FD4 | VMEbus Slave Image 7 Bound Address reg. | VSI7_BD |
| FD8 | VMEbus Slave Image 7 Translation Offset reg. | VSI7_TO |
| FDC-FEC | Reserved | |
| FF0 | VME CR/CSR Reserved | |
| FF4 | VMEbus CSR Bit Clear register | VCSR_CKR |
| FF8 | VMEbus CSR Bit Set register | VCSR_SET |
| FFC | VMEbus CSR Base Address register | VCSR_BS |

Note: 1. Avoid updating slave image registers while VME traffic is using the slave window. Doing so results in VME data errors. See ***Programming notes, Universe II***on page 215 for more information.

# Improving VME performance

There are two ways to improved VME performance:

- Special registers in the Universe II chip.
- An onboard register bit that controls signal noise filtering logic in hardware.

## Universe II specific (U2SPEC) register

The Universe II chip (Revision ID = 01 or 02) comes with a register called "Specific Register" (U2SPEC, offset 0x4FC) which is used to improve the performance of the Universe II by reducing the latency of key VMEbus timing elements. The timing adjustment provided by the U2SPEC register is intended to compensate for VME master and slave latencies introduced by buffers, transceivers, and the backplane itself. **Using the U2SPEC register may result in violation of the VME specification.**

⚠ The U2SPEC register is an unsupported feature of Universe II. Its design may not be as robust as other areas of the Universe II design, and may not be included in future revisions of the device. Improper use of the U2SPEC register may result in undesirable system behavior. **Tundra Semiconductor Corp. and Synergy Microsystems, Inc. do not recommend the manipulation of this register by users who are unfamiliar with the timing characteristics of their VME systems.**

## U2SPEC adjustable VME timing parameters

**VME DTAck\ Inactive Filter (DTKFLTR), bit 12** — In order to overcome the DTAck\ noise typical of most VME systems, the Universe II quadruple samples this signal with the 64MHz clock. While “safer”, the extra sampling results in decreased performance. User who believe their systems to have little noise on their DTAck\ lines can elect to filter this signal less and thus increase their Universe II response time.

**VME Master Parameter t11 Control (MASt11), bit 10** — According to the VME64 Specification, a VMEbus master must not drive DS0\ low until both it and DS1\ have been simultaneously high for a minimum of 40ns. The MASt11 parameter in the U2SPEC register, however, allows DS0\ to be driven low in less than 40ns.

**VME Master Parameter t27 Control (READt27), bits 8 & 9** — During read cycles, the VMEbus master must guarantee that the data lines will be valid within 25ns after DTAck\ is asserted. That is to say, the master must not latch the data and terminate the cycle for a minimum of 25ns after the falling edge of DTAck\. The READt27 parameter in the U2SPEC register allows for faster cycle termination in one of two ways. One setting allows for the data to be latched and the cycle terminated with an associated delay that is less than 25ns. The other setting results in no delay whatsoever in latching and termination.

**VME Slave Parameter t28 Control (POSt28), bit 2** — According to the VME64 Specification, VMEbus slaves must wait at least 30ns after the assertion of DS\ before driving DTAck\ low. The POSt28 parameter in the U2SPEC register, however, allows DTAck\ to be asserted in less than 30ns when executing posted writes.

**VME Slave Parameter t28 Control (PREt28), bit 0** — VMEbus slaves must wait at least 30ns after the assertion of DS\ before driving DTAck\ low. The PREt28 parameter in the U2SPEC register, however, allows DTAck\ to be asserted in less than 30ns when executing prefetched reads.

## U2SPEC register bit assignments

The U2SPEC bit assignments and descriptions are listed below.

**U2SPEC register, offset 0x4FC - bit assignments**

<table>
<tr><th>Bits</th><th colspan="8">Function</th></tr>
<tr><td>31-24</td><td colspan="8">Universe Reserved</td></tr>
<tr><td>23-16</td><td colspan="8">Universe Reserved</td></tr>
<tr><td>15-08</td><td colspan="3">Universe Reserved</td><td>DTKFLTR</td><td>Reserved</td><td>MASt11</td><td colspan="2">READt27</td></tr>
<tr><td>07-00</td><td colspan="5">Universe Reserved</td><td>POSt28</td><td>Reserved</td><td>PREt28</td></tr>
</table>

**U2SPEC bit descriptions**

| Name | Type | Reset By | Reset State | Function |
|---|---|---|---|---|
| DTKFLTR | R/W | All | 0 | ***VME DTAck\ Inactive Filter***<br>0 = Slower but better filter, 1 = Faster but poorer filter |
| MASt11 | R/W | All | 0 | ***VME Master Parameter t11 Control (DS\ hight time during BLT's and MBLT's)***<br>0 = Default, 1 = Faster |
| READt27 | R/W | All | 00 | ***VME Master Parameter t27 Control (Delay of DS\ negation after read)***<br>00 = Default, 01 = Faster, 10 = No Delay |
| POSt28 | R/W | All | 0 | ***VME Slave Parameter t28 Control (Time of DS\ to DTAck\ for posted-write)***<br>0 = Default, 1 = Faster |
| PREt28 | R/W | All | 0 | ***VME Slave Parameter t28 Control (Time of DS\ to DTAck\ for prefetch read)***<br>0 = Default, 1 = Faster |

Note: Bits marked as "Universe Reserved" must be set to zero (0).

# BLT speedup bit

A BLT speedup bit (bit 4) is provided by the onboard PowerPC-VME64 mode register at 0xFFEF_FF20. When set, it improves BLT performance by disabling Data Strobe noise filtering logic that is normally in place to ensure proper operation of the board in a heavily loaded VMEbus system.

## PowerPC-VME64 Mode register, FFEF_FF20 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *x* | *x* | *0* | *1* | *0* | *1* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Broadcast Slave Interface Select | 0 = Universe PCI-VME64 Bridge<br>1 = Direct VME interface |
| b6-b5 | Reserved | — |
| b4 | BLT Speedup, PCI-VME64 Bridge | 0 = slow BLT (default)<br>1 = fast BLT |
| b3 | Bus Release Mode | 0 = RWD (Release When Done)<br>1 = ROR (Release On Request, default) |
| b2 | FAIR Requestor | 0 = normal, non-FAR mode.<br>1 = FAIR requestor mode, wait for BRQ false before re-asserting bus request. |
| b1-b0 | Bus Request Level | 0 = Level 0<br>1 = Level 1<br>2 = Level 2<br>3 = Level 3 |

For a system with only a few boards, this bit can be set (1=fast BLT). For a heavily loaded system, this bit should be cleared (0=normal).

⚠ The BLT speedup bit is an unsupported feature of the VGM5. It may not be included in future revisions of the board. Improper use of the BLT speedup bit may result in undesirable system behavior. **Synergy Microsystems, Inc. does not recommend the manipulation of this bit by users who are unfamiliar with the timing characteristics of their VME systems.**

# Programming notes, Universe II

## Writing to non-existent VME locations

### Problem

The Universe II chip has a problem dealing with non-existent VME locations. The problem is that the Grackle will not return a machine check exception to the CPU when a VME write fails with a Bus Error.

### Solutions

Workaround #1: First perform a read of the location to verify its existence.

Workaround #2: Program the Universe II to generate an interrupt upon VME Bus Error and have this interrupt report a fatal error.

## Slave image programming

### Problem

Updating a Universe II slave window while VME traffic is using that window results in data errors **even if the register contents are the same**.

### Solution

Avoid programming a slave's image registers while VME is accessing that slave's window.

# 8 Direct PowerPC-to-VME64 Interface

This section provides information about the direct PowerPC-to-VME64 interface. This interface along with the PCI-VME64 bridge (Universe II) interface forms the VGM5 VMEbus interface.

- Using the VGM5 direct VME interface
- Master direct VME interface
- Programming notes, Master direct VME interface
- Slave direct VME interface
- Read-Modify-Write (RMW)

Note: The bit numbering of registers in this section follows the zero-on-the-right convention as opposed to the zero-on-the-left bit numbering convention used by Motorola and IBM in their PowerPC documentation.

# Using the VGM5 direct VME interface

The Direct VME interface provides a fast path between the PowerPC bus and the VMEbus for single transfers. This improves performance for multiprocessor applications that require a large amount of data sharing across the VMEbus. The Direct VME interface is an adjunct to the Universe II; the Universe II must be used for such VME operations as interrupts, mailbox and block transfers.

The table below lists the registers that control the Direct VME interface.

***Direct VME interface onboard registers***

| Register | Address | Access |
|---|---|---|
| PPC-VME64 Slave Mask Register | 0xFFEF_FF00 | D8 (RW) |
| PPC-VME64 Slave Addr Register | 0xFFEF_FF08 | D8 (RW) |
| VME64 Broadcast Addr Register | 0xFFEF_FF10 | D8 (RW) |
| PPC-VME64 Master Addr Register | 0xFFEF_FF18 | D8 (RW) |
| PPC-VME64 Mode Register | 0xFFEF_FF20 | D8 (RW) |
| PPC-VME64 RMW Register | 0xFFEF_FF28 | D8 (RW) |

# Master direct VME interface

The Direct Master VMEbus interface provides a high-speed path to the VMEbus for single read or write accesses. Since this interface connects the processor directly to the VMEbus, the latency encountered in using the Universe II via the PCI bus is not incurred.

The Master interface allows the CPU to read or write VMEbus slaves in a selected 1 Gigabyte region of the direct VMEbus' A32 address space, plus all of the A24 and A16 VME address spaces. Each of the two CPU chips has a separate master address translation register.

The Direct VME Master interface generates a particular Address Modifier code depending on the type access.

- 0x0D (Extended Supervisory Data Access) for A32 accesses
- 0x3D (Standard Supervisory Data Access) for A24 accesses
- 0x2D (Short Supervisory Access) for A16 accesses

## Direct VMEbus address map

The VMEbus is accessible in the CPU address range of 0x4000_0000–0x7FFF_FFFF. The table below lists the regions for A32, A24 and A16 direct VMEbus accesses.

**Direct VMEbus access regions**

| Address size | AM Code | Region |
|---|---|---|
| A32 | 0x0D | 4000_0000 - 77FF_FFFF |
| A24 | 0x3D | 7B00_0000 - 7BFF_FFFF |
| A16 | 0x2D | 7FFF_0000 - 7FFF_FFFF |

## Master Mode

A Mode register selects the Direct VME bus request level, arbitration mode, and bus release mode.

### PowerPC-VME64 Mode register bit description, 0xFFEF_FF20 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *x* | *x* | *0* | *1* | *0* | *1* | *1* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | Broadcast Slave Interface Select | 0 = Universe PCI-VME64 Bridge<br>1 = Direct VME interface |
| b6-b5 | Reserved | — |
| b4 | BLT Speedup, PCI-VME64 Bridge | 0 = slow BLT (default)<br>1 = fast BLT |
| b3 | Bus Release Mode | 0 = RWD (Release When Done)<br>1 = ROR (Release On Request, default) |
| b2 | FAIR Requestor | 0 = normal, non-FAIR mode.<br>1 = FAIR requestor mode, wait for BRQ false before re-asserting bus request. |
| b1-b0 | Bus Request Level | 0 = Level 0<br>1 = Level 1<br>2 = Level 2<br>3 = Level 3 |

Bits 1-0 select the bus request level used by the Direct VME Master interface. The default request level is 3. This may be changed to any of levels 0 through 3. In a system with a priority arbiter (PRI), request level 3 is the highest priority. In a system with a round-robin arbiter, all request levels have the same priority. In a system with a one-level arbiter, level 3 is the only request level supported.

Bit 2 selects Fair bus request mode for the Direct VME Master interface. Fair mode allows more equitable usage of the bus by many masters; without it, boards toward the right end of a VMEbus cardcage tend to be starved for bus access since the bus uses a daisy-chained bus grant.

Fair mode must be selected for all requesters used on a particular request level if it is selected for any requester. Fair mode is not supported by all VMEbus cards; consult the user manual for any other cards used in the system.

Bit 3 selects the bus release mode for the Direct VME Master interface. Release On Request (ROR) is more efficient in systems with less frequent bus ownership changes, and Release When Done (RWD) is more efficient in systems with frequent bus ownership changes. Experimentation will reveal the best setting.

The remainder of the bits in this register do not apply to the Master direct VME interface. Refer to ***Improving BLT performance*** on page 211 and ***PowerPC–VME64 Slave Mask register bit description*** on page 135 for more information on bits 4 and 7 respectively.

## Master Address Translation

The Direct VME A32 address window shown above is smaller than the VMEbus A32 space. To allow access to all 4 gigabytes of the VME A32 address space, the address translation register at 0xFFEF_FF18 is provided.

### PowerPC-VME64 Master Address register bit description, 0xFFEF_FF18 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *x* | *x* | *x* | *x* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b6 | Master Address bits 31-30 | Top two bits of CPU-X VME Master address have this value. |
| b5-b4 | Master Address bits 31-30 | Top two bits of CPU-Y VME Master address have this value. |
| b3-b0 | Reserved | — |

Separate translation bits are provided for each of the X and Y CPUs on a dual-CPU board. It is advisable to allow write access to this register by only one CPU, since this register is shared. On a single-CPU board, bits 4 and 5 of this register are unused.

The tables below show the mapping of CPU address to VME address.

### CPU-X, VME master address translation

| Addr Trans. Register | | | | CPU X | VME |
|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | Address | Address |
| 0 | 0 | X | X | 0x4000_0000-0x77FF_FFFF | 0x0000_0000-0x37FF_FFFF |
| 0 | 1 | X | X | 0x4000_0000-0x77FF_FFFF | 0x4000_0000-0x77FF_FFFF |
| 1 | 0 | X | X | 0x4000_0000-0x77FF_FFFF | 0x8000_0000-0xB7FF_FFFF |
| 1 | 1 | X | X | 0x4000_0000-0x77FF_FFFF | 0xC000_0000-0xF7FF_FFFF |

### CPU-Y, VME master address translation

| Addr Trans. Register | | | | CPU Y | VME |
|---|---|---|---|---|---|
| b7 | b6 | b5 | b4 | Address | Address |
| X | X | 0 | 0 | 0x4000_0000-0x77FF_FFFF | 0x0000_0000-0x37FF_FFFF |
| X | X | 0 | 1 | 0x4000_0000-0x77FF_FFFF | 0x4000_0000-0x77FF_FFFF |
| X | X | 1 | 0 | 0x4000_0000-0x77FF_FFFF | 0x8000_0000-0xB7FF_FFFF |
| X | X | 1 | 1 | 0x4000_0000-0x77FF_FFFF | 0xC000_0000-0xF7FF_FFFF |

# Programming notes, Master direct VME interface

## Improving performance, master direct VME interface

To improve performance in systems in which the bus is used mostly by one CPU at a time for several transfers, set the Direct VME bus mode register to ROR mode to allow the board to hold the VME bus ownership until another master wants to use it.

# Slave direct VME interface

The VGM5 Direct VME Slave interface provides a low-latency access path to the VGM5's onboard DRAM from other VMEbus masters. Since this interface connects the PowerPC bus directly to the VMEbus, it reduces the latency of said accesses.

The Direct VME Slave window may be placed anywhere in the VMEbus A32 address space on a 16 MB boundary. The window size is programmable from 16 MB to 256 MB. The window always maps to the DRAM address 0x0000_0000.

The Direct VME Slave interface responds to Address Modifiers 0x09, 0x0A, 0x0D, and 0x0E. These correspond to A32 Supervisory or User data or program transfers. It does not respond to Block transfer AM codes. This makes it possible to set the Universe II's slave window to respond to BLTs and the Direct VME Slave window to respond to single transfers in the same address range, thereby achieving highest performance for all transfer types.

The Direct VME Slave interface is controlled by two registers: Address and Size. The Slave Enable bit in the Size register is used to enable the Direct VME Slave interface.

⚠ Due to the possibility of encountering a deadlock condition, it is recommended that the direct slave interface not be used in conjunction with the Universe II single-transfer master feature. The symptom of this deadlock is a bus error on the Universe II master transfer.

## PowerPC-VME64 Slave Address register bit description, 0xFFEF_FF08 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7-b0 | VME64 Slave Addr bits 31-24 | 00 = 0000_0000 – 00FF_FFFF<br>01 = 0100_0000 – 01FF_FFFF<br>↓ :<br>FF = FF00_0000 – FFFF_FFFF |

Note that if the Slave Address Mask register has a bit set in bits 0-3, the corresponding bit in the Slave Address register becomes don't-care. For example, if the slave window size is set to 64 megabytes, then slave address bits 25-24 become don't-care because they fall within the slave window.

## PowerPC-VME64 Slave Mask register bit description, 0xFFEF_FF00 (RW)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *Reset value* |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7 | PowerPC-VME64 Slave Enable | 0 = Slave disabled<br>1 = Slave enabled |
| b6 | VME64 Broadcast Enable | 0 = Broadcasting slave support disabled<br>1 = Broadcasting slave support enabled |
| b5 | VME64 Broadcast, DTAck Driver Enable[1] | 0 = DTAck disabled<br>1 = DTAck enabled |
| b4 | VME64 Broadcast, Wait Line Selection | 0 = VME SerA<br>1 = VME SerB |
| b3-b0 | Slave Address Mask bits 29-24[2]<br>(Sets the size of the PowerPC-VME64 slave window. Must be less than or equal to DRAM size.) | 0 = 16 MB window size<br>1 = 32 MB window size<br>3 = 64 MB window size<br>7 = 128 MB window size<br>F = 256 MB window size |

NOTE:S 1. Set one board in a broadcast group for DTAck drive enabled; set all others disabled.
2. All other values for b3-b0 (window size) are reserved and should not be used.

Bits 0–3 set the size of the direct VME slave window. Bit 7 sets the direct VME slave enable/disable status.

The remaining bits (4–6) of the register control the optional broadcast feature which is usable with **both** Universe II and direct VME slave interfaces.

Bit 4 selects which VMEbus pin to use for the group Wait\ signal. It selects WaitA\ (VME P1 pin B21) when set to 0, or WaitB\ (VME P1 pin B22) when set to 1.

Bit 5 is set to 1 to enable this board to drive DTAck\ for the broadcast group, or to 0 to prevent this board from driving DTAck\ during broadcast transfers. This bit should be set to 1 on one board in the group; this bit should be set to 0 on the other boards in the group.

Bit 6 enables the broadcast feature when set to 1.

The Broadcast Address register is loaded with the eight MSBs of the VME address of the 16 Megabyte broadcast window. This is the address range over which the broadcast operation will take place, and provides a second direct slave window if the direct slave interface is enabled. To use broadcast with the Universe II VME slave interface, the Universe II must also be programmed to respond to the broadcast window.

Contact the factory for availability of the broadcast slave feature.

# Read-Modify-Write (RMW)

## General description

The VMEbus standard defines a type of cycle called read-modify-write (RMW). It is used to provide atomic access to a semaphore by one VME master at a time. This cycle type is based on the Motorola 68000 processor family TAS instruction. It consists of one VMEbus address cycle containing two VMEbus data cycles, the first a read and the second a write. Since it is only one address cycle, bus arbitration cannot occur, so no intervening cycles from other masters may interrupt the read-write sequence.

The PowerPC processor bus does not support atomic read-modify-write cycles, since arbitration occurs automatically before every read or write access. Instead, another mechanism is provided: a reservation bit and a pair of instructions called load-with-reservation (lwarx) and store-conditional (stwcx.). The reservation is a bit internal to the processor that indicates whether the load/store sequence has been performed atomically or not.

The reservation is set by the first of these instructions, and cleared by the second. If the reservation is still set when the store conditional is performed, the store will occur. The reservation is cleared by an intervening write to the cache line (32 byte region) containing the reservation address. If the reservation is cleared, then the store conditional will not be performed.

The lwarx/stwcx. instruction pair does not allow semaphores to exist on other VME boards, since the processor's snooping logic must be able to observe all accesses to the semaphore in order to clear the reservation, but VMEbus snooping is not performed due to performance considerations. Because of this fact, any VMEbus system that contains semaphores in RAM must use the RMW mechanism.

The Direct VME interface on the VGM5 contains hardware to allow the processor to perform a VMEbus read-modify-write cycle which is compatible with both legacy hardware and with the lwarx/stwcx. instruction pair. This chapter describes the proper use of this hardware.

## Slave RMW operations

Slave read-modify-write operations are transparently handled by the direct VME interface. There are some requirements for the onboard access to a semaphore stored in onboard RAM, however.

If the semaphore will be shared by a local contender, the local contender must use the lwarx/stwcx. instruction pair to use the semaphore. This imposes several restrictions on the semaphore's placement in memory as follows:

- Only one semaphore may exist in a 32-byte cache line.
- The semaphore must be a 32-bit value stored on an even 4-byte boundary.
- The semaphore must either be in a non-cached page or the code must take special precautions to flush the cache when updating the semaphore. The code examples show how to do this.
- The semaphore must be cleared as well as set with the lwarx/stwcx. instruction pair to maintain cache coherency.

## Master RMW operations

The Master RMW cycle is performed with the use of the Master RMW Register. This register contains one active bit. The bit is self-clearing.

The Master RMW sequence is as follows:

```
Read the VME address.
begin
    Write a 1 to the RMW mode register.
    Read the VME address.
    Read the RMW Mode register.
    If 1, leave
end
Write the VME address with the semaphore bit set.
```

The code example below, written for VxWorks and using Address Map B, illustrates RMW. Included in the example code are the intLock() and intUnlock() functions which disable and re-enable interrupts respectively. These functions are required in a multitasking environment where it is sometimes necessary to briefly disable interrupts to make sure that an instruction sequence does not get interrupted. In VxWorks these functions are part of the kernel and thus external to the actual RMW code. However, they are shown here to clarify the procedures involved in RMW.

```
/* The macro below does an eieio instruction followed by a sync instruction.
This is to ensure that the previous write to memory has completed before the next
instruction is done.
*/
# define EIEIO_SYNC __asm__(" eieio; sync")

#define VME64_RMW_ADR  0xFFEFFF28 /* Addrs of RMW register. */

/***************************************************************************
* sysMsrGet - Reads the PPC MSR register and returns the 32-bit result.
*
* Reads the MSR reg. in the PPC and returns the 32-bit result.
* RETURNS: Current contents of PPC MSR register.
*/
int sysMsrGet(   void)
   {
__asm__("   mfmsr     3");

   }

/***************************************************************************
* sysMsrWrite - Writes the PPC MSR register with the specified value.
*
* Writes the MSR reg. in the PPC.
*
* RETURNS: N/A
*/
void sysMsrWrite
   (
   int msrData
   )
   {
__asm__("   mtmsr     3");
   }
/***************************************************************************
* intLock - Disables interrupts.
*
* Clears the interrupt enable bit in the MSR
*
* RETURNS: On PPC architecture it returns 0 always.
*/
int intLock(void)
   {
   sysMsrWrite(sysMsrGet() & 0xFFFF7FFF);
   return(0);
   }
```

```
/***************************************************************************
* intUnlock - Enables interrupts.
*
* Sets the interrupt enable bit in the MSR. Takes an argument for compatibility
* with versions for non-PPC processors.
*
* RETURNS: N/A
*/
void intUnlock
   (
   int level
   )
   {
   sysMsrWrite(sysMsrGet() & 0x00008000);
   }


/***************************************************************************
*
* sysSemClear - write a 0 using stwcx. instruction.
*
* Stores a 32-bit 0 to memory using the stwcx. instruction. Called
* principally by sysBusTasClear.
*
* RETURNS: N/A
*
*/
void sysSemClear
   (
   int *ptr
   )
   {
__asm__("fail1: ");
__asm__("   lis  4, 0");
__asm__("   dcbf 0, 3");       /* flush the sema! */
__asm__("   sync");
__asm__("   lwarx     5, 0, 3");    /* reserve */
__asm__("   sync");                 /* jic */
__asm__("   stwcx.    4, 0, 3");    /* try to clear */
__asm__("   sync");
__asm__("   bne  fail1");
   }

/***************************************************************************
* synTas - Do a lwarx & stcwx to specified location.
*
* This does a "Test and Set" operation to an onboard RAM location.
* To do a "Test and Set" over the VMEbus, see sysBusTas().
*
* RETURNS: TRUE if the flag was available and is now owned, or
*          FALSE if the flag was not available.
*
*/
BOOL synTas
   (
   char *adrs /* address of the location being tested. */
   )
{
```

```
__asm__("    lis  6, 0x8000");

__asm__("    dcbf 0, 3");      /* flush the cache!*/
__asm__("    sync");
__asm__("    lwarx    7, 0, 3");     /* reserve */
__asm__("    sync");
__asm__("    cmpwi    7, 0");
__asm__("    bne  failClear");       /*sema already set; res set */
__asm__("    stwcx.   6, 0, 3");     /* try to set*/
__asm__("    sync");
__asm__("    bne  fail");            /* failed; res clear */
__asm__("    li   3, 1");
__asm__("    blr");

__asm__("failClear: ");
__asm__(" li 4, 4");
__asm__(" stwcx. 4,0,4");      /* store a 4 at 4 */

__asm__("fail: ");
__asm__("    sync");
   return(FALSE);
   }

/****************************************************************************
* sysBusTas - test and set a location across the bus
*
* This routine performs a test-and-set (TAS) instruction on the specified
* address. To prevent deadlocks, interrupts are disabled and the Gemini
* Read Modify Write feature is used.
*
* NOTE: This routine is dependent upon, not equivalent to, vxTas().
*
* RETURNS: TRUE if the value had not been set but is now, or FALSE if the
*          the value was set already.
*
* SEE ALSO: vxTas(), sysBusTasClear()
*/
BOOL sysBusTas
   (
   char * adrs   /* address to be tested and set */
   )
   {
   char temp;
   BOOL state = FALSE;     /* semaphore state */
   int lockKey;            /* interrupt lock key */

   /* lock interrupts so there will be no TAS interference */
   lockKey = intLock ();

   /* set up for next cycle to be RMW */

   /* perform the TAS */
   if(adrs < (char *)0x40000000)
        {                            /* It's onboard */
        state = synTas (adrs);
        EIEIO_SYNC;
        }
   else
        {                           /* It's offboard */
        do
            {
            *(volatile char *)VME64_RMW_ADR = 1;
            temp = *(volatile char *)VME64_RMW_ADR; /* Wait for write Post.*/

            temp = *adrs;           /* This is the VMEbus read. */
            } while( *(char *)VME64_RMW_ADR != 1); /* Check that RMW took. */

        *adrs = temp |0x80;     /* Set the msb. This is the modify & write. */
        if ((temp & 0x80) == 0) /* did we get it? i.e., was it 0 before? */
            {
            state = TRUE;
            }
        else
            {
            state = FALSE;
            }

        }

   /* unlock the interrupt */
   intUnlock (lockKey);

   /* return TAS test result */
   return (state);

   }

/*****************************************************************************
* sysBusTasClear - clear a location set by sysBusTas()
*
* This routine clears the specified location which is assumed to be a
* TAS flag. It uses a different method for onboard flags as for
* offboard flags.
*
* To prevent deadlocks, interrupts are disabled during the clear operation.
*
* RETURNS: N/A
*
* SEE ALSO: sysBusTas()
*/
void sysBusTasClear
   (
   volatile char * address /* address to be tested-and-cleared */
   )
   {
   int lockKey;

   lockKey = intLock();
   EIEIO_SYNC;
   if(address < (char *)0x40000000)
        sysSemClear((int *) address);    /* Onboard */
   else
        *address = 0;                    /* Offboard */
   intUnlock(lockKey);
   }
```

# 9 SCSI/Ethernet Controller

This section provides information about the SYM53C885 SCSI/Ethernet controller interface.

- General description
- SYM53C885 registers
- Programming notes, SYM53C885

Note: The bit numbering of registers in this section follows the zero-on-the-right convention as opposed to the zero-on-the-left bit numbering convention used by Motorola and IBM in their PowerPC documentation.

# General description

The VGM5 SCSI and Fast Ethernet interface is provided by a single multifunction device, the Symbios Logic SYM53C885 PCI-SCSI/Fast Ethernet Multifunction Controller. On the SCSI side, the SYM53C885 provides a Wide Ultra SCSI interface using a PCI bus master DMA core and Symbios Logic SCSI SCRIPTS processor. On the Ethernet side, the SYM53C885 provides a 10/100Base-T Ethernet interface with independent DMA engines for the transmit and receive channels for access to the motherboard bus and memory with little or no CPU intervention.

The Ethernet connection is available at the VGM5 front panel. The SCSI option is provided with a front panel connector. Refer to Appendix A for connector information.

