Technical Reference Manual for PP 110/01x CompactPCI® Pentium® III-M Single Board Computer

Manual Order Code 560 0009 Rev 04 June 2003

Concurrent Technologies Inc
3840 Packard Road
Suite 130
Ann Arbor, MI 48108
USA
Tel: (734) 971 6309
Fax: (734) 971 6350

Concurrent Technologies Plc
4 Gilberd Court
Newcomen Way
Colchester, Essex CO4 9WN
United Kingdom
Tel: (+44) 1206 752626
Fax: (+44) 1206 751116

E-mail: info@gocct.com

http://www.gocct.com

NOTES

Information furnished by Concurrent Technologies is believed to be accurate and reliable. However, Concurrent Technologies assumes no responsibility for any errors contained in this document and makes no commitment to update or to keep current the information contained in this document. Concurrent Technologies reserves the right to change specifications at any time without notice.

Concurrent Technologies assumes no responsibility either for the use of this document or for any infringements of the patent or other rights of third parties which may result from its use. In particular, no license is either granted or implied under any patent or patent rights belonging to Concurrent Technologies.

CONVENTIONS

Throughout this manual the following conventions will apply:
*# or * or ¯ over a name represents an active low signal. e.g. INIT* or $\overline{INIT}$ or INIT#*
h denotes a hexadecimal number. e.g. FF45h
byte represents 8-bits
word represents 16-bits
dword represents 32-bits

GLOSSARY OF TERMS

BIOS · · · · · Basic Input Output System
BIST · · · · · Built In Self Test
BMC · · · · · Baseboard Management Controller
BSB· · · · · · Back Side Bus
CCT· · · · · · Concurrent Technologies
CPCI · · · · · CompactPCI
CPU · · · · · Central Processing Unit
CRT· · · · · · Cathode Ray Tube
DIB · · · · · · Dual Independent Bus
DIMM · · · · Dual Inline Memory Module
DFP· · · · · · Digital Flat Panel
DMA · · · · · Direct Memory Access
ECC · · · · · Error Checking and Correcting
ECP· · · · · · Extended Capabilities Port
EIDE · · · · · Enhanced Integrated Drive Electronics
EPP· · · · · · Enhanced Parallel Port
EPROM· · · Electrically Programmable Read Only Memory
FSB· · · · · · Front Side Bus
ICMB· · · · · Intelligent Chassis Management Bus
IPM · · · · · · Intelligent Platform Management
IPMB· · · · · Intelligent Platform Management Bus
IPMI· · · · · · Intelligent Platform Management Interface
ISA · · · · · · Industry Standard Architecture
LFM· · · · · · Linear Feet per Minute
LPC· · · · · · Low Pin Count
LSB· · · · · · Least Significant Bit
MSB · · · · · Most Significant Bit
NMI · · · · · · Non Maskable Interrupt
PCI · · · · · · Peripheral Component Interconnect
PMC · · · · · PCI Mezzanine Card
POST · · · · Power-on Self Test
RFU· · · · · · Reserved for Future Use
RTC· · · · · · Real Time Clock
SCC · · · · · Serial Communications Controller
SCSI · · · · · Small Computer Systems Interface
SDR · · · · · Sensor Data Record
SDRAM· · · Synchronous Dynamic Random Access Memory
SEL · · · · · · System Event Log
SMC · · · · · Slave Management Controller
SMIC· · · · · System Management Interface Controller
SODIMM· · Small Outline Dual Inline Memory Module
UART · · · · Universal Asynchronous Receiver Transmitter
USB· · · · · Universal Serial Bus

NOTATIONAL CONVENTIONS

NOTE Notes provide general additional information.

WARNING Warnings provide indication of board malfunction if they are not observed.

CAUTION Cautions provide indications of board or system damage if they are not observed.

Revision	Revision History	Date
01	Preliminary Release	June 2002
02	First Full Release	July 2002
03	Changes to ISA Registers, added sections on RMI and CPCI Bridge programming, various minor corrections	April 2003
04	Updated for Rev B board	June 2003

Table of Contents

Table of Figures

Table of Tables

This page has been left intentionally blank

1.1 General

This manual is a guide and reference handbook for engineers and system integrators who wish to use the Concurrent Technologies' PP 110/01x ultra high-performance Pentium III-M single board computer. The board has been designed for high-speed multiprocessing applications using a PC-AT™ architecture operating in a CompactPCI Bus environment.

The PP 110/01x board is available in several different variants which differ by the processor speed, the amount of fitted SDRAM, the CompactPCI signaling voltage and the rear Ethernet connector configuration. Currently the board is available with either an 800MHz or 1.2GHz Pentium III-M processor designated by PP 110/010 and PP 110/012 respectively. The other configuration options are specified by a two-digit suffix to the board name; refer to the product data sheet for further details. Further details of other board options are given in Section 1.3. References to the board in this document will use the name PP 110/01x unless they apply only to a specific variant, in which case the full name will be used.

The information contained in this manual has been written to provide users with all the information necessary to configure, install and use the PP 110/01x as part of a system. It assumes that the user is familiar with the CompactPCI bus and PC-AT bus architectures and features.

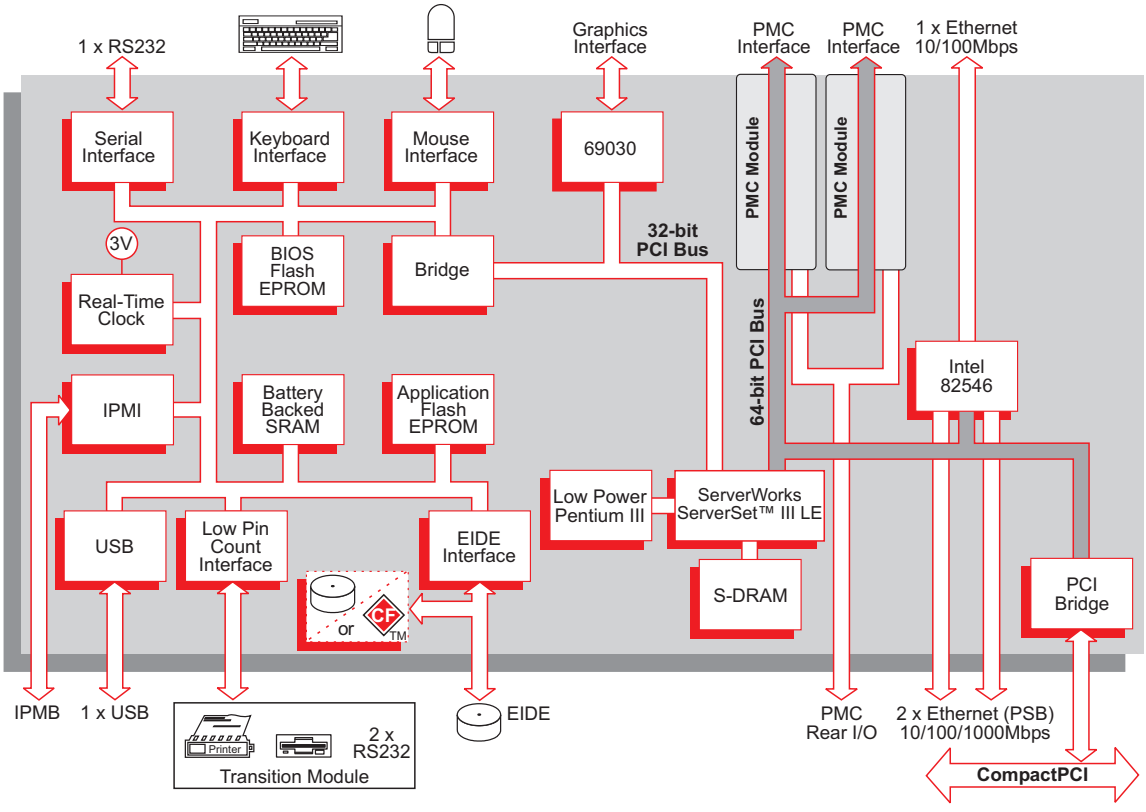


Figure 1-1 Overview

1.2 The PP 110/01x - Main Features

The PP 110/01x is a member of the Concurrent Technologies range of single-board computers for the CompactPCI bus architecture. It has been designed as a powerful single board computer based upon the Pentium III-M processor, the 82546EB dual channel Gigabit Ethernet controller, and the 69030 Graphics Controller. It also provides two IEEE 1386.1 PMC interfaces, optional on-board mass storage, and interfaces for standard PC-AT based peripherals.

1.2.1 Central Processor

The central processor used on this board is an ultra high performance low power Intel® Pentium III-M 32-bit microprocessor, operating internally at 800MHz or 1.2GHz. The processor supports the Dual Independent Bus (DIB) architecture with the backside bus connected to the on die Level 2 cache and the frontside bus connected to the memory controller at 133MHz. The processor is capable of addressing 4 Gbytes of physical memory all of which is cacheable, and 64 Terabytes of virtual memory. The Pentium III-M is upwardly code-compatible with the other members of the x86 family of microprocessors.

The processor has an in-built floating point coprocessor for compatibility with 486 and 386/387 designs.

The processor features Data Prefetch Logic that speculatively fetches data to the Level 2 cache before a Level 1 cache request occurs. This reduces latency resulting in improved performance.

1.2.2 Cache Memories

The Level 1 and Level 2 caches are both implemented on the processor die for maximum performance. The Level 1 cache is 32 Kbytes in size and the Level 2 cache is 512 Kbytes.

The Level 1 cache is organized as 4-way set associative with a 32-byte line size. It is split into a 16 Kbyte instruction cache and a 16 Kbyte write-back data cache.

The Level 2 cache is organized as 8-way set associative with a 32-byte line size. It operates at the core frequency and is based on Intel's Advanced Transfer Cache architecture. The Level 2 cache data is ECC protected.

1.2.3 Chipset

The PP 110/01x uses the ServerWorks ServerSet™ III LE chipset. This is comprised of the CNB30LE North Bridge and the CSB5 South Bridge.

The CNB30LE interfaces to the CPU host bus. It provides an SDRAM memory controller and two PCI bus bridges. It supports concurrent CPU and PCI bus operations. Pentium III burst and pipelining modes are supported to achieve a transfer rate of up to 425 Mbytes/s from SDRAM.

The CSB5 South Bridge provides a variety of peripheral functions including EIDE controllers, USB controller, LPC (Low Pin Count) Bus bridge, interrupt controller and other legacy PC-AT architectural functions. It is connected to the CNB30LE primary PCI bus.

The LPC Bus is used to connect to the on-board PC87417 Super I/O Controller and the PC87391 Super I/O controller on the AD PP5/001 Transition Module. These devices implement the floppy disk controller, serial ports, parallel port, keyboard and mouse controller and the real-time clock.

1.2.4 SDRAM

The on-board SDRAM operates at 133MHz and features ECC data protection. The board contains up to 1 Gbyte of soldered-on SDRAM. A 144-pin SODIMM socket is provided for memory expansion. This accepts a standard PC133 SDRAM module having a capacity up to 512 Mbytes. Hence a maximum of 1.5 Gbytes of SDRAM may be fitted to the board.

1.2.5 PCI Busses

There are two on-board PCI busses supported by the CNB30LE North Bridge. The secondary PCI bus is 64-bits wide and provides a high performance, up to 264 Mbytes/s, connection between the CNB30LE controller, Ethernet controller, PMC sites and the CompactPCI bridge. The Primary PCI bus is 32-bits wide and provides a lower performance, up to 132 Mbytes/s, connection between the CNB30LE, PC-AT peripherals and graphics controller.

1.2.6 EPROM

The board contains a socketed 512 Kbyte Flash EPROM, for the BIOS code and fixed data. It also contains two 16 Mbyte Flash EPROMs for Application Program storage. The EPROMs have 8-bit data paths and are connected to the CSB5 X-Bus interface.

1.2.7 Static RAM

The board contains one low power 4Mbit static RAM. This device provides 512 Kbytes of non-volatile data storage. The contents are maintained by the on-board battery when the board is not powered. Battery life is approximately 5 years at room temperature. The static RAM is connected to the CSB5 X-Bus interface.

1.2.8 EIDE Controllers

The PP 110/01x has two EIDE/Ultra ATA100 interfaces. One EIDE interface is available via the J5 connector, the other via an on-board connector for use by the optional on-board disk drive or CompactFlash™ module.

1.2.9 USB

One USB 1.0 channel is provided by the board. The USB interface is available via the J5 connector.

1.2.10 PMC Interfaces

Two PMC interfaces which support single width 64 or 32-bit PMC modules complying with the IEEE 1386.1 standard are provided. 5V or 3.3V PCI signaling environments are jumper selectable.

The PMC interfaces will also accept dual function PMC modules and Processor PMC modules. The latter will operate only in non-Monarch modes.

1.2.11 Ethernet Controllers

An Intel 82546EB dual channel Gigabit Ethernet controller is used to provide high performance PCI to Ethernet interfaces. Both channels support 10, 100 and 1000Mbits/s operation.

Channel 0 can be routed independently, under software control, to either a front panel RJ45 connector or the CompactPCI J3 connector. Channel 1 is available only on the J3 connector.

The board can either support PICMG® 2.16 backplane networking or rear panel Ethernet. This is specified by an ordering option. A suitable Transition Module, such as the AD PP5/001, is required for rear panel Ethernet.

1.2.12 Graphics Controller

The Asiliant Technologies 69030 is used to provide a high performance graphics accelerator with 4 Mbytes of integrated memory. A CRT interface is provided via a shared 26-way high density connector on the front panel. A splitter cable (part number CB 26D/124) is required to access the CRT interface.

1.2.13 CompactPCI Interface

The PP 110/01x is a CompactPCI compatible System Controller and Peripheral Board. It may also act as a Satellite board in any backplane slot. The board uses a 64-bit interface implemented with a HiNT® Corporation (HINT) HB6 PCI to PCI bridge. As a System Controller it supports PICMG 2.1 compliant Hot Swap Peripheral Boards and as a Peripheral Board it is PICMG 2.1 Hot Swap compliant. As a Satellite board it may be Hot Swapped under the control of an on-board microcontroller.

1.2.14 System Management

The PP 110/01x supports PICMG 2.9 system management. The IPMI SMIC interface is implemented and both IPMB 0 and IPMB 1 communication channels are provided. An on-board microcontroller provides the requisite IPMI functionality. The microcontroller can be configured to provide Baseboard Manager Controller (BMC) functionality.

1.2.15 Serial Communications

The PP 110/01x provides one RS232 serial data communication channel. This is implemented by the Super I/O controller and is available at a shared 26-way front panel connector. A splitter cable (part number CB 26D/124) is required to access the serial interface.

The baud rate clock is generated internally by the Super I/O Controller.

NOTE Two more serial channels are provided on the AD PP5/001-00.

1.2.16 Keyboard & Mouse

PS/2™ type keyboard and mouse interfaces are available via a shared 26-way high density front panel connector. A splitter cable (part number CB 26D/124) is required to access these interfaces. See Section 6.2 for more information about these ports.

1.2.17 Real Time Clock (RTC)

A battery backed RTC device provides PC-AT clock, calendar and configuration RAM functions. The RTC and BIOS are year 2000 compliant.

1.3 AD PP5/001 Peripheral Functions

The AD PP5/001 Transition Module contains a Super I/O controller. This device provides the following interfaces:

- Floppy disk interface for up to two floppy disk drives
- Parallel port interface
- Two RS232 serial interfaces
- Two general purpose inputs
- Two general purpose outputs
- External Reset input
- Fan sensor input

The AD PP5/001 also contains either a CompactFlash site or an on-board 1.8" hard disk drive. Only one of these options may be fitted at a time. Refer to the AD PP5/001 datasheet for ordering information.

1.4 Additional Board Options

The PP 110/01x board may be ordered with one of several different configuration options, namely,

Soldered-on SDRAM capacity

- 512 Mbytes
- 1 Gbyte

Ethernet (on J3 connector)

- Configured for PICMG 2.16 backplane networking
- Configured for rear panel Ethernet via connectors (on Transition Module)

CompactPCI Signaling Voltage

- 5V (33MHz only)
- 3.3V (33MHz or 66MHz automatic selection)

Two on-board mass storage options are available, namely:

- A 2.5” EIDE hard disk drive of at least 10GB capacity
- A dual CompactFlash carrier that supports the IBM® Microdrive™

Only one of these options may be fitted at a time. Refer to the PP 110/01x datasheet for ordering information.

NOTE On-board mass storage uses one PMC site area. Only one PMC module may be fitted when a mass storage option is fitted.

Hardware Installation

2.1 General

This chapter contains general information on unpacking and inspecting the PP 110/01x after shipment, and information on how to configure board options and install the board into a CompactPCI chassis.

CAUTION It is strongly advised that, when handling the PP 110/01x and its associated components, the user should at all times wear an earthing strap to prevent damage to the board as a result of electrostatic discharge.

The list below outlines the steps necessary to configure and install the board. Each entry in the list refers to a section in this chapter which will provide more details of that stage of the procedure. It is recommended that the board is configured in the sequence below.

1) Unpack the board - see Section 2.2.
2) Locate and familiarize yourself with the front panel indicators and controls - see Section 2.4.
3) Check that the board jumper settings match the required operating mode. See Section 2.3 for details of the default settings and where to find more information.
4) Fit the battery if necessary - see Section 2.7.
5) Fit any optional mass storage modules or DRAM - see Sections 2.5 and 2.6.
6) Fit the PMC modules(s) if required - see Section 2.8.
7) Configure the board for the correct CompactPCI operating mode and for any external sources for board or system resets - see Sections 2.9 and 2.10.
8) Check that the backplane V(I/O) voltage matches the board configuration - see Section 2.11.
9) Install the board, using Hot Swap or non-Hot Swap procedures - see Section 2.11.

2.2 Unpacking and Inspection

Immediately after the board is delivered to the user's premises the user should carry out a thorough inspection of the package for any damage caused by negligent handling in transit.

CAUTION If the packaging is badly damaged or water-stained the user must insist on the carrier's agent being present when the board is unpacked.

Once unpacked, the board should be inspected carefully for physical damage, loose components etc. In the event of the board arriving at the customer's premises in an obviously damaged condition Concurrent Technologies or its authorized agent should be notified immediately.

2.3 Default Jumper Settings

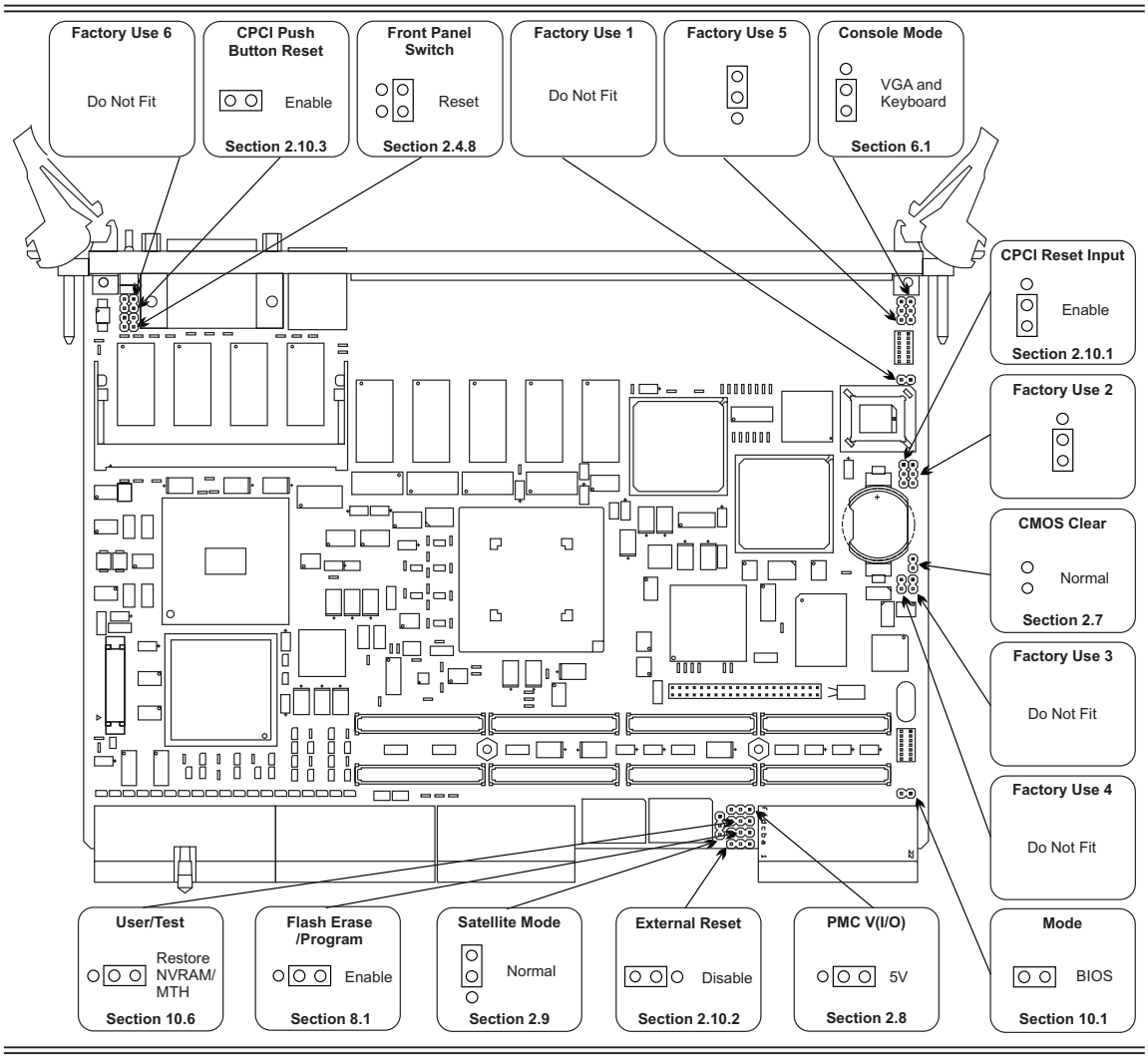


Figure 2-1 Default Jumper Settings

2.4 Front Panel Indicators and Controls

When installing or removing the board for the first time, or when checking it's operation, it can be very useful to note the behavior of the LEDs on the front panel. Figure 2-2 shows the location of the LEDs, and their purpose is outlined below.

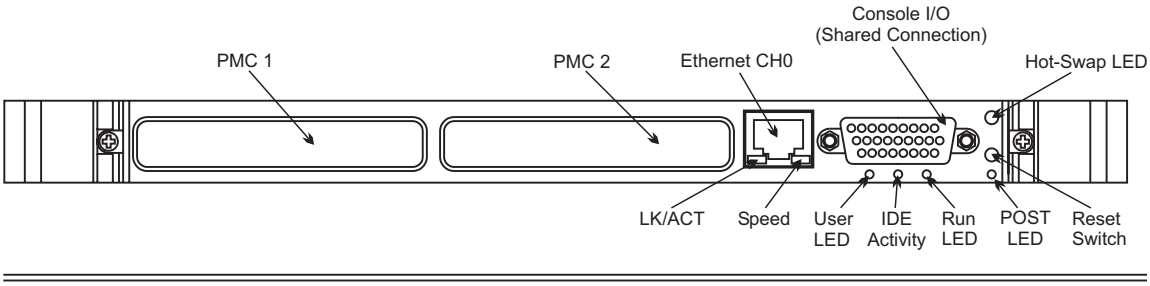


Figure 2-2 Front Panel Indicators and Controls

2.4.1 Run LED (R) Green

The run LED indicates that activity is occurring on the PCI bus. This allows the user to quickly assess how active the PCI bus is.

2.4.2 POST LED (P) Yellow

The POST LED is used to indicate that a power on self test has failed. This LED will also flash when outputting sound on the speaker.