Listed below are key features of the SYM53C885.

- Fully PCI 2.1 compliant
- Full 32-bit PCI DMA bus master
- High performance SCSI and Ethernet cores, both highly programmable
- Up to 40 MB/s synchronous Wide, Ultra SCSI transfers
- 10/100 Mb/s Ethernet operation

For detailed SYM53C885 programming information, refer to the Symbios Logic Data Manual no. T89962I 1296 15MH. This document (and other supporting documents) can be obtained by contacting:

LSI Logic Corporation
1551 McCarthy Blvd
Milpitas CA 95035
United States
Tel: 408.433.8000
FAX: 408.433.8989
Web: **http://www.lsilogic.com**

A datasheet in PDF format is available from the LSI Logic website in their Tech Library:

Document: (P01964I) SYM53C885 Data Sheet (3/98)

# SYM53C885 registers

The operating system PCI discovery routines dynamically set up the SYM53C885 base address by writing to the appropriate registers in PCI configuration address space. Note that the SYM53C885 is a multifunction PCI device. Thus, SCSI and Ethernet functions are programmed as separate entities by the PCI auto-configuration.

For more information on PCI configuration, refer to the ***PCI implementation details*** discussion in the **PCI bus** chapter in Section 3, page 62, and the ***Setting PCI device base address*** discussion in Section 4, page 103. Also refer to the ***Type 0 configuration table*** on page 110.

The following is an overview of the registers in each interface.

# SCSI registers

## PCI configuration

The PCI configuration registers for the SCSI interface are shown below. Addresses 0x40 through 0xFF are unused.

| 31 | 16 | 15 | 0 | |
|---|---|---|---|---|
| Device ID<br>0x000D | | Vendor ID<br>0x1000 | | 0x00 |
| Status | | Command 0x0000* | | 0x04 |
| Class Code = 0x010000 | | | Revision ID= 0x00 | 0x08 |
| BIST | Header Type | Latency Timer | Cache Line Size | 0x0C |
| Base Address Register 1 | | | | 0x10 |
| Base Address Register 2 | | | | 0x14 |
| Base Address Register 3 | | | | 0x18 |
| Not Supported | | | | 0x1C |
| Not Supported | | | | 0x20 |
| Not Supported | | | | 0x24 |
| Reserved | | | | 0x28 |
| Subsystem ID | | Subsystem Vendor ID | | 0x2C |
| Expansion ROM Base Address | | | | 0x30 |
| Reserved | | | | 0x34 |
| Reserved | | | | 0x38 |
| Max_Lat | Max_Gnt | Interrupt Pin | Interrupt Line | 0x3C |

*Note: Setting bit 0 or bit 1 of the PCI command register enables the SCSI interface to respond to accesses to the PCI Memory or I/O Address space respectively.

## Operating registers

The table below lists the SYM53C885 SCSI registers which are accessed through PCI memory or I/O cycles depending on operation.

**SYM53C885 SCSI registers address and descriptions**

| Addr. Offset | R/W | Label | Description |
|---|---|---|---|
| 0x00 | R/W | SCNTL0 | SCSI Control 0 |
| 0x01 | R/W | SCNTL1 | SCSI Control 1 |
| 0x02 | R/W | SCNTL2 | SCSI Control 2 |
| 0x03 | R/W | SCNTL3 | SCSI Control 3 |
| 0x04 | R/W | SCID | SCSI Chip ID |
| 0x05 | R/W | SXFER | SCSI Transfer |
| 0x06 | R/W | SDID | SCSI Destination ID |
| 0x07 | R/W | GPREG | General Purpose Bits |
| 0x08 | R/W | SFBR | SCSI First Byte Received |
| 0x09 | R/W | SOCL | SCSI Output Control Latch |
| 0x0A | R | SSID | SCSI Selector ID |
| 0x0B | R/W | SBCL | SCSI Bus Control Lines |
| 0x0C | R | DSTAT | DMA Status |
| 0x0D | R | SSTAT0 | SCSI Status 0 |
| 0x0E | R | SSTAT1 | SCSI Status 1 |
| 0x0F | R | SSTAT2 | SCSI Status 2 |
| 0x10-0x13 | R/W | DSA | Data Structure Address |
| 0x14 | R/W | ISTAT | Interrupt Status |
| 0x18 | R/W | CTEST0 | Reserved |
| 0x19 | R/W | CTEST1 | Chip Test 1 |
| 0x1A | R | CTEST2 | Chip Test 2 |
| 0x1B | R | CTEST3 | Chip Test 3 |
| 0x1C-1F | R/W | TEMP | Temporary Register |
| 0x20 | R/W | DFIFO | DMA FIFO |
| 0x21 | R/W | CTEST4 | Chip Test 4 |
| 0x22 | R/W | CTEST5 | Chip Test 5 |
| 0x23 | R/W | CTEST6 | Chip Test 6 |
| 0x24-0x26 | R/W | DBC | DMA Byte Counter |
| 0x27 | R/W | DCMD | DMA Command |
| 0x28-0x2B | R/W | DNAD | DMA Next Address For Data |
| 0x2C-0x2F | R/W | DSP | DMA Scripts Pointer |
| 0x30-0x33 | R/W | DSPS | DMA Scripts Pointer Save |
| 0x34-0x37 | R/W | SCRATCHA | General Purpose Scratch Pad A |
| 0x38 | R/W | DMODE | DMA Mode |

### SYM53C885 SCSI registers address and descriptions (cont.)

| Addr. Offset | R/W | Label | Description |
|---|---|---|---|
| 0x39 | R/W | DIEN | DMA Interrupt Enable |
| 0x3A | R/W | SBR | Scratch Byte Register |
| 0x3B | R/W | DCNTL | DMA Control |
| 0x3C-0x3F | R | ADDER | Sum Output of Internal Adder |
| 0x40 | R/W | SIEN0 | SCSI Interrupt Enable 0 |
| 0x41 | R/W | SIEN1 | SCSI Interrupt Enable 1 |
| 0x42 | R | SIST0 | SCSI Interrupt Status 0 |
| 0x43 | R | SIST1 | SCSI Interrupt Status 1 |
| 0x44 | R/W | SLPAR | SCSI Longitudinal Parity |
| 0x45 | R | SWIDE | SCSI Wide Residue Data |
| 0x46 | R/W | MACNTL | Memory Access Control |
| 0x47 | R/W | GPCNTL | General Purpose Control |
| 0x48 | R/W | STIME0 | SCSI Timer 0 |
| 0x49 | R/W | STIME1 | SCSI Timer 1 |
| 0x4A | R/W | RESPID0 | Response ID 0 |
| 0x4B | R/W | RESPID1 | Response ID 1 |
| 0x4C | R | STEST0 | SCSI Test |
| 0x4D | R | STEST1 | SCSI Test |
| 0x4E | R/W | STEST2 | SCSI Test |
| 0x4F | R/W | STEST3 | SCSI Test |
| 0x50-0x51 | R | SIDL | SCSI Input Data Latch |
| 0x52-0x53 | — | — | Reserved |
| 0x54-0x55 | R/W | SODL | SCSI Output Data Latch |
| 0x56-0x57 | — | — | Reserved |
| 0x58-0x59 | R | SBDL | SCSI Bus Data Lines |
| 0x5A-0x5B | — | — | Reserved |
| 0x5C-0x5F | R/W | SCRATCHB | General Purpose Scratch Pad B |
| 0x60-0x7F | R/W | ScratchC-J | General Purpose Scratch Pad C-J |

# Ethernet registers

## PCI configuration

The PCI configuration registers for the Ethernet interface are shown below. Addresses 0x40 through 0xFF are unused.

| 31 | 16 | 15 | 0 | |
|---|---|---|---|---|
| Device ID<br>0x0701 | | Vendor ID<br>0x1000 | | 0x00 |
| Status | | Command 0x0000* | | 0x04 |
| Class Code = 0x020000 | | | Revision ID= 0x00 | 0x08 |
| BIST | Header Type | Latency Timer | Cache Line Size | 0x0C |
| Base Address Zero (I/O), Ethernet Operating Registers | | | | 0x10 |
| Base Address One (Memory, Ethernet Operating Registers | | | | 0x14 |
| Not Supported | | | | 0x18 |
| Not Supported | | | | 0x1C |
| Not Supported | | | | 0x20 |
| Not Supported | | | | 0x24 |
| Reserved | | | | 0x28 |
| Subsystem ID | | Subsystem Vendor ID | | 0x2C |
| Expansion ROM Base Address | | | | 0x30 |
| Reserved | | | | 0x34 |
| Reserved | | | | 0x38 |
| Max_Lat | Max_Gnt | Interrupt Pin | Interrupt Line | 0x3C |

*Note: Setting bit 0 or bit 1 of the PCI command register enables the Ethernet interface to respond to accesses to the PCI Memory or I/O Address space respectively.

## Operating registers

The table below lists the SYM53C885 Ethernet registers which are accessed through PCI memory or I/O cycles depending on operation.

### SYM53C885 Ethernet registers address and descriptions

| Addr. Offset | R/W | Description |
|---|---|---|
| 0x00-0x03 | R/W | Transmit Channel Control |
| 0x04-0x05 | R/W | Transmit Channel Status |
| 0x06-0x07 | — | Reserved |
| 0x08-0x0B | — | Reserved |
| 0x0C-0x0F | R/W | Transmit CommandPtrLo |
| 0x10-0x13 | R/W | Transmit InterruptSelect |
| 0x14-0x17 | R/W | Transmit BranchSelect |
| 0x18-0x1B | R/W | Transmit WaitSelect |
| 0x1C-3F | — | Reserved |
| 0x40-0x43 | R/W | Receive ChannelControl |
| 0x44-0x45 | R | Receive ChannelStatus |
| 0x46-0x47 | — | Reserved |
| 0x48-0x4B | — | Reserved |
| 0x4C-0x4F | R | Receive CommandPtrLo |
| 0x50-0x53 | R | Receive InterruptSelect |
| 0x54-0x57 | R | Receive BranchSelect |
| 0x58-0x5B | R/W | Receive WaitSelect |
| 0x5C-0x7F | — | Reserved |
| 0x80-0x81 | R/W | EventStatus |
| 0x82-0x83 | R/W | InterruptEnable |
| 0x84-0x85 | R | InterruptClear |
| 0x86-0x87 |  | InterruptStatus |
| 0x88-0x8B | — | Reserved |
| 0x8C-0x8F |  | Chip Revision |
| 0x90-0x93 |  | DBDMA Control |
| 0x0x94 |  | TxThreshold |
| 0x95-0x97 | — | Reserved |
| 0x98-0x9B | — | Reserved |
| 0x9C-0x9D | — | Reserved |
| 0x9E |  | General Purpose |
| 0x9F |  | General Purpose Control |
| 0xA0-0xA1 | R | Configuration |
| 0xA2-0xA3 | R/W | Back-to-Back Interpacket Gap |
| 0xA4-0xA5 | R/W | Non Back-to-Back Interpacket Gap |
| 0xA6-0xA7 | R/W | MIIM Command |

### SYM53C885 Ethernet registers address and descriptions (cont.)

| Addr. Offset | R/W | Description |
|---|---|---|
| 0xA8-0xA9 | R/W | MII Address or TP_PMD Control |
| 0xAA-0xAB | R/W | MII Write Data |
| 0xAC-0xAD | R/W | MII Read Data |
| 0xAE-0xAF | R/W | MII Indicators |
| 0xB0-0xCF | R/W | Reserved |
| 0xD0-0xD1 | R/W | Address Filter |
| 0xD2-0xD3 | R/W | Station Address 0 |
| 0xD4-0xD5 | R/W | Station Address 1 |
| 0xD6-0xD7 | R/W | Station Address 2 |
| 0xD8-0xD9 | R/W | Hash Table 0 |
| 0xDA-0xDB | R/W | Hash Table 1 |
| 0xDC-0sDD | R/W | Hash Table 2 |
| 0xDE-0xDF | R/W | Hash Table 3 |
| 0xE0-0xE3 | — | Reserved |
| 0xE4-0xE5 | R/W | PHY Identifier 0 |
| 0xE6-0xE7 | R/W | PHY Identifier 1 |
| 0xE8-0xEB | — | Reserved |
| 0xEC-0xEF | — | Reserved |
| 0xF0 | R/W | EE Status |
| 0xF1 | R/W | EE Control |
| 0xF2 | R/W | EE Word Address |
| 0xF3 | R/W | EE Read Data |
| 0xF4 | R/W | EE Write Data |
| 0xF5 | R/W | EE Feature Enable |
| 0xF6-0xF7 | — | Reserved |
| 0xF8-0xFB | — | Reserved |
| 0xFC-0xFF | — | Reserved |

# Programming notes, SYM53C885

## SCSI prematurely surrendering PCI bus

### Problem

Slow SCSI transfers to SCSI drive when VME BLT's were occurring.

### Observation

The SYM53C885 SCSI chip was surrendering the PCI bus in the middle of a burst read transfer when the Universe II PCI-VME bridge needed to write data. This was slowing down the SCSI transfers in the face of VME traffic.

### Solution

The solution is to write a value of 20 decimal or greater to the SCSI chip's PCI Latency register (0x0D; or 0x0E if big-endian). The C code to do this in pSOS or VxWorks is below.

```
SetLatency( int value )
{
   int temp
   value = value & 0xFF; /* Mask value to range of 0-255 */
   temp = readMPC( 0x8000600C);
   temp = temp &~0xff00;
   writeMPC( 0x8000600C, temp | (value <<8) );
            /* numbering bytes from right to left, byte 0 is
             * the Cache Line length byte at offset 0xC,
             * and byte 1 is the Latency timer byte at offset 0xD,
             * which is defined as number of PCI clocks of latency.
             */
}
```

Note that this register should always be set with this value. There is no downside to setting the PCI Latency register to 20, since that's the length of one burst transfer. This means that it's the optimal setting for use with the Grackle.

# 10 PCI-PCI Bridge Interface

This section provides information about the PCI-PCI bridge chip that provides the P0•PCI™ bridge interface.

- General description
- Registers
- Configuration
- Software support, P0•PCI™

# General description

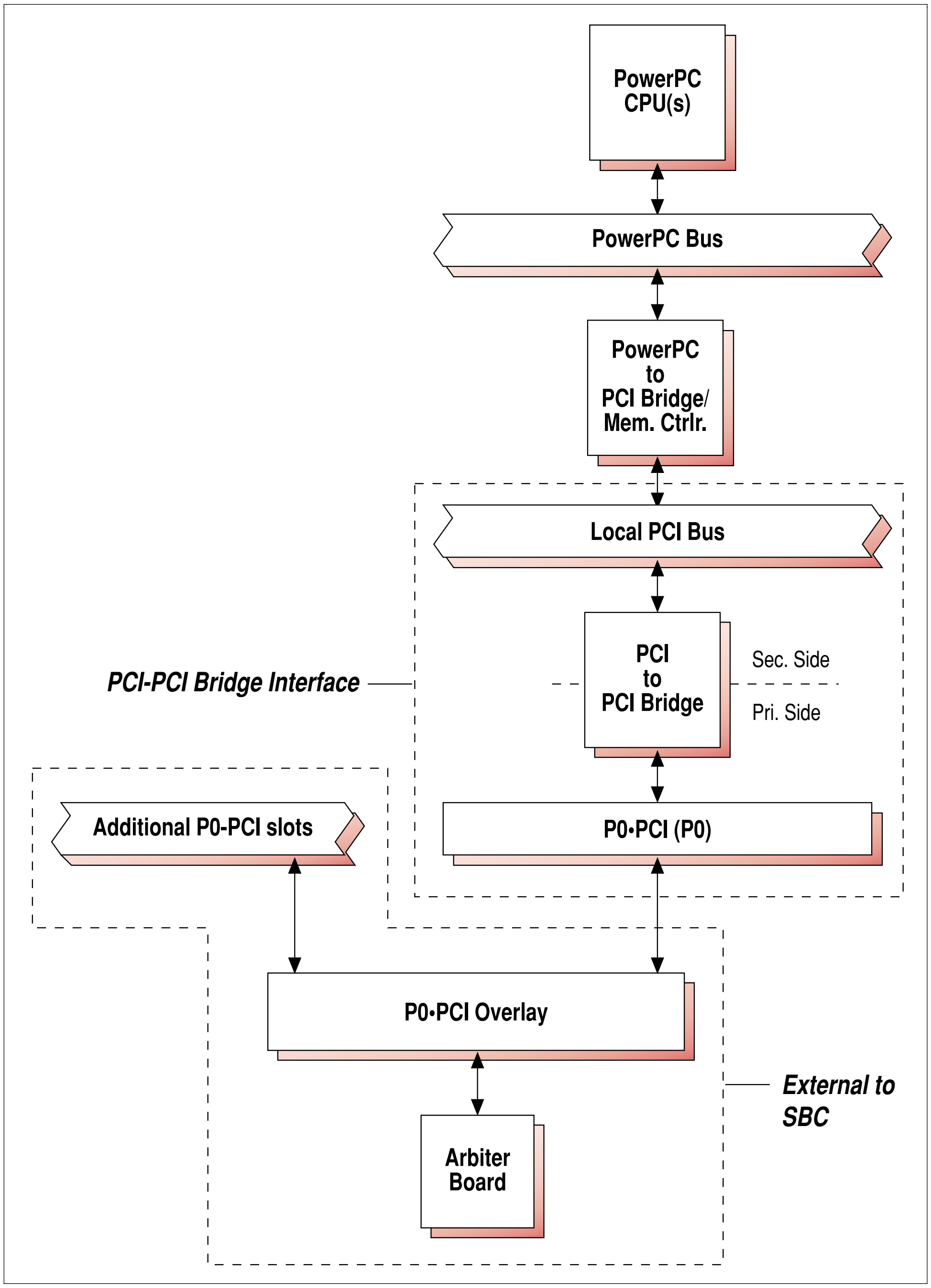
Synergy's P0•PCI™ bus interface provides the VGM5 with a sub-bus for board-to-board data transfers that are speedier than the VMEbus. The P0•PCI™ bus is 64-bits wide and operates at 33 MHz, for a theoretical maximum aggregate bandwidth of 266 MB/sec. Up to 4 pair of SBCs (8 slots) are allowed on the P0•PCI bus. A single VGM5 can write data to another VGM5 at about 72 MB/sec over the P0•PCI™ bus. However, the rest of the P0•PCI™ bandwidth is not consumed, so other transfers can occur concurrently up to the maximum bandwidth. The P0•PCI™ bus interface is based on Intel's (formerly Digital Semiconductor's) 21554 64-bit PCI-to-PCI bridge chip.

The 21554 chip has two PCI ports, primary and secondary. The primary port is connected to P0 and the secondary port is connected to the VGM5's onboard PCI bus. This connection scheme allows the PCI-PCI bridge to be used for peer-to-peer communication between multiple VGM5 boards. Up to 8 boards can be connected together using the P0•PCI™ bus.

The P0•PCI™ interface uses the VME64x P0 connector which is in-between the VME P1 and P2 connectors. Note that the VGM5's P0 connector may interfere with board insertion in non-VME64x compatible backplanes. The VGM5 P0•PCI™ interface works in conjunction with a P0•PCI™ overlay board. See the ***Installing the P0 overlay*** chapter in Section 2, page 45, for model descriptions and installation information. (For ordering information of these components, contact Synergy Microsystems' Customer Service.)

The external P0•PCI™ overlay board connects P0•PCI™ slots together from the back side of the VME64x backplane. The overlay board (with the exception of passive models) includes a plug-in arbiter board which provides clock generation and round-robin PCI bus arbitration for all interconnected boards on the P0•PCI™ bus. The arbiter board also provides power-on reset to all devices on the primary side of P0•PCI™ interface.

The diagram below shows the VGM5 PCI-PCI bridge interface in relation to the other major busses in the system.



**Block diagram VGM5 PCI-PCI bridge**

# Registers

The 21554 contains the following register groups:

- Primary and Secondary interface PCI configuration header registers
- Device-specific configuration registers
- Memory and I/O mapped control and status registers

For detailed information on the 21554 PCI-to-PCI bridge, refer to the "21554 PCI-to-PCI Bridge for Embedded Applications, Hardware Reference Manual" available from Intel. This document is available as a PDF file from Intel's developer website:

**http://developer.intel.com/design/bridge/**

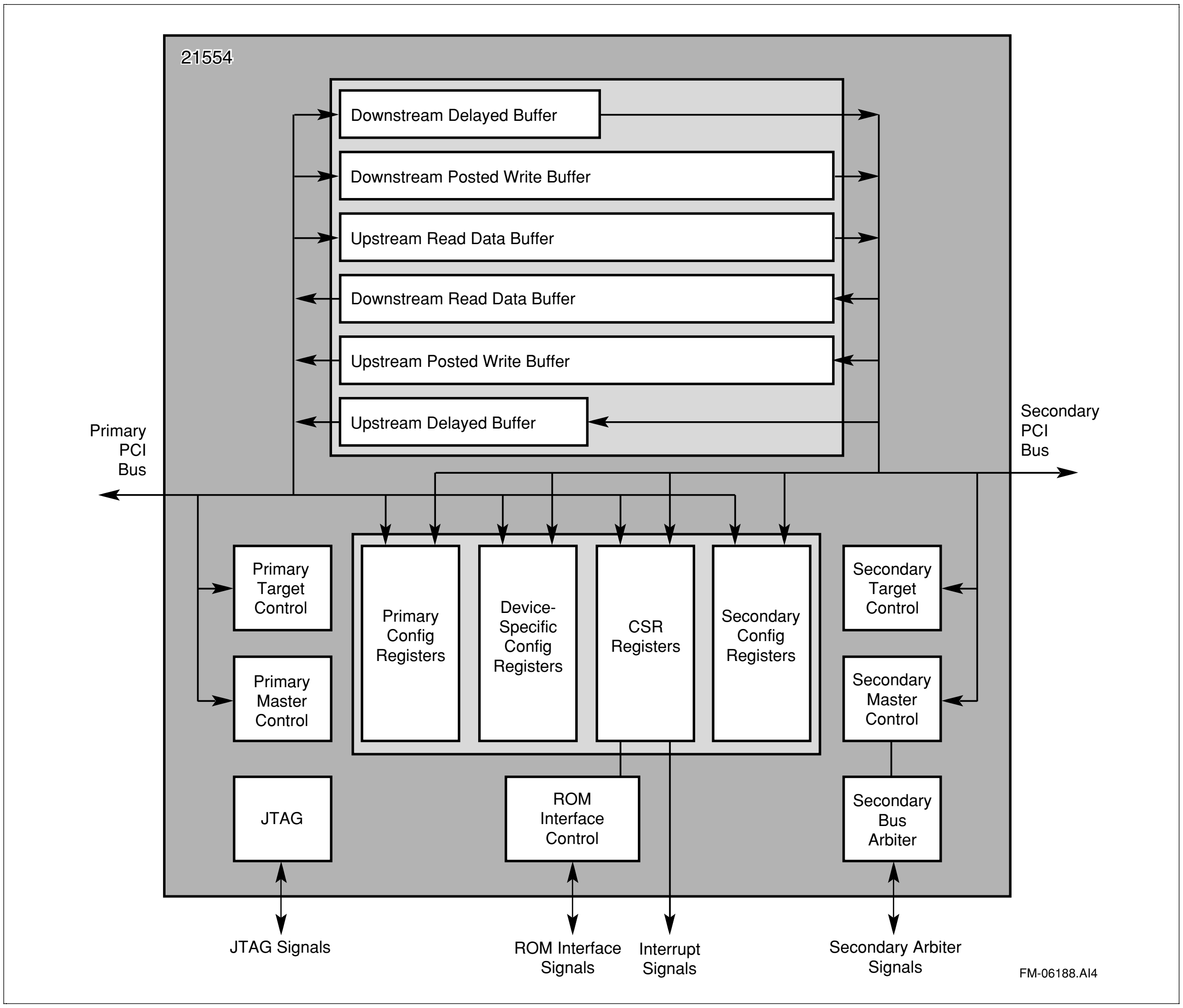
Application notes, datasheets, and manuals for Intel's PCI bridges can be found at this site.

The table below lists recommended related documents available as PDF files from the Intel developer website (online availability subject to change without notice).

***Related 21554 documents available for download***

| **Document** | **Date** | **Size** |
|---|---|---|
| Specification Update | Aug 1998 | 88 Kbytes |
| Product Preview Datasheet | | 797 Kbytes |
| Bridge Performance Optimization, AppNote | Sep 1998 | 72 Kbytes |
| Getting Started..., AppNote | Sep 1998 | 94 Kbytes |
| Issues w/ Host Processor Card..., AppNote | Oct 1998 | 68 Kbytes |
| Embedded Applications...Hardware Reference, Manual | Sep 1998 | 1075 Kbytes |

The diagram below shows the microarchitecture of the 21554 PCI-PCI bridge.



*Microarchitecture, 21554 PCI-PCI bridge chip*

# Configuration

## Introduction

The 21554 configuration space is divided into three parts:

- Primary interface configuration registers
- Secondary interface configuration registers
- Device-specific configuration registers

Both the primary and secondary interface configuration headers contain the 64-byte Type 0 configuration header corresponding to that interface. The device-specific configuration registers are specific to the 21554, some of which apply to the primary interface, others to the secondary interface, and some to other 21554 functions.

Access to the 21554 configuration registers is supported from both the primary (P0•PCI) and secondary (onboard PCI) interfaces. Normally, however, access to the 21554 configuration space is allowed from the secondary interface only. In this case, the 21554 returns target retry to all accesses initiated on the primary bus, with the exception of accesses to the Reset Control register at Dword 0xD8. Clearing the Primary Lockout Bit in the Chip Control 0 register allows access to 21554 configuration space from the primary side.

## *PCI configuration*

The operating system PCI discovery routines dynamically set up the 21554 base address by writing to the appropriate registers in PCI configuration address space.

The host accesses devices behind the bridge (i.e., devices outside the local PCI bus) with Type 1 PCI configuration accesses which contain extra data for bus number and device number. Type 1 accesses are intended for PCI-PCI bridges only. If a bridge detects that the bus number is not to the secondary bus of the bridge, the access is passed through unchanged. If the bus number matches the secondary bus number, the bridge converts the access to a Type 0 PCI configuration access. The device number is then decoded and the proper IDSel asserted to configure the device on the secondary bus.

For more information on PCI configuration, refer to the ***PCI implementation details*** discussion in the **PCI bus** chapter in Section 3, page 62, and the ***Setting PCI device base address*** discussion in Section 4, page 103. Also refer to the ***Type 0 configuration table*** on page 110.

## *P0•PCI™ configuration*

The implementation of configuring the P0•PCI™ bus for each board connected to the bus, the address ranges and any address translation, are application-specific. It is recommended that each board be allocated at least one 256 KB window into the onboard memory.

# Software support, PO•PCI™

Synergy Microsystems provides optional software to support board-to-board communications using the VGM5's P0•PCI™ interface.

- **Global Buffer Manager (GBM) Software Package** – The GBM software package allows sharing of data among processors on Synergy PPC CPU boards connected via the P0•PCI™ bus backplane. GBM is available for Linux and VxWorks (contact factory for availability).
- **Board Support Package (BSP), Linux and VxWorks** – Synergy Microsystems' Linux and VxWorks BSP distribution includes sample software for P0•PCI™.

Contact Synergy Customer Service for more information about these software options.

# 11 Warranties & Service

This section provides information on product warranty and support.

- Warranty terms and options
- Customer service

# Warranty terms & options

This chapter describes the warranty terms and options provided for the VGM5 SBC.

## Warranty terms

Synergy Microsystems, Inc. warrants all standard (off-the-shelf) and non-standard (custom) products to be free of defects in materials and workmanship under normal use for the applicable warranty period (as described below). This limited warranty is void if the failure has resulted from accident, abuse, alteration, or misapplication by the customer.

## Product returns

The following guidelines describe warranty terms for product returns.

- **Initial product acceptance** — Synergy presumes that customers will inspect products within 14 days of receipt for conformance to the specifications stated in this manual (for standard, off-the-shelf units) and/or purchasing documentation (for custom units). Products not rejected within this period are considered by Synergy to be accepted by the customer.
- **Delivery rejection** — Products that do not conform to the specifications and standards in this manual or purchase documents can be returned to Synergy for replacement/repair. Before returning products, notify Synergy of the problem and get a Return Material Authorization (RMA) number. Board rejection will not be valid unless boards are returned in the original shipping cartons within 10 days of the receipt of the RMA number.

For more information about returning products, see the next chapter on *Customer service*.

If the customer adheres to these requirements, Synergy agrees to pay shipping charges, otherwise shipment costs must be paid by the customer.

- **Delivery turnaround after rejection** — Synergy's service goal is to return new or refurbished products within 14 days of the receipt of properly rejected boards that were returned in accordance with the requirements stated above and in next chapter.
- **Product returns under warranty** — Once products have been either accepted or the initial product accept/reject period has passed, products are warranted for the applicable warranty period as described below:

  For information about returning products under warranty, see the next chapter on *Customer service*.

## *Warranty periods*

Synergy Microsystems, Inc. offers the following warranty periods:

- **90-day guarantee and limited warranty** — All standard (off-the-shelf), non-standard (custom) products, software (without a purchased maintenance agreement), and out-of-warranty RM repairs are automatically guaranteed for 90 days from the day of delivery.
- **1-year standard limited warranty** — Customers who complete payment for the product to Synergy within **30 days** of delivery receive a free warranty extension for a full year on all products covered by the payment.
- **Extended limited warranty** — If desired, Synergy offers an extended warranty for an additional charge. Contact Customer Service for details.

# Customer service

Please contact Synergy Microsystems, Inc. if you have any questions, comments, or suggestions. You can contact our **customer service** department by writing or calling:

**Synergy Microsystems, Inc.**
**9605 Scranton Rd., Suite 700**
**San Diego, CA  92121-1773**
**(858) 452-0020**
**(858) 452-0060 (FAX)**
**Web: http://www.synergymicro.com**
**E-mail: sales-info@synergymicro.com**

## Reporting problems

If you encounter any difficulty with your VGM5 board, call Synergy customer service. If possible, please have the following information available to assist our staff in assessing your problem:

- VGM5 model number (silk-screened on solder side of board)
- Serial number marked on solder side of board
- VGM5 revision level (silk-screened on the solder side of board)
- ECO level (marked on solder side of board)

## Return policies and procedures

Should it become necessary to return a board to Synergy for repair, please take the following steps.

❶ Call Synergy Microsystems, Inc. customer service for a **Return Merchandise Authorization** (RMA) number. Use this number in all communications regarding the problem boards.

❷ Provide the following information with all returned items:

- VGM5 model number (solder side of the board)
- Serial number (solder side of board)
- VGM5 revision level (solder side of board)
- ECO level (solder side of board)
- Purchase order number and billing address if the board is out of warranty.
- Customer contact name, address, and telephone number
- Complete description of the problem.

❸ Carefully package the board to protect it during shipment; be sure that it is enclosed in an anti-static bag.

❹ Mark the RMA number on the shipping container.

❺ Send the board and the requested information prepaid to Synergy at the following address:

**Synergy Microsystems, Inc.**
**9605 Scranton Rd., Suite 700**
**San Diego, CA 92121-1773**

An inspection and test charge will be applicable to all units returned for repair, unless the unit is found to be defective and under warranty. If the repair charge exceeds the inspection and test charge, we will notify you of the repair charge. The test and inspection charge will be applied to your repair charge. No repair (other than test and inspection) will be performed on products that are out of warranty until we have received your approval for the charges.

We appreciate your cooperation with these procedures. They help us give you the best possible service.

# Appendix A, Connectors & Cables

This appendix contains descriptions and diagrams of the VGM5 connectors and specialized cabling.

- VMEbus connectors (P1 & P2)
- PMC connectors (P11– P15)
- P0•PCI™ bus connector (P0)
- Memory module connector (PM22)
- Memory module connectors (PM1 & PM2)
- Wide Ultra SCSI connector (P264/P262)
- Fast Ethernet connector (P240/P238)
- Asynchronous serial connector (P347/P345)
- Serial I/O cabling options

# VMEbus connectors (P1 & P2)

The VGM5's P1 and P2 connectors provide the standard I/O interface to the VMEbus as listed in the table below:

The P2 connector shows the signals Synergy has assigned to the user-defined pins for rows A and C (and Z and D for 5-row option) on the standard VMEbus. These rows are connected to the PMC I/O (P14 & P15) connectors listed later in this chapter.

***VMEbus P1 connector pinouts***

| Pin | Row Z[1] | Row A | Row B | Row C | Row D[1] |
|---|---|---|---|---|---|
| **1** | — | D0 | BBsy\ | D8 | +5VPrecharge |
| **2** | Gnd | D1 | BClr\ | D9 | Gnd |
| **3** | — | D2 | ACFail\ | D10 | — |
| **4** | Gnd | D3 | BG0In\ | D11 | — |
| **5** | — | D4 | BG0Out\ | D12 | — |
| **6** | Gnd | D5 | BG1In\ | D13 | — |
| **7** | — | D6 | BG1Out\ | D14 | — |
| **8** | Gnd | D7 | BG2In\ | D15 | — |
| **9** | — | Gnd | BG2Out\ | Gnd | GAP\ |
| **10** | Gnd | SysClk | BG3In\ | SysFail\ | GA0\ |
| **11** | — | Gnd | BG3Out\ | BErr\ | GA1\ |
| **12** | Gnd | DS1\ | BR0\ | SysReset\ | +3.3V |
| **13** | — | DS0\ | BR1\ | LWord\ | GA2\ |
| **14** | Gnd | Write\ | BR2\ | AM5 | +3.3V |
| **15** | — | Gnd | BR3\ | A23 | GA3\ |
| **16** | Gnd | DTAck\ | AM0 | A22 | +3.3V |
| **17** | — | Gnd | AM1 | A21 | GA4\ |
| **18** | Gnd | AS\ | AM2 | A20 | +3.3V |
| **19** | — | Gnd | AM3 | A19 | — |
| **20** | Gnd | IAck\ | Gnd | A18 | +3.3V |
| **21** | — | IAckIn\ | — | A17 | — |
| **22** | Gnd | IAckOut\ | — | A16 | +3.3V |
| **23** | — | AM4 | Gnd | A15 | — |
| **24** | Gnd | A7 | IRq7\ | A14 | +3.3V |
| **25** | — | A6 | IRq6\ | A13 | — |
| **26** | Gnd | A5 | IRq5\ | A12 | +3.3V |
| **27** | — | A4 | IRq4\ | A11 | LII\ |
| **28** | Gnd | A3 | IRq3\ | A10 | +3.3V |
| **29** | — | A2 | IRq2\ | A9 | LIO\ |
| **30** | Gnd | A1 | IRq1\ | A8 | +3.3V |
| **31** | — | -12V | — | +12V | Gnd |
| **32** | Gnd | +5V | +5V | +5V | +5VPrecharge |

***Notes:*** 1. This row present only with optional wide (160-pin) VMEbus P1 & P2 connectors.

### VMEbus P2 connector pinouts

| Pin | Row Z[1,2] | Row A [1] | Row B | Row C[1] | Row D[1,2] |
|---|---|---|---|---|---|
| 1 | (UsrIO66) | (UsrIO2) | +5V | (UsrIO1) | (UsrIO65) |
| 2 | Gnd | (UsrIO4) | Gnd | (UsrIO3) | (UsrIO67) |
| 3 | (UsrIO69) | (UsrIO6) | Retry\ | (UsrIO5) | (UsrIO68) |
| 4 | Gnd | (UsrIO8) | A24 | (UsrIO7) | (UsrIO70) |
| 5 | (UsrIO72) | (UsrIO10) | A25 | (UsrIO9) | (UsrIO71) |
| 6 | Gnd | (UsrIO12) | A26 | (UsrIO11) | (UsrIO73) |
| 7 | (UsrIO75) | (UsrIO14) | A27 | (UsrIO13) | (UsrIO74) |
| 8 | Gnd | (UsrIO16) | A28 | (UsrIO15) | (UsrIO76) |
| 9 | (UsrIO78) | (UsrIO18) | A29 | (UsrIO17) | (UsrIO77) |
| 10 | Gnd | (UsrIO20) | A30 | (UsrIO19) | (UsrIO79) |
| 11 | (UsrIO81) | (UsrIO22) | A31 | (UsrIO21) | (UsrIO80) |
| 12 | Gnd | (UsrIO24) | Gnd | (UsrIO23) | (UsrIO82) |
| 13 | (UsrIO84) | (UsrIO26) | +5V | (UsrIO25) | (UsrIO83) |
| 14 | Gnd | (UsrIO28) | D16 | (UsrIO27) | (UsrIO85) |
| 15 | (UsrIO87) | (UsrIO30) | D17 | (UsrIO29) | (UsrIO86) |
| 16 | Gnd | (UsrIO32) | D18 | (UsrIO31) | (UsrIO88) |
| 17 | (UsrIO90) | (UsrIO34) | D19 | (UsrIO33) | (UsrIO89) |
| 18 | Gnd | (UsrIO36) | D20 | (UsrIO35) | (UsrIO91) |
| 19 | (UsrIO93) | (UsrIO38) | D21 | (UsrIO37) | (UsrIO92) |
| 20 | Gnd | (UsrIO40) | D22 | (UsrIO39) | (UsrIO94) |
| 21 | (UsrIO96) | (UsrIO42) | D23 | (UsrIO41) | (UsrIO95) |
| 22 | Gnd | (UsrIO44) | Gnd | (UsrIO43) | (UsrIO97) |
| 23 | (UsrIO99) | (UsrIO46) | D24 | (UsrIO45) | (UsrIO98) |
| 24 | Gnd | (UsrIO48) | D25 | (UsrIO47) | (UsrIO100) |
| 25 | (UsrIO102) | (UsrIO50) | D26 | (UsrIO49) | (UsrIO101) |
| 26 | Gnd | (UsrIO52) | D27 | (UsrIO51) | (UsrIO103) |
| 27 | (UsrIO105) | (UsrIO54) | D28 | (UsrIO53) | (UsrIO104) |
| 28 | Gnd | (UsrIO56) | D29 | (UsrIO55) | (UsrIO106) |
| 29 | (UsrIO108) | (UsrIO58) | D30 | (UsrIO57) | (UsrIO107) |
| 30 | Gnd | (UsrIO60) | D31 | (UsrIO59) | (UsrIO109) |
| 31 | (UsrIO110) | (UsrIO62) | Gnd | (UsrIO61) | Gnd |
| 32 | Gnd | (UsrIO64) | +5V | (UsrIO63) | +5VPrecharge |

***Notes:***

1. Pins in this row connect to the P14 or P15 PMC connector pin indicated in parentheses. Space is provided in these columns to write in the assigned signals, if desired. Refer to the applicable PMC module documentation for P2 pin assignments.
2. This row present only with optional wide (160-pin) VMEbus P1 & P2 connectors.

# PMC connectors (P11–P15)

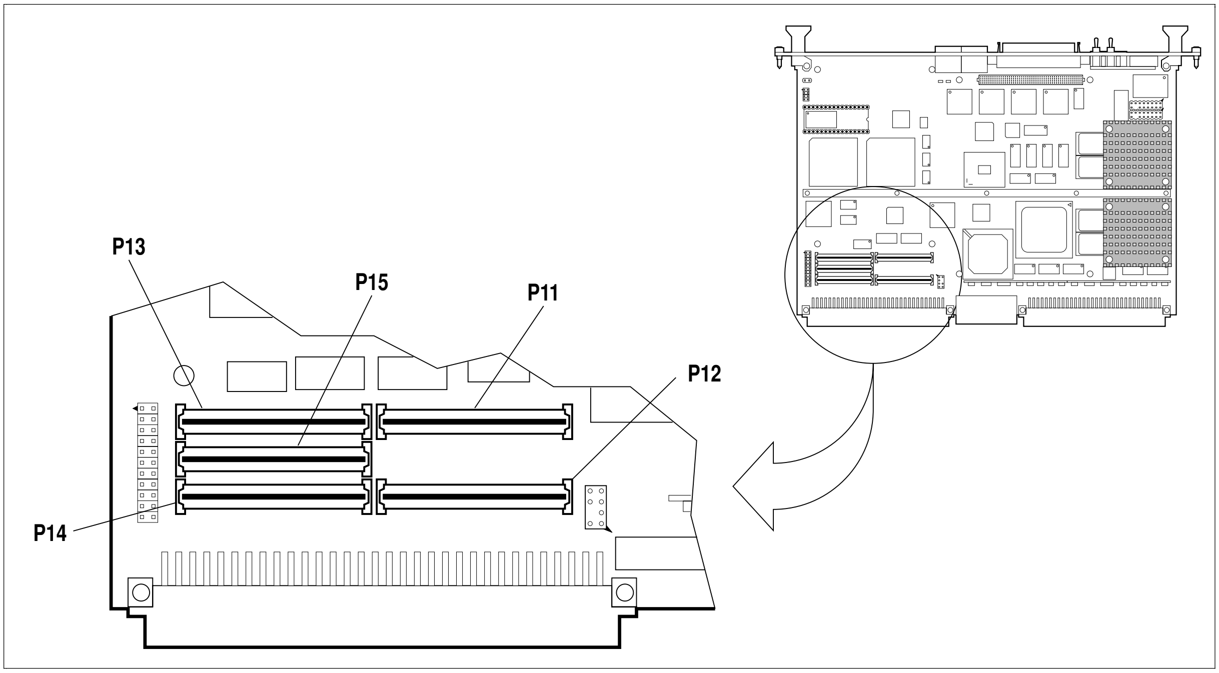
The PMC connectors provide the connection of an add-on PMC (PCI mezzanine card) card or PMC expansion board for various types of I/O options. These connectors have the following functions:

- P11, P12 — 32-/64-bit PCI
- P13 — 64-bit PCI (provided as an option)
- P14 — PMC User I/O
- P15 — Synergy-proprietary stacking and I/O (provided as an option)

Synergy Microsystems has made a few minor changes to the standard PMC connector pinout to support the PEX3 PMC Expansion card. These changes are:

- The JTAG test port is not supported. These pins are used for the bus request, grant, and clock connections to the PEX3.
- The IDSel pin is wired to AD13 on the PMC connector.
- VI/O is connected to 5V. This means that all PMC signals are at 5V logic levels.

The drawing below shows the locations of the VGM5 PMC connectors. The tables that follow list the pin assignments of these connectors. (Refer to PMC card installation instructions in Section 2 page 39.)



*PMC connectcor (P11–P15) locations*

### *PMC connector, P11 pinouts*

| Pin | Function | Pin | Function |
|---|---|---|---|
| **1** | PEX3_Gnt\ (TCK) | **2** | -12V |
| **3** | Gnd | **4** | IntA\ |
| **5** | IntB\ | **6** | IntC\ |
| **7** | — | **8** | +5V |
| **9** | IntD\ | **10** | Reserved |
| **11** | Gnd | **12** | Reserved |
| **13** | Clk2b | **14** | Gnd |
| **15** | Gnd | **16** | Gnt\ |
| **17** | Req\ | **18** | +5V |
| **19** | VI/O (5V) | **20** | AD31 |
| **21** | AD28 | **22** | AD27 |
| **23** | AD25 | **24** | Gnd |
| **25** | Gnd | **26** | CBE3\ |
| **27** | AD22 | **28** | AD21 |
| **29** | AD19 | **30** | +5V |
| **31** | VI/O (5V) | **32** | AD17 |
| **33** | Frame\ | **34** | Gnd |
| **35** | Gnd | **36** | IRdy\ |
| **37** | DevSel\ | **38** | +5V |
| **39** | Gnd | **40** | Lock\ |
| **41** | — (SDONE) | **42** | — (SBO\) |
| **43** | Par | **44** | Gnd |
| **45** | VI/O (5V) | **46** | AD15 |
| **47** | AD12 | **48** | AD11 |
| **49** | AD9 | **50** | Vcc5V |
| **51** | Gnd | **52** | CBE0\ |
| **53** | AD6 | **54** | AD5 |
| **55** | AD4 | **56** | Gnd |
| **57** | VI/O (5V) | **58** | AD3 |
| **59** | AD2 | **60** | AD1 |
| **61** | AD0 | **62** | +5V |
| **63** | Gnd | **64** | Req64B\ |

### *PMC connector, P12 pinouts*

| Pin | Function | Pin | Function |
|---|---|---|---|
| **1** | +12V | **2** | Gnd (TRst\) |
| **3** | PEX3_Req\ (TMS) | **4** | — (TDO) |
| **5** | PEX3_Clk (TDI) | **6** | Gnd |
| **7** | Gnd | **8** | Reserved |
| **9** | Reserved | **10** | Reserved |
| **11** | — | **12** | +3.3V |
| **13** | Rst\ | **14** | — |
| **15** | Vcc3V | **16** | — |
| **17** | Reserved | **18** | Gnd |
| **19** | AD30 | **20** | AD29 |
| **21** | Gnd | **22** | AD26 |
| **23** | AD24 | **24** | +3.3V |
| **25** | AD13 (IDSel) | **26** | AD23 |
| **27** | +3.3V | **28** | AD20 |
| **29** | AD18 | **30** | Gnd |
| **31** | AD16 | **32** | CBE2\ |
| **33** | Gnd | **34** | Reserved |
| **35** | TRdy\ | **36** | +3.3V |
| **37** | Gnd | **38** | Stop\ |
| **39** | PErr\ | **40** | Gnd |
| **41** | +3.3V | **42** | SErr\ |
| **43** | CBE1\ | **44** | Gnd |
| **45** | AD14 | **46** | AD13 |
| **47** | Gnd | **48** | AD10 |
| **49** | AD8 | **50** | +3.3V |
| **51** | AD7 | **52** | Reserved |
| **53** | +3.3V | **54** | Reserved |
| **55** | Reserved | **56** | Gnd |
| **57** | Reserved | **58** | Reserved |
| **59** | Gnd | **60** | Reserved |
| **61** | Ack64B\ | **62** | +3.3V |
| **63** | Gnd | **64** | Reserved |

### PMC connector, P13 pinouts

| Pin | Function | Pin | Function |
|---|---|---|---|
| **1** | Reserved | **2** | Gnd |
| **3** | Gnd | **4** | CBE7\ |
| **5** | CBE6\ | **6** | CBE5\ |
| **7** | CBE4\ | **8** | Gnd |
| **9** | VI/O (5V) | **10** | Par64 |
| **11** | AD63 | **12** | AD62 |
| **13** | AD61 | **14** | Gnd |
| **15** | Gnd | **16** | AD60 |
| **17** | AD59 | **18** | AD58 |
| **19** | AD57 | **20** | Gnd |
| **21** | VI/O (5V) | **22** | AD56 |
| **23** | AD55 | **24** | AD54 |
| **25** | AD53 | **26** | Gnd |
| **27** | Gnd | **28** | AD52 |
| **29** | AD51 | **30** | AD50 |
| **31** | AD49 | **32** | Gnd |
| **33** | Gnd | **34** | AD48 |
| **35** | AD47 | **36** | AD46 |
| **37** | AD45 | **38** | Gnd |
| **39** | VI/O (5V) | **40** | AD44 |
| **41** | AD43 | **42** | AD42 |
| **43** | AD41 | **44** | Gnd |
| **45** | Gnd | **46** | AD40 |
| **47** | AD39 | **48** | AD38 |
| **49** | AD37 | **50** | Gnd |
| **51** | Gnd | **52** | AD36 |
| **53** | AD35 | **54** | AD34 |
| **55** | AD33 | **56** | Gnd |
| **57** | VI/O (5V) | **58** | AD32 |
| **59** | Reserved | **60** | Reserved |
| **61** | Reserved | **62** | Gnd |
| **63** | Gnd | **64** | Reserved |

### PMC connector, P14 pinouts

| Pin | Function[1] | Pin | Function[1] |
|---|---|---|---|
| 1 | UsrIO1 | 2 | UsrIO2 |
| 3 | UsrIO3 | 4 | UsrIO4 |
| 5 | UsrIO5 | 6 | UsrIO6 |
| 7 | UsrIO7 | 8 | UsrIO8 |
| 9 | UsrIO9 | 10 | UsrIO10 |
| 11 | UsrIO11 | 12 | UsrIO12 |
| 13 | UsrIO13 | 14 | UsrIO14 |
| 15 | UsrIO15 | 16 | UsrIO16 |
| 17 | UsrIO17 | 18 | UsrIO18 |
| 19 | UsrIO19 | 20 | UsrIO20 |
| 21 | UsrIO21 | 22 | UsrIO22 |
| 23 | UsrIO23 | 24 | UsrIO24 |
| 25 | UsrIO25 | 26 | UsrIO26 |
| 27 | UsrIO27 | 28 | UsrIO28 |
| 29 | UsrIO29 | 30 | UsrIO30 |
| 31 | UsrIO31 | 32 | UsrIO32 |
| 33 | UsrIO33 | 34 | UsrIO34 |
| 35 | UsrIO35 | 36 | UsrIO36 |
| 37 | UsrIO37 | 38 | UsrIO38 |
| 39 | UsrIO39 | 40 | UsrIO40 |
| 41 | UsrIO41 | 42 | UsrIO42 |
| 43 | UsrIO43 | 44 | UsrIO44 |
| 45 | UsrIO45 | 46 | UsrIO46 |
| 47 | UsrIO47 | 48 | UsrIO48 |
| 49 | UsrIO49 | 50 | UsrIO50 |
| 51 | UsrIO51 | 52 | UsrIO52 |
| 53 | UsrIO53 | 54 | UsrIO54 |
| 55 | UsrIO55 | 56 | UsrIO56 |
| 57 | UsrIO57 | 58 | UsrIO58 |
| 59 | UsrIO59 | 60 | UsrIO60 |
| 61 | UsrIO61 | 62 | UsrIO62 |
| 63 | UsrIO63 | 64 | UsrIO64 |

Note: 1. The function of pins labeled ‘UsrIOxxx’ depends on the add-on card installed on the board. Space is provided in these columns to write the assigned signals if desired.

### PMC connector, P15 pinouts

| Pin | Function[1] | Pin | Function[1] |
|---|---|---|---|
| 1 | UsrIO65 | 2 | UsrIO66 |
| 3 | UsrIO67 | 4 | Gnd |
| 5 | UsrIO68 | 6 | UsrIO69 |
| 7 | UsrIO70 | 8 | — |
| 9 | UsrIO71 | 10 | UsrIO72 |
| 11 | UsrIO73 | 12 | Gnd |
| 13 | UsrIO74 | 14 | UsrIO75 |
| 15 | UsrIO76 | 16 | — |
| 17 | UsrIO77 | 18 | UsrIO78 |
| 19 | UsrIO79 | 20 | Gnd |
| 21 | UsrIO80 | 22 | UsrIO81 |
| 23 | UsrIO82 | 24 | — |
| 25 | UsrIO83 | 26 | UsrIO84 |
| 27 | UsrIO85 | 28 | Gnd |
| 29 | UsrIO86 | 30 | UsrIO87 |
| 31 | UsrIO88 | 32 | — |
| 33 | UsrIO89 | 34 | UsrIO90 |
| 35 | UsrIO91 | 36 | Gnd |
| 37 | UsrIO92 | 38 | UsrIO93 |
| 39 | UsrIO94 | 40 | — |
| 41 | UsrIO95 | 42 | UsrIO96 |
| 43 | UsrIO97 | 44 | Gnd |
| 45 | UsrIO98 | 46 | UsrIO99 |
| 47 | UsrIO100 | 48 | — |
| 49 | UsrIO101 | 50 | UsrIO102 |
| 51 | UsrIO103 | 52 | Gnd |
| 53 | UsrIO104 | 54 | UsrIO105 |
| 55 | UsrIO106 | 56 | — |
| 57 | UsrIO107 | 58 | UsrIO108 |
| 59 | UsrIO109 | 60 | Gnd |
| 61 | Gnd | 62 | UsrIO110 |
| 63 | +5VPrecharge | 64 | — |

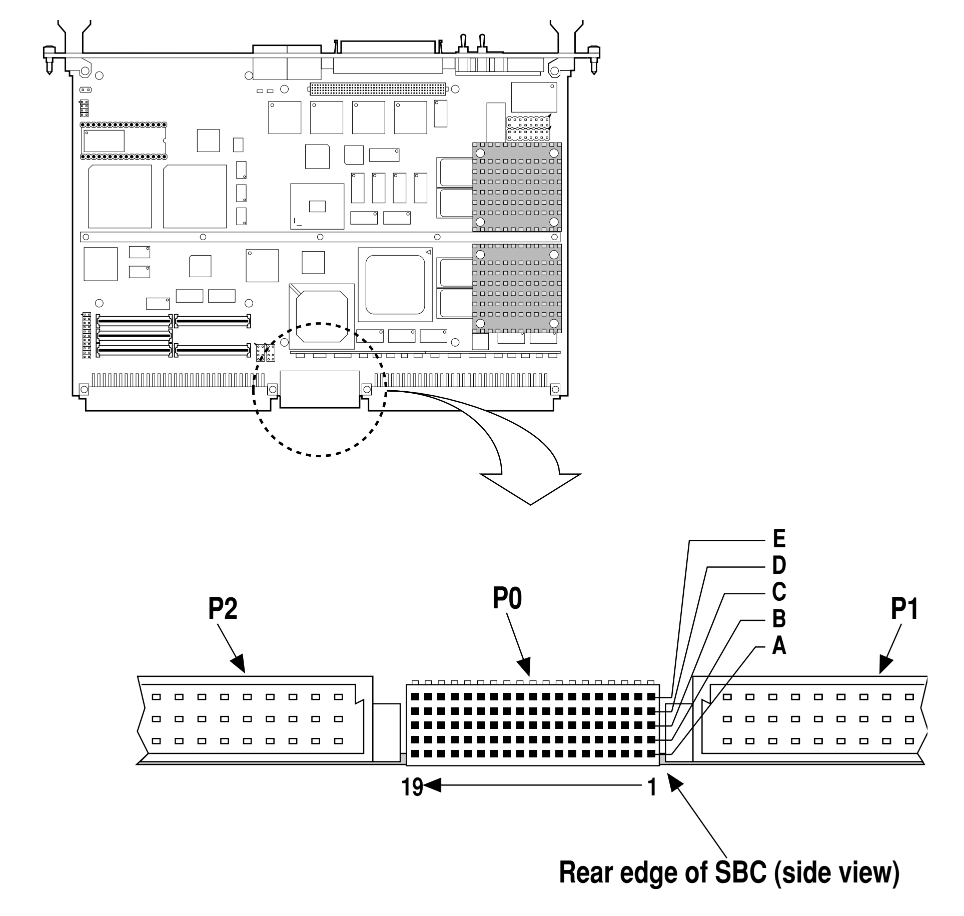
Note: 1. The function of pins labeled ‘UsrIOxxx’ depends on the add-on card installed on the board. Space is provided in these columns to write the assigned signals if desired.

# P0 • PCI™ bus connector (P0)

The VGM5's P0 • PCI™ connector provides an additional PCI bus connection to the VGM5's PCI bus for board-to-board communications and additional expansion of PCI devices. Boards with the P0 • PCI™ interface option require installation in a VME64x compatible backplane that includes the P0 backplane connector.

⚠ Because the VME64x P0 connector has user-defined pins, make sure that your backplane's P0 connections are compatible with the VGM5's secondary PCI bus connector before plugging the board in. **Failure to observe this warning can cause the complete destruction of many on-board components and also voids the product warranty.**

The drawing below shows the location and pin orientation of the VGM5's P0•PCI™ connector. The table that follows lists the pin assignments of this connector.