2.4.3 Ethernet Speed LED (Speed) Yellow

This LED indicates the operating speed of the front panel Ethernet interface, as follows:

- Off = 10Mbits/s
- Steady On = 100Mbits/s

NOTE The LED stays Off if the Ethernet interface is routed to the rear panel.

2.4.4 Link/Activity LED (LK/ACT) Green

This LED lights when connection has been made on the Ethernet interface. It will flash to indicate link activity, and during periods of high Ethernet activity the LED may switch off for several seconds.

NOTE The LED stays Off if the Ethernet interface is routed to the rear panel.

2.4.5 User LED (U) Red

This LED is available for use by user software. It is manipulated using RMI registers (see Section 10.7 for details).

2.4.6 Hot-Swap LED (H) Blue

This LED is used when the board is hot-swapped, and lights to indicate when the board can be safely removed from the chassis.

2.4.7 EIDE Activity LED (I) Green

This LED lights when there is activity on the on-board (secondary) EIDE interface.

NOTE The AD PP5/001 Transition Module contains a LED to indicate activity on the off-board (primary) EIDE interface.

2.4.8 Switch (SW)

The reset or NMI function is selected by the setting of the Front Panel Reset and NMI Switch jumper shown in Figure 2-3.

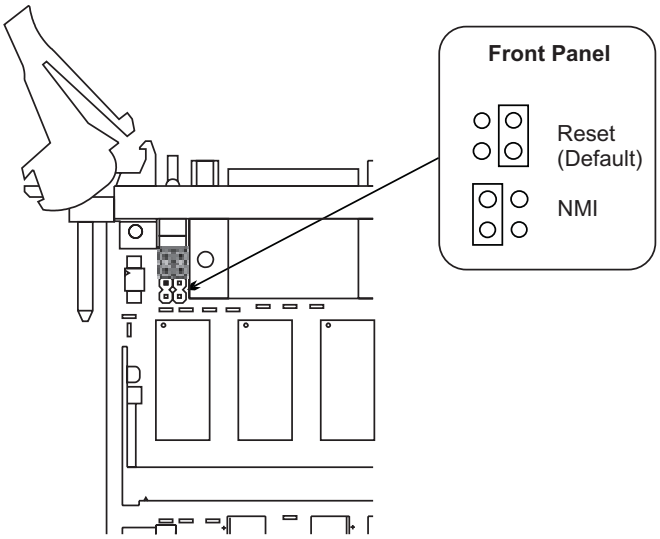


Figure 2-3 Front Panel Reset and NMI Switch Jumper

Selecting the Reset jumper position setting will cause the board to be reset when the front panel switch is operated. If the board is in the System Controller Slot, it will also assert RST# on the CompactPCI backplane and hence reset the other boards in the chassis. If the board is operating as a Peripheral or Satellite, it will respond to front panel resets but will not reset the other boards in the chassis.

Selecting the NMI jumper position setting configures the switch to generate NMI when operated. No reset is generated in this case.

2.5 Installation of On-Board Mass Storage

If an on-board mass storage option has been ordered, it will be necessary to install the option at this time.

The mass storage option plugs into connector P2 and is secured via screws and spacers using the four mounting holes as shown in Figure 2-4 below.

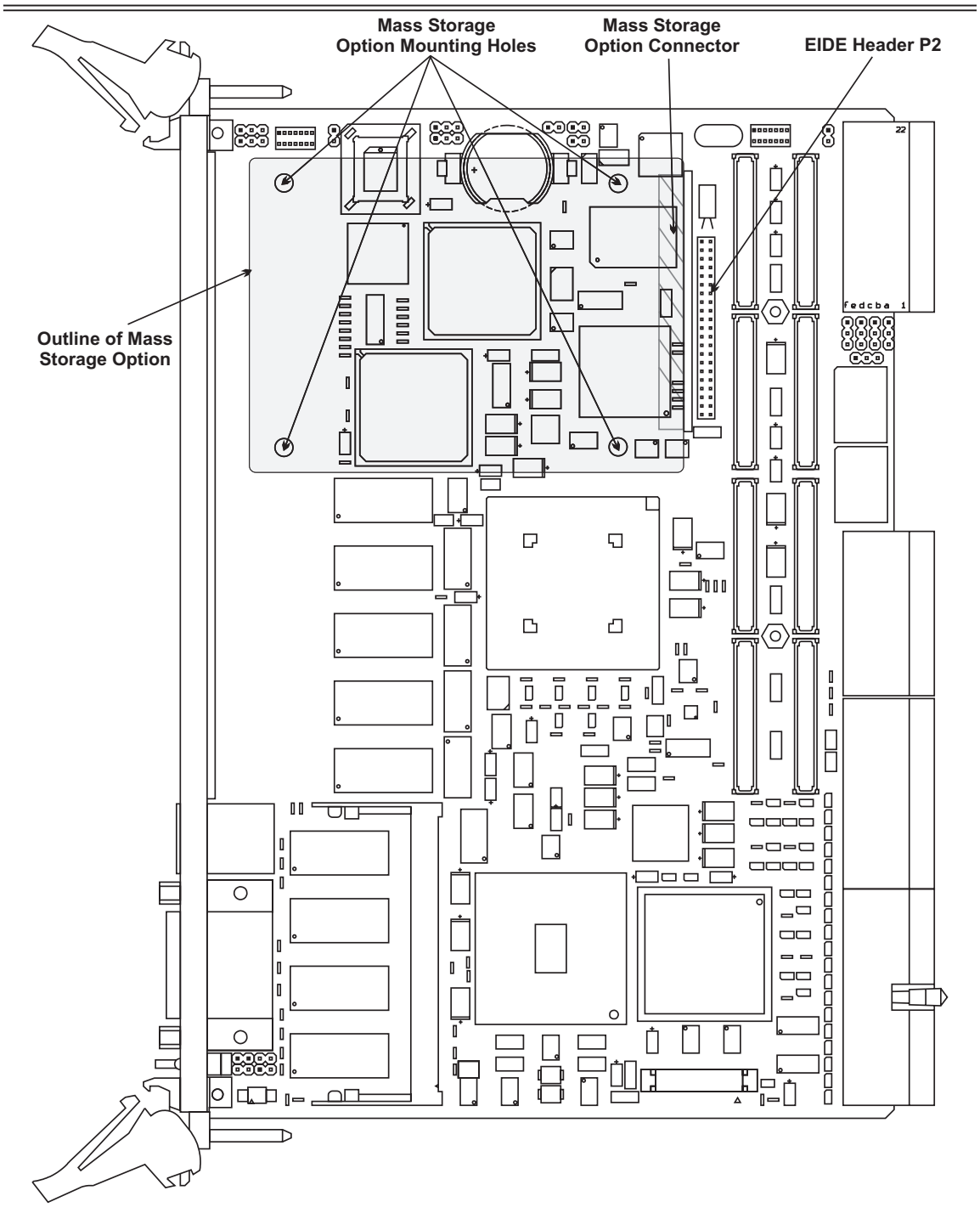


Figure 2-4 Mass Storage Connector and Fixing Holes

2.5.1 Hard Disk Storage Kit (AD CP1/DR1)

The option kit comprises:

- A 2.5” EIDE disk drive
- A ribbon cable assembly
- Four M3 x 10mm screws
- Four M3 x 5mm spacers

The ribbon cable assembly has a 50-way connector at one end and a 44-way connector at the other end. The 50-way connector plugs into the disk drive and the 44-way plugs into P2 on the PP 110/01x.

1) Plug the 50-way connector into the disk drive as shown in Figure 2-5 below, note the orientation.

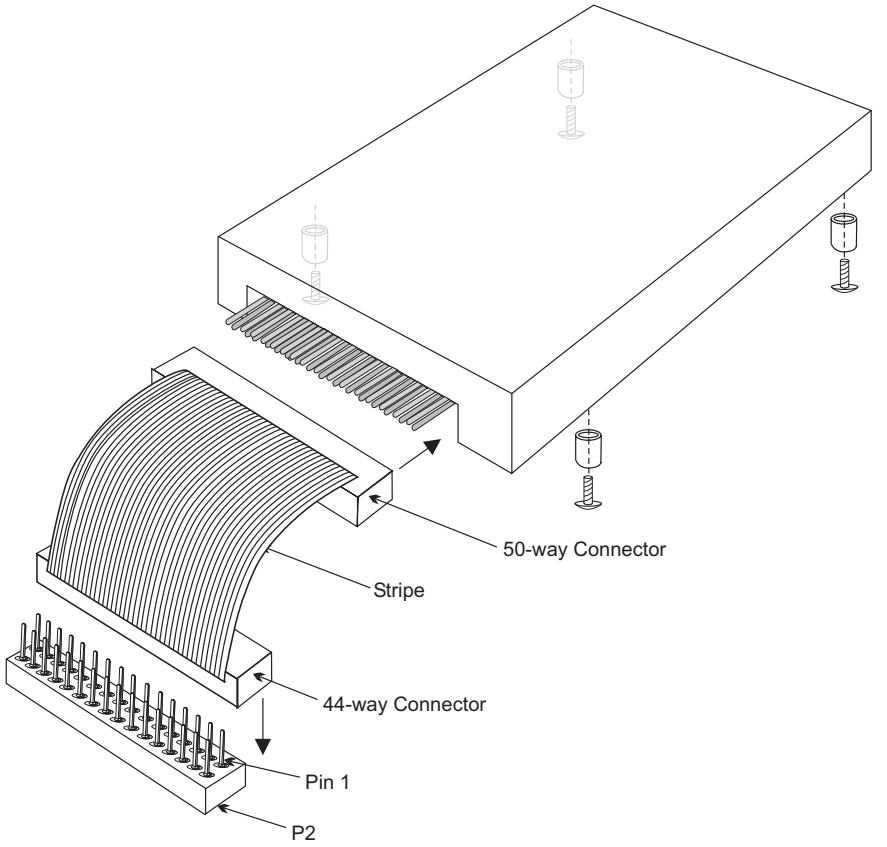


Figure 2-5 Disk Drive Cable Installation

2) Plug the 44-way connector into P2, note the orientation.

3) Fix the disk drive into position using the four screws and spacers provided. Do not over tighten the screws.

NOTE If the board is likely to be subjected to mechanical vibration a suitable thread lock compound applied to the screws should be considered.

2.5.2 CompactFlash Storage Kit (AD 200/001)

The option kit comprises:

- A CompactFlash carrier module
- Four M3 panhead screws

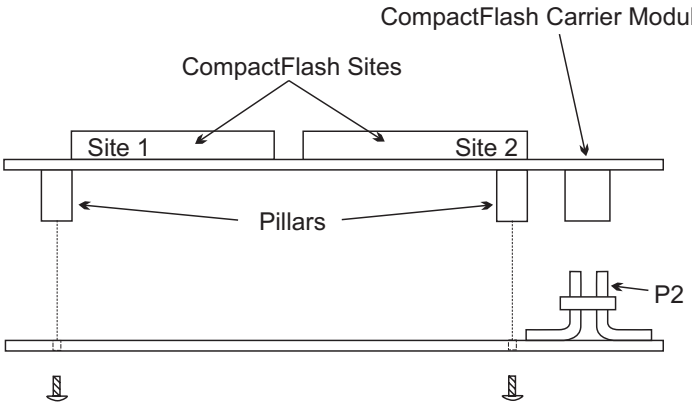


Figure 2-6 CompactFlash Carrier Module Installation

1) The M3 panhead screws may be loosely screwed into the end of the pillars, if so unscrew them.

NOTE Do not unscrew the countersunk screws attaching the pillars to the circuit board.

2) Position the connector of the CompactFlash carrier module over P2. Ensure that the pins are correctly aligned, then press the module down on to the pins of P2 until the four pillars are touching the PP 110/01x circuit board.

3) Fix the module into position using the four panhead screws referred to earlier. Do not over tighten the screws.

NOTE If the board is likely to be subjected to mechanical vibration a suitable thread lock compound applied to the screws should be considered.

The CompactFlash sites are labeled CompactFlash 1 and CompactFlash 2.

If a single CompactFlash card is fitted, it should always go into site 1. Site 2 should be used only when two CompactFlash cards are fitted.

The CompactFlash card(s) may be retained in position by fitting short M3 screws and spacers into the holes near the long edge of the carrier. This will protect against accidental removal due to vibration or deliberate but unauthorized removal.

NOTE If more than one CompactFlash module is fitted, the module in the CompactFlash 2 site must support operation as a Slave device.

The DIL switch on the AD 200/001 should be set as shown in Figure 2-7.

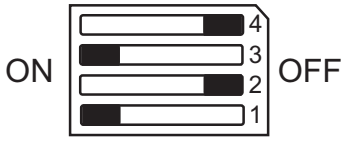


Figure 2-7 AD 200/001 DIL Switch Settings

2.6 Adding or Replacing DRAM Modules

The PP 110/01x accepts standard 144-pin SODIMM modules fitted with 3.3V PC133 DRAM. One socket is provided and will accommodate SODIMMs of 256 Mbytes or 512 Mbytes capacities.

NOTE SODIMMs using 256Mbit DRAMs with 8K refresh are required.

Figure 2-8 shows shows the way in which SODIMMs are fitted or removed. No other changes are necessary when a SODIMM is added or removed.

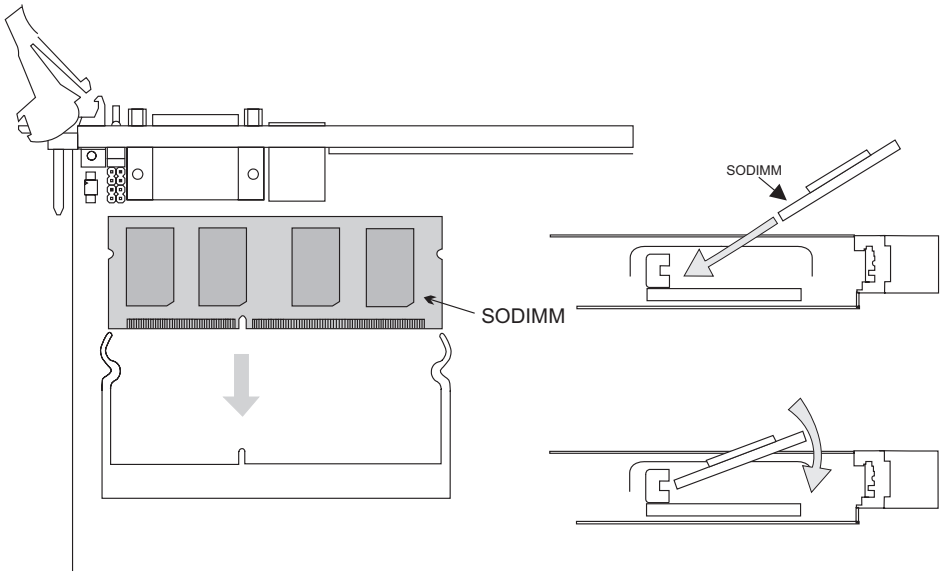


Figure 2-8 DRAM Module Replacement

2.7 Battery Installation/Replacement

The on-board Real-Time Clock and CMOS memory used by the PC BIOS firmware are powered by a 3.3V Lithium battery when the board is powered OFF. This battery also powers the static RAM device. It is advisable, though not essential, for the battery to be fitted prior to using the board. Figure 2-9 shows how to do this. One battery is supplied with the board, but it is not normally fitted.

If the board is operated without a battery, the User Selectable NVRAM Defaults feature can be used to override the factory default NVRAM settings. See Section 10.6 for further details.

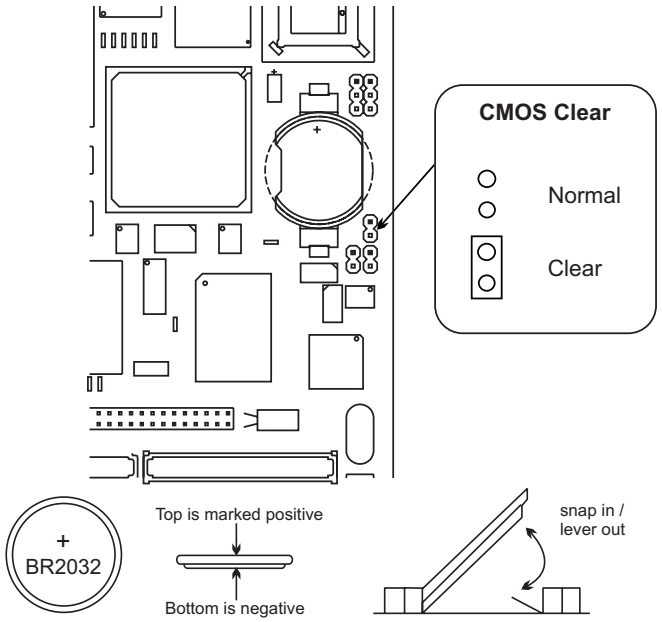


Figure 2-9 Battery Fitting and CMOS Clear Jumper

The battery should be replaced when the voltage falls below 2.8V. Depending on the way in which the board is operated and stored, battery life should be in excess of 5 years. The life expectancy will fall if the battery is subjected to long periods at temperatures of 45°C or above. It will also fall if the battery is fitted to a board that is stored in it's conductive bag even at room temperature.

CAUTION When replacing the battery, proper anti-static precautions must be observed.

WARNING Dispose of battery properly. DO NOT BURN. The date and time settings will need to be initialized if the battery is disconnected.

If the BIOS setup screens have been used to set up the board for an invalid configuration, or in other fault conditions, it may be useful to be able to reset the contents of the CMOS RAM and Real-Time Clock. In this case, the CMOS Clear Jumper can be used.

To clear the CMOS RAM to a known state, fit the CMOS Clear jumper and apply power. When the board is next powered down remove the jumper, otherwise CMOS RAM will again be reset.

2.8 Installing or Removing a PMC Module

Before installing a PMC module, check that the PP 110/01x board PMC V(I/O) voltage is configured to match the requirements of the PMC module. If two PMC modules are fitted, their V(I/O) requirements must be the same.

CAUTION If the PP 110/01x is not correctly configured to match the PMC module V(I/O) requirements, it may result in damage to the module or the PP 110/01x.

Setting the the correct PMC V(I/O) voltage on the PP 110/01x requires the positioning of a detachable polarizing key for each PMC site, and the setting of a board jumper. Figure 2-10 shows the location of the key for both 5V and 3.3V V(I/O) configurations, and how to fit the PMC module to the PP 110/01x board. Figure 2-11 shows the location and settings for the PMC V(I/O) jumper.

NOTE The positions of the polarizing keys must correspond with the setting of the PMC V(I/O) jumper.

NOTE PMC Site 1 does not have provision for a 3.3V polarizing key.

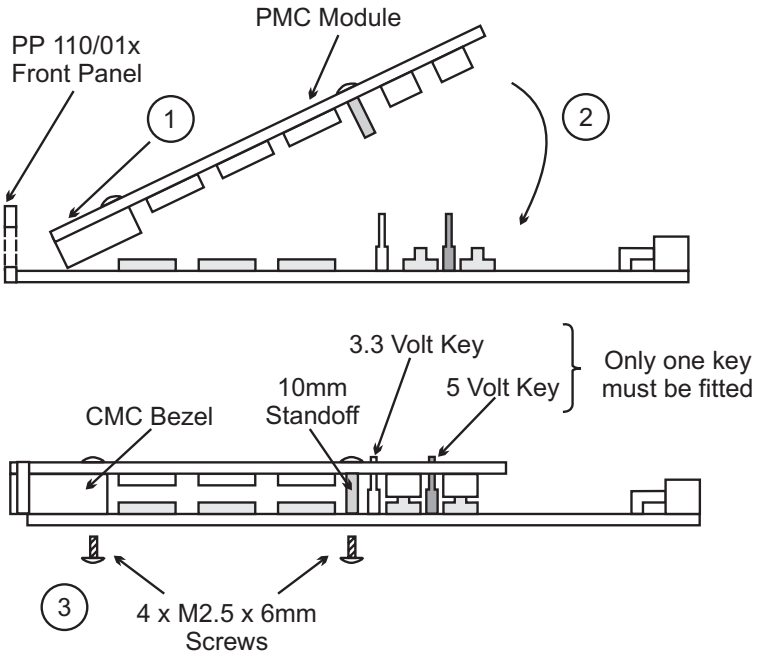


Figure 2-10 PMC Installation Diagram

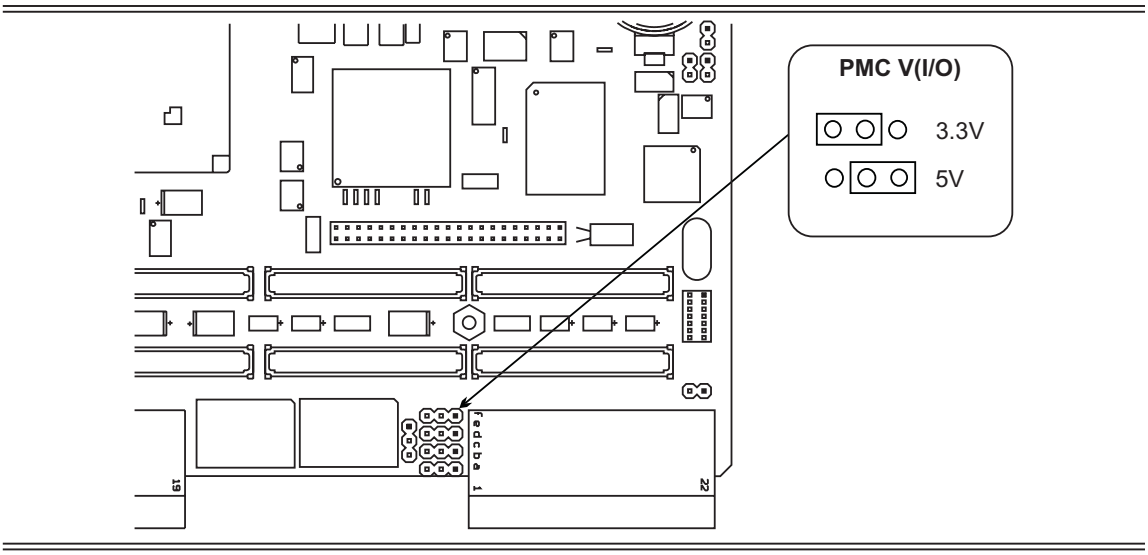


Figure 2-11 PMC V(I/O) Jumper

2.9 CompactPCI Operating Mode Selection

This is normally automatic and depends only on what type of slot the board is installed into, as detailed below:

Slot	Mode
System Controller	System Controller
Bussed Peripheral	Peripheral
Non-Bussed Peripheral	Satellite

A jumper is provided to force Satellite mode operation in any slot. The settings are shown in Figure 2-12. In Satellite mode the board cannot communicate on the CompactPCI bus.