*P0•PCI™ bus connector, P0*

### *P0 • PCI™ bus connector (P0) pinouts*

| Pin Number | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| **1** | IntA | AD44 | IntB/Gt | IntC/Rq | IntD/Ck | Gnd |
| **2** | AD45 | AD39 | AD37 | AD32 | AD34 | Gnd |
| **3** | AD47 | AD46 | AD42 | AD33 | AD36 | Gnd |
| **4** | AD50 | AD51 | AD48 | AD38 | AD41 | Gnd |
| **5** | AD52 | AD54 | AD53 | AD35 | AD40 | Gnd |
| **6** | AD56 | AD55 | Vcc5V | AD43 | Par64 | Gnd |
| **7** | AD59 | AD63 | AD60 | AD49 | AD57 | Gnd |
| **8** | AD61 | CBE6\ | CBE4\ | AD58 | AD62 | Gnd |
| **9** | CBE5\ | AD1 | AD0 | Req64\ | CBE7\ | Gnd |
| **10** | Ack64\ | AD2 | AD6 | AD5 | AD4 | Gnd |
| **11** | AD3 | AD7 | AD8 | CBE0\ | AD9 | Gnd |
| **12** | AD11 | AD12 | AD10 | Par | AD13 | Gnd |
| **13** | AD14 | SErr\ | AD15 | AD28 | Lock | Gnd |
| **14** | Req\ | Stop\ | SysCon | CBE2\ | IRdy\ | Gnd |
| **15** | CBE1\ | PErr\ | TRdy\ | AD23 | AD20 | Gnd |
| **16** | DevSel\ | AD19 | AD17 | AD24 | AD29 | Gnd |
| **17** | Frame\ | CBE3\ | AD22 | AD27 | AD26 | Gnd |
| **18** | AD16 | AD21 | IDSel | AD30 | AD31 | Gnd |
| **19** | Gnt\ | AD18 | AD25 | Rst\ | Clk | Gnd |

# Memory module connector (PM22)

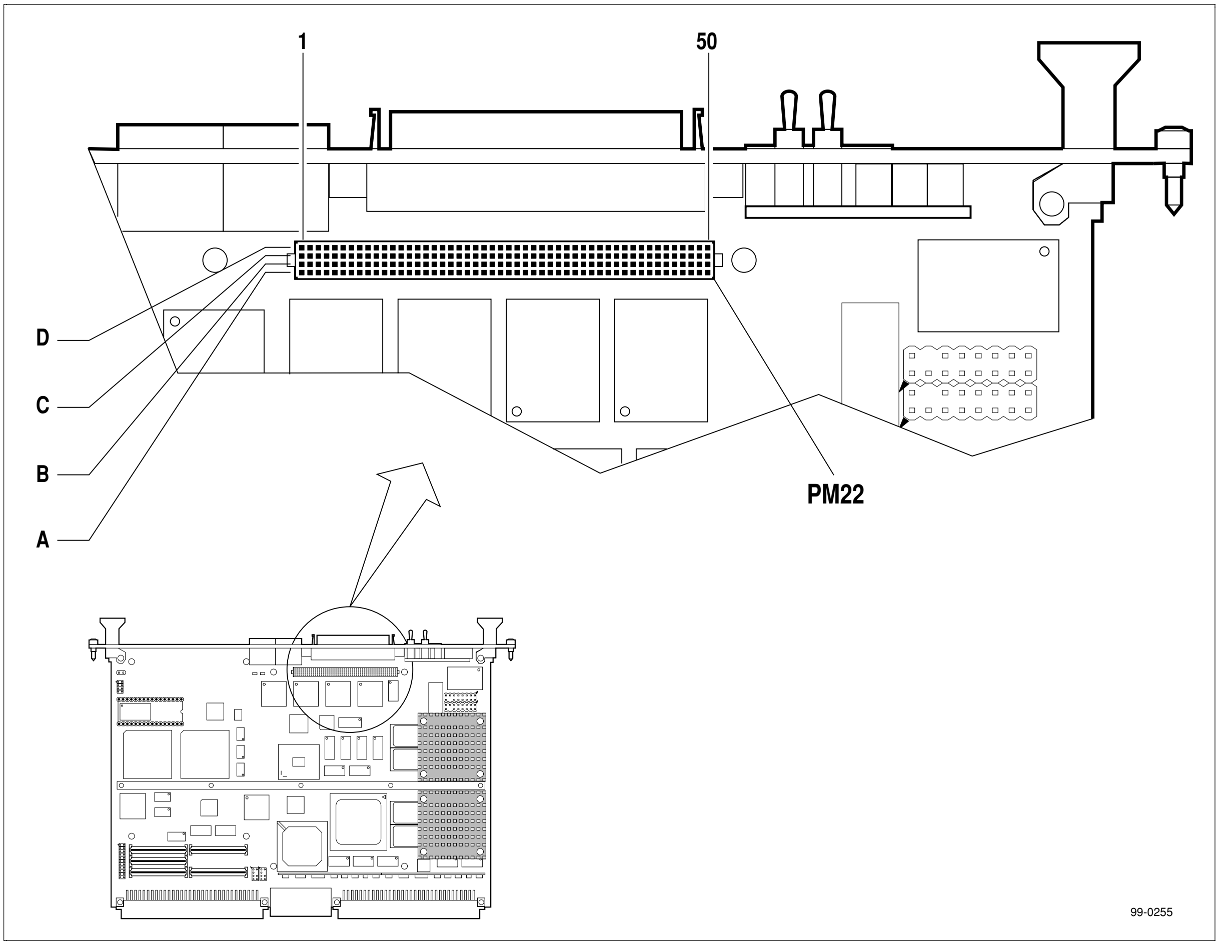
The field-replaceable RGS1/RGS2 memory module plugs into the VGM5 board (rev. D or earlier boards) via connector PM22. The drawing below shows the location of this connector. The table that follows lists the PM22 pin assignments.

Note This connector accepts the following memory module board(s):

**RGS1**: 16, 32, or 64 MB of SDRAM

**RGS2**: 32, 64, 128, 256 or 512 MB of SDRAM

Note that these modules are for use with VGM5 rev. D or earlier boards only.



Memory module connector (PM22) location

### Memory module connector, PM22 pinouts

| Row A | Function | Row B | Function | Row C | Function | Row D | Function |
|---|---|---|---|---|---|---|---|
| 1 | PpcDP0 | 1 | PpcDP1 | 1 | PpcDP2 | 1 | PpcDP3 |
| 2 | PpcDP4 | 2 | PpcDP5 | 2 | PpcDP6 | 2 | PpcDP7 |
| 3 | Gnd | 3 | Vcc3 | 3 | Vcc3 | 3 | Gnd |
| 4 | PpcDH0 | 4 | PpcDH1 | 4 | PpcDH2 | 4 | PpcDH3 |
| 5 | PpcDH4 | 5 | PpcDH5 | 5 | PpcDH6 | 5 | PpcDH7 |
| 6 | Gnd | 6 | Vcc3 | 6 | Vcc3 | 6 | Gnd |
| 7 | PpcDH8 | 7 | PpcDH9 | 7 | PpcDH10 | 7 | PpcDH11 |
| 8 | PpcDH12 | 8 | PpcDH13 | 8 | PpcDH14 | 8 | PpcDH15 |
| 9 | Gnd | 9 | Vcc3 | 9 | Vcc3 | 9 | Gnd |
| 10 | PpcDH16 | 10 | PpcDH17 | 10 | PpcDH18 | 10 | PpcDH19 |
| 11 | PpcDH20 | 11 | PpcDH21 | 11 | PpcDH22 | 11 | PpcDH23 |
| 12 | Gnd | 12 | Vcc3 | 12 | Vcc3 | 12 | Gnd |
| 13 | PpcDH24 | 13 | PpcDH25 | 13 | PpcDH26 | 13 | PpcDH27 |
| 14 | PpcDH28 | 14 | PpcDH29 | 14 | PpcDH30 | 14 | PpcDH31 |
| 15 | Gnd | 15 | Vcc3 | 15 | Vcc3 | 15 | Gnd |
| 16 | PpcDL0 | 16 | PpcDL1 | 16 | PpcDL2 | 16 | PpcDL3 |
| 17 | PpcDL4 | 17 | PpcDL5 | 17 | PpcDL6 | 17 | PpcDL7 |
| 18 | Gnd | 18 | Vcc3 | 18 | Vcc3 | 18 | Gnd |
| 19 | PpcDL8 | 19 | PpcDL9 | 19 | PpcDL10 | 19 | PpcDL11 |
| 20 | PpcDL12 | 20 | PpcDL13 | 20 | PpcDL14 | 20 | PpcDL15 |
| 21 | Gnd | 21 | Vcc3 | 21 | Vcc3 | 21 | Gnd |
| 22 | PpcDL16 | 22 | PpcDL17 | 22 | PpcDL18 | 22 | PpcDL19 |
| 23 | PpcDL20 | 23 | PpcDL21 | 23 | PpcDL22 | 23 | PpcDL23 |
| 24 | Gnd | 24 | Vcc3 | 24 | Vcc3 | 24 | Gnd |
| 25 | PpcDL24 | 25 | PpcDL25 | 25 | PpcDL26 | 25 | PpcDL27 |
| 26 | PpcDL28 | 26 | PpcDL29 | 26 | PpcDL30 | 26 | PpcDL31 |
| 27 | Gnd | 27 | Vcc3 | 27 | Vcc3 | 27 | Gnd |
| 28 | — | 28 | — | 28 | — | 28 | HReset\ |
| 29 | MemID0\ | 29 | MemID1\ | 29 | MemID2\ | 29 | MemID3\ |
| 30 | Gnd | 30 | Vcc3 | 30 | Vcc3 | 30 | Gnd |
| 31 | MemID4\ | 31 | MemID5\ | 31 | MemID6\ | 31 | MemID7\ |
| 32 | MemB Enb\ | 32 | — | 32 | — | 32 | — |
| 33 | Gnd | 33 | Vcc3 | 33 | Vcc3 | 33 | Gnd |
| 34 | MemRamWE | 34 | MemRamRE\ | 34 | BClkRamWr | 34 | BClkRamRd |
| 35 | MemWE\ | 35 | MemSDCAS\ | 35 | MemSDRAS\ | 35 | MemCKE |
| 36 | Gnd | 36 | Vcc3 | 36 | Vcc3 | 36 | Gnd |
| 37 | MemCS0\ | 37 | MemCS1\ | 37 | MemCS2\ | 37 | MemCS3\ |
| 38 | MemCS4\ | 38 | MemCS5\ | 38 | MemCS6\ | 38 | MemCS7\ |
| 39 | Gnd | 39 | Vcc3 | 39 | Vcc3 | 39 | Gnd |
| 40 | MemDQM0 | 40 | MemDQM1 | 40 | MemDQM2 | 40 | MemDQM3 |
| 41 | MemDQM4 | 41 | MemDQM5 | 41 | MemDQM6 | 41 | MemDQM7 |
| 42 | Gnd | 42 | Vcc3 | 42 | Vcc3 | 42 | Gnd |
| 43 | — | 43 | — | 43 | — | 43 | MemMA0 |
| 44 | MemSDBA0 | 44 | MemSDMA1 | 44 | MemSDMA2 | 44 | MemSDMA3 |
| 45 | Gnd | 45 | Vcc3 | 45 | Vcc3 | 45 | Gnd |
| 46 | MemSDMA4 | 46 | MemSDMA5 | 46 | MemSDMA6 | 46 | MemSDMA7 |
| 47 | MemSDMA8 | 47 | MemSDMA9 | 47 | MemSDMA10 | 47 | MemSDMA11 |
| 48 | Gnd | 48 | Vcc3 | 48 | Vcc3 | 48 | Gnd |
| 49 | BClkRamBnk0 | 49 | BClkRamBnk1 | 49 | BClkRamBnk2 | 49 | BClkRamBnk3 |
| 50 | BClkRamBnk4 | 50 | BClkRamBnk5 | 50 | BClkRamBnk6 | 50 | BClkRamBnk7 |

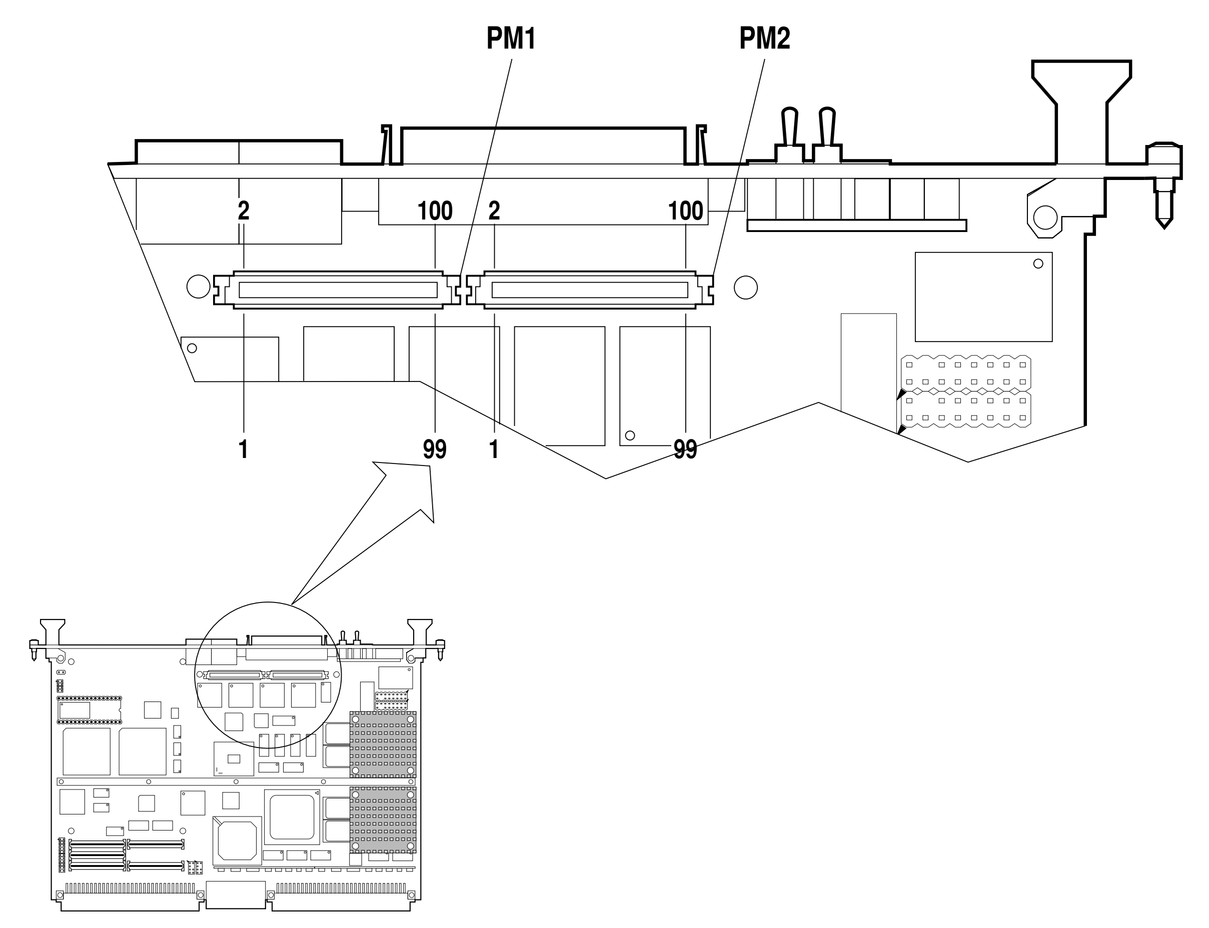
# *Memory module connectors (PM1 & PM2)*

The field-replaceable RGS3 memory module plugs into the VGM5 (rev. E or higher boards) via connectors PM1 and PM2. The drawing below shows the locations of these connectors. The table that follows lists the PM1 and PM2 pin assignments.

This connector accepts the following memory module board(s):

**RGS3**: 32, 64, 128, 256 or 512 MB of SDRAM

Note that this module is for use with VGM5 rev. E or higher boards only.



*Memory module connector (PM1 & PM2) locations*

### Memory module connector, PM1 pinouts

| Odd Row Pin No. | Function | Even Row Pin No. | Function |
|---|---|---|---|
| 1 | Gnd | 2 | PpcDP0 |
| 3 | PpcDP1 | 4 | +3.3V |
| 5 | PpcDP2 | 6 | PpcDP3 |
| 7 | Gnd | 8 | PpcDP4 |
| 9 | PpcDP5 | 10 | +3.3V |
| 11 | PpcDP6 | 12 | PpcDP7 |
| 13 | Gnd | 14 | PpcDH0 |
| 15 | PpcDH1 | 16 | +3.3V |
| 17 | PpcDH2 | 18 | PpcDH3 |
| 19 | Gnd | 20 | PpcDH4 |
| 21 | PpcDH5 | 22 | +3.3V |
| 23 | PpcDH6 | 24 | PpcDH7 |
| 25 | Gnd | 26 | PpcDH8 |
| 27 | PpcDH9 | 28 | +3.3V |
| 29 | PpcDH10 | 30 | PpcDH11 |
| 31 | Gnd | 32 | RamClk7 |
| 33 | RamClk3 | 34 | +3.3V |
| 35 | PpcDH12 | 36 | PpcDH13 |
| 37 | Gnd | 38 | PpcDH14 |
| 39 | PpcDH15 | 40 | +3.3V |
| 41 | PpcDH16 | 42 | PpcDH17 |
| 43 | Gnd | 44 | PpcDH18 |
| 45 | PpcDH19 | 46 | +3.3V |
| 47 | PpcDH20 | 48 | PpcDH21 |
| 49 | Gnd | 50 | PpcDH22 |
| 51 | PpcDH23 | 52 | +3.3V |
| 53 | PpcDH24 | 54 | PpcDH25 |
| 55 | Gnd | 56 | PpcDH26 |
| 57 | PpcDH27 | 58 | +3.3V |
| 59 | PpcDH28 | 60 | PpcDH29 |
| 61 | Gnd | 62 | PpcDH30 |
| 63 | PpcDH31 | 64 | +3.3V |
| 65 | RamClk6 | 66 | RamClk2 |
| 67 | Gnd | 68 | PpcDL0 |
| 69 | PpcDL1 | 70 | +3.3V |
| 71 | PpcDL2 | 72 | PpcDL3 |
| 73 | Gnd | 74 | PpcDL4 |
| 75 | PpcDL5 | 76 | +3.3V |
| 77 | PpcDL6 | 78 | PpcDL7 |
| 79 | Gnd | 80 | PpcDL8 |
| 81 | PpcDL9 | 82 | +3.3V |
| 83 | PpcDL10 | 84 | PpcDL11 |
| 85 | Gnd | 86 | PpcDL12 |
| 87 | PpcDL13 | 88 | +3.3V |
| 89 | PpcDL14 | 90 | PpcDL15 |
| 91 | Gnd | 92 | PpcDL16 |
| 93 | PpcDL17 | 94 | +3.3V |
| 95 | PpcDL18 | 96 | PpcDL19 |
| 97 | Gnd | 98 | PpcDL20 |
| 99 | PpcDL21 | 100 | +3.3V |

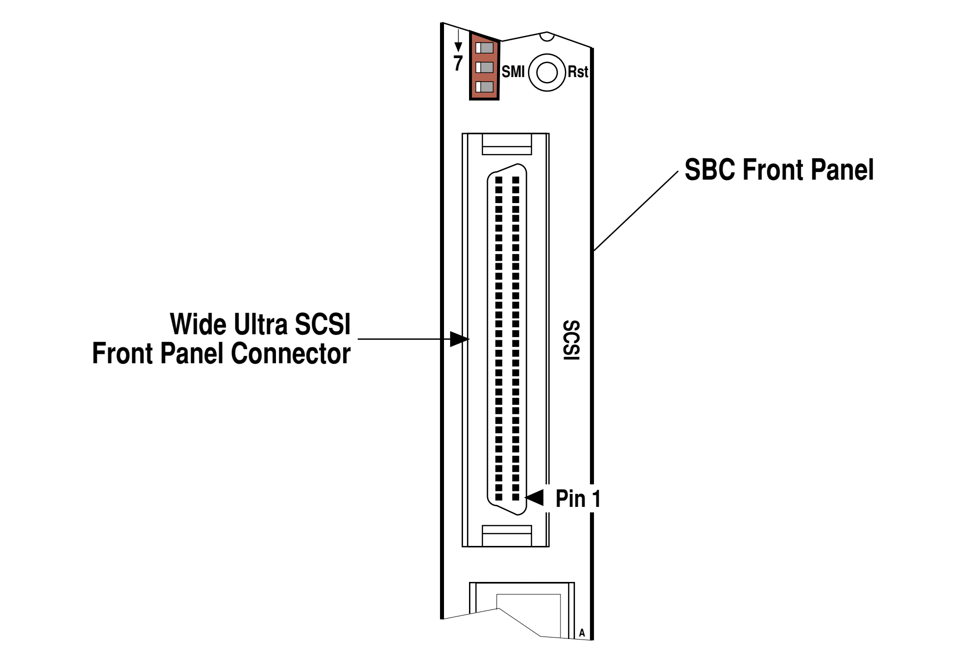
### Memory module connector, PM2 pinouts

| Odd Row Pin No. | Function | Even Row Pin No. | Function |
|---|---|---|---|
| 1 | PpcDL22 | 2 | PpcDL23 |
| 3 | Gnd | 4 | PpcDL24 |
| 5 | PpcDL25 | 6 | +3.3V |
| 7 | PpcDL26 | 8 | PpcDL27 |
| 9 | Gnd | 10 | PpcDL28 |
| 11 | PpcDL29 | 12 | +3.3V |
| 13 | PpcDL30 | 14 | PpcDL31 |
| 15 | Gnd | 16 | MemRamWE |
| 17 | RamWrClk | 18 | +3.3V |
| 19 | MemRamRE\ | 20 | RamRdClk |
| 21 | Gnd | 22 | MemWE\ |
| 23 | MemSDCAS\ | 24 | +3.3V |
| 25 | MemSDRAS\ | 26 | MemCKE |
| 27 | Gnd | 28 | MemCS7\ |
| 29 | MemCS3\ | 30 | +3.3V |
| 31 | RamClk5 | 32 | RamClk1 |
| 33 | Gnd | 34 | MemCS6\ |
| 35 | MemCS2\ | 36 | +3.3V |
| 37 | MemCS5\ | 38 | MemCS1\ |
| 39 | Gnd | 40 | MemCS4\ |
| 41 | MemCS0\ | 42 | +3.3V |
| 43 | MemDQM7 | 44 | MemDQM3 |
| 45 | Gnd | 46 | MemDQM6 |
| 47 | MemDQM2 | 48 | +3.3V |
| 49 | MemDQM5 | 50 | MemDQM1 |
| 51 | Gnd | 52 | MemDQM4 |
| 53 | MemDQM0 | 54 | +3.3V |
| 55 | MemSDBA0 | 56 | MemSDBA1 |
| 57 | Gnd | 58 | MemSDMA1 |
| 59 | MemSDMA2 | 60 | +3.3V |
| 61 | MemSDMA3 | 62 | MemSDMA4 |
| 63 | Gnd | 64 | MemSDMA5 |
| 65 | MemSDMA6 | 66 | +3.3V |
| 67 | RamClk4 | 68 | RamClk0 |
| 69 | Gnd | 70 | MemSDMA7 |
| 71 | MemSDMA8 | 72 | +3.3V |
| 73 | MemSDMA9 | 74 | MemSDMA10 |
| 75 | Gnd | 76 | MemSDMA11 |
| 77 | MemSDMA12 | 78 | +3.3V |
| 79 | — | 80 | — |
| 81 | Gnd | 82 | MemID7\ |
| 83 | MemID3\ | 84 | +3.3V |
| 85 | MemID6\ | 86 | MemID2\ |
| 87 | Gnd | 88 | MemID5\ |
| 89 | MemID1\ | 90 | +3.3V |
| 91 | MemID4\ | 92 | MemID0\ |
| 93 | Gnd | 94 | MbxWrD\ |
| 95 | MbxWrC\ | 96 | +5V |
| 97 | MbxWrB\ | 98 | MbxWrA\ |
| 99 | Gnd | 100 | MbxLE\ |

# *Wide Ultra SCSI connector (P264/P262)*

A front panel, 68-pin, high-density D-connector provides the connection to the optional onboard Wide Ultra SCSI port.

The drawing below shows this connector (P264, Rev. D or lower; P262, Rev. E or higher) and its pin 1 orientation. The table that follows lists the P264/P262 pinouts.



*Wide Ultra SCSI front panel connector (P264/P262)*

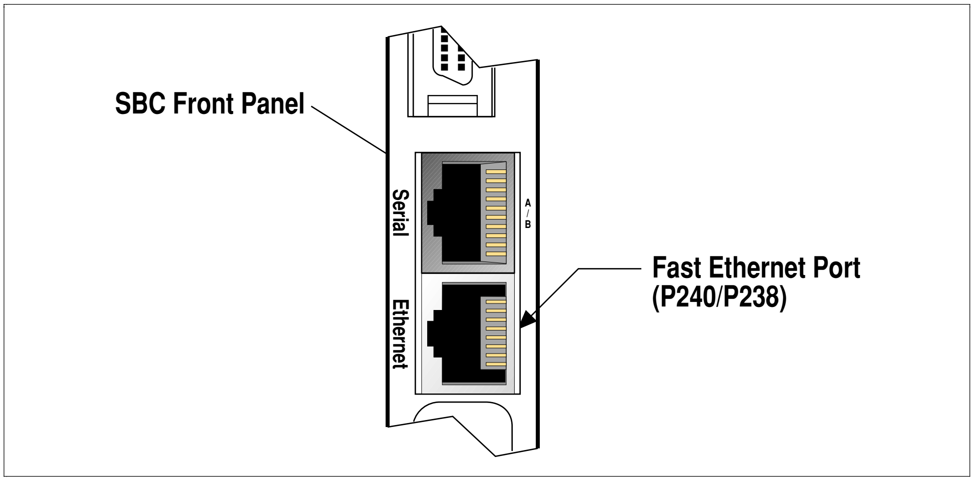
### *Wide Ultra SCSI connector, P264/P262 pinouts*

| Pin No. | Function | Pin No. | Function |
|---|---|---|---|
| **1** | Gnd | **35** | Data12- |
| **2** | Gnd | **36** | Data13- |
| **3** | Gnd | **37** | Data14- |
| **4** | Gnd | **38** | Data15- |
| **5** | Gnd | **39** | Parity1- |
| **6** | Gnd | **40** | Data0- |
| **7** | Gnd | **41** | Data1- |
| **8** | Gnd | **42** | Data2- |
| **9** | Gnd | **43** | Data3- |
| **10** | Gnd | **44** | Data4- |
| **11** | Gnd | **45** | Data5- |
| **12** | Gnd | **46** | Data6- |
| **13** | Gnd | **47** | Data7- |
| **14** | Gnd | **48** | Parity0- |
| **15** | Gnd | **49** | Gnd |
| **16** | Gnd | **50** | Gnd |
| **17** | Term Power | **51** | Term Power |
| **18** | Term Power | **52** | Term Power |
| **19** | — | **53** | — |
| **20** | Gnd | **54** | Gnd |
| **21** | Gnd | **55** | Atn- |
| **22** | Gnd | **56** | Gnd |
| **23** | Gnd | **57** | Bsy- |
| **24** | Gnd | **58** | Ack- |
| **25** | Gnd | **59** | Rst- |
| **26** | Gnd | **60** | Msg- |
| **27** | Gnd | **61** | Sel- |
| **28** | Gnd | **62** | C/D- |
| **29** | Gnd | **63** | Req- |
| **30** | Gnd | **64** | I/O- |
| **31** | Gnd | **65** | Data8- |
| **32** | Gnd | **66** | Data9- |
| **33** | Gnd | **67** | Data10- |
| **34** | Gnd | **68** | Data11- |

# Fast Ethernet connector (P240/P238)

The VGM5 board's Fast Ethernet port connects to the Ethernet 10Base/100Base-TX network via the front panel RJ-45 jack as shown in the figure below. This chapter lists the pinout for this connector.

The Ethernet connector provided on the VGM5 may be connected to a hub using a standard (straight-wired) cable, or to another Ethernet port using a crossover cable. A crossover cable has the Transmit and Receive pairs swapped on one end.



*Fast Ethernet front panel cable connector*

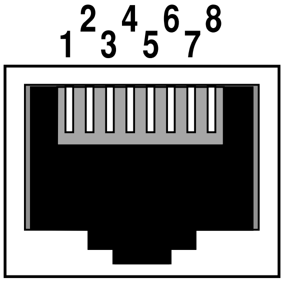
The figure and table below identify the pinout numbers and signals for the VGM5 front panel Fast Ethernet connector (P240 for Rev. D or lower boards, P238 for Rev. E or higher boards).

The VGM5 Fast Ethernet port supports 10Base-T and 100Base-TX. The other two variations in the 100Base-T standard are not supported (100Base-T4 and 100Base-FX).

For a 10Base-T Ethernet network, use a Category 3 or higher UTP (unshielded twisted pair) cable to connect the VGM5 to the 10Base-T hub.

For a 100Base-TX Ethernet network, use a Category 5 UTP or Type 1 STP (shielded twisted pair) cable to connect the VGM5 board to the Fast Ethernet hub.

Pre-assembled, twisted-pair Ethernet cables in a variety of lengths and colors are available from various electronic and computer supply houses.



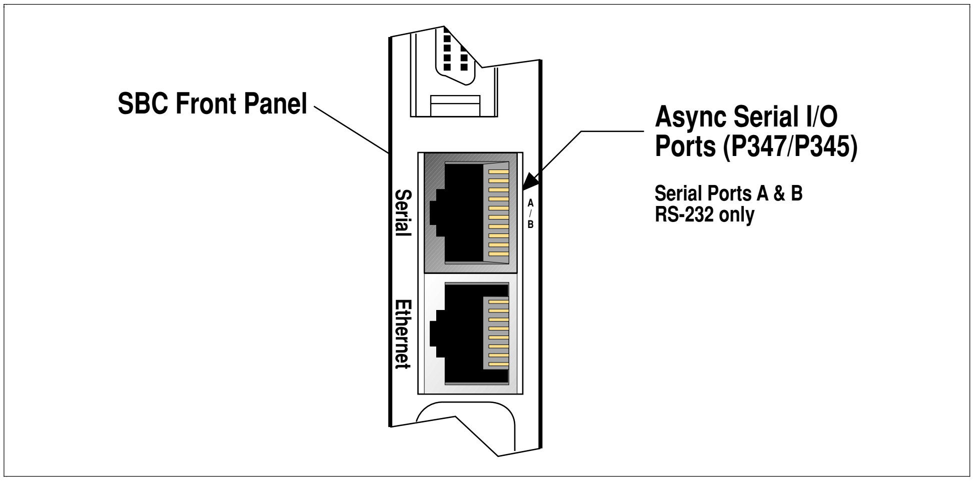
*Ethernet 10Base/100Base-TX connector pin numbering*

*Ethernet 10Base/100Base-TX port (P240/P238) pin assignments*

| Pin | Function |
|---|---|
| 1 | *Transmit Data+* |
| 2 | *Transmit Data–* |
| 3 | *Receive Data+* |
| 4 | Shield |
| 5 | Shield |
| 6 | *Receive Data–* |
| 7 | Shield |
| 8 | Shield |

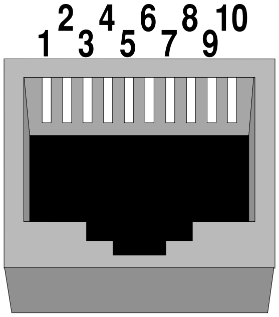
# Asynchronous serial connector (P347/P345)

The VGM5 board's two asynchronous serial ports are brought out to a single 10-pin RJ-69 modular connector (shown below). This chapter lists the pinout for this connector. See the next chapter for serial interface cabling options.



***Front panel serial I/O ports cable connector***

The figure and table identify the pin numbers and signals for the serial port RJ-69 connector on the VGM5 front panel (P347 for Rev. D or lower boards, P345 for Rev. E or higher boards).



***Asynchronous serial connector pin numbering***

***Serial Ports A & B (P347/P345) pin assignments***

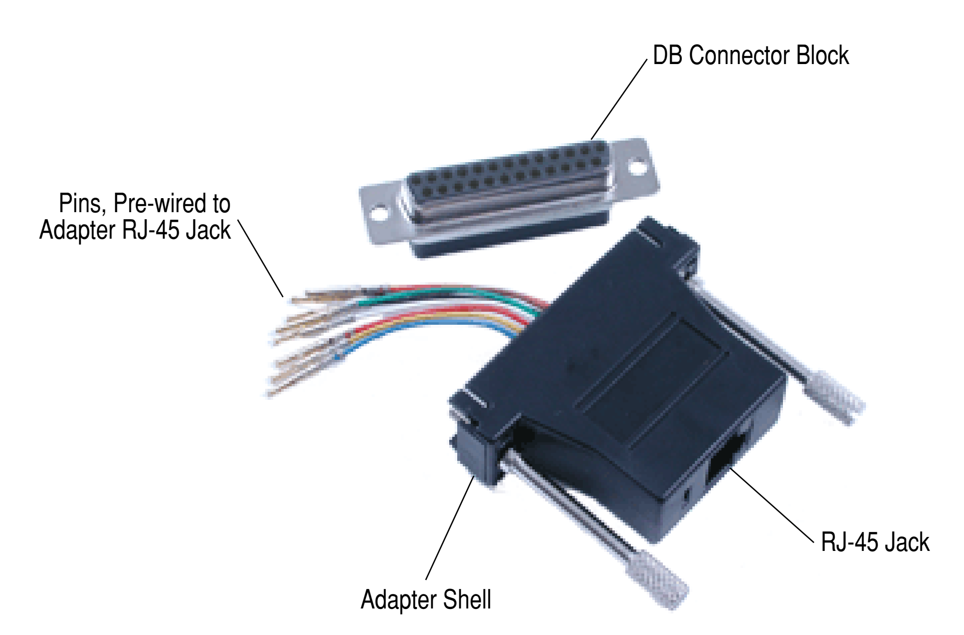
| Pin | Function |
|---|---|
| 1 | Port B, ***Request to Send*** (RTS) |
| 2 | Port A, ***Request to Send*** (RTS) |
| 3 | Port B, ***Transmit Data*** (TxD) |
| 4 | Port A, ***Transmit Data*** (TxD) |
| 5 | ***Ground*** (Gnd) |
| 6 | ***Ground*** (Gnd) |
| 7 | Port A, ***Receive Data*** (RxD) |
| 8 | Port B, ***Receive Data*** (RxD) |
| 9 | Port A, ***Clear to Send*** (CTS) |
| 10 | Port B, ***Clear to Send*** (CTS) |

# Serial I/O cabling options

As described in the last chapter, the VGM5 provides a front panel RJ-69 modular jack for connection of the two asynchronous serial channels. This chapter discusses cabling options for using one or both channels from the front panel serial connector.

## Using only Serial Port A

A single modular cable in conjunction with a readily-available Modular-to-D connector adapter kit (see picture below) connects **one** serial port, Serial Port A, to RS-232 terminals and communications devices with a DB-9 or DB-25 serial connector. In this scheme a standard straight (that is, not reversed or crossover-type) modular cable is used to connect the VGM5 board's modular connector to the adapter and then the adapter routes the signal to the appropriate pin on the D-type connector.



***Serial port modular adapter (DB-25F shown)***

The Modular-to-D connector adapters generally consist of an empty D connector shell and a separate modular jack that has pigtails with D pins crimped to them. The adapter is constructed by plugging the appropriate pigtail pins into the correct D shell holes and then assembling the adapter shell. The tables below list the proper adapter wiring needed for various D-type connectors and communications protocols.

***10-pin Modular-to-D null-modem adapter pinout (for connection to a DTE device)***

| Modular connector (female) | | | Signal name | D-type connector (M or F) | |
|---|---|---|---|---|---|
| 4-pin conn. | 8-pin conn. | 10-pin conn. | | DB-9<br>IBM PC<br>RS-232 | DB-25<br><br>RS-232 |
| — | — | 1 | — | — | — |
| — | 1 | 2 | Request-to-Send (RTS) | 7 | 4 |
| — | 2 | 3 | — | — | — |
| 1 | 3 | 4 | Transmit Data (Tx) | 3 | 2 |
| 2 | 4 | 5 | Ground (Gnd) | — | — |
| 3 | 5 | 6 | Ground (Gnd) | 5 | 7 |
| 4 | 6 | 7 | Receive Data (Rx) | 2 | 3 |
| — | 7 | 8 | — | — | — |
| — | 8 | 9 | Clear-to-Send (CTS) | 8 | 5 |
| — | — | 10 | — | — | — |

**10-pin Modular-to-D modem adapter pinout**
**(for connection to a DCE device)**

| Modular connector (female) | | | Signal name | D-type connector (M or F) | |
|---|---|---|---|---|---|
| 4-pin conn. | 8-pin conn. | 10-pin conn. | | DB-9<br>IBM PC<br>RS-232 | DB-25<br><br>RS-232 |
| — | — | 1 | — | — | — |
| — | 1 | 2 | Request-to-Send (RTS) | 8 | 5 |
| — | 2 | 3 | — | — | — |
| 1 | 3 | 4 | Transmit Data (Tx) | 2 | 3 |
| 2 | 4 | 5 | Ground (Gnd) | — | — |
| 3 | 5 | 6 | Ground (Gnd) | 5 | 7 |
| 4 | 6 | 7 | Receive Data (Rx) | 3 | 2 |
| — | 7 | 8 | — | — | — |
| — | 8 | 9 | Clear-to-Send (CTS) | 7 | 4 |
| — | — | 10 | — | — | — |

Note Many serial DB-25 devices use non-standard pin assignments. If the wiring for the RS-232/DB-25 adapter listed above does not work, try shorting pin 6 (DTR) to pin 20 (DCD) on the DB-25. If still not successful, consult the device manual.

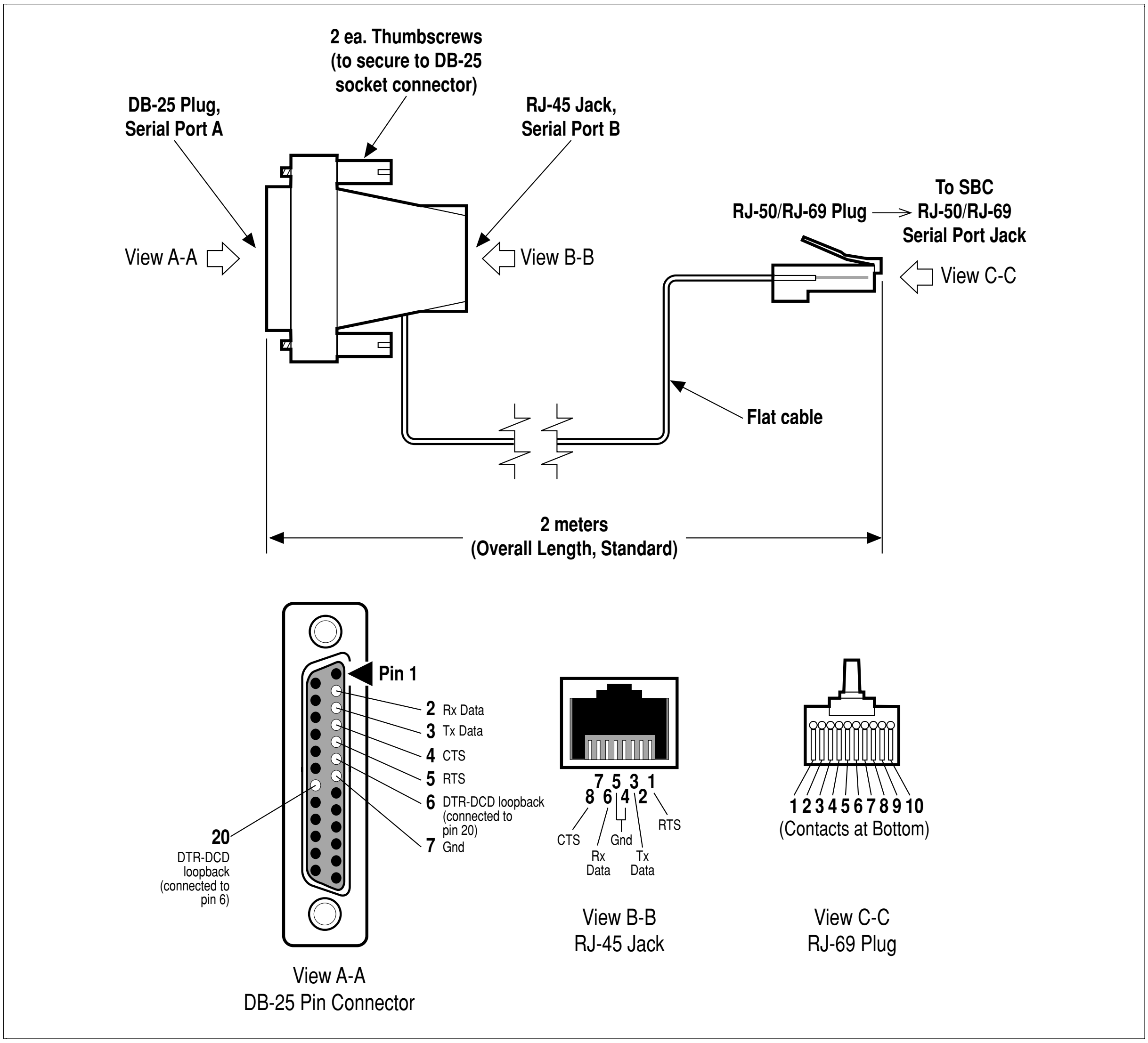
## Using both serial ports via Synergy serial cable adapters

The use of both VGM5 serial ports requires special cable adapters that bring out the two serial ports from the single front panel RJ-69 serial I/O jack. Two types of adapters are available from Synergy (contact Customer Service for details on ordering these adapters):

- RJ-69 to male DB-25/RJ-45 jack adapter, Synergy part no. **Cbl/J10J8D25M2m** — RJ-69 plug to adapter head containing one DB-25 male connector for Serial Port A and RJ-45 connector jack for Serial Port B. This adapter uses a 2-meter (standard length) flat cable. A custom length can be ordered in which case the last two characters in the cable part number will reflect the custom length (e.g., “4m” for a 4-meter cable, Cbl/J10J8D25M4m).

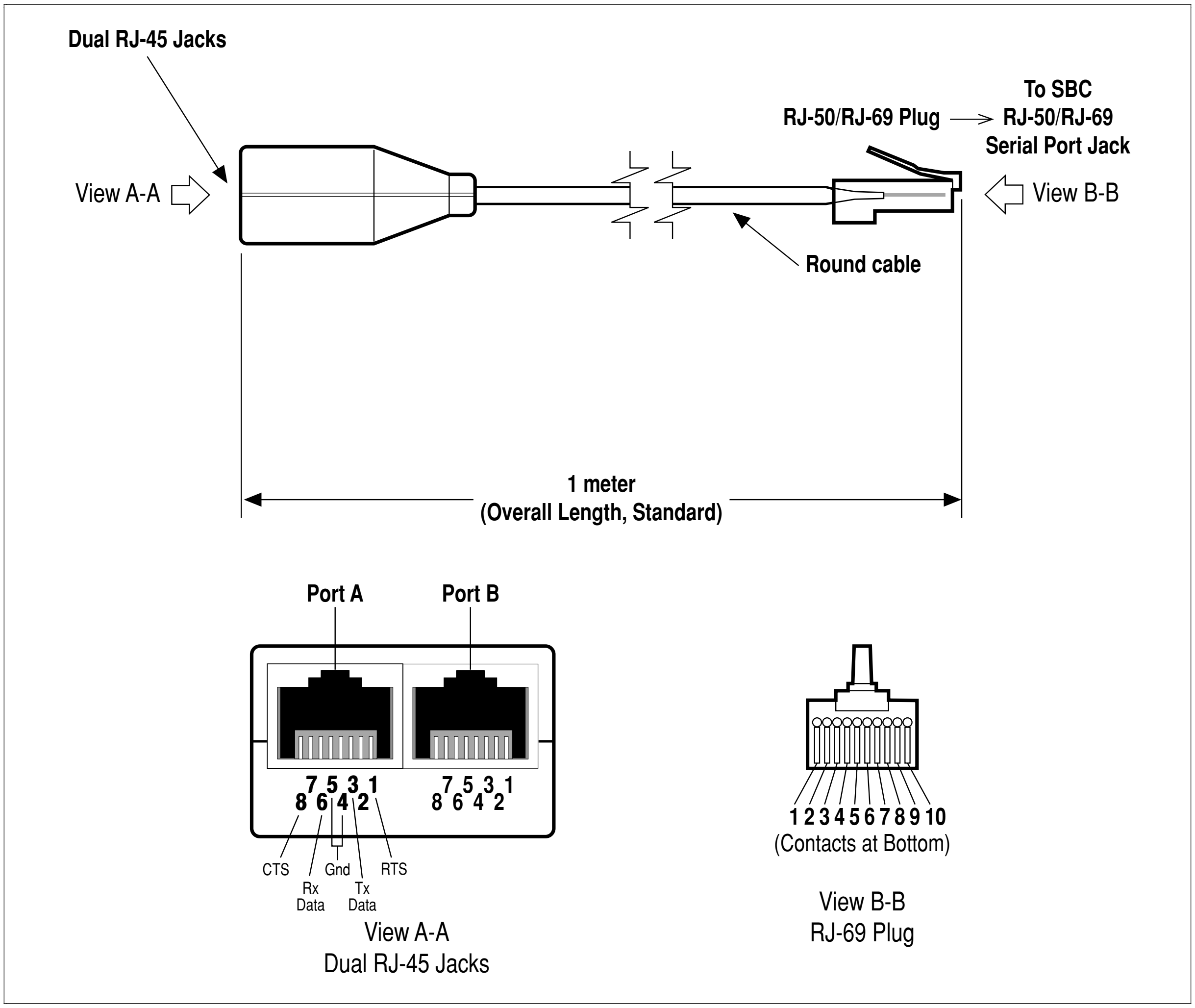
- RJ-69 to dual RJ-45 jack adapter, Synergy part no. **Cbl/J10J8J81m** — RJ-69 plug to adapter head containing two RJ-45 connectors, one for each serial port. This adapter uses a 1 meter (standard length) round cable. A custom length can be ordered in which case the last two characters in the cable part number will reflect the custom length (e.g., “3m” for a 3-meter cable, Cbl/J10J8J83m).

The drawing below shows the **Cbl/J10J8D25M2m** cable and pin assignments of the connectors on the peripheral end.


2 ea. Thumbscrews (to secure to DB-25 socket connector)
DB-25 Plug, Serial Port A
RJ-45 Jack, Serial Port B
To SBC RJ-50/RJ-69 Serial Port Jack
RJ-50/RJ-69 Plug
View A-A
View B-B
View C-C
Flat cable
2 meters (Overall Length, Standard)
Pin 1
2 Rx Data
3 Tx Data
4 CTS
5 RTS
6 DTR-DCD loopback (connected to pin 20)
7 Gnd
20 DTR-DCD loopback (connected to pin 6)
7 5 3 1
8 6 4 2
CTS
Rx Data
Gnd
Tx Data
RTS
1 2 3 4 5 6 7 8 9 10
(Contacts at Bottom)
View A-A DB-25 Pin Connector
View B-B RJ-45 Jack
View C-C RJ-69 Plug

*Cbl/J10J8D25M2m serial cable adapter*

The drawing below shows the **Cbl/J10J8J81m** cable and pin assignments of the connectors on the peripheral end.



***Cbl/J10J8J81m serial cable adapter***

# Appendix B, Specifications

The VGM5 SBC conforms to the following set of specifications and standards.

## VMEbus compliance

IEEE 1014 VMEbus Specification; Rev C.1 & D.1

| | |
|---|---|
| **Master:** | A32,A24,A16/D32,D16,D08(EO):RMW<br>RWD,ROR,FAIR:UAT,<br>BLT32, BLT64. |
| **Slave:** | A32,A24:D32,D16,D08(EO):RMW:UAT,<br>BLT32, BLT64. |
| **Interrupter:** | I (1-7):D08(O):ROAK. |
| **Interrupt handler:** | IH(1-7):D08(O). |

## Physical dimensions

The VGM5 printed circuit board conforms to **VME 6U** requirements for form factor, board spacing, and board thickness:

| | |
|---|---|
| **Board Size:** | 6U: 6.4"x 9.19"x 0.8" minus front panel |
| **Board Thickness:** | 0.062 +/- 0.005 inches or 15.24 +/- 0.51 mm |

## Weight

**VGM5:** **15** ounces (425g)
Weight (approx.) for dual processor board with EMI front panel, hot insertion, wide VME P1/P2 connectors, no PMC card and no memory module.

## *Power requirements*

VGM5 typical power consumption (no PMC card installed):

***Typical power consumption***

| | |
|---|---|
| **Dual G3/466 MHz** | +5.0V +/-5%, 5.8 A typical @ 5.00V (29 W)<br>±12V ±5%, 50 mA for -12V, 150 mA for +12V |
| **Dual G4/466 MHz** | +5.0V +/-5%, 7.8 A typical @ 5.00V (39 W).<br>±12V ±5%, 50 mA for -12V, 150 mA for +12V |

**Voltages must be kept within these tolerances** to ensure proper operation.

## *Operating environment*

**Temperature:** Operating: 0 to +55 °C ambient with forced air cooling (at sea level: minimum, 300 LFM;; recommended, 400 LFM)
Non-operating/Storage: -20 to +70 °C

Board configurations that provide a wider operating temperature range are available. Contact Customer Service for a listing of Thermal Capability options.

**Humidity:** 10% to 90% RH, non-condensing
**Altitude:** up to 50,000 ft. max. with capacitor backup option

## *Capacitor backup option*

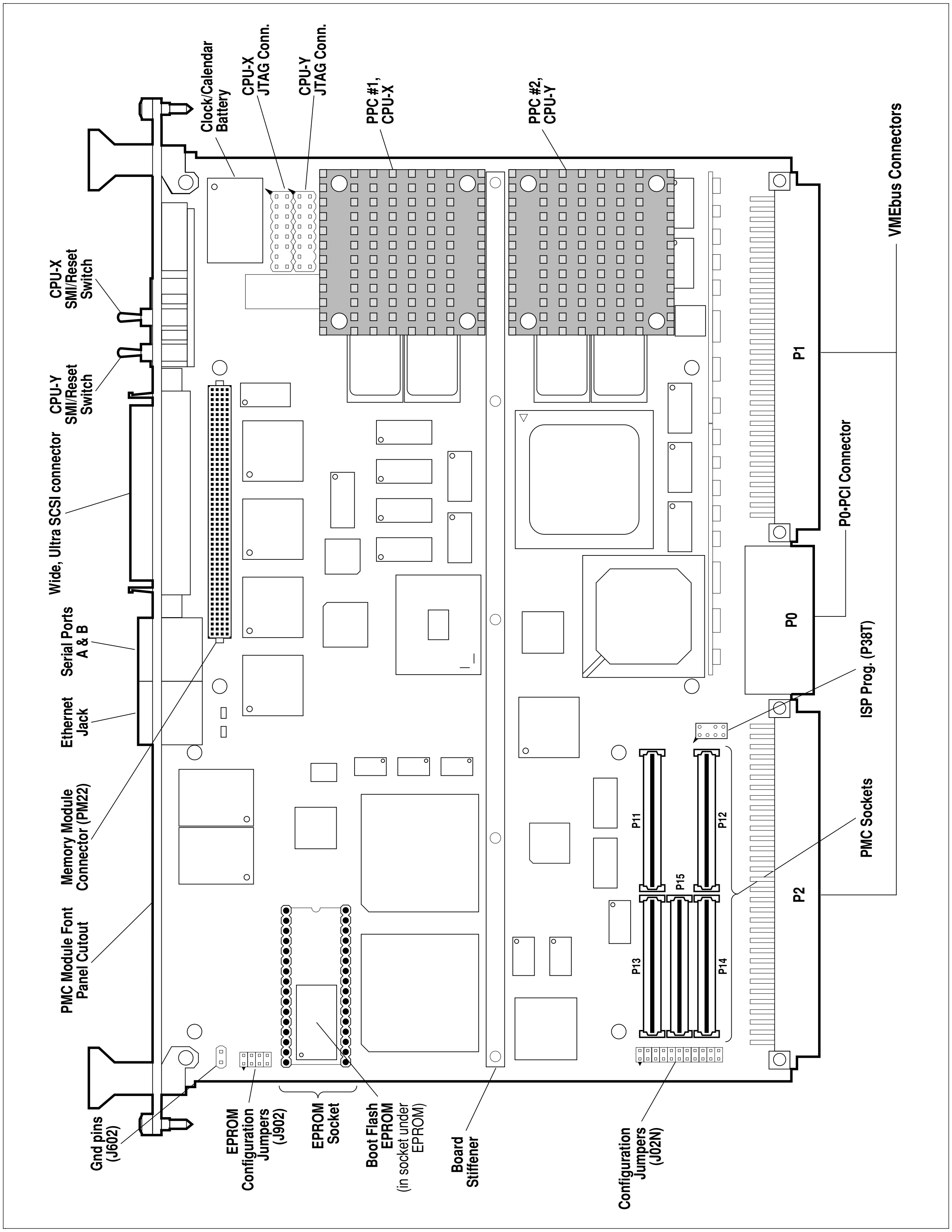
**Time to charge:** 2 hours minimum (capacitor with residual charge);
12 hours maximum (capacitor fully discharged)
**Backup duration:** 12 days @ 20 °C, typical

## *Number of VME slots used*

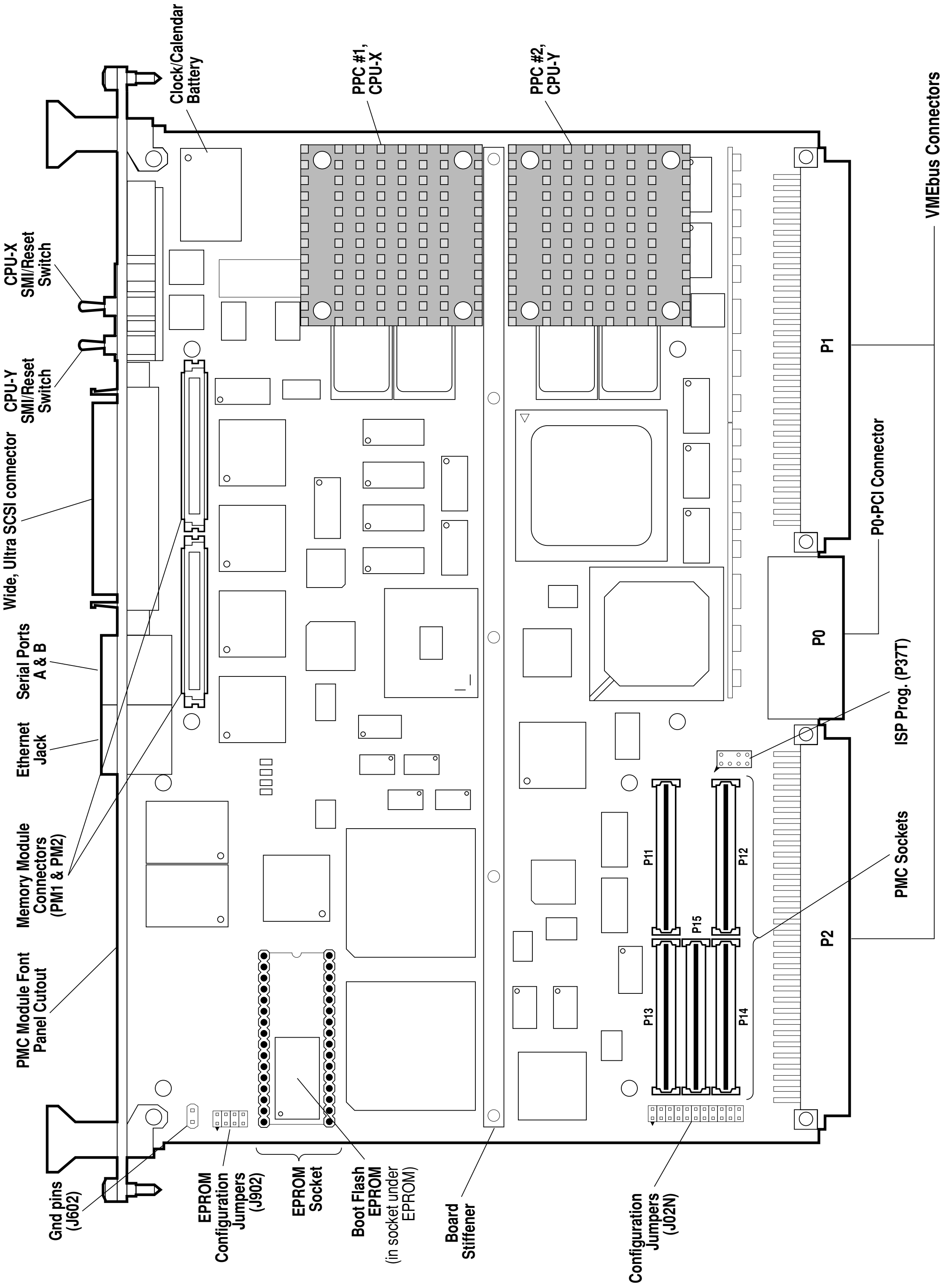
**1**

## *Board layout*

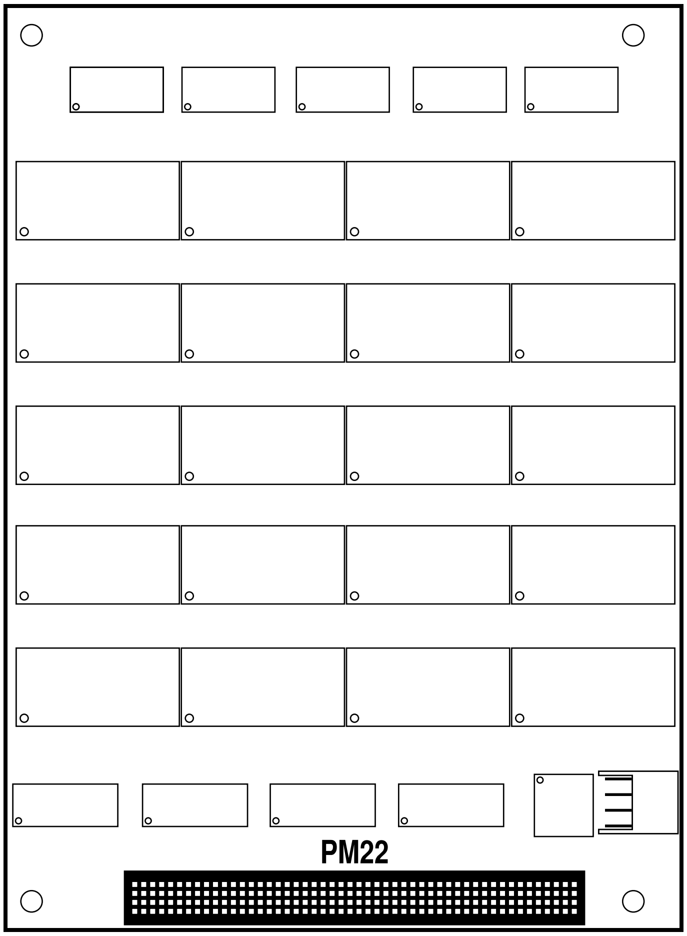
**See drawings below for VGM5 and memory module layout.**



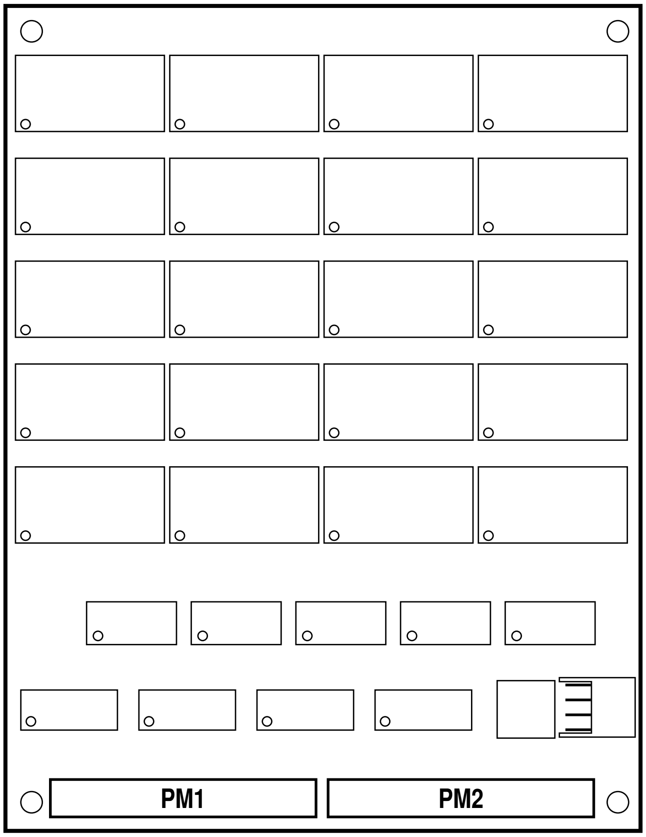
*VGM5 board layout (Assy. Rev. D or lower)*



*VGM5 board layout (Assy. Rev. E or higher)*



*Memory module RGS1/RGS2 board layout*



*Memory module RGS3 board layout*

# Appendix C, Board revision summary

This appendix summarizes major changes made to the VGM5 SBC and memory modules affecting form, fit, and/or function. The paragraphs below list the changes pertaining to the revision shown.