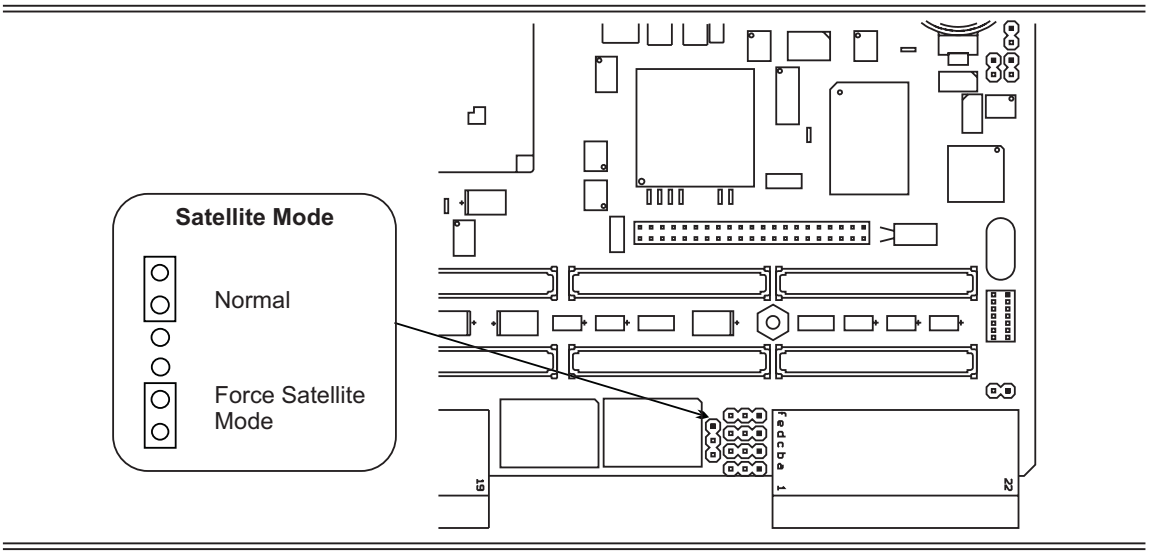


Figure 2-12 Satellite Mode Jumper

2.10 Reset Sources

In addition to the front panel switch described in Section 2.4.8, the board may be reset from several external sources, as described below. Table 2-1 outlines how board and system resets can be achieved using the available jumper options.

Mode	Board Level Reset Sources	System Reset Sources	Section
System Controller	Front Panel Switch CompactPCI Push Button Reset External Reset	Front Panel Switch CompactPCI Push Button Reset External Reset	2.4.8 2.10.3 2.10.2
Peripheral	Front Panel Switch External Reset (AD PP5) CompactPCI RST# signal (mandatory)	No option	2.4.8 2.10.2 2.10.1
Satellite	Front Panel Switch External Reset (AD PP5) CompactPCI RST# signal (optional)	No option	2.4.8 2.10.2 2.10.1

Table 2-1 Reset Configuration Options

NOTE Resets generated by hardware or software which cause a local PCI bus reset will also activate the CompactPCI backplane reset if the board is the System Controller.

2.10.1 CompactPCI Reset

The CompactPCI Reset signal is generated by the System Controller and is routed to all bussed peripheral slots.

If the board is in Peripheral mode it must respond to the CompactPCI Reset signal. However, if it is in Satellite mode, it may be preferable for it to ignore the Reset signal.

A jumper is provided to facilitate this choice. The settings are shown in Figure 2-13.

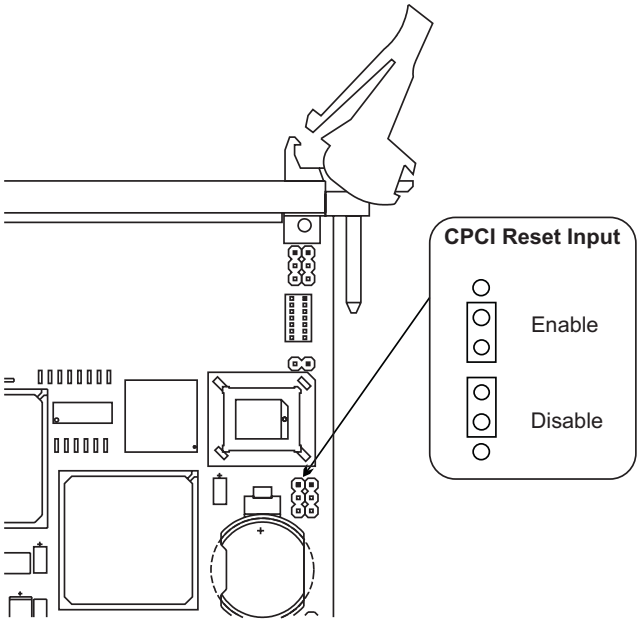


Figure 2-13 CompactPCI Reset Input Jumper

2.10.2 External Reset

When the PP 110/01x board is used with an AD PP5/001 rear transition module, a local (board-level) reset may be generated from a connector on the AD PP5/001 (see that board's Technical Reference Manual for details). The action of that connector input is controlled on the PP 110/01x by the External Reset jumper shown in Figure 2-14. This jumper has no function when the PP 110/01x is used without a rear transition module.

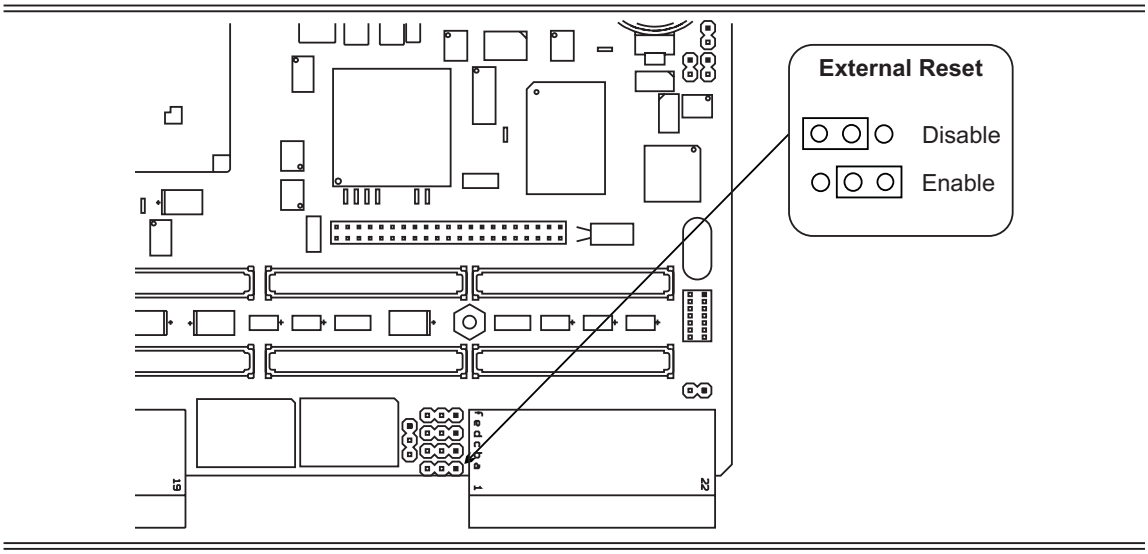


Figure 2-14 External Reset Jumper

NOTE Unlike the Push Button Reset, this facility is available when the board is installed in any slot.

2.10.3 CompactPCI Push Button Reset

The Push Button Reset signal available on the J2 connector (PRST#) will cause a board reset if the board is in the System Controller slot. This input can be driven from an open collector TTL output (or discrete transistor) or normally open switch/relay contacts. To initiate the reset pull this input to 0V. This input is filtered and protected from overshoots/undershoots so no external contact debouncing is required.

This signal is not wired to Peripheral or Satellite slots.

A jumper determines whether or not assertion of this signal will reset the board. The settings are shown in Figure 2-15.

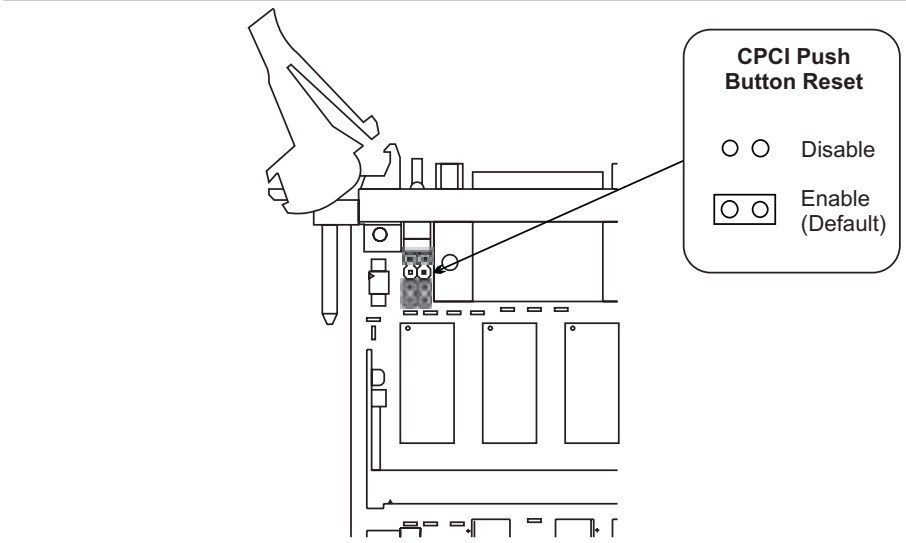


Figure 2-15 CompactPCI Push Button Reset Jumper

2.11 Installation and Power-up

Before the board is installed in a CompactPCI chassis, check the following points:

- The backplane V(I/O) configuration. The PP 110/01x board can be ordered pre-configured for either 5V or 3.3V V(I/O). Some CompactPCI backplanes are pre-wired for a particular voltage and others can be configured by the user.

CAUTION The board configuration must match the backplane configuration, otherwise one or both may be damaged.

WARNING V(I/O) must be wired on the backplane. If it is not wired the board will hang during POST.

- The Power Supply Unit current capabilities. The board draws current primarily from the +5V and 3.3V rails, and the details are provided in Section A.4.
- Front panel keys. PICMG 2.10 describes a method of keying CompactPCI board front panels to individual chassis slots. Keying pins for this purpose are supplied with the board but are not fitted. The user is referred to PICMG 2.10 for information on using these keys.

The board can be installed as a System Controller in the system slot or as a peripheral or satellite board in a peripheral slot. In satellite mode the CompactPCI interface is disabled and hence the board will not be recognized by the system controller. When acting as the System Controller, the board cannot be Hot Swapped.

2.11.1 Non Hot Swap Procedure - Installing

If your system requires the use of the EMC spring contact strips provided, fit strips into the slots on the long edges of the front panel.

The board is installed and powered up as follows:

1) Make sure that system power is turned OFF.
2) Slide the board into the designated slot, making sure that the board fits neatly into the runners.
3) Push the board into the card-cage until the J1 ... J5 connectors are firmly located. Use the injector/ejector handles for the final push.
4) Screw the ejector handle retaining bolts into the holes in the chassis.
5) Connect the I/O cables to the connectors on the board's front panel and fix in place with the connectors' retaining screws.
6) If using a Transition Module, install it at the rear of the backplane and connect the I/O cables.
7) Power-up the system. The following sequence of events should then occur:
   - The green "RUN" LED and the yellow "POST" LED on the front panel will light.
   - The yellow "POST" LED will switch OFF.

   If power-up does not follow the sequence described above this will indicate that the board is not operational.

NOTE This sequence of events assumes the PP 110/01x has Concurrent Technologies standard BIOS firmware and that the board is configured to the factory setting described in Section 2.3.

2.11.2 Non Hot Swap Procedure - Removing

To remove the board, shut down the application and operating system software before powering down the system, opening the ejector handles and extracting the board.

2.11.3 Hot Swap Procedure - Installing

The board is installed and powered up as follows:

1) Slide the board into the designated slot, making sure that the board fits neatly into the runners.
2) Push the board into the card-cage until the J1 ... J5 connectors are firmly located. Use the injector/ejector handles for the final push.
3) The following sequence of events should then occur:
   - The blue “Hot Swap” LED will flash once.
   - The green “RUN” LED and the yellow “POST” LED on the front panel will light.
   - The yellow “POST” LED will switch OFF.

   If power-up does not follow the sequence described above this will indicate that the board is not operational.
4) Screw the ejector handle retaining bolts into the holes in the chassis.
5) Connect the I/O cables to the connectors on the board’s front panel and fix in place with the connectors’ retaining screws.
6) If using a Transition Module, install it at the rear of the backplane and connect the I/O cables.

2.11.4 Hot Swap Procedure - Removing

To remove the board:

1) Open the lower ejector handle and wait for the blue “Hot Swap” LED to switch on. This may take a few seconds. Newer handles require a red button to be pressed in order to open the handle.
2) Open both ejector handles and remove the board.

Software Installation

In most cases, installing operating system software on the PP 110/01x board follows the same sequence as installing on a PC. However, there are some additional points to note. The sections below summarize the special actions required for a few common operating systems.

3.1 Starting up for the first time

Many operating systems running on the board will want to use the standard Real-Time Clock hardware. To maintain the date and time settings, and several other settings recorded by the PC BIOS, the battery must be fitted. When the board is first powered up, or at the first power-up after changing the battery, carry out the following steps to set up the board.

1) If required fit a battery as shown in Section 2.7.
2) Make sure that the Console Mode jumper is set to the correct state for the console device which will be used (VGA monitor and keyboard, or serial terminal). Most operating systems which install on the target hardware will require a monitor and keyboard during installation, even if they can subsequently be re-configured to use only a serial terminal. See Section 6.1 for details of how to configure the board for this option.
3) Connect any additional modules and peripherals especially any mass storage devices.
4) Connect the console device and power up the board. Wait for the PC BIOS to sign on and run its memory test.
5) When the test finishes, the BIOS may report a setup or date/time setting error. If this occurs, press the <F2> key as soon as possible after the error is reported, and carry out the following:

a) Set the time and date by using the cursor keys to move around the screen and reading the help information in the right-hand screen panel.
b) When the time and date have been set, move the cursor to any other field on the same screen, then press the <F4> key to exit.
c) Press the ‘y’ key to accept the changes and restart.

The BIOS will then completely restart and re-run its memory test. This time it should complete and begin bootloading. To proceed with software installation, check that all necessary mass storage devices are connected before continuing with one of the sequences below.

3.2 Bootloading from CD-ROM

Operating systems which install on the target hardware will generally install from CD-ROM, or may require both a CD-ROM and floppy disk. Bootloading from floppy disk requires no special steps other than to connect the drive using an appropriate cable. To bootload from CD-ROM, use the following procedure:

1) While the BIOS is running its memory test, press the <ESC> key.
2) Wait for the pop-up boot device menu to be displayed.
3) Select the CD-ROM drive using the cursor keys, then press the <Enter> key.

3.3 Installing Windows NT® 4.0

To install Windows NT from CD-ROM, set up the board initially using the steps outlined in Sections 3.1 and 3.2 above, ensuring that all the necessary drives are connected. Then follow the procedure below.

1) Obtain the Ethernet driver from the Intel web site, starting from the following address:

   http://www.intel.com

   Click “Support”, “resources for developers”, then follow the links for: “Networking Drivers”, “Adapters for Desktop and Server Systems”. Click on “Latest drivers” for the “Intel PRO/1000 MT Desktop RJ45”.

   Select the “Windows NT 4.0” option and look for the “Windows NT 4.0 Network Adapter Driver Set”. Download the driver PRONT4.EXE file and run it to extract the files. Read the “readme” file in this package to find out how to run the “makedisk” program. Run this program to create an NT driver disk (2 floppy disks are required for this).

2) Power up the system and insert the Windows NT CD.
3) Allow Windows to boot and follow the normal setup procedure up to the point where Windows prompts to know if “the computer is to participate on a network”.
4) Select “this computer will participate on a network” then click the “Next” button.
5) When the Network Adapter screen is displayed click the “Select From List” button.
6) On the Select Network Adapter screen click the “Have Disk” button.
7) Insert the first floppy disk containing the Intel PRO/1000 Gigabit Adapter driver and click the “OK” button.
8) When the Select OEM Options screen is displayed select the “Intel PRO/1000 Adapter” from the list then click the “OK” button.
9) When returned to the Network Adapter screen click the “Next” button.
10) Insert the second floppy disk containing the Intel PRO/1000 Gigabit Adapter Driver when prompted.
11) Continue with the installation of Windows NT in the normal way.

3.4 Installing Windows® 2000

To install Windows 2000 from CD-ROM, set up the board initially using the steps outlined in Sections 3.1 and 3.2 above, ensuring that all the necessary drives are connected. Then follow the procedure below.

1) Obtain the Ethernet driver from the Intel web site, starting from the following address:

   http://www.intel.com

   Click “Support”, “resources for developers”, then follow the links for: “Networking Drivers”, “Adapters for Desktop and Server Systems”. Click on the “Latest drivers” for the “Intel PRO/1000 MT Desktop RJ45”.

   Select the “Windows 2000” option followed by the “Windows 2000 and XP Network Adapter Base Driver”. Download the driver 1000DISK.EXE file and run it to extract the files. Read the “readme” file in this package to find out to run the “makedisk” program. Run this program to create a Windows 2000 driver disk.

2) Power up the system and insert the Windows 2000 CD.
3) Allow Windows to boot and follow the normal setup procedure up to the point where Windows restarts.
4) Logon, right-click “My Computer” and select “Properties”.
5) Select the “Hardware” tab and click the “Device Manager” button.
6) On the Device Manager tree view, double-click the first Ethernet device located in the “Other Devices” branch.
7) Click “Reinstall Driver” to start the Device Driver Wizard.
8) Click the “Next” button when the Welcome screen is displayed.
9) Choose the “Search” option and click the “Next” button.
10) Select the Floppy disk drive option on the Locate Driver screen and click the “Next” button.
11) When prompted, insert the floppy disk containing the Intel PRO/1000 Gigabit Adapter driver, then click the“OK” button.
12) The necessary files will be installed and the Device Driver Wizard will display a “Completed” message. Click the “Finish” button.
13) Repeat the above procedure for the second Ethernet device.
14) Restart Windows 2000.
15) When Windows 2000 has restarted, open the Device Manager again and test that the Intel 82546EB-based MT Gigabit Adapters, now located in the Network adapter branch, are operational.

3.5 Installing RedHat® Linux® 7.1

To install RedHat Linux 7.1 from CD-ROM, set up the board initially using the steps outlined in Sections 3.1 and 3.2 above, ensuring that all the necessary drives are connected. Then follow the procedure below.

1) Obtain the Ethernet driver from the Intel web site, starting at the address:

   http://www.intel.com

   Click “Support”, “resources for developers”, then follow the links for: “Networking Drivers”, “Adapters for Desktop and Server Systems”. Click on the “Latest drivers” for the “Intel PRO/1000 MT Desktop RJ45”.

   Select the “Linux” option followed by the “Linux Gigabit Adapter Base Driver” file. Copy this file to a floppy disk. Download the “readme” file and keep a copy to use.

2) Follow the standard RedHat installation instructions, but at the screen following the selection of monitor type, ensure that a “Text” login type is selected. This prevents the system from automatically starting the X11 window software.

3) Proceed through the remaining installation sequence. The installer will not allow configuration of the network adapters at this stage.

4) After the installation is complete and the board has been rebooted, login as the super user (login name **root**).

5) Create a working directory and copy the files from the Gigabit Ethernet driver floppy disk created in Step 1.

6) Follow the instructions in the “readme” file also downloaded in Step 1, to rebuild and install the new Gigabit Ethernet driver.

7) Reboot Linux.

8) During the boot sequence, the Ethernet Controllers will be detected and a prompt will appear to allow their configuration. Fill out the forms for network parameters appropriately for the network being used. When this is complete, reboot the operating system to enable the new settings.

9) For full control of the system configuration use the **linuxconf** utility. This is not installed by the RedHat installer but can be manually installed from the RedHat CD.

   NOTE It is not essential to install the **linuxconf** utility.

   To install the **linuxconf** utility, insert CD 2 of 2 into the CD-ROM drive and enter the following commands:

```
mount /dev/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm -i linuxconf-1*
```

   When the **linuxconf** installation is complete, the CD can be unmounted and removed from the drive:

```
umount /mnt/cdrom
```

   Type **linuxconf** and follow the on screen forms and help for system configuration.

10) If the X11 window system will be used, there are some additional steps to take. The video driver included in RedHat 7.1 is not fully compatible with the Asiliant 69030 graphics controller. Concurrent Technologies can provide on request an updated driver which corrects this problem. The new driver must be placed in the **/usr/X11R6/lib/modules/drivers** directory.

    By default RedHat configures the XFree86 X11 version 3 X server. To use the new video driver, XFree86 version 4 must be used. To enable the version 4 X server, login as the super user (**root**) and type the following commands:

```
rm /etc/X11/X
ln -s /usr/X11R6/bin/XFree86 /etc/X11/X
```

3.6 Using VxWorks 5.4 with Tornado 2

Applications using this operating system are not developed on the target hardware. Concurrent Technologies can supply on request a separate Board Support Package (BSP) for this board and many others. Read the “readme” file provided with this package for details of how to configure and run VxWorks on the PP 110/01x board.

Mass Storage Interfaces

The PP 110/01x board has two interfaces which can be used to attach mass storage devices:

- a Primary EIDE (ATA100) interface is accessible via the CompactPCI J5 connector
- a Secondary EIDE (ATA100) interface supporting on-board Mass Storage option kits

In addition, the AD PP5/001 Transition Module provides a floppy disk interface. SCSI may be provided by fitting PMC SCSI modules.

The order in which the PC BIOS firmware tries to bootload from these drives can be changed via the BIOS Setup screen for **Boot**.

4.1 EIDE Interfaces

The board supports two EIDE (ATA100) interfaces.

The Primary EIDE interface connects via the CompactPCI J5 connector of the PP 110/01x board, or through the Transition Module. Up to two EIDE peripherals may be connected to this interface. The BIOS Setup screens, for **Main | Primary Master** and **Main | Primary Slave** allow the user to see what is connected to this interface, and to select some characteristics of the drives manually. Normally the PC BIOS firmware will automatically determine the drive characteristics from the drives themselves.

The Secondary EIDE interface connects only to the optional Hard Disk or CompactFlash Storage Kits. The Hard Disk kit will appear as the Secondary Master drive, and the CompactFlash cards on the CompactFlash kit will appear as the Secondary Master and Secondary Slave drives. The BIOS Setup screens for **Main | Secondary Master** and **Main | Secondary Slave** allow the user to see what is connected to this interface, and to select some characteristics of the drives manually.

Note that when using faster EIDE drives the overall cable length from the PP 110/01x board to the drive furthest from the board must be kept as low as possible, and in any case no more than 18 inches or 450 mm. If this is not practical, it may be necessary to reduce the interface performance using the **UltraDMA Mode** and **Transfer Mode** fields of the BIOS Setup screens indicated above.

To achieve the faster speeds (above ATA33) via the Primary EIDE Interface it will also be necessary to use the correct (80-way) type of EIDE cable, and to manually select the **User** configuration and **UDMA 4** or **UDMA 5** speeds for the drive using the BIOS Setup screens for **Main|Primary Master** and **Main|Primary Slave** as appropriate. Selection of the fastest speed for the Secondary (on-board) EIDE interface is automatic.

This page has been left intentionally blank

Ethernet Interfaces

The PP 110/01x board is fitted with two independent 1Gbit/s Ethernet interfaces, implemented with an Intel 82546EB controller. These interfaces can connect in different ways, depending primarily on the build configuration of the board. This configuration is indicated by the first digit of the board name suffix. For example, the name PP 110/012-12 indicates a 1.2GHz processor board with 1.5 Gbytes of DRAM, V(I/O) set for 5V, and built for Rear Ethernet. The name PP 110/012-22 indicates the same board but built for backplane networking (PICMG 2.16).

5.1 Rear Ethernet Configuration

If the board is built for operation with Rear Ethernet, both interfaces connect via J3 to a Transition Module which provides the isolating “magnetics” to support standard RJ45 connectors and Category 5 cable. Ethernet channel 0 may be switched to route the signals to either the front panel connector or to the connector on the Transition Module. This switching may be performed by appropriate software running on the board, but can also be set up using the BIOS Setup screen for **Advanced** configuration. Neither interface can be connected simultaneously via the front panel and Transition Module connections. When operating via the front panel connector, the Ethernet line speed is limited to 10 or 100Mbits/s. When operating via the Transition Module, 1Gbit/s operation is supported.