The revision levels for each feature represents the revision level when the listed feature was added to the standard design. Some boards with older revision levels may have had some of these features added during previous rework/upgrades.

Contact Synergy customer service for upgrade information.

## VGM5

These paragraphs describe changes made to the VGM5 main board.

***Revision F*** Incorporated the following additions: support for PowerPC 7400 2 MB L2 caches, support for more Flash now 64 Megs (on back), capability for Sci Sub Library Enable, capability for enabling VME broadcasting, resistors for configuring/using the 21555 Bridge chip, and Auto Speed detection for P0•PCI bus. Incorporated the following changes: removed 3.3V inputs from P1 connector, new P0 pinout with Req64 switch, switched to LXT971 Phyceiver, and switched to PLL for 24/25 and 40MHz clock sources.

***Revision E*** Incorporated the following additions: clock/calendar/NVRAM capacitor backup option, G4 PowerPC 7400 support, 2 MB L2 cache support, and watchdog timer. Changed Ethernet PHYceiver to LXT970 and memory module support to accept new RGS3 memory module only.

***Revision D*** Fixed mailbox 1 data inputs, and PMC stacking and BUSMODE signals.

## VGM5 (continued)

***Revision C*** Incorporated various electrical improvements. Fixed SCSI termination disable jumpering. Added LED lamp test feature.

***Revision B*** Incorporated various electrical and produceability improvements. Increased NVRAM from 32KB to 128KB.

***Revision A*** Initial board release.

## RGSx memory modules

### RGS1

These paragraphs describe RGS1 memory module changes.

***Revision D*** Added mailbox support feature. Incorporated minor board layout improvements.

***Revision C*** Changed data buffer configuration from transparent flow-through to registered to improve data setup and hold-times for 66.67MHz bus speed.

***Revision B*** Reduced board dimensions, swapped data buffers port A and B, fixed RamOE\ and PPEn\ on U100, removed zero-delay clock driver and adjusted pinlist to support MPC106 rev 4.0.

***Revision A*** Initial board release.

### RGS2

These paragraphs describe RGS2 memory module changes.

***Revision B*** Added mailbox support feature. Incorporated minor board layout improvements.

***Revision A*** Initial board release.

### RGS3

These paragraphs describe RGS3 memory module changes.

***Revision B*** PCB notched to clear DC-DC converter module.

***Revision A*** Initial board release.

# Appendix D,
# PEX3 PMC expansion option

The Synergy Microsystems' PEX3 is an optional 6U board that provides PMC and memory expansion to certain model Synergy SBCs (using the Grackle or Chaparral PCI bridge). PEX3 provides three single-width PMC slots and up to 256 MB of SDRAM and up to 128 MB of Flash. The PEX3 has PCI bus master capability for fast, direct communication between the PMCs/onboard memory and devices on the host SBC.

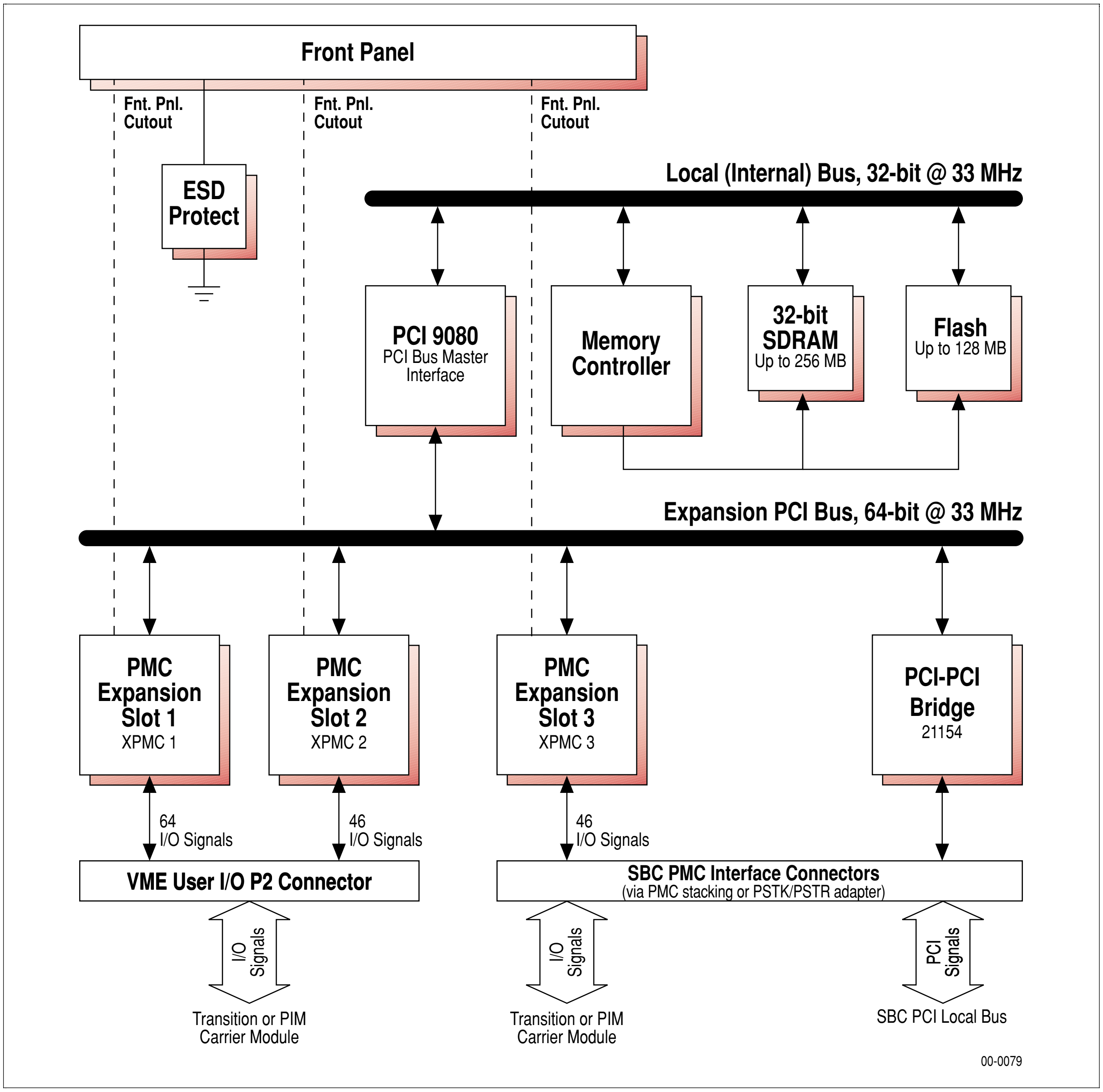
The PEX3 option replaces the regular SBC front panel with a double-wide front panel to form a double-wide module assembly. The PEX3 connects to the host SBC through a stackable Synergy PMC card or adapter. The stacking design lets the SBC use a PMC/memory expansion solution without giving up the use of a PMC.

## Features

- Supports three single-width PMCs, one single-width and one double-width PMC, or one triple-width PMC
- SDRAM (capable of streaming data) up to 256 MB
- Flash up to 128 MB
- PCI bus mastering capability with 2 DMA channels
- ESD protection

## Block diagram

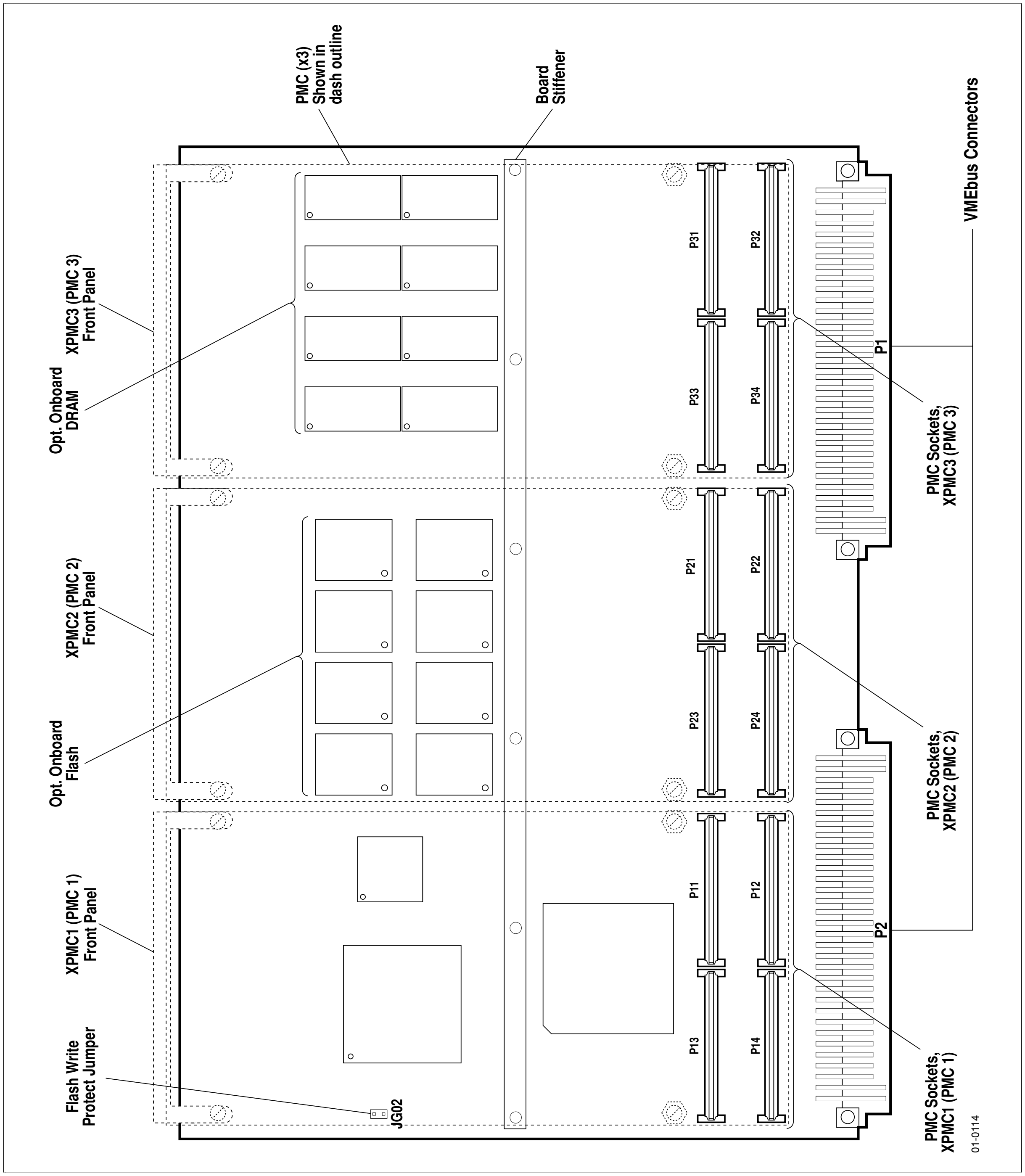
A functional block diagram of the optional PEX3 expansion board is shown below.



*PEX3 functional block diagram*

## Board layout

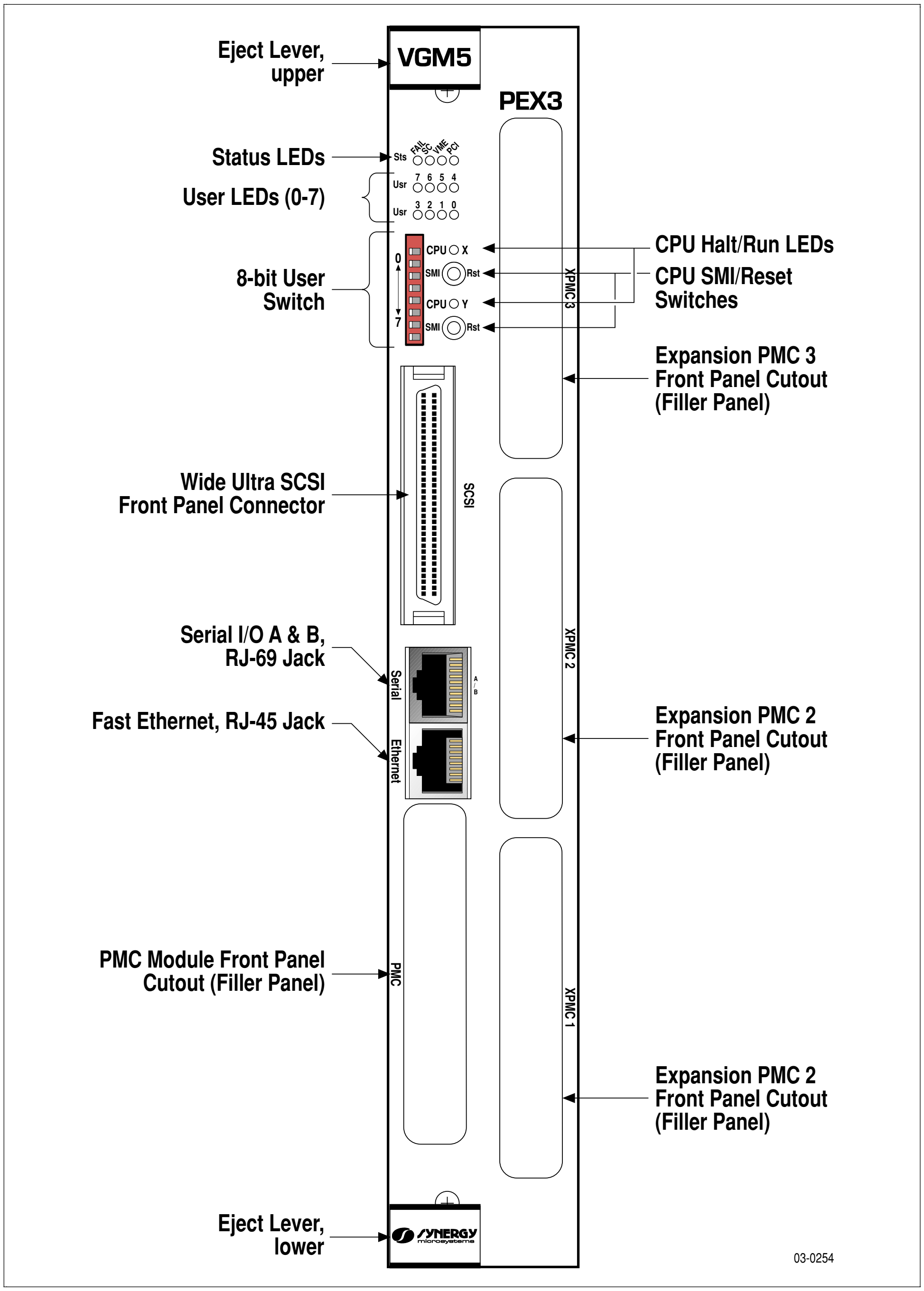
The drawing below shows the PEX3 components and PMC modules.


PMC (x3) Shown in dash outline
Board Stiffener
VMEbus Connectors
XPMC3 (PMC 3) Front Panel
Opt. Onboard DRAM
XPMC2 (PMC 2) Front Panel
Opt. Onboard Flash
XPMC1 (PMC 1) Front Panel
Flash Write Protect Jumper
JG02
P31
P32
P33
P34
P21
P22
P23
P24
P11
P12
P13
P14
P1
P2
PMC Sockets, XPMC3 (PMC 3)
PMC Sockets, XPMC2 (PMC 2)
PMC Sockets, XPMC1 (PMC 1)
01-0114

*PEX3 board layout*

## Front panel layout

The drawing below shows the VGM5/PEX3 front panel layout.

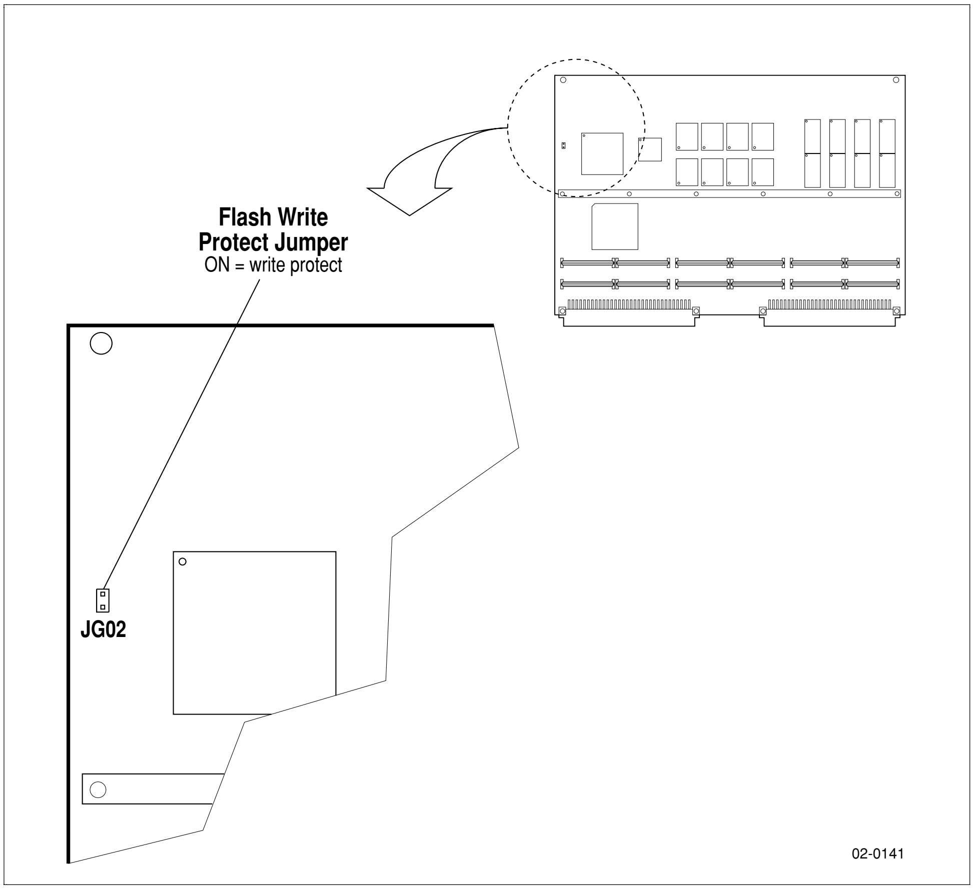

Eject Lever, upper
VGM5
PEX3
Status LEDs
Sts
FAIL
SC
VME
PCI
User LEDs (0-7)
Usr
7 6 5 4
Usr
3 2 1 0
8-bit User Switch
0
7
CPU X
SMI Rst
CPU Y
SMI Rst
CPU Halt/Run LEDs
CPU SMI/Reset Switches
XPMC 3
Expansion PMC 3 Front Panel Cutout (Filler Panel)
Wide Ultra SCSI Front Panel Connector
SCSI
Serial I/O A & B, RJ-69 Jack
Serial
A/B
Fast Ethernet, RJ-45 Jack
Ethernet
XPMC 2
Expansion PMC 2 Front Panel Cutout (Filler Panel)
PMC Module Front Panel Cutout (Filler Panel)
PMC
XPMC 1
Expansion PMC 2 Front Panel Cutout (Filler Panel)
Eject Lever, lower
SYNERGY microsystems
03-0254

*VGM5/PEX3 option front panel*

# Configuration

## Flash write protect

Flash Write protect is the only configuration option for PEX3. After programming the Flash, it can be protected from writes by installing a .100" jumper shunt on JG02. Install this jumper as required for your application. See drawing below.



*PEX3 Flash write protect jumper, JG02*

# Installation

## Installing PMC cards

The PEX3 expansion option lets the VGM5 use up to 3 additional single-width PMC cards. Perform the following steps to install a PMC card onto a PEX3 expansion board. Refer to the PMC Card/PEX3 Expansion Board Installation drawing below for details.

The PEX3 expansion board and its PMC cards are assembled as a sub-unit prior to mating to the SBC. This allows the PMC cards to be secured to the expansion board.

The PEX3 option converts the SBC into a double-wide module. Ensure that space is available in the card cage for your SBC/PEX3 combo.

⚠ PEX3 does not support the VITA 32 extension (Processor PCI Mezzanine Cards, including Second Agent support) to IEEE 1386.1. Therefore, PMCs that use IDSelB (e.g., PMCs with two PCI devices onboard) are not supported by PEX3.

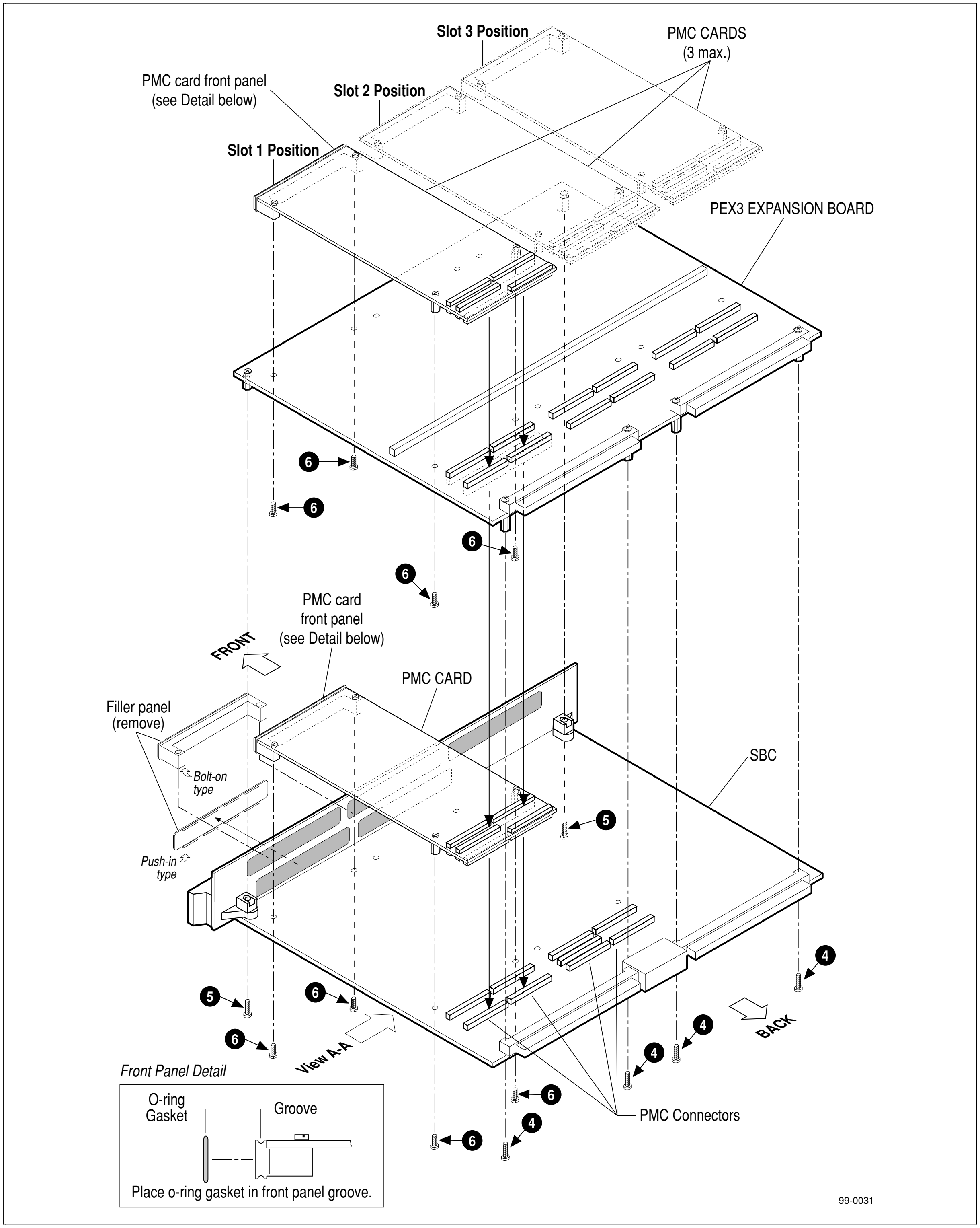
***PMC/PEX3 installation — Required hardware***

| Item locator | Quantity in assy | Synergy part number | Item description |
|---|---|---|---|
| ❸ | 4 | Fas/HSM25F3S1HA | Hex standoff, M2.5 thread, female, 3/16 OD, 1/2 inch long |
| ❹ | 8 | Fas/SwM25PS12S | Screw, M2.5 thread, pan head, slotted, 12 mm long, steel |
| ❺ | 2 | Fas/SwM25PS14S | Screw, M2.5 thread, pan head, slotted, 14 mm long, steel |
| ❻ | 16* | Fas/SwM25PS6S | Screw, M2.5 thread, pan head, slotted, 6 mm long, steel |

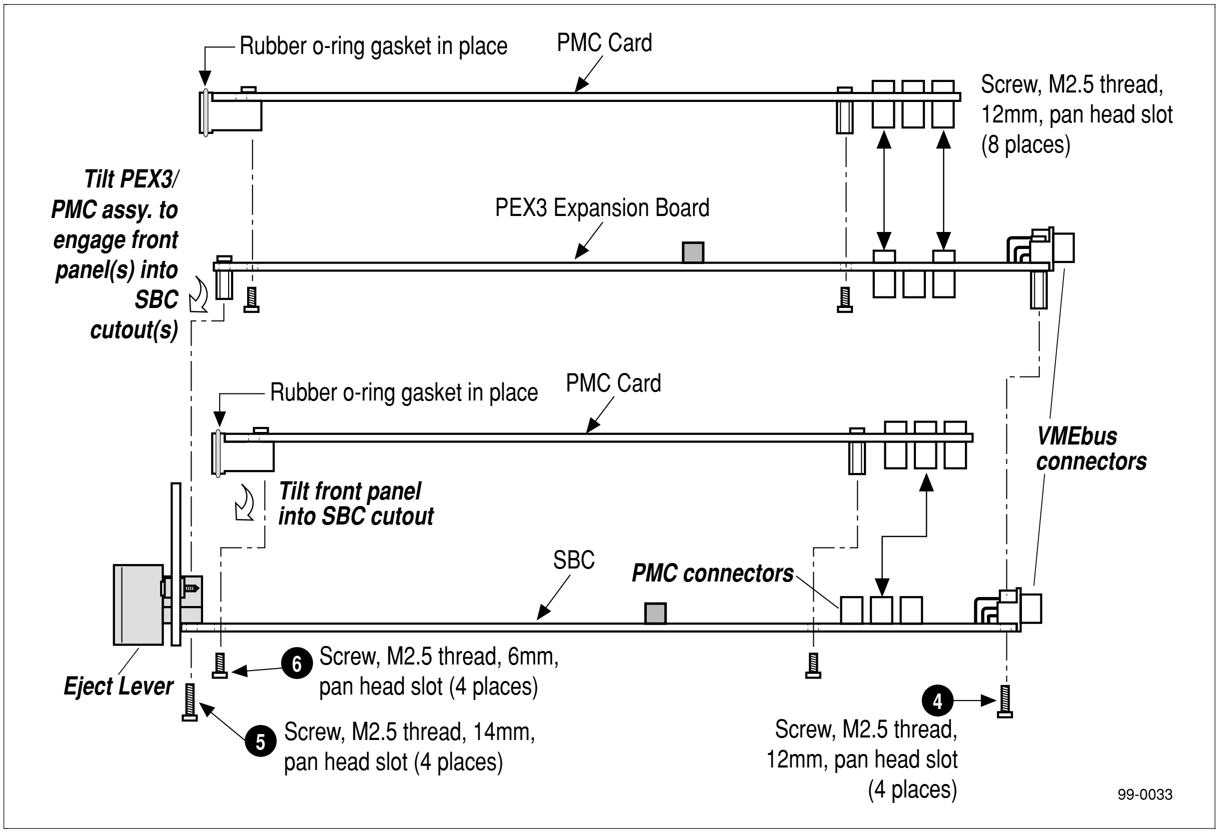
* Qty. is 4 ea. per PMC card. Allow 2 ea. to secure the PSTK or PSTR adapter.

## PSTK/PSTR adapters

The PEX3 expansion carrier board can stack on top of an installed Synergy PMC for additional PMC expansion. The Synergy PSTK or PSTR adapter lets the PEX3 connect to the SBC without the need for a Synergy PMC. Refer to ***PMC stacking and P2 I/O routing*** (page 322) for details on stacking and the use of the PSTK/PSTR adapter.



*PMC card/PEX3 expansion board installation*


Rubber o-ring gasket in place
PMC Card
Screw, M2.5 thread, 12mm, pan head slot (8 places)
Tilt PEX3/ PMC assy. to engage front panel(s) into SBC cutout(s)
PEX3 Expansion Board
Rubber o-ring gasket in place
PMC Card
VMEbus connectors
Tilt front panel into SBC cutout
SBC
PMC connectors
Eject Lever
6
Screw, M2.5 thread, 6mm, pan head slot (4 places)
5
Screw, M2.5 thread, 14mm, pan head slot (4 places)
4
Screw, M2.5 thread, 12mm, pan head slot (4 places)
99-0033

*View A-A, Side View, PMC card/PEX3 expansion board installation*

❶ **Power-down and remove SBC from card cage** — Power-down the system and remove the VGM5 SBC from the card cage.

> ⚠ Synergy SBCs contain static-sensitive devices. Make sure you are properly grounded (by putting on a ground-strap, touching a system ground such as a metallic chassis or case, etc.) before removing and handling the board. Use an ESD-protected workstation for module removal and installation work.

❷ **Remove PEX3 board from SBC** — If not already removed, remove the PEX3 from the SBC by removing 4 ea. screws (item 4) from PEX3's VME connectors and 2 ea. screws (item 5) from underneath the SBC at the front (eject lever).

❸ **Mount PMC card(s) to PEX3 board** (refer to installation drawings above for assembly details):

a. Place PEX3 face-up on a flat surface of an ESD-protected workstation.

b. If not already on, install PMC card's front panel O-ring gasket (included with PMC card) by slipping gasket into groove around front panel.

c. Grasp PMC at sides and place card over PEX3 board's PMC connectors (at slot position A, B or C). Ensure both PMC and PEX3 connectors are aligned then press down over PMC connector area to fully engage PEX3 board connectors.

d. Turn PEX3 board assembly over.

e. Install four 6 mm M2.5 slot-head screws (item 6) from rear (solder) side of PEX3 board. Two screws engage standoffs on the PMC card. The other two screws engage threaded holes in PMC card front panel.

f. Repeat steps 'b' through 'e' for each PMC card to be installed on PEX3 board.

❹ **Remove appropriate PMC filler panel(s) from SBC/PEX3 front panel** — The filler panel will be one of two types. The first type simply snaps in place — remove by pushing from the inside. The second type is an actual blank PMC front panel — remove 2 ea. 6 mm M2.5 slot head securing screws from solder side of board to remove (see PMC card securing screws location drawing above).

❺ **Ensure that stackable Synergy PMC card or PSTK/PSTR adapter is installed on SBC** — The PEX3 board connects to the SBC via a stackable Synergy PMC card or a Synergy PSTK/PSTR adapter. Both of these items have connectors on the opposite side of the SBC connectors for mating with the PEX3 board's plug connectors.

To install a PMC card onto the SBC, refer to the PMC card installation instructions in Section 2 (page 41).

To install the PSTK/PSTR adapter, plug the adapter onto the SBC PMC connectors, then secure with 2 ea. screws (item 6) as if securing the connector portion of a PMC card. (See *PMC card/PEX3 board installation* drawing.)

❻ **Remove rear stiffener bar (if so equipped)** — If the SBC is being fitted with the PEX3 for the first time, a stiffener bar will be installed over the VGM5 VME P1 and P2 connectors. Remove 4 ea. 16 mm securing screws from the VME P1 and P2 connectors. Set aside this hardware plus the 4 ea. 5 mm standoffs and stiffener bar for possible future use. In place of the removed hardware, install four ea. standoffs (item 3) on SBC's VME P1 and P2 connectors using 4 ea. screws (item 4).

❼ **Mount PEX3 board assembly to SBC** — Tilt PEX3 board assembly to engage PMC front panel(s) to SBC front panel cutout(s). Each O-ring gasket on PMC must engage chamfer in front panel cutout. Once all front panels are in place, align PEX3 board connector at bottom of Expansion PMC Slot 1 with SBC's PMC (or PSTK/PSTR stacking adapter if no PMC is used), then press down to seat connector.

*Pre-existing installation* — secure PEX3 board by using two ea. screws (item 5) in front (underneath SBC) to engage PEX3 standoffs and four ea. screws (item 4) in back to engage standoffs on SBC. See *PMC card/PEX3 expansion board installation* drawing for details.

*New installation* — Remove screw and nut securing each SBC eject handle to board. Replace removed hardware with two ea. screws (item 5). Secure PEX3 in front by screwing in these screws to PEX3 standoffs. Finish installation by securing rear of PEX3 with four ea. screws (item 4) which engage standoffs on SBC VME connectors. See *PMC card/PEX3 expansion board installation* drawing for details.

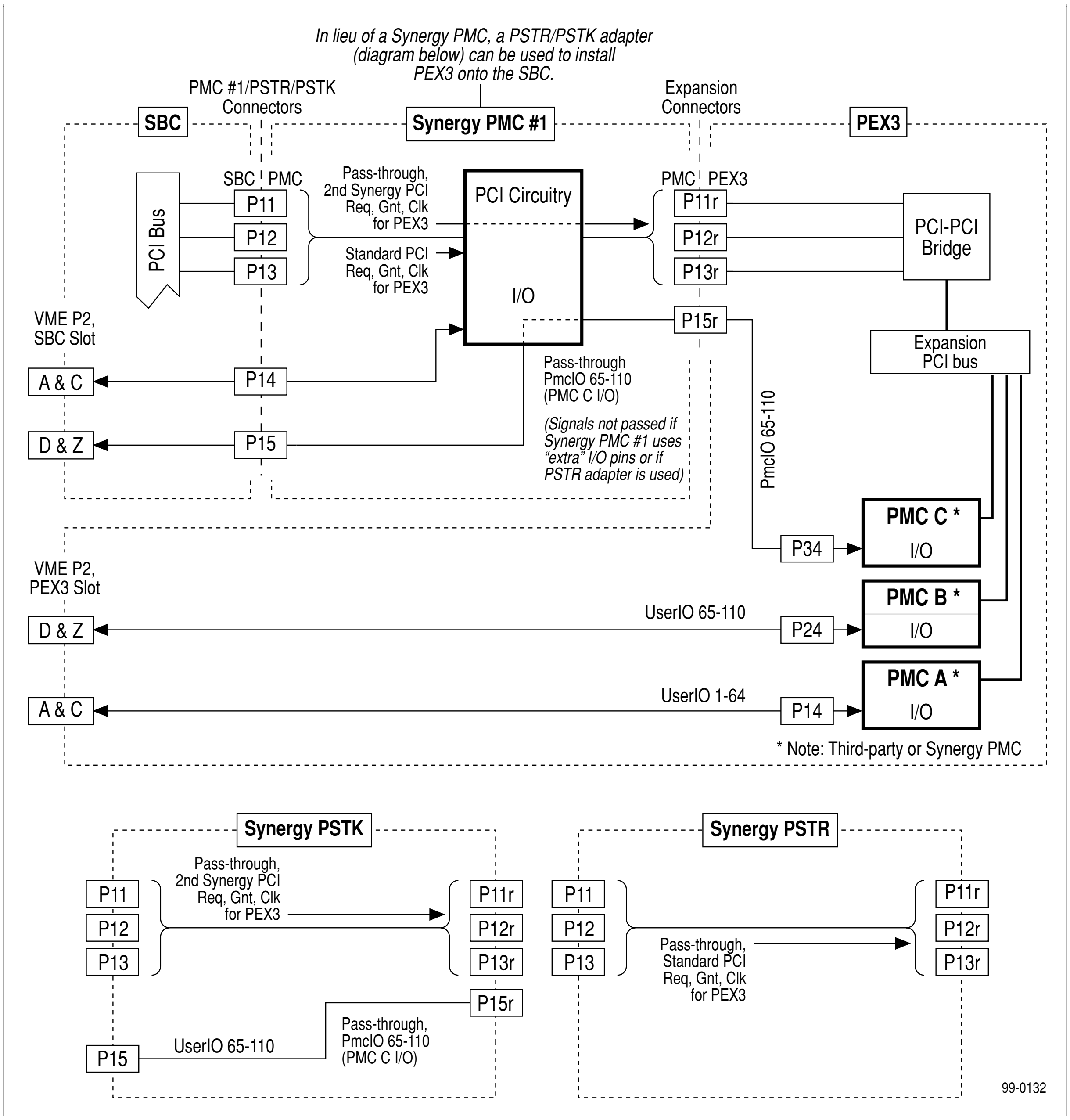
## PMC stacking and P2 I/O routing

The schematic below shows the onboard PMC and PMC expansion PCI bus connections and the I/O routing through the VME P2 user I/O pins. Use this diagram to plan your PMC installation and P2 I/O wiring.

## PMC P2 I/O restriction

The Slot C (or XPMC 3) PMC card on the optional PEX3 expansion module cannot use the host SBC's P2 I/O if the onboard Synergy PMC card uses extended P2 I/O pins (i.e., uses SBC P2 D & Z rows) or if the PSTR adapter is used. The Slot C PMC card can use the SBC's P2 I/O if the onboard Synergy PMC uses standard P2 I/O (e.g., uses only rows A and C) or if the PSTK (not PSTR) adapter is used.

Note that all PMC cards in the system are free to use front panel I/O without restriction.


In lieu of a Synergy PMC, a PSTR/PSTK adapter (diagram below) can be used to install PEX3 onto the SBC.
PMC #1/PSTR/PSTK Connectors
SBC
Synergy PMC #1
Expansion Connectors
PEX3
PCI Bus
SBC
PMC
P11
P12
P13
Pass-through, 2nd Synergy PCI Req, Gnt, Clk for PEX3
Standard PCI Req, Gnt, Clk for PEX3
PCI Circuitry
I/O
PMC
PEX3
P11r
P12r
P13r
P15r
PCI-PCI Bridge
Expansion PCI bus
VME P2, SBC Slot
A & C
P14
D & Z
P15
Pass-through PmcIO 65-110 (PMC C I/O)
(Signals not passed if Synergy PMC #1 uses "extra" I/O pins or if PSTR adapter is used)
PmcIO 65-110
PMC C *
I/O
P34
VME P2, PEX3 Slot
D & Z
UserIO 65-110
P24
PMC B *
I/O
A & C
UserIO 1-64
P14
PMC A *
I/O
* Note: Third-party or Synergy PMC
Synergy PSTK
P11
P12
P13
Pass-through, 2nd Synergy PCI Req, Gnt, Clk for PEX3
P11r
P12r
P13r
P15r
P15
UserIO 65-110
Pass-through, PmcIO 65-110 (PMC C I/O)
Synergy PSTR
P11
P12
P13
Pass-through, Standard PCI Req, Gnt, Clk for PEX3
P11r
P12r
P13r
99-0132

**PMC P2 I/O routing/stacking**

# Operation

## Address map

The table below lists the PEX3 onboard memory and register addresses.

**PEX3 memory address map**

| PCI 9080 Space | Local Address | Device | Access |
|---|---|---|---|
| Space 0 | 0x0000_0000 – 0x3FFF_FFFF | Flash | D8, D16, D32 (R)<br>D32 (W*) |
| Space 1 | 0x4000_0000 – 0x43FF_FFFF | SDRAM | D8, D16, D32 (RW) |
| Space 0 | 0xC000_0000 | Board Type register | D8 (RO) |
| Space 0 | 0xC000_0004 | Revision and ECO Level register | D8 (RO) |
| Space 0 | 0xC000_0008 | Flash Configuration register | D8 (RO) |
| Space 0 | 0xC000_000C | DRAM Configuration register | D8 (RO) |

*Note: Flash write width depends on device type, mode, and address.

## PMC PCI interrupts

A PEX3 PMC responds to and generates certain PCI interrupts depending on the PMC slot in which it is installed. The table below lists the PEX3 PMC slots' associated PCI interrupt lines.

**PEX3 PCI interrupts**

| Slot 1 (A) | Slot 2 (B) | Slot 3 (C) | SBC PCI Bus |
|---|---|---|---|
| IntD | IntC | IntB | IntA |
| IntA | IntD | IntC | Int B |
| IntB | IntA | IntD | IntC |
| IntC | IntB | IntA | IntD |

For example, if a board in PEX3's Slot #1 drives its IntA line, it will be mapped to the SBC's PCI IntB input. This is an interrupt rotation of 1. Slot #2 rotates 2 (IntA assertion gets mapped to SBC's PCI IntC input) and Slot #3 rotates 3 (IntA assertion gets mapped to SBC's PCI IntD input).

## PCI Type 0 configuration and address

The table below lists the PEX3's PCI configuration.

**PEX3 PCI Type 0 configuration**

| Device Number | ID Select | Bus Master no. | Owner | PCI Config. Address[1] |
|---|---|---|---|---|
| 0 | 16 | 0 | PMC Slot 1 (A) | 0x800n_0000 |
| 1 | 17 | 1 | PMC Slot 2 (B) | 0x800n_0800 |
| 2 | 18 | 2 | PMC Slot 3 (C) | 0x800n_1000 |
| 3 | 19 | 3 | Flash | 0x800n_1800 |

Notes: 1. n = PCI Secondary Bus number for PEX3 21154 bridge.

## PCI configuration

The PEX3's 9080 PCI bus mastering interface chip is automatically configured via PCI configuration accesses during system startup. The board's 9080 driver and the onboard serial EEPROM work in tandem to set up the chip.

The table below shows the PCI 9080 configuration space.

**PEX3, PCI 9080 PCI configuration registers**

| 31 | 16 | 15 | 0 | PCI Access |
|---|---|---|---|---|
| Device ID = 0x9080 | | Vendor ID = 0x10B5 | | 0x00[1] |
| Status | | Command | | 0x04[2] |
| Class Code = 0x068000[1] | | | Rev. ID = 0x05 | 0x08 |
| BIST | Header Type | PCI Bus Latency Timer | Cache Line Size[2] | 0x0C |
| PCI Base Addr 0, Memory Mapped Config. Registers (PCIBAR0) | | | | 0x10[2] |
| PCI Base Addr 1, I/O Mapped Config. Registers (PCIBAR1) | | | | 0x14[2] |
| PCI Base Addr 2, Local Address Space 0 (PCIBAR2, **PEX3 Flash**) | | | | 0x18[2] |
| PCI Base Addr 3, Local Address Space 1 (PCIBAR3, **PEX3 SDRAM**) | | | | 0x1C[2] |
| Unused Base Address (PCIBAR4) | | | | 0x20 |
| Unused Base Address (PCIBAR5) | | | | 0x24 |
| Cardbus CIS Pointer (not supported) | | | | 0x28 |
| Subsystem ID = 0x2321 | | Subsystem Vendor ID = 0x80F6 | | 0x2C[1] |
| PCI Base Address for Local Expansion ROM | | | | 0x30 |
| Reserved | | | | 0x34 |
| Reserved | | | | 0x38 |
| Max_Lat[1] | Min_Gnt[1] | Interrupt Pin[1] | Interrupt Line[1, 2] | 0x3C |

**Notes**: Shaded = Register NOT USED for PEX3
1. EEPROM Writeable
2. PCI Writeable

## PCI 9080 basic set up

The PCI 9080 set up for PEX3 is summarized below. Since PCI 9080 setup is done by Synergy's SMon software and/or BSP OS (Linux, VxWorks), the following is provided for information only.

The PEX3 PCI 9080 operates in C bus mode (separate nonmultiplexed 32-bit address & data busses) with no local masters, i.e.; no Req used on the local bus.

Note Refer to the PLX PCI 9080 datasheet for detailed device information. The PCI 9080 datasheet is available as a PDF file from the PLX Technology website:

**www.plxtech.com**

The PCI 9080 has three local address spaces of which two are used by the PEX3, Space 0 and Space 1.

Space 0 is used for Flash memory in operation mode, and capability register space at driver initialization time.

- BTerm\ and Burst disabled and READY# Input enabled (hardware wait states)
- Read-ahead disabled during programming, enabled during operation (if so configured)
- Local base address = 0x0000_0000 for Flash; 0xC000_0000 for registers

Space 1 is used for SDRAM memory.

- BTerm\ and Burst enabled and READY# Input enabled (hardware wait states)
- Read-ahead enabled if so configured
- Local base address = 0x4000_0000

## Registers

In addition to the PCI configuration registers, the 9080 chip has these groups of registers.

- Local Configuration Registers
- Runtime Registers (typically not used for PEX3)
- DMA Registers
- Messaging Queue Registers (typically not used for PEX3)

Refer to the PLX PCI 9080 documentation for detailed 9080 register information.

The PEX3 is also provided with the following onboard capability registers.

- Board Type Register
- Revision and ECO Level Register
- Flash Configuration Register
- SDRAM Configuration Register

The following describes PEX3's onboard read-only registers. These registers are intended to be read during driver initialization time using Space 0. This information configures the driver accordingly.

The register bit description uses the notation listed below in each bit position to show the register's value after a board reset (i.e., power cycling or system reset).

**Register bit description notations for reset value**

| Notation | What it means |
|---|---|
| *x* | Unused bit; set to 0 for future compatibility |
| — | Read-only bit |
| *1* | Set to 1 upon reset |
| *0* | Set to 0 upon reset |

## Board type register, 0xC000_0000 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *0* | *0* | *0* | *0* | *0* | *0* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b4 | Board Type | 0x0 = PEX3 |
| b3–b0 | Reserved | — |

A byte read of this register indicates board type.

## Revision and ECO level register, 0xC000_0004 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | — | — | — | — | — | — | — | — | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b4 | Board Revision | 0x0 = a<br>0x1 = b<br>0x2 = c<br>↓ :<br>0xF = p |
| b3–b0 | ECO Level | 0x0 = none<br>0x1 = 1<br>0x2 = 2<br>↓ :<br>0xF = 15 |

A byte read of this register reveals the board revision level (higher order nibble) and ECO level (lower order nibble).

## Flash configuration register, 0xC000_0008 (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *x* | *x* | *0* | *0* | *x* | *x* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b6 | Reserved | — |
| b5–b3 | Flash ROM Chip Size | 0x0 = 2x16Mb chips or 4 MB per bank<br>0x1 = 2x32Mb chips or 8 MB per bank<br>0x2 = 2x64Mb chips or 16 MB per bank<br>0x3 = 2x128Mb chips or 32 MB per bank |
| b3–b2 | Reserved | — |
| b1–b0 | Number of Flash ROM banks installed | 0x0 = No Flash installed<br>0x1 = 1 bank installed<br>0x2 = 2 banks installed<br>0x3 = 4 banks installed |

Init code is to read the contents of this register to determine how Flash (if installed) is to be set up.

## DRAM configuration register, 0xC000_000C (RO)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | *0* | *0* | *x* | *x* | *0* | *0* | *x* | *x* | ***Reset value*** |

Bit assignments:

| Bit(s) | Function | Values |
|---|---|---|
| b7–b6 | Reserved | — |
| b5–b3 | DRAM Chip Size | 0x0 = 2x64Mb chips or 16 MB per bank<br>0x1 = 2x128Mb chips or 32 MB per bank<br>0x2 = 2x256Mb chips or 64 MB per bank<br>0x3 = Reserved for future use |
| b3–b2 | Reserved | — |
| b1–b0 | Number of DRAM banks installed | 0x0 = No DRAM installed<br>0x1 = 1 bank installed<br>0x2 = 2 banks installed<br>0x3 = 4 banks installed |

Init code is to read the contents of this register to determine how DRAM (if installed) is to be set up.

## Using PEX3 memory

The PEX3's Flash and SDRAM each have a PCI window pointed to by the 9080's PCI Base Address Registers (BAR) 2 and 3, respectively. Upon reset, the device reads its configuration EEPROM to get the initial values for all of its registers. The PCI windows are then normally assigned to particular addresses by PCI auto-initialization code, such as that run by SMon, the Synergy ROM monitor. Software can then determine where the PCI windows have been allocated by issuing a find-PCI-device (or similar) system call and reading the device's BARs.

Once the memory is mapped into PCI space, it is accessible to the host processor(s) and other PCI devices in the system. The dual DMA channels of the PCI interface allows PCI bus mastering for fast memory accesses. Typical uses include any combination of:

- Expansion RAM
- Expansion Flash
- Buffer for a PMC I/O board with DMA
- Buffer for a passive (target-only) PMC I/O board

If a given PCI window is at least as large as the memory behind it, no windowing control is needed: all of the memory is directly accessible. For smaller windows, the 9080's Local Base Address register must be adjusted.

Reading the Flash is similar to reading PEX3 SDRAM (i.e., both are accessed through a window). Writing to Flash, however, requires special code. The PowerPC Series SMon monitor includes a full set of Flash commands. The Flash driver is also available from Synergy. Contact Customer Service for more information.

### *VxWorks BSP PEX3 driver*

Synergy's VxWorks BSP includes a PEX3 driver named 'exmem'. The header file for it is exmem.h. To include the driver in the kernel, define INCLUDE_EXMEM. The driver is initialized on startup.

The PEX3 Flash and DRAM are accessed through a PCI window. The window size is stored in the PEX3's PCI-configuration EEPROM. Use the wrEEPEX3() SMon routine to change the size of this window as required. Access the DRAM window by first calling exmemDRamAdr() with a DRAM offset. This points the window to that section of DRAM and returns the actual access address.

Note that if the DRAM needs to be accessed by multiple threads, use a wrapper routine and a window-ownership semaphore to manage it.

## *PEX3 connector pinouts*

⚠ Because the VME P2 connectors have user-defined pins, make sure that your backplane's P2 connectors are compatible with the PEX3's P2 wiring before powering up the board. **Failure to observe this warning can cause the complete destruction of many on-board components and also voids the product warranty.**

## VMEbus connectors (P1 & P2)

### VMEbus P1 connector pinouts

| Pin | Row Z[1] | Row A | Row B | Row C | Row D[1] |
|---|---|---|---|---|---|
| 1 | — | — | — | — | +5V |
| 2 | Gnd | — | — | — | Gnd |
| 3 | — | — | — | — | — |
| 4 | Gnd | — | BG0\ | — | — |
| 5 | — | — | BG0\ | — | — |
| 6 | Gnd | — | BG1\ | — | — |
| 7 | — | — | BG1\ | — | — |
| 8 | Gnd | — | BG2\ | — | — |
| 9 | — | Gnd | BG2\ | Gnd | — |
| 10 | Gnd | — | BG3\ | — | — |
| 11 | — | Gnd | BG3\ | — | — |
| 12 | Gnd | — | — | — | +3.3V |
| 13 | — | — | — | — | — |
| 14 | Gnd | — | — | — | +3.3V |
| 15 | — | Gnd | — | — | — |
| 16 | Gnd | — | — | — | +3.3V |
| 17 | — | Gnd | — | — | — |
| 18 | Gnd | — | — | — | +3.3V |
| 19 | — | Gnd | — | — | — |
| 20 | Gnd | — | Gnd | — | +3.3V |
| 21 | — | IAck\ | — | — | — |
| 22 | Gnd | IAck\ | — | — | +3.3V |
| 23 | — | — | Gnd | — | — |
| 24 | Gnd | — | — | — | +3.3V |
| 25 | — | — | — | — | — |
| 26 | Gnd | — | — | — | +3.3V |
| 27 | — | — | — | — | — |
| 28 | Gnd | — | — | — | +3.3V |
| 29 | — | — | — | — | — |
| 30 | Gnd | — | — | — | +3.3V |
| 31 | — | -12V | — | +12V | Gnd |
| 32 | Gnd | +5V | +5V | +5V | — |

***Notes:*** 1. This row present only with optional wide (160-pin) VMEbus P1 & P2 connectors.

### VMEbus P2 connector pinouts

| Pin | Row Z[1,3] | Row A [2] | Row B | Row C[2] | Row D[1,3] |
|---|---|---|---|---|---|
| 1 | (UsrIO66) | (UsrIO2) | +5V | (UsrIO1) | (UsrIO65) |
| 2 | Gnd | (UsrIO4) | Gnd | (UsrIO3) | (UsrIO67) |
| 3 | (UsrIO69) | (UsrIO6) | — | (UsrIO5) | (UsrIO68) |
| 4 | Gnd | (UsrIO8) | — | (UsrIO7) | (UsrIO70) |
| 5 | (UsrIO72) | (UsrIO10) | — | (UsrIO9) | (UsrIO71) |
| 6 | Gnd | (UsrIO12) | — | (UsrIO11) | (UsrIO73) |
| 7 | (UsrIO75) | (UsrIO14) | — | (UsrIO13) | (UsrIO74) |
| 8 | Gnd | (UsrIO16) | — | (UsrIO15) | (UsrIO76) |
| 9 | (UsrIO78) | (UsrIO18) | — | (UsrIO17) | (UsrIO77) |
| 10 | Gnd | (UsrIO20) | — | (UsrIO19) | (UsrIO79) |
| 11 | (UsrIO81) | (UsrIO22) | — | (UsrIO21) | (UsrIO80) |
| 12 | Gnd | (UsrIO24) | Gnd | (UsrIO23) | (UsrIO82) |
| 13 | (UsrIO84) | (UsrIO26) | +5V | (UsrIO25) | (UsrIO83) |
| 14 | Gnd | (UsrIO28) | — | (UsrIO27) | (UsrIO85) |
| 15 | (UsrIO87) | (UsrIO30) | — | (UsrIO29) | (UsrIO86) |
| 16 | Gnd | (UsrIO32) | — | (UsrIO31) | (UsrIO88) |
| 17 | (UsrIO90) | (UsrIO34) | — | (UsrIO33) | (UsrIO89) |
| 18 | Gnd | (UsrIO36) | — | (UsrIO35) | (UsrIO91) |
| 19 | (UsrIO93) | (UsrIO38) | — | (UsrIO37) | (UsrIO92) |
| 20 | Gnd | (UsrIO40) | — | (UsrIO39) | (UsrIO94) |
| 21 | (UsrIO96) | (UsrIO42) | — | (UsrIO41) | (UsrIO95) |
| 22 | Gnd | (UsrIO44) | Gnd | (UsrIO43) | (UsrIO97) |
| 23 | (UsrIO99) | (UsrIO46) | — | (UsrIO45) | (UsrIO98) |
| 24 | Gnd | (UsrIO48) | — | (UsrIO47) | (UsrIO100) |
| 25 | (UsrIO102) | (UsrIO50) | — | (UsrIO49) | (UsrIO101) |
| 26 | Gnd | (UsrIO52) | — | (UsrIO51) | (UsrIO103) |
| 27 | (UsrIO105) | (UsrIO54) | — | (UsrIO53) | (UsrIO104) |
| 28 | Gnd | (UsrIO56) | — | (UsrIO55) | (UsrIO106) |
| 29 | (UsrIO108) | (UsrIO58) | — | (UsrIO57) | (UsrIO107) |
| 30 | Gnd | (UsrIO60) | — | (UsrIO59) | (UsrIO109) |
| 31 | (UsrIO110) | (UsrIO62) | Gnd | (UsrIO61) | Gnd |
| 32 | Gnd | (UsrIO64) | +5V | (UsrIO63) | — |

***Notes:***

1. Pins in this row connect to Expansion PMC 2's P24 connector pin indicated in parentheses. Space is provided in these columns to write in the assigned signals, if desired. Refer to the applicable PMC module documentation for P2 pin assignments.
2. Pins in this row connect to Expansion PMC 1's P14 connector pin indicated in parentheses. Space is provided in these columns to write in the assigned signals, if desired. Refer to the applicable PMC module documentation for P2 pin assignments.
3. This row present only with optional wide (160-pin) VMEbus P1 & P2 connectors.