5.2 PICMG 2.16 Configuration

If the board is built for operation with backplane networking (PICMG 2.16), both interfaces connect via J3, but the isolating “magnetics” are now also built into the PP 110/01x board itself. This allows the board to connect directly to fabric boards via the J3 sub-plane of a PICMG 2.16 backplane. If a Transition Module is used, it must be wired or jumpered such that it does not connect to the pins on J3 which provide the Ethernet signals. The AD PP5/001 Transition Module is fitted with switches to provide this isolation where necessary. Ethernet channel 0 may be switched between front panel and backplane connections, using the same methods as for the Rear Ethernet configuration. The speed of operation is again limited to 10 or 100Mbits/s via the front panel connector, but 1Gbit/s via the backplane.

This page has been left intentionally blank

Other Interfaces

Many additional standard interfaces are provided on the PP 110/01x board. These interfaces consist primarily of those found in a regular desktop or mobile PC, and are outlined below.

6.1 Serial Ports

One RS232 serial interface is provided on the PP110/01x board, and connects via the front panel using a shared 26-way high density connector. A splitter cable (part number CB 26D/124) is required to access the serial interface. The serial port is implemented in the PC chipset used on the board, using a standard 16550-style device. The serial line may be configured for speeds up to 115kbaud.

With some operating systems, or in some applications, it is preferable to use a serial terminal as an operator console device for the board. In this case, it will be necessary to configure the board for operation with a Serial Console. When configured in this mode, the PC BIOS firmware will re-direct its output to the COM1 port, and similarly will take its input from this port, rather than using the VGA screen and PC keyboard. A board jumper must be set to select this mode, and is shown in Figure 6-1 below.

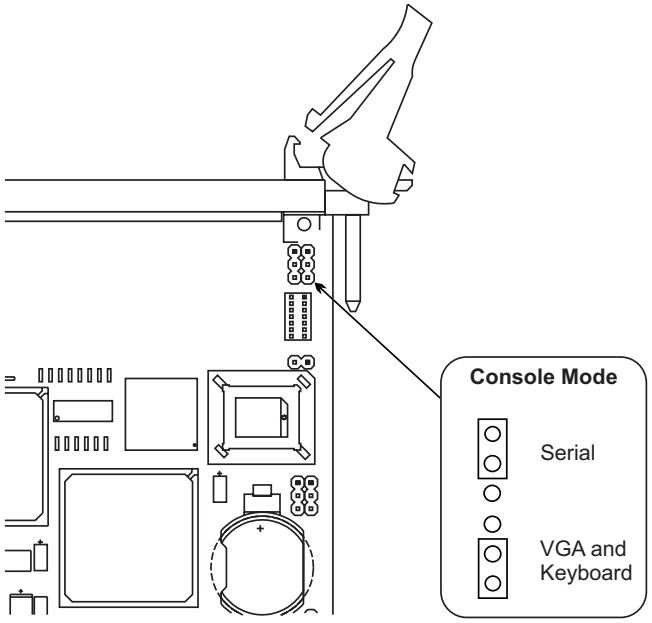


Figure 6-1 Console Jumper

The serial line speed used for the Serial Console mode may be selected from the BIOS Setup screen for **`Main`** configuration.

6.2 Keyboard and Mouse Ports

A shared 26-way high density PS/2 connector on the front panel of the board provides connections for a PC keyboard and a PS/2 mouse. The pin-out of the front panel connector is detailed in Section A.5.1. A splitter cable (part number CB 26D/124) is required to access the keyboard and mouse interfaces.

Power for the keyboard and mouse interfaces is protected by a 0.75A self-resetting current limiting circuit. To reset this circuit, cycle the power to the board off and on again.

6.3 Graphics (VGA) Controller

An Asiliant Technologies 69030 graphics controller chip is fitted to this board. This chip includes 4 Mbytes of graphics RAM, and supports many different modes of operation, including VGA and SVGA, with resolutions up to 1600x1200 and color depths up to 24 bits (16 million colors). The CRT interface is available on a shared 26-way high density front panel connector. A splitter cable (part number CB 26D/124) is required to access this interface.

See also Section 6.1 for information about how to switch the PC BIOS console device between VGA screen and keyboard, and a serial port.

6.4 Real-Time Clock

A conventional PC Real-Time Clock is included on this board. This is Year 2000 compliant and can be powered by an additional Lithium battery when main power to the board is removed. See Section 2.7 for more details of how to fit or replace the battery. The Clock device also provides 242 bytes of CMOS RAM, in which the PC BIOS keeps much of its setup screen data and other information.

6.5 Universal Serial Bus (USB)

One USB 1.0 interface is provided on this board. Channel 1 is connected via the CompactPCI J5 connector or a Transition Module. This channel can operate at 1.5Mbits/s or 12Mbits/s.

6.6 Power On Self Test LED/Speaker

The Power On Self Test (POST) LED is connected to the PC Speaker port. A connection for this port is also made available via the CompactPCI J5 connector. This connection is driven by an open-collector device and can drive a speaker of 32 or 64 impedance.

Intelligent Platform Management Interface

7.1 Introduction

The Intelligent Platform Management Interface (IPMI) is an industry-standard environment which allows centralized monitoring and control of a computer system. The features of the IPMI support the management of CompactPCI or other systems containing multiple intelligent modules and other standard features such as sensors for system temperature, fan failure or chassis intrusion. The IPMI specifications define the protocols used by multiple intelligent devices conforming to these specifications to communicate with each other, and in some cases with external systems used for remote monitoring and control. Provision is made in these specifications for non-volatile storage of inventory, control and status information, and to allow the IPMI subsystem to implement other system-specific features.

Although the IPMI specifications can be applied to many different types of system, the remainder of this chapter considers only how they apply to CompactPCI systems containing the PP 110/01x board. In addition, the IPMI features of this board would normally be used with System Management Software, and the specifications and design of this software are beyond the scope of this manual. For an in-depth discussion of IPMI, the reader is referred to the original IPMI specifications. At the time of writing, these specifications are available from the World Wide Web, at the address:

http://developer.intel.com/design/servers/ipmi/index.htm

In a practical implementation, the features of an IPMI may be used by System Management Software to implement a complete control and monitoring application. The specifications and design of this software are beyond the scope of this manual. The System Management Software will normally be run on the CompactPCI system controller board and in this role the PP 110/01x board's local IPMI subsystem will be termed the Baseboard Management Controller (BMC). In this role the board acts as the interface between the System Management Software and the CompactPCI system chassis. The BIOS Setup screen for **Advanced** allows the user to force the board to act either as a BMC or as a Slave Management Controller (SMC) in any slot.

The remainder of this chapter outlines the functions provided by this board via IPMI, and details some ways in which these features can be used. For a more complete description of the IPMI protocols and implementation on Concurrent Technologies' boards, refer to the document: "Intelligent Platform Management Interface for Concurrent Technologies Boards", order code 555 0042.

7.2 IPMI Compatibility

This board implements a version of IPMI compatible with revision 1.5 of the IPMI specifications. It includes the mandatory elements of the PCI Industrial Computer Manufacturer's Group (PICMG) specification 2.9.

The IPMI facilities supported by the PP 110/01x board are implemented using a microcontroller and its resident firmware with non-volatile operational data stored in EEPROM. A total of 8 Kbytes of non-volatile storage is provided for inventory, sensor and event data recording. This implementation provides a hardware interface which conforms to the Server Management Interface Controller (SMIC) type as defined in the IPMI specifications.

7.3 IPMI Overview

The PP 110/01x board includes hardware and firmware which implements an IPMI as a separate resource to the main Pentium III-M processor. The processor communicates with the IPMI subsystem through a standardized hardware interface using multi-byte message sequences, transferring one byte at a time using a handshaking protocol. Instructions can be sent to the IPMI subsystem with a command and response message pair. The IPMI subsystem may in turn communicate with additional hardware in the chassis using similar message sequences transferred over one or more Intelligent Platform Management Busses (IPMB). The IPMB 0 bus provides a 2-wire interface based on the Philips® I^2C protocol, and in the CompactPCI chassis connects to all boards on the CompactPCI backplane via the J1 connector.

The IPMB 0 bus, and, in some cases, other busses managed by the Baseboard Management Controller, can connect to additional intelligent or non-intelligent devices which may act as sensors or simply data sources and sinks. These devices may include fan or power supply monitoring hardware, temperature sensors or perhaps just non-volatile memory. The IPMI specifications define data structures for these sensors which are stored in non-volatile memory, and describe both control and run-time status (event) information that can be retrieved by System Management Software.

As the IPMI subsystem on the PP 110/01x board is separated from the main processor, many of its features can operate when main power is removed, provided that power is supplied to the board through the IPMB_PWR pin of the J1 connector. This allows the board to be interrogated via the IPMB even in a power-down state, which may be useful when the board is operating in Satellite mode.

The programming interface to the IPMI subsystem is via the System Management Interface Controller (SMIC). This is a set of I/O registers accessed using a polled handshaking protocol. Example software for driving this interface is provided in Section 7.5.1 of this manual. When the board is acting as an SMC, software running on the local processor may only access local IPMI resources through the SMIC interface. When the board is acting as the BMC, software running on the local processor may access both on-board IPMI resources and those elsewhere on the IPMB.

The implementation of IPMI on this board provides the following functions.

7.3.1 Message Passing

The flow of information in an IPMI compliant system is achieved by using messages. A transaction consists of a request and a response message pair. The interface between the System Management Software and the Baseboard Management Controller uses simple messages which contain enough information to allow a response to be generated. The “Get Message” and “Send Message” commands are used to pass information to the IPMB and embed simple messages with channel routing information.

7.3.2 Events

Events can be generated whenever a system failure is detected. These events are stored in the System Event Log and can be retrieved by the System Management Software which can process the events and determine what, if any, corrective action can be taken.

7.3.3 System Event Log

Events are stored in the System Event Log which is held in non-volatile memory. The System Management Software can access the System Event Log and by analyzing the events may be able to determine the sequence of events that caused the system failure.

7.3.4 Sensors

The PP 110/01x board is capable of monitoring the temperature of the board and the processor chip, the levels of the main power supply voltage rails, the status of a system fan and the board's Geographic Address (effectively its CompactPCI bus slot number). These sensors can be configured to generate events when, for example, the temperature of the processor chips exceeds a previously configured value. Sensor Data Records are used to contain information pertaining to sensors.

7.3.5 Sensor Data Records

Sensor Data Records (SDRs) are used to describe the configuration and characteristics of sensors and can also be used to describe the operating characteristics of the Intelligent Platform Management Device. These Sensor Data Records are stored in a repository held in non-volatile memory. The PP 110/01x board is supplied with 10 preconfigured SDRs.

7.3.6 Field Replaceable Unit Inventory Data

The Field Replacement Unit (FRU) Inventory Data is stored in non-volatile memory and allows the System Management Software to determine the identity of Intelligent Platform Management devices in the system. Typically this data comprises the manufacturer's name, board name, part number, serial number, etc.

7.3.7 Watchdog

The Watchdog function allows the System Management Software to protect the system from errant programs. For example, a critical program taking 3 seconds to complete may set the watchdog to expire in 5 seconds. If the watchdog expires then the system management software can take the appropriate corrective action.

7.4 Supported Commands

The PP 110/01x board supports a subset of the command messages as defined in the IPMI specification. All commands, unless explicitly stated, expect a response which is generally retrieved through the SMIC interface. Commands that are not supported receive a “Invalid Command” response.

The following commands are supported and their purpose and basic operation are outlined in the following sections.

- Get Device ID.
- Broadcast Get Device ID.
- Cold Reset.
- Warm Reset.
- Get Self Test Results.
- Get Chassis Capabilities.
- Set BMC Global Enables.
- Get BMC Global Enables.
- Clear Message Flags.
- Get Message Flags.
- Get Message.
- Send Message.
- Set Watchdog Timer.
- Get Watchdog Timer.
- Reset Watchdog Timer.
- Set Event Receiver.
- Get Event Receiver.
- Platform Event Message.
- Get SEL Information.
- Get SEL Allocation Information.
- Reserve SEL.
- Clear SEL.
- Get SEL Entry.
- Add SEL Entry.
- Delete SEL Entry.
- Get SEL Time.
- Set SEL Time.
- Get Sensor Reading.
- Get SDR Repository Information.
- Get SDR Repository Allocation Information.
- Reserve SDR Repository.
- Get SDR.
- Add SDR.
- Clear SDR Repository.
- Get SDR Repository Time.
- Get FRU Inventory Area Information.
- Read FRU Inventory Data.
- Write FRU Inventory Data.
- Master Write-Read.

The sub-sections below indicate board specific characteristics of these functions. For complete description of the commands, refer to the documents indicated in Section 7.1.

7.4.1 Get Self Test Results Command

On this board, no self test results are available via this command. The PC BIOS runs a Power On Self Test (POST) sequence, but the results of these tests are reported only via on-screen messages and flashes of the POST LED.

The IPMI command returns codes of 55h (meaning not supported) in the Self-Test: Results field and 00h in the Self-Test: Supplemental Information field of the response message.

7.4.2 Watchdog Commands

The basic timing function of the watchdog is controlled by a 16-bit timer with 100ms ticks giving a watchdog timing period between 0 and 6553.5 seconds. The watchdog can optionally be programmed to generate either NMI or SMI interrupts at a fixed interval (specified in seconds) before the watchdog expires. This is termed a pre-interrupt. The watchdog when it expires can be programmed to perform no action, reset the board and either power-off or power cycle the board. The watchdog can also generate an internal event which is stored in the System Event Log.

NOTE The watchdog is disabled following a power on or a reset

Example source code for controlling the Watchdog timer is provided in Section 7.5.2.

7.4.3 SEL and SDR Commands

The SEL and SDR Repository Timestamp counters are the same on this board. This means that operations affecting one of these counters also affect the other.

7.4.4 Sensor Commands

Table 7-1 lists the sensors fitted to this board, alongside the sensor number and type information found in their SDRs. These sensors are scanned by the IPMI subsystem at a rate of approximately 10Hz. The values returned by the Get Sensor Reading command are the values obtained in the most recent scan.

The Sensor LUN is 00h.

The IPMI subsystem will create an IPMB Management Controller Locator SDR and an IPMB Management Controller Confirmation SDR. These records provide status information indicating the capability of the board depending on its position in the CompactPCI chassis, and are used primarily by other IPMB devices which want to retrieve information about the other devices on the IPMB bus.

Sensor Function	Sensor Number	Sensor Type	Event Type
Board Temperature	00h	01h	01h
CPU Temperature	01h	01h	01h
Fan Monitor	80h	0Ah	03h
Voltage, +12V	10h	02h	01h
Voltage, +5V	11h	02h	01h
Voltage, +3.3V	12h	02h	01h
Voltage, -12V	13h	02h	01h
cPCI Slot Number	30h	C0h	6Fh
cPCI BDSEL Signal	31h	C1h	03h
cPCI SYSEN Signal	32h	C1h	03h
cPCI PRESENT Signal	33h	C1h	03h
Hot Swap Ctrl Power Good	34h	02h	06h
Hot Swap Ctrl Power Fail	35h	02h	06h
Vcore & Vtt	36h	02h	06h
Vaux	38h	02h	06h
Vbat	39h	02h	06h
Ejector Handle Status	3Ah	C1h	03h

Table 7-1 IPMI Sensors

7.4.4.1 Board Temperature Sensor

This sensor is located in the position shown in Figure A-1. A Maxim 1617 or 1617A device is used to measure this reading, and the result is returned by the IPMI subsystem in degrees Celsius. The sensor has a quoted accuracy of +/- 3°C over the operating temperature range of the board. The resolution is 1°C.

7.4.4.2 CPU Temperature Sensor

This sensor is located inside the CPU chip itself. A Maxim 1617 or 1617A device is used to measure this reading, and the result is returned by the IPMI subsystem in degrees Celsius. The sensor has a quoted accuracy of +/- 3°C over a range of +60°C to +100°C, and of +/- 5°C outside this range. The resolution is 1°C.

7.4.4.3 Fan Monitor

Fan failure is detected by a single change in the state of the fan monitor input on the J5 connector. The signal state which indicates failure is set in the Sensor Data Record (SDR) for this sensor. The Assertion Event Mask / Lower Threshold Reading Mask and Deassertion Event Mask / Upper Threshold Reading Mask fields are used to set these parameters. The default settings for these fields select an active low fan failure indication. To select an active high fan failure indication, the pre-defined SDR must be deleted and replaced by one which sets these fields to 0002h.

7.4.4.4 Voltage, +12V Sensor

The sensor reading returned for this sensor must be converted to a voltage reading by the application software. Using the default parameters for this sensor, multiply by 0.0708 to convert the reading to a voltage.

7.4.4.5 Voltage, +5V Sensor

The sensor reading returned for this sensor must be converted to a voltage reading by the application software. Using the default parameters for this sensor, multiply by 0.0244 to convert the reading to a voltage.

7.4.4.6 Voltage, +3.3V Sensor

The sensor reading returned for this sensor must be converted to a voltage reading by the application software. Using the default parameters for this sensor, multiply by 0.0244 to convert the reading to a voltage.

7.4.4.7 Voltage, -12V Sensor

The sensor reading returned for this sensor must be converted to a voltage reading by the application software. Using the default parameters for this sensor, multiply by (-0.0781) to convert the positive sensor reading into a voltage in the range -12V to 0V.

7.4.4.8 CompactPCI Slot Number

This sensor reading indicates board slot number which is read from geographic address pins on the CPCI backplane. The possible values are from 0 to 31.

7.4.4.9 CompactPCI BDSEL Signal

This sensor reading indicates status of the BDSEL# pin on the CPCI backplane. The board is fully seated if the value is 1. If the value is 0, the board is not fully seated.

7.4.4.10 CompactPCI SYSEN Signal

This sensor reading indicates status of the SYSEN# pin on the CPCI backplane. The board is in the system controller slot if the value is 1. If the value is 0, the board is not in the system controller slot.

7.4.4.11 CompactPCI PRESENT Signal

This sensor reading indicates status of the PCI_PRESENT# pin on the CPCI backplane. The PCI bus is present at this slot if the value is 1. If the value is 0, the PCI bus is not present at this slot. This sensor is provided mainly for use with PICMG 2.16 backplanes.

7.4.4.12 Hot Swap Control Power Good

This sensor reading indicates status of power supply voltages from the backplane. If the value is 0, the power supply voltages are within specification. If the value is 1, one or more voltages are out of specification.

7.4.4.13 Hot Swap Control Power Fail

This sensor reading indicates status of power supply currents from the backplane. If the value is 0, the power supply currents are good. If the value is 1, an overcurrent has occurred.

7.4.4.14 Vcore & Vtt

This sensor reading indicates the status of the CPU core and Vtt regulator voltages. If the value is 0, the voltage is good. If the value is 1, the voltage is outside the correct operating range.

7.4.4.15 Vaux

This sensor reading indicates the status of 1.5V and 2.5V voltages. If the value is 0, the voltage is good. If the value is 1, the voltage is outside the correct operating range.

7.4.4.16 Vbat

This sensor reading indicates the status of the battery voltage. If the value is 0, the voltage is good. If the value is 1, the voltage is outside the correct operating range.

7.4.4.17 Ejector Handle

This sensor reading indicates the status of the ejector handle switch. If the value is 0, the handle is closed. If the value is 1, the handle is open.

7.4.5 FRU Inventory Data

The FRU Inventory Area shares the non-volatile memory with the repository of Sensor Data Records and the System Event Log and contains information about the board, including, for example, the manufacturer's name, part number and serial number.

The FRU Inventory Area comprises, at most, six information areas. Each area, if used, is always a multiple of 8 bytes in length. The Common Header Area is always included and the PP 110/01x board also includes the Board Area. The Internal Use, Chassis Information, Product Information and Multi-Record areas are, currently, not used.

7.4.5.1 Common Header Area

The Common Header Area is always included in the FRU Inventory Area and is used to specify the location of the other areas in the FRU Inventory Area. The offset of any area in the FRU Inventory Area must be a multiple of 8h. The Common Header Area always has an offset of 0000h and is arranged as shown in Table 7-2.

FRU Inventory Area Offset	Value	Meaning
0000h	01h	Common Header Format Version
0001h	00h	Offset to Internal Use Area
0002h	00h	Offset to Chassis Information Area
0003h	01h	Offset to Board Area
0004h	00h	Offset to Product Information Area
0005h	00h	Offset to Multi-Record Area
0006h	00h	Reserved
0007h	FEh	Common Header Area Checksum

Table 7-2 FRU Inventory Area : Common Header Area Data

7.4.5.2 Board Area

The Board Area, in the FRU Inventory, contains information about the PP 110/01x board. Its offset in the FRU Inventory Area is calculated by multiplying its offset value in the Common Header Area by 8. The Board Area is arranged as shown in Table 7-3.

Board Area Offset	Value	Comments
0000h	01h	Board Area format version
0001h	09h	Board Area length
0002h	19h	Language code
0003h	xxxxxxh	Manufacturing date and time
0006h	D7h	ASCII+LATIN 1 character set and 23 characters
0007h	“Concurrent Technologies”	Manufacturer’s name
001Eh	CAh	ASCII+LATIN 1 character set and 10 characters
001Fh	“PP 110/01x”	Board’s name
0029h	CAh	ASCII+LATIN 1 character set and 10 characters
002Ah	“M0001/999”	Typical serial number
0034h	CBh	ASCII+LATIN 1 character set and 11 characters
0035h	“760-6009-xx”	Board’s part number
0040h	C0h	No FRU file ID record is define
0041h	C1h	No more records
0042h	00h	Reserved
0043h	00h	Reserved
0044h	00h	Reserved
0045h	00h	Reserved
0046h	00h	Reserved
0047h	xxh	Board Area checksum

Table 7-3 FRU Inventory Area : Board Area Data

7.4.5.2.1 Board Area Format Version

This field defines the format of the Board Area and is always set to 01h.

7.4.5.2.2 Board Area Length

This size of the Board Area in the FRU Inventory Area is computed by multiplying this field by 8.

7.4.5.2.3 Language Code

The language code is always set to 19h (English).

7.4.5.2.4 Manufacturing Date and Time

This is defined as the number of minutes since 00:00 hours on the 1st January 1996.

7.4.5.2.5 Manufacturer’s Name

This is a string of characters and is set to “Concurrent Technologies”. It is not zero terminated.

7.4.5.2.6 Board's Name

This is a string of characters representing the board's name and is set to "PP 110/01x". It is not zero terminated.

7.4.5.2.7 Board's Part Number

This is a string of characters representing the board's internal part number. It is not zero terminated.

7.4.5.2.8 Serial Number

This is a string of characters matching the serial number imprinted on the board. It is not zero terminated. An example of a serial number is "M3144/003".

7.4.5.2.9 FRU File ID

This field is used to store manufacturing information but is currently unused.