## PMC connectors

### PMC connector, P11, P21, & P31 pinouts

| Pin | Function (P11, P21, P31) | Pin | Function (P11, P21, P31) |
|---|---|---|---|
| 1 | Gnd | 2 | -12V |
| 3 | Gnd | 4 | (IntB\, IntC\, IntD\)[1] |
| 5 | (IntC\, IntD\, IntA\)[1] | 6 | (IntD\, IntA\, IntB\)[1] |
| 7 | — | 8 | +5V |
| 9 | (IntA\, IntB\, IntC\)[1] | 10 | — |
| 11 | Gnd | 12 | — |
| 13 | Clk | 14 | Gnd |
| 15 | Gnd | 16 | Gnt\ |
| 17 | Req\ | 18 | +5V |
| 19 | VI/O | 20 | AD31 |
| 21 | AD28 | 22 | AD27 |
| 23 | AD25 | 24 | Gnd |
| 25 | Gnd | 26 | CBE3\ |
| 27 | AD22 | 28 | AD21 |
| 29 | AD19 | 30 | +5V |
| 31 | VI/O | 32 | AD17 |
| 33 | Frame\ | 34 | Gnd |
| 35 | Gnd | 36 | IRdy\ |
| 37 | DevSel\ | 38 | +5V |
| 39 | Gnd | 40 | Lock\ |
| 41 | — | 42 | — |
| 43 | Par | 44 | Gnd |
| 45 | VI/O | 46 | AD15 |
| 47 | AD12 | 48 | AD11 |
| 49 | AD9 | 50 | +5V |
| 51 | Gnd | 52 | CBE0\ |
| 53 | AD6 | 54 | AD5 |
| 55 | AD4 | 56 | Gnd |
| 57 | VI/O | 58 | AD3 |
| 59 | AD2 | 60 | AD1 |
| 61 | AD0 | 62 | +5V |
| 63 | Gnd | 64 | Req64\ |

Note: 1. PMC interrupts set in hardware as follows: PEX3 PMC 1 Int rotates by 1 (e.g., IntA\ = IntB\); PEX 3 PMC 2 rotates by 2 (e.g., IntA\ = IntC\); PEX 3 PMC 3 rotates by 3 (e.g., IntA\ = IntD\). See page 324.

### *PMC connector, P12, P22, & P32 pinouts*

| Pin | Function (P12, P22, P32) | Pin | Function (P12, P22, P32) |
|---|---|---|---|
| **1** | +12V | **2** | Gnd |
| **3** | +3.3V | **4** | — |
| **5** | +3.3V | **6** | Gnd |
| **7** | Gnd | **8** | — |
| **9** | — | **10** | — |
| **11** | +3.3V | **12** | +3.3V |
| **13** | Rst\ | **14** | Gnd |
| **15** | +3.3V | **16** | Gnd |
| **17** | — | **18** | Gnd |
| **19** | AD30 | **20** | AD29 |
| **21** | Gnd | **22** | AD26 |
| **23** | AD24 | **24** | +3.3V |
| **25** | (AD16, AD17, AD18) IDSel | **26** | AD23 |
| **27** | +3.3V | **28** | AD20 |
| **29** | AD18 | **30** | Gnd |
| **31** | AD16 | **32** | CBE2\ |
| **33** | Gnd | **34** | — |
| **35** | TRdy\ | **36** | +3.3V |
| **37** | Gnd | **38** | Stop\ |
| **39** | PErr\ | **40** | Gnd |
| **41** | +3.3V | **42** | SErr\ |
| **43** | CBE1\ | **44** | Gnd |
| **45** | AD14 | **46** | AD13 |
| **47** | Gnd | **48** | AD10 |
| **49** | AD8 | **50** | +3.3V |
| **51** | AD7 | **52** | — |
| **53** | +3.3V | **54** | — |
| **55** | — | **56** | Gnd |
| **57** | — | **58** | — |
| **59** | Gnd | **60** | — |
| **61** | Ack64\ | **62** | +3.3V |
| **63** | Gnd | **64** | — |

### PMC connector, P13, P23, & P33 pinouts

| Pin | Function (P13, P23, P33) | Pin | Function (P13, P23, P33) |
|---|---|---|---|
| 1 | — | 2 | Gnd |
| 3 | Gnd | 4 | CBE7\ |
| 5 | CBE6\ | 6 | CBE5\ |
| 7 | CBE4\ | 8 | Gnd |
| 9 | VI/O | 10 | Par64 |
| 11 | AD63 | 12 | AD62 |
| 13 | AD61 | 14 | Gnd |
| 15 | Gnd | 16 | AD60 |
| 17 | AD59 | 18 | AD58 |
| 19 | AD57 | 20 | Gnd |
| 21 | VI/O | 22 | AD56 |
| 23 | AD55 | 24 | AD54 |
| 25 | AD53 | 26 | Gnd |
| 27 | Gnd | 28 | AD52 |
| 29 | AD51 | 30 | AD50 |
| 31 | AD49 | 32 | Gnd |
| 33 | Gnd | 34 | AD48 |
| 35 | AD47 | 36 | AD46 |
| 37 | AD45 | 38 | Gnd |
| 39 | VI/O | 40 | AD44 |
| 41 | AD43 | 42 | AD42 |
| 43 | AD41 | 44 | Gnd |
| 45 | Gnd | 46 | AD40 |
| 47 | AD39 | 48 | AD38 |
| 49 | AD37 | 50 | Gnd |
| 51 | Gnd | 52 | AD36 |
| 53 | AD35 | 54 | AD34 |
| 55 | AD33 | 56 | Gnd |
| 57 | VI/O | 58 | AD32 |
| 59 | — | 60 | — |
| 61 | — | 62 | Gnd |
| 63 | Gnd | 64 | — |

### PMC connector, P14 pinouts (PMC 1)

| Pin | Function[1] | Pin | Function[1] |
|---|---|---|---|
| 1 | UsrIO1 | 2 | UsrIO2 |
| 3 | UsrIO3 | 4 | UsrIO4 |
| 5 | UsrIO5 | 6 | UsrIO6 |
| 7 | UsrIO7 | 8 | UsrIO8 |
| 9 | UsrIO9 | 10 | UsrIO10 |
| 11 | UsrIO11 | 12 | UsrIO12 |
| 13 | UsrIO13 | 14 | UsrIO14 |
| 15 | UsrIO15 | 16 | UsrIO16 |
| 17 | UsrIO17 | 18 | UsrIO18 |
| 19 | UsrIO19 | 20 | UsrIO20 |
| 21 | UsrIO21 | 22 | UsrIO22 |
| 23 | UsrIO23 | 24 | UsrIO24 |
| 25 | UsrIO25 | 26 | UsrIO26 |
| 27 | UsrIO27 | 28 | UsrIO28 |
| 29 | UsrIO29 | 30 | UsrIO30 |
| 31 | UsrIO31 | 32 | UsrIO32 |
| 33 | UsrIO33 | 34 | UsrIO34 |
| 35 | UsrIO35 | 36 | UsrIO36 |
| 37 | UsrIO37 | 38 | UsrIO38 |
| 39 | UsrIO39 | 40 | UsrIO40 |
| 41 | UsrIO41 | 42 | UsrIO42 |
| 43 | UsrIO43 | 44 | UsrIO44 |
| 45 | UsrIO45 | 46 | UsrIO46 |
| 47 | UsrIO47 | 48 | UsrIO48 |
| 49 | UsrIO49 | 50 | UsrIO50 |
| 51 | UsrIO51 | 52 | UsrIO52 |
| 53 | UsrIO53 | 54 | UsrIO54 |
| 55 | UsrIO55 | 56 | UsrIO56 |
| 57 | UsrIO57 | 58 | UsrIO58 |
| 59 | UsrIO59 | 60 | UsrIO60 |
| 61 | UsrIO61 | 62 | UsrIO62 |
| 63 | UsrIO63 | 64 | UsrIO64 |

Note: 1. The function of pins labeled ‘UsrIOxxx’ depends on the add-on card installed on the board. Space is provided in these columns to write the assigned signals if desired.

### PMC connector, P24 (PMC 2) & P34 (PMC 3) pinouts

| Pin | Function (P24, P34)[1] | Pin | Function (P24, P34)[1] |
|---|---|---|---|
| 1 | UsrIO65 | 2 | UsrIO66 |
| 3 | UsrIO67 | 4 | UsrIO68 |
| 5 | UsrIO69 | 6 | UsrIO70 |
| 7 | UsrIO71 | 8 | UsrIO72 |
| 9 | UsrIO73 | 10 | UsrIO74 |
| 11 | UsrIO75 | 12 | UsrIO76 |
| 13 | UsrIO77 | 14 | UsrIO78 |
| 15 | UsrIO79 | 16 | UsrIO80 |
| 17 | UsrIO81 | 18 | UsrIO82 |
| 19 | UsrIO83 | 20 | UsrIO84 |
| 21 | UsrIO85 | 22 | UsrIO86 |
| 23 | UsrIO87 | 24 | UsrIO88 |
| 25 | UsrIO89 | 26 | UsrIO90 |
| 27 | UsrIO91 | 28 | UsrIO92 |
| 29 | UsrIO93 | 30 | UsrIO94 |
| 31 | UsrIO95 | 32 | UsrIO96 |
| 33 | UsrIO97 | 34 | UsrIO98 |
| 35 | UsrIO99 | 36 | UsrIO100 |
| 37 | UsrIO101 | 38 | UsrIO102 |
| 39 | UsrIO103 | 40 | UsrIO104 |
| 41 | UsrIO105 | 42 | UsrIO106 |
| 43 | UsrIO107 | 44 | UsrIO108 |
| 45 | UsrIO109 | 46 | UsrIO110 |
| 47 | — | 48 | — |
| 49 | — | 50 | — |
| 51 | — | 52 | — |
| 53 | — | 54 | — |
| 55 | — | 56 | — |
| 57 | — | 58 | — |
| 59 | — | 60 | — |
| 61 | — | 62 | — |
| 63 | — | 64 | — |

Note: 1. The function of pins labeled ‘UsrIOxxx’ depends on the add-on card installed on the board. Space is provided in these columns to write the assigned signals if desired.

# Glossary

The paragraphs below define and describe some of the terms used in this manual. The definition entries observe the following conventions:

- Terms in definitions that appear (*in italics and in parentheses*) are related and/or alternative terms or acronym translations for the term being defined.
- Terms in definitions that appear in **boldface** in definitions are defined elsewhere in the glossary.

***10Base-T*** a type of Ethernet that uses unshielded twisted-pair (UTP) cable and modular RJ-45 connectors for LAN connections in a star configuration (i.e., each network node connects to a common hub). Data rate is the same as standard Ethernet: 10 Mbps.

***100Base-T*** similar to 10Base-T except that the data rate is 100 Mbps. 100Base-TX uses two pairs of a Category 5 cable. 100Base-T4 uses 4 pairs of a Category 3 cable. Also called **Fast Ethernet**.

***A16/D16*** specifies a microprocessor bus' address and data bus size. This value specifies a 16-bit wide address bus and 16-bit wide data bus.

***A16/D32*** specifies a 16-bit wide address bus and 32-bit wide data bus

***A24/D16*** specifies a 24-bit wide address bus and 16-bit wide data bus.

***A24/D32*** specifies a 24-bit wide address bus and 32-bit wide data bus.

***A32/D16*** specifies a 32-bit wide address bus and 16-bit wide data bus.

***A32/D32*** specifies a 32-bit wide address bus and 32-bit wide data bus.

***AM code bits*** *(Address Modifier)* code bits, AM0–AM5, used by the VMEbus to identify the size of address being expressed (A16, A24, A32, A40 or A64) and the type of transfer (Address-only, program, data, BLT32, BLT64 or IAck) being performed.

***banner*** a message displayed on a **CRT** screen when a debug monitor or operating system is starting.

***base address*** the lowest address in a range of addresses. Usually the lowest address of a memory window, or of a set of peripheral registers

***BCD*** *(Binary Coded Decimal)* a coding system in which four binary (1s and 0s) digits represent each digit in a decimal (0 through 9) value.

***Big-Endian*** see **Endian**.

***bit*** the smallest unit of data represented as either a 1 (ON or true) or 0 (OFF or false).

***BLT*** (*Block Transfer*) a data transfer method for moving large amounts (blocks) of data. A BLT cycle is faster and more efficient than a regular R/W cycle because the address to start the transfer of multiple bytes is presented only once.

***bridge*** a chip that connects two different busses together. A bridge may be either transparent, meaning that it does not translate the addresses passing through it, or it may be non-transparent, meaning that it translates addresses.

***byte*** a unit of data eight bits in length.

***cache line*** the amount of memory read into or out of cache in a single operation. This is 32 bytes in 60x and 7xx PowerPC processors.

***cache memory*** special RAM memory that provides the processor with quicker, more direct access to data. The use of cache memory increases performance as time is saved by not having to access the relatively slower main memory circuits for data. See also **L1 cache** and **L2 cache**.

***category 3*** unshielded twisted-pair cable specification that functions at 10 Megabits per second on each pair.

***category 5*** unshielded twisted-pair cable specification that functions at 100 Megabits per second on each pair.

***clock/calendar*** a device that records the progress of the time and date and makes this information available to programs running on the computer system.

***collision*** the simultaneous transmission of Ethernet packets by two or more Ethernet nodes, resulting in a garbled transmission. A collision occurs when two Ethernet nodes attempt to send a packet at the same time. Collisions are handled with the CSMA/CD protocol.

***CPU*** *(Central Processing Unit)* central controlling device in a computer system.

***CRT*** *(Cathode Ray Tube)* normally refers to a viewing screen; also used as a synonym for **terminal**.

***data broadcasting*** a bus communications technique in which a single CPU board can send data to multiple CPU boards at the same time.

***DB-9*** a “D” shaped serial interface connector for **I/O** cabling that provides access to up to 9 separate lines or pins on a matching connector.

***DB-25*** a “D” shaped serial interface connector for **I/O** cabling that provides access to up to 25 separate lines or pins on a matching connector.

***DCE*** *(Data Communications Equipment)* the end of a serial communications link that is, or mimics, a modem (opposite of **DTE**).

***differential*** a method of signaling in which two wires are used, each carrying opposite versions of the signal information. This is done to increase maximum cable drive and to increase noise immunity. For example, a pair of signals are called SD0+ and SD0-. A 1 data bit may be represented by +5V on SD0+ and 0V on SD0-, and a 0 bit by 0V on SD0+ and +5V on SD0-. See **single-ended**.

***DMA*** *(Direct Memory Access)* a data transfer method in which data can pass between peripheral devices and memory without intervention by the **CPU**.

***DRAM*** *(Dynamic Random Access Memory)* high density fast access memory storage media that must be refreshed at continuous intervals. Also simply referred to as RAM

***DTE*** *(Data Terminal Equipment)* the end of a serial communications link that is, or mimics, a terminal or printer (opposite of **DCE**).

***DUART*** *(Dual Universal Asynchronous Receiver/ Transmitter)* see **UART.**

***dynamic RAM*** see **DRAM**.

***ECO*** *(Engineering Change Order)* an engineering document that describes and orders a change to a released product.

***Endian*** refers to the addressing of individual bytes within a 16, 32 or 64-bit number. Byte ordering that begins with the highest order byte as Byte 0 is referred to as **Big-Endian**. Byte ordering that begins with the lowest order byte as Byte 0 is referred to as **Little-Endian**. The programming community borrowed the Endian terms from the story “Gulliver’s Travels” by Jonathan Swift. In Swift’s novel, there were two ways of breaking eggs before eating them. People who broke their eggs from the large end were called Big-Endians; people who broke their eggs from the small end were called Little-Endians.

***EPROM*** *(Erasable Programmable Read Only Memory)* a special type of **PROM** whose programming can be erased by exposure to ultraviolet light and then reprogrammed.

***Ethernet*** a high speed (10Mb/sec) communications protocol and cable standard for computer networks.

***Fast Ethernet*** see **100BASE-T**.

***FIFO*** *(First-In-First-Out)* a data storage technique in which the first item stored in memory is also the first item on the stack of items for retrieval. Also a piece of hardware that stores data in such a manner.

***Flash memory*** a nonvolatile, random access, and rewritable solid-state storage technology that is ideal for field-upgradable code storage. Flash memory is electrically erased and programmed in-circuit.

***floating point*** method to represent numbers using the significant digits (mantissa) multiplied by the base of the number raised to the appropriate power (exponent). Values expressed in floating point form are similar in structure to number expressed in “scientific notation.”

***FPU*** *(Floating Point Unit)* a floating point co-processor.

***GPS receiver*** a radio receiver that locks onto the GPS (Global Positioning System) satellites in orbit around the earth. Using a GPS receiver, you can pinpoint your exact location anywhere on earth and use the GPS satellite’s onboard atomic clock as a time reference.

***IAck\*** *(Interrupt Acknowledge)* a VMEbus signal used by a Master to indicate that an interrupt was received.

***IBM*** manufacturer of the Selectric typewriter and inventor of the 80-column punched card. Collaborated with Motorola and Apple Computer in 1991 to invent the **PowerPC** based on IBM’s RISC design called POWER (**P**erformance **O**ptimization **W**ith **E**nhanced **R**ISC).

***interrupter*** a circuit that sources interrupts, usually at the behest of peripherals. An interrupter must drive an interrupt line and provide a vector number during an interrupt acknowledge cycle. In VMEbus devices, it may cease driving the interrupt line upon the interrupt being acknowledged (**ROAK**) or wait until a register access to the peripheral explicitly removes the request (**RORA**).

***interrupt handler*** a circuit (usually in conjunction with a CPU) that acknowledges and handles interrupts.

***I/O*** *(Input/Output)*

***ISP*** *(In-System Programmable logic)* a high density programmable logic device that can be programmed while the device is in the circuit. ISP logic can be upgraded easily in the field using a standard PC and a simple adapter cable.

***JEDEC*** *(Joint Electronic Device Engineering Council)* a body that sets standards for chip packages and pinouts.

***L1 cache*** a type of **cache memory** that is most closely coupled to the CPU core. It is built into the processor chip and is typically smaller and faster than **L2 cache**.

***L2 cache*** a type of **cache memory** that is external to the processor chip inbetween the CPU core and main memory. It is typically larger and slower than **L1 cache**.

***LED*** *(Light Emitting Diode)* a diode that emits light when forward biased, commonly used for displays and indicators.

***Little-Endian*** see **Endian**.

***longword*** a unit of data 32 bits in length.

***mailbox*** mechanism to allow any CPU or other Master to interrupt any other CPU of its choice.

***Master*** a device that initiates and controls the transfer of addresses and data across a bus. The opposite of Slave.

***mem protect*** *(Memory protect)* a bit that can be set or cleared in the Mode register. It usually is set to disable write accesses to the board and cleared to enable them, but its meaning can be changed via PALs or **ISP**s.

***MMU*** *(Memory Management Unit)* a circuit that provides address translation and access control services for a CPU.

***µs*** *(microsecond)* one millionth ($10^{-6}$) of a second.

***ms*** *(millisecond)* one thousandth ($10^{-3}$) of a second.

***MSB*** *(Most Significant Bit).*

***nibble*** a unit of data four bits in length. Sometimes spelled “nybble.”

***ns*** *(nanosecond)* one billionth ($10^{-9}$) of a second.

***NVRAM*** *(Non-Volatile RAM)* RAM that retains its data even without external power.

***object code*** output from a compiler or assembler that is in machine language but still must be linked to other object code to form an executable program.

***P1*** the mandatory 96-pin VMEbus connector. It carries all the signals to allow transfers up to A24 and D16. On a 3U board, it is the only connector.

***P2*** the secondary 96-pin VMEbus connector on 6U of 9U boards. Thirty two of its pins carry the signals necessary to allow A32 and D32 transfers. The other 64 pins are user definable.

***page*** the smallest unit of memory which is mapped by the **MMU**.

***P cable*** a 68-pin, high density connector SCSI cable with 50 mil (1.27mm) pin spacing. The P cable, which is defined in the SCSI-3 specification, comes in two varieties: external (MiniD68M connector) and internal (high density ribbon cable).

***PCI*** *(Peripheral Connect Interface)* an electrical specification describing a 32-bit wide multiplexed data/address bus, which is commonly used to connect peripheral chips to a processor through a **bridge** chip.

***PMC*** *(PCI Mezzanine Card)* a type add-on mezzanine I/O card that plugs into a VMEbus Single Board Computer’s PCI bus.

***PowerPC*** a microprocessor or architecture based on Motorola/IBM’s 32-bit, RISC design CPU core.

***PROM*** *(Programmable Read-Only Memory)* a memory storage media that can be programmed using electrical pulses. Once programmed, the PROM is read-only but does not need power or refresh to maintain the stored data.

***RAM*** *(Random Access Memory)* high speed, randomly accessible memory that can be easily read and written to by the processor.

***requester*** a circuit that requests Mastership of a bus.

***read-modify-write*** see **RMW.**

***RMA*** *(Return Merchandise Authorization)* a number assigned by Synergy for returning defective products.

***RMW*** *(Read-Modify-Write)* a read memory access followed by a write access performed in such a way that no other access is allowed to the location between the read and write.

***ROAK*** *(Release On AcKnowledge)* a type of VMEbus Interrupter module that deasserts its Interrupt Request to the VMEbus during reception of a valid IACK cycle for its interrupt level.

***ROR*** *(Release On Request)* a requester strategy that once granted the bus asserts continued Mastership of the bus even if not currently needed, until another requestor requests the bus. Opposite of RWD.

***RORA*** *(Release On Register Access)* a type of VMEbus Interrupter module that deasserts its Interrupt Request to the VMEbus during reception of a VME slave access cycle to one of its (vendor-specific) control registers.

***round robin*** a bus sharing method that engages each device or process in a group at its turn in a fixed cycle.

***RS-232*** an industry standard for serial communications using ±12V signals at up to 19.2 kb/sec for distances up to 50 ft.

***RWD*** *(Release When Done)* a requester strategy that once granted the bus asserts Mastership only as long as actually needed. Opposite of ROR.

***SBC*** *(Single Board Computer)* a printed circuit board containing microprocessor and support devices that provide **CPU**, ROM, RAM and peripheral interfaces.

***SCSI*** *(Small Computer Systems Interface)* an industry standard parallel interface bus that provides host computers with device independence of add-on peripherals such as disk drives, tape drives, CD-ROM drives, etc. The standard began as an 8-bit parallel data interface with a max. transfer rate of 5 MB/S (SCSI-1). The next SCSI standard, SCSI-2 (1994), doubled the 8-bit bus' transfer rate to 10 MB/S. The SCSI Trade Organization (STA) has since categorized higher performing, 16-bit SCSI types such as Wide Ultra SCSI and Wide Ultra2 SCSI. The T10 standards committee expanded the scope of the SCSI interface with SCSI-3 which is not a standard but a collection of spec documents describing additional connector and cabling options, protocol extensions, and transmission schemes (high performance serial and fiber data channel).

***SDRAM*** *(Synchronous Dynamic Random Access Memory)* a type of DRAM that operates in step with the CPU clock which allows the processor to perform more instructions over a given time.

***SIMD*** *(Single Instruction Multiple Data)* a processor performance enhancement that speeds multimedia applications by letting one microinstruction operate at the same time on multiple data items.

***single-ended*** a method of signaling in which one wire is used per signal, referenced to a common Ground signal. This is the most cost efficient signaling method for short cable runs. See **differential**.

***Slave*** a device connected to a bus that responds to commands from a Master.

***SMI*** *(System Management Interrupt)* for PowerPC, an asynchronous, maskable exception that is signaled to the processor by assertion of the SMI\ signal. On Synergy PowerPC SBCs, this interrupt is asserted via the front panel SMI switch.

***spurious interrupt*** an interrupt whose acknowledge cycle received no response. Usually caused by late acknowledgement of periodic interrupters such as timers. But may be caused by interrupt request that was aborted before being acknowledged.

***SRAM*** *(Static Random Access Memory)* a memory storage media that needs no refresh cycle. SRAM is faster and of lower density than DRAM.

***supervisor*** *(supervisor mode)* a Motorola processor execution mode in which the CPU enjoys all its privileges.

***SysClk \*** *(System Clock)* a signal driven by the system controller to all boards of a Multibus or VMEbus system.

***SysFail \*** *(System Failure)* a signal that can be driven by any board of a VMEbus system. Traditionally used to indicate a failure to one or more boards or devices on a bus.

***SysRes \*** *(System Reset)* a signal driven by the system controller to reset all the cards on the system bus.

***system controller*** on VMEbus, a group of circuits on the #1 slot VMEbus board that prioritize the bus-requests, provide a system clock, and provide system timeouts.

***terminal*** a keyboard and display monitor (**CRT**) attached to a computer to allow communications between the user and a computer.

***UART*** *(Universal Asynchronous Receiver/Transmitter)* a device able to translate between parallel and asynchronous serial communications signals for transmission and reception between a parallel processor bus and a serial communications port.

***VMEbus*** *(Versa Module Eurocard bus)* a microcomputer architecture whose physical and electrical characteristics are defined in the IEC 821 and IEEE 1014-1987 specifications. The VMEbus supports separate address and data lines of up to 32 bits each. This bus uses a backplane in which VMEbus modules are interconnected using DIN-41612 connectors.

***watchdog*** an on-board timer that can automatically reset the board if not accessed on a regular basis. Used to reset the board in response to a software loop and/or malfunction or a CPU halt.

***window size*** the range of contiguous addresses that the board responds to is called the window. The number of addresses in the window is called the window size. The board will respond to addresses from base to base+window size.

***word*** typically, a unit of data 16 bits in length. In the PowerPC environment, however, a word is 32 bits while a half-word is 16 bits and a double-word is 64 bits.

***WWV*** call letters for the National Bureau of Standards radio station in Ft. Collins, Colorado. WWV broadcasts technical services including timing signals, audio frequencies, and radio-propagation disturbance warnings at the 2.5, 5, 10, 15, and 25 MHz carrier bands. Canada provides similar services on CHU.

# Index