7.5 Programming Examples

7.5.1 Using the SMIC Interface

The PP 110/01x single board utilizes the Server Management Interface Controller (SMIC) as its interface with the I/O ports at the standard addresses (i.e. 0CA9h, 0CAAh and 0CABh). The IPMI specification has a complete description of the SMIC interface. The following C program fragment reads and writes IPMI messages using the SMIC interface:

```
/* Addresses of SMIC registers */

#define SMIC_DATA       0x0CA9  /* Data Register */
#define SMIC_CONTROL    0x0CAA  /* Control & Status Register */
#define SMIC_FLAGS      0x0CAB  /* Flag Register */

/* SMS Transfer Stream Control Codes */

#define CC_SMS_GET_STATUS   0x40
#define CC_SMS_WR_START     0x41
#define CC_SMS_WR_NEXT      0x42
#define CC_SMS_WR_END       0x43
#define CC_SMS_RD_START     0x44
#define CC_SMS_RD_NEXT      0x45
#define CC_SMS_RD_END       0x46

/* SMS Transfer Stream Status Codes */

#define SC_SMS_RDY          0xC0
#define SC_SMS_WR_START     0xC1
#define SC_SMS_WR_NEXT      0xC2
#define SC_SMS_WR_END       0xC3
#define SC_SMS_RD_START     0xC4
#define SC_SMS_RD_NEXT      0xC5
#define SC_SMS_RD_END       0xC6

/* Masks for SMIC Flags Registers Bits */

#define FLAG_RX_DATA_RDY    0x80
#define FLAG_TX_DATA_RDY    0x40
#define FLAG_SMI            0x10
#define FLAG_EVT_ATN        0x08
#define FLAG_SMS_ATN        0x04
#define FLAG_BUSY           0x01

#define FLAGS_BUSY          0x01
#define FLAGS_TX_DATA_READY 0x40
#define FLAGS_RX_DATA_READY 0x80

/* macros */

#define bReadSmicFlags  (inb (SMIC_FLAGS))
#define bReadSmicStatus (inb (SMIC_CONTROL))
#define bReadSmicData   (inb (SMIC_DATA))

#define vWriteSmicControl(data)     outb(SMIC_CONTROL, data)
#define vWriteSmicData(data)        outb(SMIC_DATA, data)
#define vWriteSmicFlags(data)       outb(SMIC_FLAGS, data)

/*****************************************************************************
 *
 * vBmcSmicSmsMessageWrite
 *
 * This function writes a SMS (System Managment Software) messages
 * to the IPMI using the standard BMC-SMIC interface.
```

```
 *
 * Returns: N/A
 *
 */
void vBmcSmicSmsMessageWrite
(
    const unsigned char *pbMessage, /* request */
    unsigned char bLength           /* request length */
)
{
    unsigned char bMessageStage;


    while (((bReadSmicFlags) & FLAGS_BUSY) == FLAGS_BUSY)
        ; /* do nothing ... */

    vWriteSmicControl (CC_SMS_WR_START);
    vWriteSmicData (*pbMessage++);
    vWriteSmicFlags (FLAGS_BUSY);
    bLength--;

    do
    {
        while (((bReadSmicFlags) & FLAGS_TX_DATA_READY) != FLAGS_TX_DATA_READY)
            ; /* do nothing ... */

        while (((bReadSmicFlags) & FLAGS_BUSY) == FLAGS_BUSY)
            ; /* do nothing ... */

        vWriteSmicControl ((bLength > 1) ? CC_SMS_WR_NEXT : CC_SMS_WR_END);
        vWriteSmicData (*pbMessage++);
        vWriteSmicFlags (FLAGS_BUSY);
        bLength--;

        while (((bReadSmicFlags) & FLAGS_BUSY) == FLAGS_BUSY)
            ; /* do nothing ... */
    }
    while (bLength > 0);

}


/*****************************************************************************
 *
 * vBmcSmicSmsMessageRead
 *
 * This function reads a SMS (System Managment Software) message
 * from the IPMI using the standard BMC-SMIC interface.
 *
 * Returns: N/A
 *
 */
void vBmcSmicSmsMessageRead
(
    unsigned char *pbMessage,       /* received response */
    unsigned char *bMessageLength   /* response length */
)
{
    unsigned char bReadControl;
    unsigned char bReadStatus;
    unsigned char bReadData;

    *bMessageLength = 0;

    bReadControl = CC_SMS_RD_START;
```

```
    do
    {
        while (((bReadSmicFlags) & FLAGS_RX_DATA_READY) != FLAGS_RX_DATA_READY)
            ; /* do nothing ... */

        while (((bReadSmicFlags) & FLAGS_BUSY) == FLAGS_BUSY)
            ; /* do nothing ... */

        vWriteSmicControl (bReadControl);
        vWriteSmicFlags (FLAGS_BUSY);
        while (((bReadSmicFlags) & FLAGS_BUSY) == FLAGS_BUSY)
            ; /* do nothing ... */

        bReadStatus = bReadSmicStatus;
        bReadData   = bReadSmicData;

        switch (bReadStatus)
        {
            case SC_SMS_RD_START :
                *pbMessage++ = bReadData;
                (*bMessageLength)++;
                bReadControl = CC_SMS_RD_NEXT;
                break;
            case SC_SMS_RD_NEXT  :
                *pbMessage++ = bReadData;
                (*bMessageLength)++;
                break;
            case SC_SMS_RD_END   :
                *pbMessage++ = bReadData;
                (*bMessageLength)++;
                bReadControl = CC_SMS_RD_END;
                break;
            default :
                break;
        }
    }
    while (bReadStatus != SC_SMS_RDY);

    return;
}
```

7.5.2 Using the Watchdog Timer

The IPMI provides a standardized watchdog facility which is fully described in the IPMI specification. The following C program fragment sets and resets the watchdog facility:

```
/* network function codes */

#define NFC_APP_REQUEST         0x06
#define NFC_APP_RESPONSE        0x07

/* watchdog commands */

#define CMD_WATCHDOG_RESET  0x22    /* Reset Watchdog Timer */
#define CMD_WATCHDOG_SET    0x24    /* Set Watchdog Timer */
#define CMD_WATCHDOG_GET    0x25    /* Get Watchdog Timer */

/* definition of watchdog operational constants */

#define USAGE_BIOS_FRB2     1
#define USAGE_BIOS_POST     2
#define USAGE_OS_LOAD       3
#define USAGE_SMS_OS        4
#define USAGE_OEM           5

#define FLAG_BIOS_FRB2      (1 << (USAGE_BIOS_FRB2))
#define FLAG_BIOS_POST      (1 << (USAGE_BIOS_POST))
#define FLAG_OS_LOAD        (1 << (USAGE_OS_LOAD))
#define FLAG_SMS_OS         (1 << (USAGE_SMS_OS))
#define FLAG_OEM            (1 << (USAGE_OEM))

#define PRE_TO_INT_NONE         0
#define PRE_TO_INT_SMI          1
#define PRE_TO_INT_NMI          2
#define PRE_TO_INT_MESSAGE      3

#define TO_ACTION_NONE          0
#define TO_ACTION_RESET         1
#define TO_ACTION_POWER_DOWN    2
#define TO_ACTION_POWER_CYCLE   3


/* Completion codes */

#define COMPLETION_OK           0       /* Completion code OK */

/* error codes */

#define E_OK            0           /* OK */
#define E_COMPLETION    0x400       /* wrong completion code */

/* forward declarations */

void vBmcSmicSmsMessageWrite (const unsigned char *pbMessage,
                              unsigned char bLength);
void vBmcSmicSmsMessageRead (unsigned char *pbMessage,
                              unsigned char *bMessageLength);



/*****************************************************************************
 *
 * wSetWatchdog
 *
 * This function defines operation of the IPMI watchdog which is initiated
 * by the Reset Watchdog Command issued the the wResetWatchdog function.
```

```
 *
 * RETURNS: E_OK if it is OK, or error code
 *
 */
unsigned short int wSetWatchdog
(
    unsigned char bDontLog,                 /* FALSE to log event */
    unsigned short int wTimeoutInterval,    /* multiples of 100ms */
    unsigned char bTimeoutAction,
    unsigned char bPreTimeoutInterval,      /* multiples of 1s */
    unsigned char bPreTimeoutInterrupt,
    unsigned char bTimerUse,
    unsigned char bTimerUseClearFlags
)
{
    unsigned char abRequest  [10];
    unsigned char abResponse [10];
    unsigned char bLength;
    unsigned short int wStatus = E_OK;

    abRequest [0] = NFC_APP_REQUEST << 2;
    abRequest [1] = CMD_WATCHDOG_SET;       /* command */
    abRequest [2] = ((bDontLog) ? 0x80 : 0) | bTimerUse & 0x07;
    abRequest [3] = ((bPreTimeoutInterrupt & 0x07) << 4)
                    | (bTimeoutAction & 0x07);
    abRequest [4] = bPreTimeoutInterval;
    abRequest [5] = bTimerUseClearFlags;
    abRequest [6] = (unsigned char ) (wTimeoutInterval & 0x00FF);
    abRequest [7] = (unsigned char ) (wTimeoutInterval >> 8);

    vBmcSmicSmsMessageWrite (abRequest, 8);
    vBmcSmicSmsMessageRead (abResponse, &bLength);

    if (abResponse [2] != COMPLETION_OK)
        wStatus = E_COMPLETION;
}


/*****************************************************************************
 *
 * wResetWatchdog
 *
 * This function starts / restarts the IPMI watchdog.
 *
 * RETURNS: E_OK if it is OK, or error code
 *
 */
unsigned short int wResetWatchdog (void)
{
    unsigned char abRequest  [10];
    unsigned char abResponse [10];
    unsigned char bLength;
    unsigned short int wStatus = E_OK;

    abRequest [0] = NFC_APP_REQUEST << 2;
    abRequest [1] = CMD_WATCHDOG_RESET; /* command */

    vBmcSmicSmsMessageWrite (abRequest, 2);
    vBmcSmicSmsMessageRead (abResponse, &bLength);

    if (abResponse [2] != COMPLETION_OK)
        wStatus = E_COMPLETION;
        return wStatus;
}
```

The following example shows how to set the watchdog to power cycle if the watchdog is not restarted within 20 seconds, generate an NMI if there is less than 10 seconds before the watchdog expires, define its use as an “operating system load” watchdog, clear any OEM expiration flags, log the watchdog failure in the event log:

```
wSetWatchdog (FALSE, 200, TO_ACTION_POWER_CYCLE,
                     10, PRE_TO_INT_NMI,
                     USAGE_OS_LOAD, FLAG_OEM);
```

The **wSetWatchdog** function does not start the watchdog facility and the **wResetWatchdog** function must be used. This function simply restarts from the internal watchdog timer to the value specified in the **vSetWatchdog** function. The **wResetWatchdog** function is used thus:

```
wResetWatchdog ();
```

The **vSetWatchdog** function is used to disable the watchdog facilities and is used thus:

```
wSetWatchdog (FALSE, 0, 0, 0, 0, 0, 0);
```

7.5.3 Reading Sensors

The IPMI specification supports many commands to manage and interrogate sensors. The following program fragment illustrates how the current reading of a sensor can be obtained.

```
/* network function codes */

#define NFC_SENSOR_EVENT_RQ 0x04

/* commands */

#define CMD_GET_SENS_RD     0x2D    /* Get sensor reading */

/* Completion codes */

#define COMPLETION_OK       0       /* Completion code OK */

/* error codes */

#define E_OK            0           /* OK */
#define E_COMPLETION    0x400       /* wrong completion code */

/* forward declarations */

void vBmcSmicSmsMessageWrite (const unsigned char *pbMessage,
                              unsigned char bLength);
void vBmcSmicSmsMessageRead (unsigned char  *pbMessage,
                              unsigned char *bMessageLength);



/* response data structure provided to wGetSensorReadingCmd()  */

struct SENSOR_READING
{
    unsigned char bData;
    unsigned char bStatus;
};


/*****************************************************************************
 *
 * wGetSensorReadingCmd
 *
 * This function gets current sensor reading.
 *
 * RETURNS: E_OK if it is OK, or error code
 *
 */
unsigned short int wGetSensorReadingCmd
(
    unsigned char bSensorId,
    struct SENSOR_READING *psSensorReading
)
{
    unsigned char bLength;
    unsigned char abRequest  [10];
    unsigned char abResponse [10];
    unsigned short int wStatus = E_OK;

    /* Send Request */

    abRequest [0] = NFC_SENSOR_EVENT_RQ << 2;
```

```
    abRequest [1] = CMD_GET_SENS_RD;        /* command */
    abRequest [2] = bSensorId;

    vBmcSmicSmsMessageWrite (abRequest, 3);
    vBmcSmicSmsMessageRead (abResponse, &bLength);

    if (abResponse [2] != COMPLETION_OK)
        wStatus = E_COMPLETION;
    else
    {
        psSensorReading->bData   = abResponse [3];
        psSensorReading->bStatus = abResponse [4];
    }

    return wStatus;
}
```

The following example shows how to read the CPU temperature sensor and voltage sensors:

```
/* read CPU temperature, sensor ID = 01h */

wStatus = wGetSensorReadingCmd (0x01, &sSensorReading);
if wStatus==E_OK)
      printf(”CPU temperature is %dC\n”, sSensorReading.bData);

/* read +12V Power Supply Voltage, sensor ID = 10h */

wStatus = wGetSensorReadingCmd (0x10, &sReading);
if (wStatus == E_OK)
    printf (“Power Supply +12V = %fV\n”, sReading.bData * 0.0708);

/* read +5V Power Supply Voltage, sensor ID = 11h */

wStatus = wGetSensorReadingCmd (0x11, &sReading);
if (wStatus == E_OK)
   printf (“Power Supply +5V = %fV\n”, sReading.bData * 0.0244);

/* read +3.3V Power Supply Voltage, sensor ID = 12h */

wStatus = wGetSensorReadingCmd (0x12, &sReading);
if (wStatus == E_OK)
    printf (“Power Supply +3.3V = %fV\n”, sReading.bData * 0.0244);

/* read -12V Power Supply Voltage, sensor ID = 13h */

wStatus = wGetSensorReadingCmd (0x13, &sReading);
if (wStatus == E_OK)
    printf (“Power Supply -12V = %fV\n”, sReading.bData / (-12.8));
```

Flash EPROM, SRAM and DRAM

8.1 Flash EPROM

The PP 110/01x is fitted with three Flash EPROM parts: the first is installed in a socket and is programmed at the factory with PC BIOS firmware. This EPROM will not normally be reprogrammed by the user, but Concurrent Technologies has programming software which allows BIOS updates to be carried out in the field when necessary, perhaps to add new features. Contact Concurrent Technologies for a copy of this software, and for the BIOS reprogramming information, if you believe that such an update is required.

The other two Flash EPROMs are soldered to the board and are intended for Application Program storage. As standard, 32 Mbytes of 8-bit wide Application Flash is provided. This memory is accessed as several 512 Kbyte pages with an I/O register being used to select the current page. Refer to Section 9.5 for details of this register.

The top page of Application Flash device 2 is used by Concurrent Technologies for storage of factory test firmware. The user may overwrite this page if desired, but should be aware that it may be re-instated if the board is ever returned to Concurrent Technologies for repair or upgrading.

A jumper is provided to protect the Application Flash devices against accidental erasure or programming. In order to erase or program the Application Flash, the jumper must be in the Enable position. See Figure 8-1 for the location and settings of this jumper.

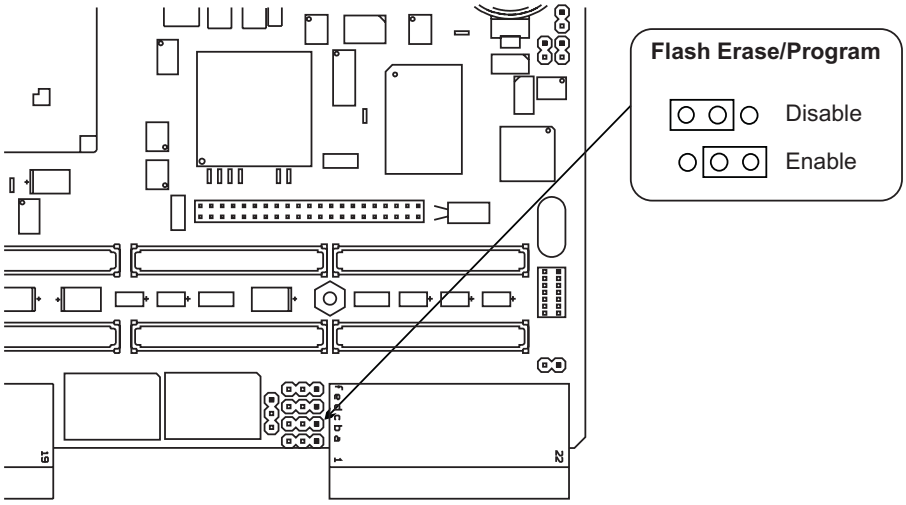


Figure 8-1 Flash Erase/Program Jumper

8.2 Static RAM

The PP 110/01x is fitted with a 4Mbit low power static RAM which provides 512 Kbytes of 8-bit wide non-volatile data storage. The battery must be fitted for this data to be retained. (See Section 2.7).

8.3 DRAM

The PP 110/01x board supports a very large amount of ECC SDRAM. Up to 1 Gbyte is soldered onto the board, and a single 144-pin SODIMM site allows an additional 256 or 512 Mbytes to be fitted either at the factory or in the field, giving a maximum size of 1.5 Gbytes. Section 2.6 describes how to fit this SODIMM, and details the types supported. All this DRAM can be accessed from both the local PCI bus and the CompactPCI backplane.

This page has been left intentionally blank

Additional Local I/O Functions

The PP 110/01x supports a variety of I/O functions whose addresses are summarized in Table 9-1.

NOTE A second LPC Super I/O controller (designated TM Super I/O in Table 9-1) is located on the AD PP5/001 Transition Module. This device provides some of the legacy peripheral functions. The parallel port and floppy disk controller sections of the on-board Super I/O controller (designated Local Super I/O in Table 9-1) are not used.

I/O Address Range	Description
0000-000Fh	Master DMA Controller (CSB5 LPC host)
0020-0021h	Master Interrupt Controller (CSB5)
002E-002Fh	Config'n Index & Data Reg's (Local Super I/O)
0040-0043h	Timers 0-2 (CSB5)
004E-004Fh	Config'n Index & Data Reg's (TM Super I/O)
0060h	Keyboard Controller (Local Super I/O)
0061h	NMI Status (CSB5)
0064h	Keyboard Controller (Local Super I/O)
0070h	NMI Enable/RTC Address (CSB5)
0078-007Bh	BIOS Timer (CSB5)
0080h	Debug Port
0092h	Port 92 (CSB5)
00A0-00A1h	Slave Interrupt Controller (CSB5)
00C0-00DFh	Slave DMA Controller (CSB5 LPC host)
00F0h	Math Coprocessor Error
0210-021Fh	Control & Status Registers & RMI
02F8-02FFh	COM2 Serial (TM Super I/O)
03BC-03BFh	Parallel Port LPT1 (TM Super I/O)
03E8-03EFh	COM3 Serial (TM Super I/O)
03F0-03F7h	Floppy Controller (TM Super I/O)
03F8-03FFh	COM1 Serial (Local Super I/O)
04D0-04D1h	Interrupt Control (CSB5)
0C14h	PCI Error Status (CSB5)
0C6Fh	Flash EPROM Write Protect (CSB5)
0CA9-0CABh	IPMI SMIC Interface
0CF8-0CFFh	PCI Configuration Registers (CNB30LE)
0D00-FFFFh	PCI Free I/O Space

Table 9-1 I/O Address Map

NOTE I/O addresses in the range 0000h-0CFFh which are not listed above should not be accessed. The effects of such accesses are unpredictable.

Most of the addresses are standard PC-AT compatible values, but at addresses 0210h - 021Fh the board provides custom Status and Control registers for the board specific features.

There are 17 byte wide status and control registers. They fall into four groups, namely, general-purpose registers, RMI interface registers, Long Duration Timer registers and IPMI SMIC interface registers. They are accessed at the following addresses:

- 210h for Status & Control Register 0.
- 211h for Status & Control Register 1.
- 212h for Status & Control Register 2.
- 213h for General Purpose I/O Register.
- 214h for Application Flash Paging Register.
- 215h for Interrupt Control Register.
- 216h for RMI Index/Flag Register.
- 217h for RMI Data Register.
- 218h for Long Duration Timer LS byte.
- 219h for Long Duration Timer Mid Low byte.
- 21Ah for Long Duration Timer Mid High byte.
- 21Bh for Long Duration Timer MS byte.
- 21Ch for Long Duration Timer Status & Control Register.
- 21Dh for Interrupt Configuration Register.
- 0CA9h for SMIC Data Register.
- 0CAAh for SMIC Control/Status Register.
- 0CABh for SMIC Flags Register.

9.1 Status & Control Register 0

This register is at I/O address 210h.

NOTE Bits 3 - 0 have different read and write functions.

7	6	5	4	3	2	1	0
PMC2	PMC1	ENUM# INTERRUPT ENABLE	FORCE SATELLITE	CPCI 66MHz/ DEVICE LOCK	PMC 66MHz/ DEVICE LOCK	PCI_P PRESENT/ DEVICE LOCK	RFU/ DEVICE LOCK

Read Bit 0: Reserved

Read Bit 1: CompactPCI PCI_PRESENT# Pin Status (Read Only)

This bit indicates whether or not the PCI bus is present at this backplane slot.

0 = PCI bus not present
1 = PCI bus present

Read Bit 2: Main PCI (PMC) Bus Frequency (Read Only)

0 = 33MHz
1 = 66MHz

Read Bit 3: CompactPCI Bus Frequency (Read Only)

0 = 33MHz
1 = 66MHz

Write Bits 3-0: Device lock (Write Only)

These bits control the Device Lock function. The device lock forces various status and control register bits to the clear (i.e. zero) state following a power-on or reset. To unlock the device, software should write 0X5h then 0XAh to this register.

Bit 4: Force Satellite Jumper (Read Only)

Used to force Satellite Mode irrespective of which backplane slot the board is plugged into.

0 = Normal Mode selection
1 = Force Satellite Mode

Bit 5: Enable Interrupt Generation from ENUM# (Read/Write)

0 = disable Interrupt generation
1 = enable Interrupt generation

Bit 6: PMC Site 1 PMC Mode 1 Status (Read Only)

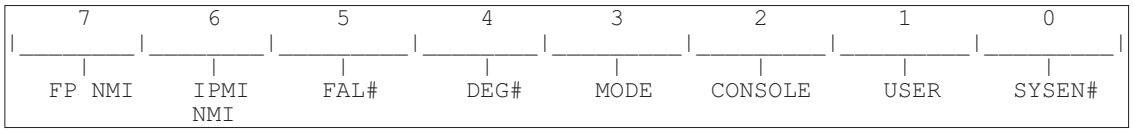
0 = PCI compliant module not fitted
1 = PCI compliant module fitted

Bit 7: PMC Site 2 PMC Mode 1 Status (Read Only)

0 = PCI compliant module not fitted
1 = PCI compliant module fitted

9.2 Status & Control Register 1

This register is at I/O address 211h.



Bit 0: CompactPCI SYSEN# Pin Status (Read Only)

This bit indicates whether or not the board is plugged into a System Controller slot.

0 = not System Controller
1 = System Controller

Bit 1: User Jumper (Read Only)

Available for user defined purposes in BIOS mode (but see Section 10.6). In CPSA (factory test) mode this jumper selects between MTH and Soak operation.

0 = MTH
1 = Soak

Bit 2: Console Jumper (Read Only)

Used to define the BIOS default standard input/output mode

0 = input/output via serial port
1 = input via keyboard/output via VGA adapter

Bit 3: Mode Jumper (Read Only)

Used to define the operating mode following a reset.

0 = BIOS operation
1 = CPSA operation

Bit 4: CompactPCI 'DEG#' Signal is the cause of NMI (Read Only)

0 = event has not occurred
1 = event has occurred

Bit 5: CompactPCI 'FAL#' Signal is the cause of NMI (Read Only)

0 = event has not occurred
1 = event has occurred

NOTE DEG# & FAL# only generate an NMI when first asserted. They will not generate another interrupt until they have cycled false then true again. Bits 4 & 5 can be used as monitoring bits for these signals as they may be asserted for some time. The clearing of these bits is PSU dependant and beyond the scope of this document.

Bit 6: IPMI is the cause of NMI (Read/Clear)

0 = event has not occurred
1 = event has occurred

Writing zero to this bit will clear it to zero, writing one will leave it unchanged.

Bit 7: Front Panel Switch is the cause of NMI (Read/Clear)

0 = event has not occurred
1 = event has occurred

Writing zero to this bit will clear it to zero, writing one will leave it unchanged.

9.3 Status & Control Register 2

This register is at I/O address 212h.

NOTE Bits 2 and 0 of this register are device locked.

7	6	5	4	3	2	1	0
EREADY	REV2	REV1	REV0	AUX V OK	SPEED STEP	RFU	ETH REAR

Bit 0: Ethernet Channel 0 Routing (Read/Write)

0 = Ethernet 0 routed to front panel RJ45
1 = Ethernet 0 routed to J3 connector

Bit 1: Reserved

Bit 2: SpeedStep

0 = low speed (battery optimized mode)
1 = high speed (performance mode)

This bit controls the logic that changes the processor operating frequency and voltage. This bit will only respond to the first write to this register following a power-on or reset. Subsequent writes to this register will not affect this bit.

NOTE This feature is reserved for use by the BIOS only. User software may read this bit (to determine the operating frequency) but may not change it.

Bit 3: Auxiliary Voltage Status

0 = auxiliary voltages (1.5V and 2.5V) not OK
1 = auxiliary voltages both OK

This bit reports the status of the auxiliary (1.5V and 2.5V) voltages used to power some of the on-board peripheral controllers.

Bit 6-4: Hardware Revision Strapping (Read Only)

000 = Rev A,
001 = Rev B etc…

Write Bit 7: PMC EREADY Detection (Write Only)

0 = read EREADY status
1 = read as 1

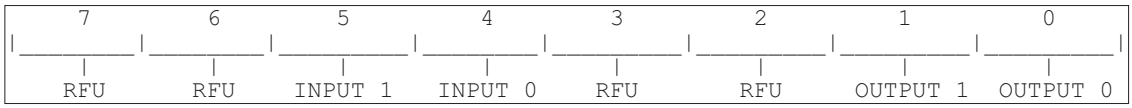
Read BIT 7: PMC EREADY Status (Ready Only)

0 = Processor PMC module(s) ready for enumeration
1 = Processor PMC module(s) not ready for enumeration

NOTE Early versions of the board did not implement PMC EREADY monitoring. On such boards Bit 7 always reads as 0.

9.4 General Purpose I/O Register

This register is at I/O address 213h.



Bits 1 - 0: General Purpose Outputs to Transition Module (Read/Write)

0 = set output line to 0
1 = set output line to 1

Bits 3 - 2: Reserved

Bits 5 - 4: General Purpose Inputs from Transition Module (Read Only)

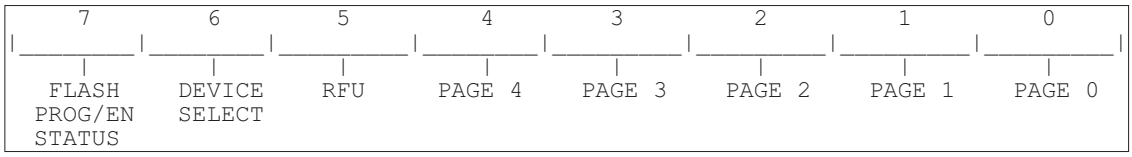
0 = input line is at 0
1 = input line is at 1

Bits 7 - 6: Reserved

NOTE A multiplexed serial I/O scheme is used to connect these register bits to the I/O pins on the Transition Module. Because of this, there is an output latency of 64µs and the maximum input frequency (to avoid losing input transitions) is approximately 8kHz.

9.5 Application Flash Paging Register

This register is at I/O address 214h. It controls the Application Flash devices.



Bits 4 - 0: Application Flash Page (Read/Write)

Pages are in the range 00h - 1Fh. Bit 5 is reserved for future 256Mbit devices.

Bit 5: Reserved

Bit 6: Flash Device Select (Read/Write)

0 = device 1
1 = device 2

Bit 7: Application Flash Program/Enable Status (Read Only)

0 = device busy
1 = device ready

9.6 Interrupt Control Register

This register is at I/O address 215h. It provides control over interrupts from the IPMI.

7	6	5	4	3	2	1	0
RFU	GPI INT FLAG	SMS_ATN INT FLAG	SMIC NOT BUSY INT FLAG	RFU	GPI INT ENA	SMS_ATN INT ENA	SMIC NOT BUSY INT ENA

Bit 0: SMIC Not Busy Interrupt Enable (Read/Write)

This bit allows an interrupt to be generated when the P89C664 microcontroller clears the SMIC BUSY flag.

0 = interrupt disabled
1 = interrupt enabled

Bit 1: SMS_ATN Interrupt Enable (Read/Write)

This bit allows an interrupt to be generated when the P89C664 microcontroller sets the SMIC SMS_ATN bit.

0 = interrupt disabled
1 = interrupt enabled

Bit 2: General Purpose Interrupt (GPI) Interrupt Enable (Read/Write)

This bit allows an interrupt to be generated when the P89C664 microcontroller sets the GPI INT FLAG bit.

0 = GPI not enabled
1 = GPI enabled

Bit 3: Reserved

Bit 4: SMIC Not Busy Interrupt Flag (Read/Clear)

0 = event has not occurred
1 = event has occurred

Writing zero to this bit will clear it to zero, writing one will leave it as is.

Bit 5: SMS_ATN Interrupt Flag (Read/Clear)

0 = event has not occurred
1 = event has occurred

Writing zero to this bit will clear it to zero, writing one will leave it as is.

Bit 6: General Purpose Interrupt (GPI) Interrupt Flag (Read/Clear)

The IPMI microcontroller sets this bit to request an interrupt. If the GPI Interrupt Enable bit is also set, an interrupt (INT 5) will be generated. Note that if the interrupt is not required, the microcontroller can use this bit to signal status to the processor.

0 = no interrupt request
1 = interrupt request

Writing zero to this bit will clear it to zero, writing one will leave it as is.

Bit 7: Reserved

9.7 RMI Registers

The PP 110/01x board includes hardware which implements a Concurrent Technologies proprietary set of registers termed the Resource Manager Interface (RMI). These registers provide several diagnostic and other features used at the factory to test the boards.

One commonly-required feature of the board which may be used by application software is control of the red USER LED. This is provided in an RMI register, but the RMI register list can vary with board revisions. The position of the control bit within the correct register does not change. To access the USER LED control bit, the first step is to determine the location of the correct register. The program fragment below shows some functions which will do this.

```
/*
 * constants (I/O addresses and register fields)
 */

#define rmiStatus           0x0216
#define rmiBusy                0x01
#define rmiAddress          0x0216
#define rmiData             0x0217

/*
 * functions to read / write bytes using the Resource Manager Interface
 */

/*****************************************************************************
 * rmiRead - reads a byte from the specified address in the Resource Manager
 *           Interface.
 *
 * RETURNS: data from the specified address in the Resource Manager Interface
 */

unsigned char rmiRead (unsigned char Address)
{
    /*
     * monitor the "rmiBusy" bit of "rmiStatus" to determine if the Resource
     * Manager Interface is busy and if so wait until it is no longer busy.
     */

    while ((inbyte (rmiStatus) & rmiBusy) == rmiBusy)
        ; /* wait and do nothing ... */
    outbyte (rmiAddress, Address);

    /*
     * monitor the "rmiBusy" bit of "rmiStatus" to determine if the Resource
     * Manager Interface is busy and if so wait until it is no longer busy.
     */

    while ((inbyte (rmiStatus) & rmiBusy) == rmiBusy)
        ; /* wait and do nothing ... */
    return inbyte (rmiData);
}

/*****************************************************************************
 * rmiWrite - writes the specified byte to the specified address in the
 *            Resource Manager Interface.
 *
 * RETURNS: n/a
 */

void rmiWrite (unsigned char Address, unsigned char Data)
{
```

```
    /*
     * monitor the "rmiBusy" bit of "rmiStatus" to determine if the Resource
     * Manager Interface is busy and if so wait until it is no longer busy.
     */

    while ((inbyte (rmiStatus) & rmiBusy) == rmiBusy)
        ; /* wait and do nothing ... */
    outbyte (rmiAddress, Address);

    /*
     * monitor the "rmiBusy" bit of "rmiStatus" to determine if the Resource
     * Manager Interface is busy and if so wait until it is no longer busy.
     */

    while ((inbyte (rmiStatus) & rmiBusy) == rmiBusy)
        ; /* wait and do nothing ... */
    outbyte (rmiData, Data);
}

/*
 * Function to locate specific records in the Resource Manager Interface
 */

/*****************************************************************************
 * rmiFindRecord - This function finds the specified record in the Resource
 *                 Manager Interface.
 *
 * RETURNS: The offset of the specified record in the Resource Manager
 *          Interface (0 is returned if the record is not found or the END
 *          record is specified.
 */

unsigned int rmiFindRecord (unsigned char RecordType)
{
    unsigned  int Address;
    unsigned char Record;

    Address = 32; /* first record is after the header which is 32 bytes long */
    Record = rmiRead (Address);

    while ((RecordType != 0xFF) && (Record != RecordType))
    {
        Address += rmiRead (Address + 1) + 2;
        Record   = rmiRead (Address);
    }
    return (Record != 0xFF) ? Address : 0;
}


/*****************************************************************************
 * vTurnLEDOn/vTurnLEDOff - sets or clears the USER LED control bit in the RMI
                            register.
 *
 * RETURNS: n/a
 */
unsigned int wLEDRegister = 0;


void vTurnLEDOn (void)
{
    if (wLEDRegister == 0)
    {
        wLEDRegister = rmiFindRecord (0xFE);
        if (wLEDRegister == 0)
            return;
```

```
            wLEDRegister += 8;
        }

        rmiWrite (wLEDRegister, rmiRead (wLEDRegister) | 0x04);
    }


    void vTurnLEDOff (void)
    {
        if (wLEDRegister == 0)
        {
            wLEDRegister = rmiFindRecord (0xFE);
            if (wLEDRegister == 0)
                return;
            wLEDRegister += 8;
        }

        rmiWrite (wLEDRegister, rmiRead (wLEDRegister) & ~0x04);
    }
```

9.8 Long Duration Timer / Periodic Interrupt Timer

The Long Duration Timer (LDT) consists of a 32-bit free running counter with a 32-bit holding register and a status & control register. It may be used by user software to timestamp events to a resolution of 1 microsecond. The counter bytes are laid out in little-endian format to permit multi-byte read/write operations. The status & control register controls the operation of the LDT.

A 32-bit holding register is provided to ensure stable count values are read. Read operations return the holding register byte values. A read operation on the low byte of the counter causes the count value to be transferred to the holding register. Hence, the low byte should be read first to ensure a stable count value.

The counter may be preset by writing to the registers. The counter bytes may be written independently. The counter should be stopped before writing to it or the outcome may be indeterminate. The counter registers are cleared at power-on, but not by subsequent reset operations. If necessary, the LDT can be cleared by writing zero to all four counter bytes.

An interrupt may be generated when the counter rolls over (from FFFFFFFFh to zero). This occurs approximately every 72 minutes (1MHz clock).

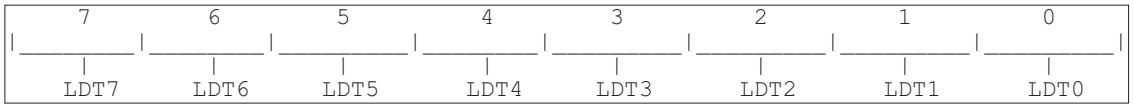
The LDT doubles as a simple Periodic Interrupt Timer (PIT). It offers 7 fixed interrupt rates, namely: 100, 200, 500, 1,000, 2,000, 5,000 and 10,000Hz (1MHz clock). The mode / interrupt rate is set by three bits in the LDT status & control register.

NOTE If the LDT clock frequency is not 1MHz (i.e. HF Clock = 4MHz), the above rates should be scaled accordingly.

In PIT mode, the counter counts up to a pre-determined maximum value and then goes back to zero. To ensure a full first interval, the low and mid-low bytes of the counter should be cleared before the counter is started.

9.8.1 LDT / PIT Low Byte Register

This register is at I/O address 218h.



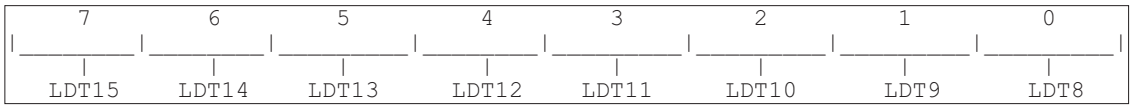
Bits 7 - 0: Low Byte of LDT / PIT (Read/Write)

Reading this register causes the current value of the LDT to be transferred to a holding register. This allows a stable 4-byte count to be read. The low byte of the holding register is returned by the read.

Writing to this register loads a value into the low byte of the LDT / PIT counter. The counter should be stopped when writing or the result will be indeterminate.

9.8.2 LDT / PIT Mid-low Byte Register

This register is at I/O address 219h.



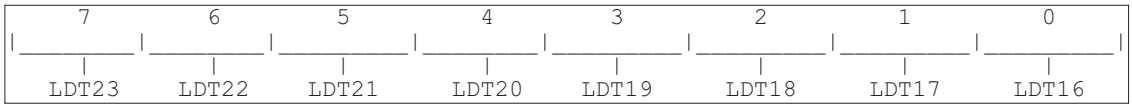
Bits 7 - 0: Mid-Low Byte of LDT / PIT (Read/Write)

Reading this register returns the mid-low byte of the holding register.

Writing to this register loads a value into the mid-low byte of the LDT / PIT counter. The counter should be stopped when writing or the result will be indeterminate.

9.8.3 LDT Mid-high Byte Register

This register is at I/O address 21Ah.



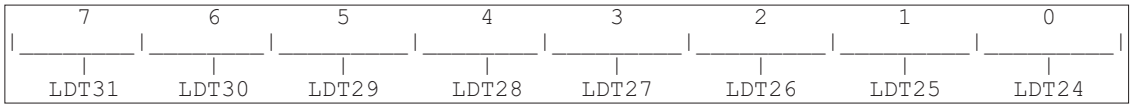
Bits 7 - 0: Mid-High Byte of LDT (Read/Write)

Reading this register returns the mid-high byte of the holding register.

Writing to this register loads a value into the mid-high byte of the LDT counter. The counter should be stopped when writing or the result will be indeterminate.

9.8.4 LDT High Byte Register

This register is at I/O address 21Bh.



Bits 7 - 0: High Byte of LDT (Read/Write)

Reading this register returns the high byte of the holding register.

Writing to this register loads a value into the high byte of the LDT counter. The counter should be stopped when writing or the result will be indeterminate.

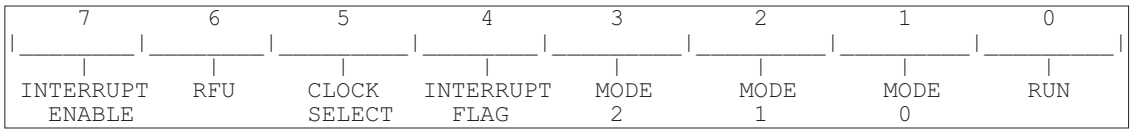
9.8.5 LDT / PIT Status & Control Register

This register is at I/O address 21Ch. It controls the operation of the LDT.

The LDT clock frequency is selectable from two sources, namely the Local Super I/O HF Clock output or LF Clock output. The HF Clock is derived from an internal 48MHz signal. The frequency is selectable via a programmable divider and is divided by 4 externally before being fed to the LDT. The LF Clock is derived from the 32kHz RTC clock. The following clock frequencies are available:

HF Clock / 4: 12MHz, 6MHz, 4MHz, 3MHz, 2MHz, 1.5MHz, 1MHz, 0.75MHz.
LF Clock: 32.768kHz, 1Hz.

NOTE The BIOS will configure the Local Super I/O and LDT for a 1MHz clock frequency.



Bit 0: LDT / PIT Run (Read/Write)

This bit controls whether the LDT / PIT runs or is stopped.

0 = Stop (default)
1 = Run.

Bits 3 - 1: LDT / PIT Mode (Read/Write)

These bits set the mode of the timer as follows:

0 0 0 = LDT
0 0 1 = PIT 100Hz
0 1 0 = PIT 200Hz
0 1 1 = PIT 500Hz
1 0 0 = PIT 1,000Hz
1 0 1 = PIT 2,000Hz
1 1 0 = PIT 5,000Hz
1 1 1 = PIT 10,00Hz

NOTE The above PIT rates require LDT CSR bit 5 = 0 and the HF Clock output from the Local Super I/O controller to be 4MHz. This results in an LDT clock frequency of 1MHz.

Bit 4: LDT / PIT Interrupt Flag (Read/Clear)

The interrupt flag is active whenever the timer is running. It is set if the LDT RUN bit is set AND either the LDT rolls over or the PIT interval expires. It can be cleared by writing to the register with a zero in its bit position. This should be done in the LDT / PIT interrupt service routine.

0 = LDT / PIT interrupt has not occurred
1 = LDT / PIT interrupt has occurred

Bit 5: LDT / PIT Clock Source (Read/Write)

This bit selects the clock source for the LDT / PIT as follows:

0 = Super I/O HF Clock / 4 (1MHz)
1 = Super I/O LF Clock (32.768kHz)

Bit 6: Reserved

Bit 7: LDT / PIT Interrupt Enable

This bit enables the LDT / PIT interrupt output. It has no effect on the interrupt flag.

0 = interrupt disabled
1 = interrupt enabled

9.8.6 Programming the LDT/PIT

The following code fragments illustrate how the system software, by using the on-board hardware, can create accurate time delays and measure elapsed times, accurate to 1 s, irrespective of the CPU's operating frequency.

The LDT and PIT control registers and operational modes are defined thus:

```
#define TIMER_BYTE_0            (0x0218U)
#define TIMER_BYTE_1            (0x0219U)
#define TIMER_BYTE_2            (0x021AU)
#define TIMER_BYTE_3            (0x021BU)
#define CONTROL_STATUS          (0x021CU)

#define INTERRUPT_MASK              (0x10U)
#define INTERRUPT_ENABLE            (0x10U)
#define INTERRUPT_DISABLE           (0x00U)
#define INTERRUPT_SET               (0x10U)
#define INTERRUPT_RESET             (0x00U)

#define TIMER_ROLLOVER              (0x10U)

#define MODE_MASK                   (0x0EU)
#define MODE_PIT_10000Hz            (0x0EU)
#define MODE_PIT_5000Hz             (0x0CU)
#define MODE_PIT_2000Hz             (0x0AU)
#define MODE_PIT_1000Hz             (0x08U)
#define MODE_PIT_500Hz              (0x06U)
#define MODE_PIT_200Hz              (0x04U)
#define MODE_PIT_100Hz              (0x02U)
#define MODE_LDT                    (0x00U)

#define MODE_RUN_MASK               (0x01U)
#define MODE_RUN_GO                 (0x01U)
#define MODE_RUN_STOP               (0x00U)
```

The following code fragment illustrates how a simple delay of 10ms is implemented.

```
outbyte (CONTROL_STATUS, MODE_RUN_STOP);
outbyte (TIMER_BYTE_0, 0);
outbyte (TIMER_BYTE_1, 0);
outbyte (TIMER_BYTE_2, 0);
outbyte (TIMER_BYTE_3, 0);
outbyte (CONTROL_STATUS, MODE_PIT_100Hz | MODE_RUN_GO);

/* wait until the PIT rolls over ... */

while (inbyte (CONTROL_STATUS) & TIMER_ROLLOVER) == 0)
    ; /* do nothing ... */

/* reset the PIT "rollover" flag ... */

outbyte (CONTROL_STATUS, MODE_RUN_STOP);
```

It is possible to implement delays of 5ms, 2ms, 1ms, 500 s, 200 s and 100 s by utilizing other PIT modes.

The PIT can generate an interrupt whenever the PIT rolls over. The system programmer must initialize the interrupt vector, enable PIC interrupts, etc. The following code fragment shows the basic interrupt handling function.

```
static volatile signed long int dCounter;

#pragma interrupt (vInterruptHandler)
static void far vInterruptHandler (void)
{
    /*
     * clear the source of the interrupt by resetting the rollover
     * flag, thus:
     */

    outbyte (CONTROL_STATUS, inbyte (CONTROL_STATUS) & ~INTERRUPT_MASK);

    /*
     * perform the relevant actions to acknowledge the interrupt
     * in the PIC, etc ...
     */

    dCounter--;
}
```

The following code fragment used in conjunction with the previous code fragment illustrates another method of implementing a timed delay. The dCounter variable is declared to be volatile which prevents any C compilers, which conform to the ANSI standard, from optimizing accesses to the dCounter variable.

```
outbyte (CONTROL_STATUS, MODE_RUN_STOP);
outbyte (TIMER_BYTE_0, 0);
outbyte (TIMER_BYTE_1, 0);
outbyte (TIMER_BYTE_2, 0);
outbyte (TIMER_BYTE_3, 0);
outbyte (CONTROL_STATUS, MODE_PIT_100Hz | MODE_RUN_GO);
dCounter = 500; /* 500 * (1 / 100) == 5 seconds */

/*
 * install the interrupt for the PIT counter, modify the
 * PIC settings, etc. and ensure interrupts are enabled.
 */

while (dCounter > 0)
    ; /* do nothing ... */

outbyte (CONTROL_STATUS, MODE_RUN_STOP);
```

The following code fragment uses the LDT to measure the elapsed time to a resolution of 1 s. In this example, the LDT is zeroed at the start of the test and so there is no need to subtract the LDT's initial value from its final value.

```
static UINT32 dElapsedTime;

outbyte (CONTROL_STATUS, MODE_RUN_STOP);
outbyte (TIMER_BYTE_0, 0);
outbyte (TIMER_BYTE_1, 0);
outbyte (TIMER_BYTE_2, 0);
outbyte (TIMER_BYTE_3, 0);
outbyte (CONTROL_STATUS, MODE_LDT | MODE_RUN_GO);

/*
 * perform action to be timed ...
 */
```

```
outbyte (CONTROL_STATUS, MODE_STOP);
dElapsedTime  =  (UINT32) inbyte (TIMER_BYTE_0);
dElapsedTime |= ((UINT32) inbyte (TIMER_BYTE_1)) << 8;
dElapsedTime |= ((UINT32) inbyte (TIMER_BYTE_2)) << 16;
dElapsedTime |= ((UINT32) inbyte (TIMER_BYTE_3)) << 24;
printf ("Elapsed time = %u.%06u seconds\n",
        dElapsedTime / 1000000U, dElapsedTime % 1000000U);
```

The `TIMER_BYTE_0, TIMER_BYTE_1, TIMER_BYTE_2` and `TIMER_BYTE_3` control registers are at successive addresses and form a 32-bit register in “little endian” format. It is possible to read and write the timer’s value in a single 32-bit I/O operation. For example, to read the timer’s value, the following C statement suffices.

```
DCounterValue = inlong (TIMER_BYTE_0);
```

9.8.7 Interrupt Configuration Register

This register is at I/O address 21Dh.

NOTE This register is not present in early versions of the board. Software can toggle bit 0 and read it back to test if this register is present. On early versions of the board this register will read as all 1's.

7	6	5	4	3	2	1	0
RFU	RFU	PMC INTERRUPT MODE1	PMC INTERRUPT MODE0	M66EN INTERRUPT FLAG	M66EN INTERRUPT ENABLE	ENUM# STATUS	ENUM# ROUTING

Bit 0: CompactPCI ENUM# Interrupt Routing (Read/Write)

This bit selects which interrupt the CompactPCI ENUM# signal is routed to. This interrupt is only relevant when the board is System Controller.

0 = NMI
1 = PCI bus interrupt

NOTE Bit 5 of Status and Control Register 0 enables this interrupt.

Bit 1: CompactPCI ENUM# Pin Status (Read Only)

0 = ENUM# is not asserted
1 = ENUM# is asserted

Bit 2: CompactPCI M66EN High-Low Transition Interrupt Enable (Read/Write)

When the board is System Controller it monitors the CompactPCI M66EN signal for high to low transitions during normal operation. A PCI bus interrupt may be generated if such a transition occurs. This bit enables that interrupt. The setting of bit 0 has no effect on this interrupt.

0 = transition interrupt is disabled
1 = transition interrupt is enabled

Bit 3: CompactPCI M66EN High-Low Transition Flag (Read Only)

This bit reports whether or not there has been a high to low transition on the CompactPCI M66EN signal.

0 = a high to low transition has not occurred
1 = a high to low transition has occurred

Bit 3: CompactPCI M66EN High-Low Transition Flag (Write Only)

This flag can be cleared by writing to the register with a zero in this bit position

0 = clear flag
1 = leave flag as is

Bit 4: PMC Interrupt Mode 0 (Read/Write)

Bit 5: PMC Interrupt Mode 1 (Read/Write)

The PMC Interrupt Mode bits support a customer application in which the Ethernet interrupts are routed to the PMC sites and are handled by one or two processor PMC modules. The settings are:

Bits 5, 4 Mode

0 0 Normal

0 1 Ethernet interrupts for channels 0 and 1 to PMC1 INTB and INTD respectively (PMC2 INTB and INTD are not available)

1 0 Ethernet interrupts for channel 0 and 1 to PMC2 INTB and INTD respectively (PMC1 INTB and INTD are not available)

1 1 Ethernet interrupt for channel 0 to PMC1 INTB and Ethernet interrupt for channel 1 to PMC2 INTB (PMC1 INTD and PMC2 INTD are not available)

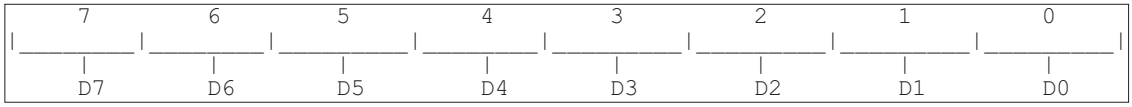
Bits 7-6: Reserved

9.9 IPMI SMIC Interface

The IPMI hardware on this board is accessed by the local CPU using the SMIC interface (see Chapter 7 for an explanation of these terms and of the purpose of IPMI). The following sections outline the register contents, and example code for using this interface is provided in Section 7.5.

9.9.1 SMIC Data Register

This register is at I/O address 0CA9h. It is dual-ported. Both the CPU and the P89C664 microcontroller can read it or write to it. The SMIC software protocol ensures that no contentions will occur.



Bits 7-0: SMIC Data Value (Read/Write)

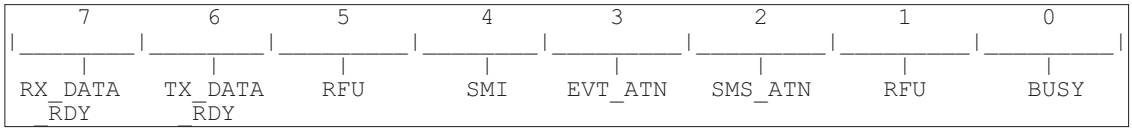
9.9.2 SMIC Control/Status Register

This register is at I/O address 0CAAh. It is dual-ported. Both the CPU and the P89C664 microcontroller can read it or write to it. The SMIC software protocol ensures that no contentions will occur.

Bits 7-0: SMIC Control/Status Value (Read/Write)

9.9.3 SMIC Flags Register

This register is at I/O address 0CABh. It reports the status of various SMIC flag bits.



Bit 0: BUSY (Read/Set)

Bit 1: Reserved

Bit 2: SMS_ATN (Read Only)

Bit 3: EVT_ATN (Read Only)

Bit 4: SMI (Read Only)

Bit 5: Reserved

Bit 6: TX_DATA_RDY (Read Only)

Bit 7: RX_DATA_RDY (Read Only)

9.10 P.O.S.T. LED / SPEAKER

The P.O.S.T. LED is controlled via the speaker port. The P.O.S.T. LED replaces a PC speaker and is programmed in the same way a speaker would be programmed. The board also outputs the speaker port via a high current open collector driver on the CompactPCI J5 connector for connection to an external speaker if required.

9.11 PORT 80

A header has been provided for monitoring data written to I/O Port 80. The PC BIOS writes status bytes to Port 80 that indicate a boot progress status and/or highlight any faults found. Data written to this port can be monitored using a Logic State Analyzer (LSA) or seven segment hexadecimal displays. See Section A.5.10 for details of the connector used for this port.

After boot-up this port can be used to monitor other status bytes written to port 80, which can be useful for debug purposes.

The PP 110/01x board is fitted with PC BIOS firmware that performs many of the functions of a standard desktop PC. It also includes additional features specifically tailored for the CompactPCI bus environment. In addition to the core BIOS firmware, the board is fitted with BIOS Extensions for remote bootload capability via either of the on-board Ethernet channels. To improve the flexibility of the board, some of these features may be selectively enabled or disabled by an operator using BIOS setup menus. Many of the features provided by the PC BIOS are unlikely to be adjusted by the user, but there are several options which many users will find helpful. Some of these are already referenced in other sections of this manual, but the remainder of this chapter will describe some other commonly-used options. More information about each of the options available is provided in the Help box of the BIOS setup menus.

10.1 Entering the PC BIOS

The startup mode of the board may be selected using the MODE jumper, but can be either of the following: PC BIOS mode (the factory default setting), which generally follows the behavior of a desktop PC, and CPSA mode (a flexible testing mode primarily for use at the factory), which can be used for system or board testing. CPSA mode operation and features are not described in this manual. Figure 10-1 shows the location of the jumper on the board and its settings.

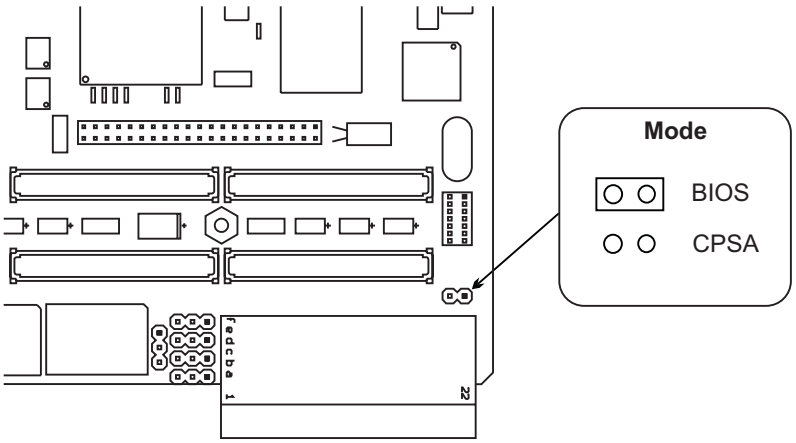


Figure 10-1 Mode Jumper

CPSA mode may be exited either by operator command, or by allowing the board to proceed through the CPSA startup sequence without intervention. In either case, the board will enter PC BIOS mode and continue as if this mode had been selected with the jumper. When the board is reset, it will generally restart in the operating mode selected by the MODE jumper. However, a reset caused by a keyboard <CTRL-ALT-DEL> keystroke combination, or by a programmed reset using one of several different I/O access sequences, will only cause a PC BIOS restart. A complete board or system reset (using the front panel switch or through the CompactPCI bus PCI_RST signal) will cause the board to restart in the mode selected by the MODE jumper setting.

Operator communication with the PC BIOS is usually through a VGA display connected to the on-board graphics interface and a separate keyboard. This can be reconfigured with a board jumper to use a serial terminal connected to the COM1 port. Section 6.1 describes the location and settings for this jumper. A VT100-compatible serial terminal or emulator program should be used. By default the serial line is programmed to operate at 9600 Baud with 8 data bits, 1 stop bit and no parity (8N1). There is no flow control. For fast terminals, the baud rate can be increased via the **`Serial Console Baud Rate`** field of the **`Main`** Setup menu.

To allow correct operation of the BIOS Setup menus via the serial interface, the terminal or emulator program should be configured to automatically wrap long lines.

10.2 The PC BIOS Startup Sequence

When the board starts up without operator intervention, it will run a basic Power-On Self-Test (POST) sequence, including ECC DRAM initialization and a DRAM test. The full DRAM test will be omitted on subsequent restarts if the BIOS configuration settings have not been changed. Once the DRAM test has completed, the board will try to bootload application software from any attached mass storage medium or through one or both of the Ethernet interfaces.

WARNING If the V(I/O) supply voltage is not wired on the backplane the CompactPCI bridge device will not operate correctly. As a result, the CompactPCI bus device scan will cause the board to hang.

When the PC BIOS starts after changing the battery, losing battery power or after using the CMOS CLEAR jumper, it may report a CMOS Checksum Error or some other problem. This will be following by a prompt to the operator to press <F1> to continue or <F2> to enter Setup mode. If no key is pressed within approximately five seconds, the PC BIOS will continue with its normal startup sequence. It will also re-calculate the CMOS Checksum to prevent this error occurring again at a subsequent restart.

Pressing the <F2> key at any time during the PC BIOS startup sequence will result in the BIOS Setup menu being entered. The Setup menu is quite extensive, and is provided with context-sensitive help information which is displayed in the right-hand panel on screen.

NOTE When the <F2> key is pressed, a few seconds may elapse before the BIOS Setup menu appears. The PC BIOS will always run BIOS Extensions for any PMC modules it detects before responding to the keypress.

10.3 Boot device selection

The order in which the PC BIOS searches for a bootable medium is pre-configured but may be altered by the operator using the **Boot** setup menu. When the order is changed using this menu it will be retained in non-volatile memory so that the order is maintained after a restart. It is also possible to specify a one-time override of the boot device when the board starts, by pressing the <ESC> key. This will result in a pop-up menu appearing. The appropriate boot device may be selected from a list by using the cursor keys and pressing <ENTER>, but this is not retained in non-volatile memory, so the correct device must be re-selected if necessary at a subsequent restart.

NOTE When the <ESC> key is pressed, a few seconds may elapse before the boot device selection menu appears. The PC BIOS will always run BIOS Extensions for any PMC modules it detects before responding to the keypress.

The on-board PMC Sites and Ethernet channels require their BIOS Extension firmware to be enabled before they can be used as boot devices. BIOS setup options are provided to control whether or not the board runs the BIOS Extensions for the Ethernet channels or the on-board PMC sites. The **Option ROM Scan** field of the appropriate device menu must be used to enable or disable the BIOS Extension. The device menus are accessible from:

Ethernet channel 0	**Advanced | PCI Device Configuration | Ethernet Channel 0**
Ethernet channel 1	**Advanced | PCI Device Configuration | Ethernet Channel 1**
On-board PMC site 1	**Advanced | PCI Device Configuration | PMC Site 1**
On-board PMC site 2	**Advanced | PCI Device Configuration | PMC Site 2**

Booting via the Ethernet channels can use either the Pre-Boot Execution (PXE) protocol, or BOOTP/TFTP protocol. By default PXE is used, but this may be changed using the setup option for **Main | Boot Options | Network Book Protocol**. Further information on PXE can be obtained from the Intel web site at **http://developer.intel.com**. Further information on BOOTP/TFTP can be obtained from the Etherboot web site at **http://etherboot.sourceforge.net**.

NOTE The BIOS has limited space available for Extension ROMs. If a PMC module containing extension firmware is fitted to the board, it may be necessary to disable one or more of the on-board firmware extensions before the PMC firmware can be loaded.

10.4 PCI Bus Resource Management

The bus structure of the PP 110/01x is quite complex. There are two on-board PCI busses: a 64-bit bus which connects to the Ethernet controller, the PMC sites and the CompactPCI bridge chip, and a second 32-bit bus which connects to the remaining on-board peripherals (CSB5, graphics). Associated with these busses are a number of interrupt lines.

The 32-bit bus operates at 33MHz with 3.3V signaling levels. The 64-bit bus can operate at 33MHz with either 3.3V or 5V signaling levels. It normally runs at 66MHz, but may be slowed to 33MHz if a 5V only or 33MHz PMC module is fitted.

10.4.1 PCI Resource Allocation

The PC BIOS initializes all devices on the local PCI bus, and allocates appropriate memory address ranges, I/O address ranges, and interrupt routings for all these devices. This process is automatic as part of the BIOS "Plug-and-play" setup. When the board is System Controller it will also allocate memory, I/O or interrupt resources to devices found on the CompactPCI bus. The ServerWorks chipset allows for a flexible allocation of many PCI bus interrupts to the available interrupt inputs on the PC-compatible interrupt controllers provided on the board. The PC BIOS uses this feature to program default settings which it considers appropriate for the combination of on-board devices and any device fitted to the PMC site. In some configurations, depending on the operating system being used and the capability of the relevant device drivers, it may be necessary for the user to modify this default configuration, to minimize the sharing of interrupt lines. The PC BIOS Setup screen for **`Advanced | PCI Device Configuration`** allows this.

This screen allows the user to override the PC BIOS default selections for interrupt allocation, but care must be taken when doing this to avoid conflicts which may result in operating system or even BIOS "crashes". To allow maximum flexibility of choice for the user, the PC BIOS performs limited checks on the user's interrupt allocation. In the event that there is a problem, it may be necessary to clear the CMOS memory (see Section 2.7), or even to reset the Extended System Configuration Data via the **`Reset Configuration Data`** field of the BIOS Setup screen for **`Advanced`** configuration settings. The PC BIOS does not allow the user to override the allocation of memory and I/O address ranges.

NOTE When reallocating interrupts using the BIOS Setup screens, try to avoid allocating the PMC interrupts to ones also allocated to other devices. This sharing of interrupts can cause problems with some operating systems where device drivers do not correctly handle shared interrupts.

Table 10-1 lists the configurable interrupts for this board. The actual allocation of PCI bus interrupts to available interrupt controller inputs will depend on both the default “Plug-and-play” settings programmed by the PC BIOS, and the way in which the user has overridden them using the Setup screens. When more than one PCI bus interrupt is routed to the same interrupt controller input, that input will remain active while any of the sources connected to it are active.

CompactPCI INTA
CompactPCI INTB
CompactPCI INTC
CompactPCI INTD
PMC 1 INTA
PMC 1 INTB
PMC 1 INTC
PMC 1 INTD
PMC 2 INTA
PMC 2 INTB
PMC 2 INTC
PMC 2 INTD
Ethernet Channel 0
Ethernet Channel 1
VGA Controller
CompactPCI Bridge
CompactPCI ENUM#

Table 10-1 Configurable PCI Bus Interrupts

NOTE CompactPCI Interrupts INTA - INTD are only relevant when the board is System Controller.

NOTE The CompactPCI Bridge interrupt is only relevant when the board is a peripheral.

10.4.2 PCI Device IDs

Each PCI bus, and each device on an individual PCI bus, has a unique ID. For the PP 110/01x, the bus and device IDs are listed in Table 10-2. The ServerWorks chipset includes two PCI bus bridges to interface to the 64-bit and 32-bit on-board PCI busses, and these bridges are identified by the same PCI device ID but with different function codes.

PCI Bus Number	PCI Device ID	PCI Function Code	Device Name Description
0	0	0	Bridge to PCI bus 0 (32-bits)
0	0	1	Bridge to PCI bus 1 (64-bits)
0	3	0	VGA Controller
0	15	0	South Bridge
0	15	1	EIDE Controller
0	15	2	USB Controller
0	15	3	LPC Bus Controller
0	15	4	XIOAPIC0 (interrupt controller)
0	15	5	XIOAPIC1 (interrupt controller)
0	15	6	XIOAPIC2 (interrupt controller)
1	5	0	CompactPCI bridge
1	6	0	PMC Site 1 Primary Function
1	7	0	PMC Site 2 Primary Function
1	8	0	Ethernet Channel 0
1	8	1	Ethernet Channel 1
1	10	0	PMC Site 1 Secondary Function
1	11	0	PMC Site 2 Secondary Function

Table 10-2 PCI Device Numbers

10.5 CompactPCI Bridge Configuration

The BIOS provides Setup menus that allow configuration of certain features of the HiNT® HB6 CompactPCI Bridge, these can be found under the **CompactPCI** top level menu. The fields available via Setup depend on the board's operating mode.

10.5.1 System Controller Mode

In System controller mode only one Setup option is provided. This allows the user to specify whether the CompactPCI ENUM interrupt drives NMI or a selectable IRQ via a PCI bus interrupt (Section 10.4.1 describes how to select the IRQ).

10.5.2 Satellite Mode

In Satellite mode the board does not connect to the CompactPCI interface, therefore no Setup menu is provided.

10.5.3 Peripheral Mode

When the board is configured for peripheral mode operation, the BIOS provides two sub-menus for configuring the upstream and downstream windows of the HB6 bridge.

10.5.3.1 Upstream Windows

Upstream windows give the peripheral mode PP 110/01x board access to resources on the CompactPCI bus. The HB6 bridge provides three upstream windows: upstream window 0 can map either memory or I/O addresses; windows 1 and 2 are for memory addresses only.

The BIOS Setup menu allows the user to specify the size of each window and the CompactPCI address that will be accessed via the window. The local address of the window is configured automatically by the BIOS.

NOTE The local address of upstream windows is not fixed and may change if the board configuration is modified. Application software should always read a window's base address from PCI configuration space.

10.5.3.2 Downstream Windows

Downstream windows give devices on the CompactPCI bus access to resources on the peripheral mode PP 110/01x board. The HB6 bridge provides three downstream windows: downstream window 0 can map either memory or I/O addresses; windows 1 and 2 are for memory addresses only.

The BIOS allows the user to specify the size of each window and the local address that will be accessed via the window. The System Controller assigns the window's address on the CompactPCI bus.

10.5.4 Peripheral Mode Window-Size Limitations

The HB6 Bridge forms PCI addresses by concatenating the least significant bits from the CPU generated address and the most significant bits from the translation base address; the contribution from each part is fixed and depends on the window type. However, the BIOS always aligns base addresses according to their resource size, to achieve optimal packing.

If a window is defined to be smaller than the translation base address granularity, the BIOS assigned base address may result in configuration where offset 0h into the resource window does not map to offset 0h from the translation base address. The example below illustrates this point:

Configured I/O window size:	256 bytes	
HB6 I/O window granularity:	4096 bytes	
BIOS assigned base address:	12345600h	(i.e. 256 byte aligned)
Translation base address:	ABCDE000h	(i.e. 4096 byte aligned)
Address generated by HB6:	ABCDE6xxh	(xx = offset into window)

In this example, accesses to offset 0h into the window result in accesses to offset 600h from the translated base address.

10.5.4.1 I/O Windows

For I/O windows, the translation base address granularity, and hence minimum practical size, is 4 Kbytes.

10.5.4.2 Memory Mapped Windows

For memory mapped windows, the translation base address granularity, and hence minimum practical size, is 1 Mbyte.

10.6 User Selectable NVRAM Defaults

The BIOS provides a facility through which the user can save preferred setup option settings to Flash memory (NVRAM). Then, if the BIOS detects that the contents of NVRAM is corrupt, the user can elect to restore the contents from the saved settings, rather than loading factory configured defaults. This facility also allows the board to operate without fitting the battery, but with NVRAM settings different to the factory defaults.

The **`Save User Defaults`** option in the BIOS Setup **`Exit`** menu is used to write the current NVRAM settings to Flash memory. When settings are saved to Flash memory, the current BIOS date and time will also be stored; this allows the board to start operating with an appropriate date and time.

NOTE Saving settings to Flash memory may take number of seconds to complete.

The NVRAM restore feature is controlled by the User/Test Jumper (see Figure 10-2). When the User/Test Jumper is in the “Restore NVRAM/MTH” position and the BIOS detects that the NVRAM contents are corrupt, NVRAM settings will be restored from Flash memory (provided that valid settings have been saved). When the switch is in the Disable/Soak position, the factory-configured defaults will be used.

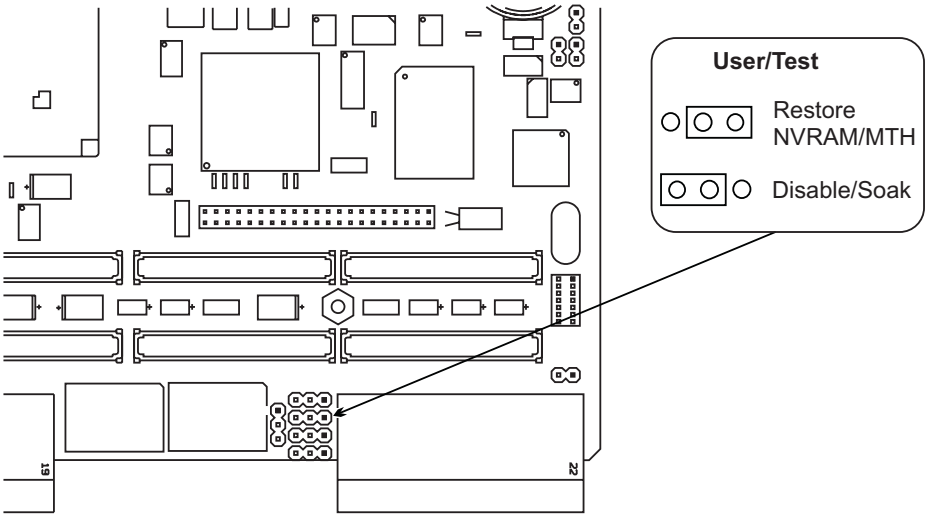


Figure 10-2 User/Test Jumper

To configure and save preferred NVRAM settings:

- Configure the BIOS in the usual manner via the Setup menus
- Reboot the board and allow it to proceed through to the bootloader without error
- Reboot the board and enter Setup
- From the **`Exit`** menu, select **`Save User Defaults`**
- Set the User/Test Jumper to enable the restore feature

NOTE The User/Test has a different function if the board is configured for CPSA (Factory Test) mode. If the board is ever returned to the factory for repair, this jumper will be changed and the User Selectable NVRAM settings may be affected.

Specifications

A.1 Functional Specification

Processor:	• 800MHz or 1.2GHz Pentium III-M with 32 Kbyte Level 1 cache.
Level 2 Cache:	• 512 Kbytes on-die RAM operating at core frequency.
Memory:	• 512 Kbytes Flash EPROM for PC BIOS using socketed 29LV040 device. • 32 Mbytes Flash EPROM for Application program storage. • SDRAM 0.5, 1 or 1.5 Gbytes defined by order number. Processor burst and cache support. • 512 Kbytes non-volatile Static RAM.
Interfaces:	• 64-bit CompactPCI interface utilizing a PCI to PCI bridge. The board supports 5V or 3.3V CPCI signaling levels, depending on order number. • One RS232 serial channel using 16550 compatible UART. • EIDE/Ultra ATA100 interface via J5 and on-board mass storage option interface. • USB interface via J5 connector. Both 1.5 and 12Mbit/s interfaces supported. • Two single-width PMC sites supporting 64/32-bit PCI interfaces with 5V or 3.3V signaling. Both 5V and 3.3V power rails are provided. • VGA CRT interface with 4 Mbyte local RAM and supporting resolutions up to 1600 x 1200. • PS/2 compatible keyboard interface. • PS/2 compatible mouse interface. • Two Gigabyte Ethernet interfaces using 82546EB PCI to Ethernet device. One 10/100Mbit/s RJ45 connection via front panel or two 10/100/1000Mbits/s connections via J3. Support for PICMG 2.16 backplane networking. Optional rear panel Ethernet I/O via RJ45 connections on transition module. • External Reset input via J2.
Peripherals:	• ServerWorks CSB5 device provides standard PC-AT architecture peripherals. • PC AT Real Time Clock. • IPMI/IPMB interface support by on-board microcontroller.

A.2 Environmental Specification

A.2.1 Temperature Range

Operating . . . . . 0 to +55ºC @ 400LFM air flow
Storage . . . . . -40 to +70ºC

NOTE If the on-board disk drive option is fitted, the operating temperature range will be restricted to +5 to +55ºC and the storage temperature range will be restricted to -40 to +65ºC.

NOTE If the battery is fitted, the storage temperature range will be restricted to 0 to +70ºC. This is because the Super I/O device is partially operational when the battery is connected. The battery life will be reduced by storage at high temperatures due to increased self-discharge. It is therefore recommended that the battery be removed during storage.

A.2.2 Humidity

Operating . . . . . 10% to 90% non-condensing
Storage . . . . . 10% to 90% non-condensing

A.3 Dimensions

Height 23.3cm
Depth 16.0cm
Width 2.0cm
Weight . . . . . 500g (without Mass Storage Kit fitted)

A.4 Electrical Specification

A.4.1 Power Supply Requirements

VOLTAGE (V)	PROCESSOR SPEED	REGULATION	CURRENT (Typical)
+5.0V	1.2GHz	+5%, -3%	5.0A
+5.0V	800MHz	+5%, -3%	3.1A
+3.3V	ALL	+5%, -3%	3.9A
+12.0V	ALL	+/-5%	0.02A
-12.0V	ALL	+/-10%	0.01A

NOTE This is for a Pentium III-M CPU with 512 Mbytes SDRAM and with no Mass Storage Kit and no PMC modules fitted.

NOTE +/- 12V supplies are provided primarily for the PMC interface, and are limited to 500mA for +12V and 200mA for -12V. These supplies must be present for the PP 110/01x board to operate correctly.

A.5 Connectors

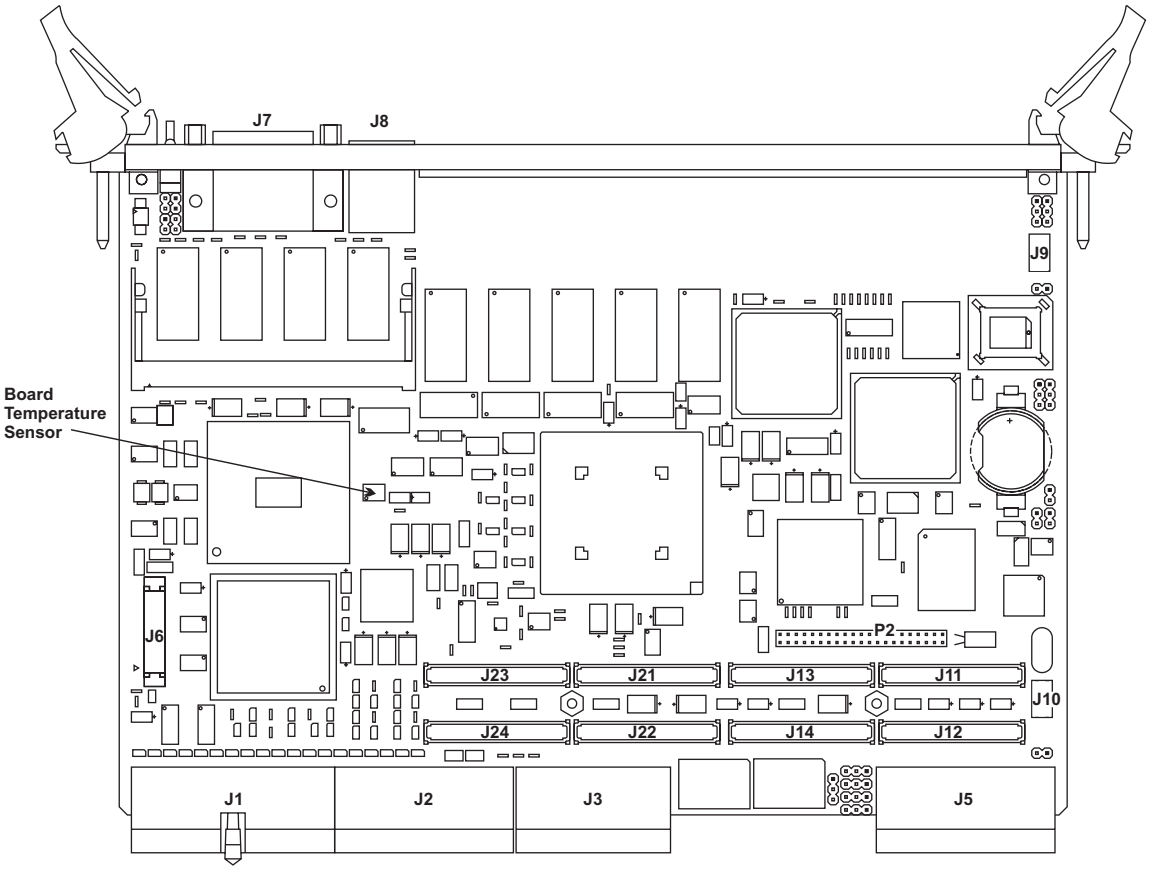


Figure A-1 Connector Layout

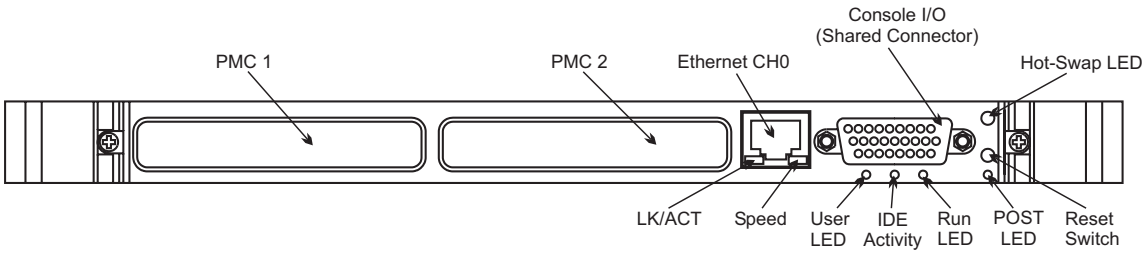


Figure A-2 Front Panel Connectors

A.5.1 Shared Front Panel Connector Pin-outs

This connector provides access to the keyboard, mouse, VGA, CRT and COM1 serial port interfaces. It is a female 26-way high density D-type connector. The pin-out is as follows.

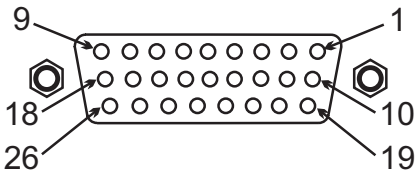


Figure A-3 Shared I/O Connector (Front View)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Interface
26	KBD/Mouse VCC	18	KBD DATA	9	Mouse DATA	KBD/Mouse
25	KBD/Mouse	17	KBD Clock	8	Mouse Clock	KBD/Mouse
24	RI	16	Tx	7	Rx	COM1 Serial
23	CD	15	RTS	6	CTS	COM1 Serial
22	DDC Clock	14	DTR	5	DSR	COM1 Serial
21	VYSNC	13	GND	4	GND	VGA
20	HYNSC	12	Blue GND	3	Blue	VGA
19	DDC DATA	11	Green GND	2	Green	VGA
		10	Red GND	1	Red	VGA

Table A-1 Shared Front Panel Connector Pin-outs

A.5.2 CompactPCI Interface (J1) Pin-outs

The CompactPCI interface connector J1 consists of a 150-pin connector with pins assigned as follows:

Pin	A	B	C	D	E	F	
25	5V	REQ64#	ENUM#	3.3V	5V	GND	
24	AD[1]	5V	V(I/O)	AD[0]	ACK64#	GND	
23	3.3V	AD[4]	AD[3]	5V	AD[2]	GND	
22	AD[7]	GND	3.3V	AD[6]	AD[5]	GND	
21	3.3V	AD[9]	AD[8]	M66EN	C/BE[0]#	GND	
20	AD[12]	GND	V(I/O)	AD[11]	AD[10]	GND	
19	3.3V	AD[15]	AD[14]	GND	AD[13]	GND	J
18	SERR#	GND	3.3V	PAR	C/BE[1]#	GND	1
17	3.3V	IPMB_SCL	IPMB_SDA	GND	PERR#	GND	
16	DEVSEL#	GND	V(I/O)	STOP#	LOCK#	GND	C
15	3.3V	FRAME#	IRDY#	BD_SEL#	TRDY#	GND	O
12-14			KEY AREA				N
11	AD[18]	AD[17]	AD[16]	GND	C/BE[2]#	GND	N
10	AD[21]	GND	3.3V	AD[20]	AD[19]	GND	E
9	C/BE[3]#	IDSEL	AD[23]	GND	AD[22]	GND	C
8	AD[26]	GND	V(I/O)	AD[25]	AD[24]	GND	T
7	AD[30]	AD[29]	AD[28]	GND	AD[27]	GND	O
6	REQ0#	PCI_PRESENT#	3.3V	CLK	AD[31]	GND	R
5	NC	NC	PCI_RST#	GND	GNT0#	GND	
4	IPMB_PWR	HEALTHY#	V(I/O)	INTP	INTS	GND	
3	INTA#	INTB#	INTC#	5V	INTD#	GND	
2	NC	5V	NC	NC	NC	GND	
1	5V	-12V	NC	+12V	5V	GND	
Pin	**A**	**B**	**C**	**D**	**E**	**F**	

Table A-2 CompactPCI J1 Interface Pin-outs

NOTE PICMG 2.16 defines pin B6 as PCI_PRESENT#. In earlier PICMG specifications this pin is defined as GND.

A.5.3 CompactPCI Interface (J2) Pin-outs

The CompactPCI interface connector J2 consists of a 132-pin connector with pins assigned as follows:

Pin	A	B	C	D	E	F	
22	GA4	GA3	GA2	GA1	GA0	GND	
21	CLK6	GND	NC	NC	NC	GND	
20	CLK5	GND	NC	GND	NC	GND	
19	GND	GND	IPMB1_SDA	IPMB1_SCL	IPMB1_ALERT#	GND	
18	NC	NC	NC	GND	NC	GND	
17	NC	GND	PRST#	REQ6#	GNT6#	GND	J
16	NC	NC	DEG#	GND	NC	GND	2
15	NC	GND	FAL#	REQ5#	GNT5#	GND	
14	AD[35]	AD[34]	AD[33]	GND	AD[32]	GND	C
13	AD[38]	GND	V(I/O)	AD[37]	AD[36]	GND	O
12	AD[42]	AD[41]	AD[40]	GND	AD[39]	GND	N
11	AD[45]	GND	V(I/O)	AD[44]	AD[43]	GND	N
10	AD[49]	AD[48]	AD[47]	GND	AD[46]	GND	E
9	AD[52]	GND	V(I/O)	AD[51]	AD[50]	GND	C
8	AD[56]	AD[55]	AD[54]	GND	AD[53]	GND	T
7	AD[59]	GND	V(I/O)	AD[58]	AD[57]	GND	O
6	AD[63]	AD[62]	AD[61]	GND	AD[60]	GND	R
5	C/BE[5]#	64EN#	V(I/O)	C/BE[4]#	PAR64	GND	
4	V(I/O)	NC	C/BE[7]#	GND	C/BE[6]#	GND	
3	CLK4	GND	GNT3#	REQ4#	GNT4#	GND	
2	CLK2	CLK3	SYSEN#	GNT2#	REQ3#	GND	
1	CLK1	GND	REQ1#	GNT1#	REQ2#	GND	
Pin	**A**	**B**	**C**	**D**	**E**	**F**	

Table A-3 CompactPCI J2 Interface Pin-outs

A.5.4 CompactPCI Interface (J3) Pin-outs

The CompactPCI interface I/O connector J3 consists of a 114-pin connector with pins assigned as follows:

Pin	A	B	C	D	E	F	
19	GND	GND	GND	GND	GND	GND	
18	LPa_DA	LPa_DA#	GND	LPa_DC	LPa_DC#	GND	
17	LPa_DB	LPa_DB#	GND	LPa_DD	LPa_DD#	GND	Ethernet
16	LPb_DA	LPb_DA#	GND	LPb_DC	LPb_DC#	GND	
15	LPb_DB	LPb_DB#	GND	LPb_DD	LPb_DD#	GND	
14	+3.3V	+3.3V	+3.3V	+5V	+5V	GND	TM POWER
13	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	GND	
12	I/O 10	I/O 9	I/O 8	I/O 7	I/O 6	GND	
11	I/O 15	I/O 14	I/O 13	I/O 12	I/O 11	GND	
10	I/O 20	I/O 19	I/O 18	I/O 17	I/O 16	GND	
9	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21	GND	
8	I/O 30	I/O 29	I/O 28	I/O 27	I/O 26	GND	
7	I/O 35	I/O 34	I/O 33	I/O 32	I/O 31	GND	PMC 2
6	I/O 40	I/O 39	I/O 38	I/O 37	I/O 36	GND	I/O
5	I/O 45	I/O 44	I/O 43	I/O 42	I/O 41	GND	
4	I/O 50	I/O 49	I/O 48	I/O 47	I/O 46	GND	
3	I/O 55	I/O 54	I/O 53	I/O 52	I/O 51	GND	
2	I/O 60	I/O 59	I/O 58	I/O 57	I/O 56	GND	
1	SPKR Out	I/O 64	I/O 63	I/O 62	I/O 61	GND	
Pin	**A**	**B**	**C**	**D**	**E**	**F**	

Table A-4 CompactPCI J3 Interface Pin-outs

A.5.5 CompactPCI Interface (J5) Pin-outs

The CompactPCI interface I/O connector J5 consists of a 132-pin connector with pins assigned as follows:

Pin	A	B	C	D	E	F	
22	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	GND	
21	I/O 10	I/O 9	I/O 8	I/O 7	I/O 6	GND	
20	I/O 15	I/O 14	I/O 13	I/O 12	I/O 11	GND	
19	I/O 20	I/O 19	I/O 18	I/O 17	I/O 16	GND	
18	I/O 25	I/O 24	I/O 23	I/O 22	I/O 21	GND	
17	I/O 30	I/O 29	I/O 28	I/O 27	I/O 26	GND	PMC1
16	I/O 35	I/O 34	I/O 33	I/O 32	I/O 31	GND	I/O
15	I/O 40	I/O 39	I/O 38	I/O 37	I/O 36	GND	
14	I/O 45	I/O 44	I/O 43	I/O 42	I/O 41	GND	
13	I/O 50	I/O 49	I/O 48	I/O 47	I/O 46	GND	
12	I/O 55	I/O 54	I/O 53	I/O 52	I/O 51	GND	
11	I/O 60	I/O 59	I/O 58	I/O 57	I/O 56	GND	
10	GND	I/O 64	I/O 63	I/O 62	I/O 61	GND	
9	TR RST#	GND	GND	GND	GND	GND	
8	DD5	DD9	DD6	DD8	DD7	GND	Primary
7	DD12	DD3	DD11	DD4	DD10	GND	EIDE
6	DD0	DD14	DD1	DD13	DD2	GND	&
5	IORDY	DIOR#	DIOW#	DRQ	DD15	GND	MUX I/O
4	DA2	DA0	DA1	INTRQ	DACK#	GND	
3	LDRQ1#	MUXIO_DATA	MUXIO_CLK	CS3#	CS1#	GND	
2	LAD3	LAD2	LAD1	LAD0	GND	GND	LPC Bus
1	USBD1#	USBD1	SERIRQ	LFRAME#	TM_CLK	GND	& USB
Pin	**A**	**B**	**C**	**D**	**E**	**F**	

Table A-5 CompactPCI J5 Interface Pin-outs

A.5.6 On-Board Mass Storage Option Connector (P2) Pin-outs

Pin No.	Signal Name	Pin No.	Signal Name
1	IDE_RST#	2	GND
3	SDD7	4	SDD8
5	SDD6	6	SDD9
7	SDD5	8	SDD10
9	SDD4	10	SDD11
11	SDD3	12	SDD12
13	SDD2	14	SDD13
15	SDD1	16	SDD14
17	SDD0	18	SDD15
19	GND	20	NC
21	SDREQ	22	GND
23	SDIOW#	24	GND
25	SDIOR#	26	GND
27	SIORDY	28	NC
29	SDDACK#	30	GND
31	INT15#	32	NC
33	SDA1	34	PDIAG
35	SDA0	36	SDA2
37	SDCS1#	38	SDCS3#
39	ACTIVITY#	40	GND
41	+5V	42	+5V MOTOR
43	GND	44	NC

Table A-6 On-Board Mass Storage Option Interface Pin-outs

A.5.7 PMC Site Connectors (J11 - J14 and J21 - J24) Pin-outs

Signal assignments on the PMC connectors are shown in Tables A-9, A-10 and A-11.

Pin No.	Signal Name	Pin No.	Signal Name
1	-	2	-12V
3	GND	4	INTA#
5	INTB#	6	INTC#
7	BUSMODE#1	8	+5V
9	INTD#	10	-
11	GND	12	+3.3V††
13	CLK	14	GND
15	GND	16	GNT#
17	REQ#	18	+5V
19	V (I/O)	20	AD(31)
21	AD(28)	22	AD(27)
23	AD(25)	24	GND
25	GND	26	C/BE(3)#
27	AD(22)	28	AD(21)
29	AD(19)	30	+5V
31	V (I/O)	32	AD(17)
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	V (I/O)	46	AD(15)
47	AD(12)	48	AD(11)
49	AD(09)	50	+5V
51	GND	52	C/BE(0)#
53	AD(06)	54	AD(05)
55	AD(04)	56	GND
57	V (I/O)	58	AD(03)
59	AD(02)	60	AD(01)
61	AD(00)	62	+5V
63	GND	64	REQ64#

# denotes active low, † pulled high via 1KOhm resistor,
†† pulled high via 10KOhm resistor.

Table A-7 PMC J11 and J21 Connector Pin-outs

Pin No.	Signal Name	Pin No.	Signal Name
1	+12V	2	-
3	-	4	-
5	-	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	AD(30)	20	AD(29)
21	GND	22	AD(26)
23	AD(24)	24	+3.3V
25	IDSEL	26	AD(23)
27	+3.3V	28	AD(20)
29	AD(18)	30	GND
31	AD(16)	32	C/BE(2)#
33	GND	34	IDSEL B
35	TRDY#	36	+3.3V
37	GND	38	STOP#
39	PERR#	40	GND
41	+3.3V	42	SERR#
43	C/BE(1)#	44	GND
45	AD(14)	46	AD(13)
47	M66EN	48	AD(10)
49	AD(08)	50	+3.3V
51	AD(07)	52	REQ B#
53	+3.3V	54	GNT B#
55	PMC-RSVD	56	GND
57	PMC-RSVD	58	EREADY††
59	GND	60	-
61	ACK64#	62	+3.3V
63	GND	64	+3.3V††

# denotes active low, † pulled high via 1KOhm resistor,
†† pulled high via 10KOhm resistor.

Table A-8 PMC J12 and J22 Connector Pin-outs

NOTE IDSEL B, REQ B# and GNT B# are provided for use by dual-function PMC modules or Processor-PMC modules.

NOTE Pins 58 and 64 are pulled high to suit Processor-PMC modules.

Pin No.	Signal Name	Pin No.	Signal Name
1	-	2	GND
3	GND	4	C/BE(7)#
5	C/BE(6)#	6	C/BE(5)#
7	C/BE(4)#	8	GND
9	V(I/O)	10	PAR64
11	AD(63)	12	AD(62)
13	AD(61)	14	GND
15	GND	16	AD(60)
17	AD(59)	18	AD(58)
19	AD(57)	20	GND
21	V(I/O)	22	AD(56)
23	AD(55)	24	AD(54)
25	AD(53)	26	GND
27	GND	28	AD(52)
29	AD(51)	30	AD(50)
31	AD(49)	32	GND
33	GND	34	AD(48)
35	AD(47)	36	AD(46)
37	AD(45)	38	GND
39	V(I/O)	40	AD(44)
41	AD(43)	42	AD(42)
43	AD(41)	44	GND
45	GND	46	AD(40)
47	AD(39)	48	AD(38)
49	AD(37)	50	GND
51	GND	52	AD(36)
53	AD(35)	54	AD(34)
55	AD(33)	56	GND
57	V(I/O)	58	AD(32)
59	-	60	-
61	-	62	GND
63	GND	64	-

Table A-9 PMC J13 and J23 Connector Pin-outs

Pin No.	Signal Name	Pin No.	Signal Name
1	-	2	GND
3	GND	4	C/BE(7)#
5	C/BE(6)#	6	C/BE(5)#
7	C/BE(4)#	8	GND
9	V(I/O)	10	PAR64
11	AD(63)	12	AD(62)
13	AD(61)	14	GND
15	GND	16	AD(60)
17	AD(59)	18	AD(58)
19	AD(57)	20	GND
21	V(I/O)	22	AD(56)
23	AD(55)	24	AD(54)
25	AD(53)	26	GND
27	GND	28	AD(52)
29	AD(51)	30	AD(50)
31	AD(49)	32	GND
33	GND	34	AD(48)
35	AD(47)	36	AD(46)
37	AD(45)	38	GND
39	V(I/O)	40	AD(44)
41	AD(43)	42	AD(42)
43	AD(41)	44	GND
45	GND	46	AD(40)
47	AD(39)	48	AD(38)
49	AD(37)	50	GND
51	GND	52	AD(36)
53	AD(35)	54	AD(34)
55	AD(33)	56	GND
57	V(I/O)	58	AD(32)
59	-	60	-
61	-	62	GND
63	GND	64	-

Table A-10 PMC J14 and J24 Connector Pin-outs

A.5.8 Ethernet Interface (J9) Pin-out

The front panel Ethernet Interface uses an 8-way RJ45 connector with the following pin-out:

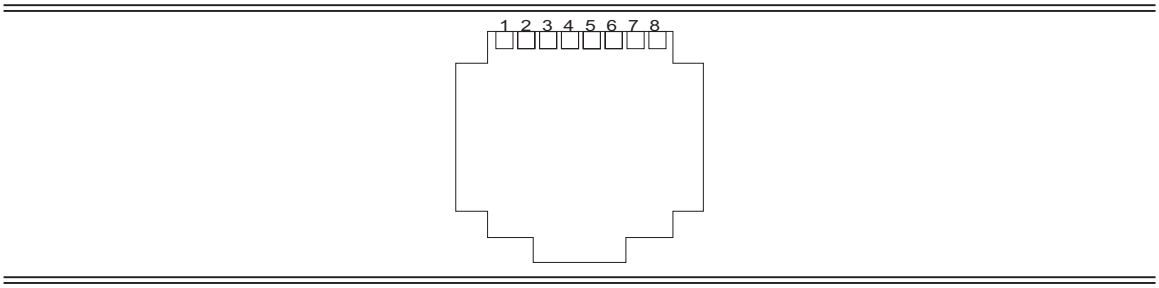


Figure A-4 Ethernet RJ-45 Connector (Front View)

Pin No.	Signal Name
1	Transmit (+)
2	Transmit (-)
3	Receive (+)
4	Not used
5	Not used
6	Receive (-)
7	Not used
8	Not used

Table A-11 Ethernet RJ-45 Connector Pin-outs

NOTE The front panel Ethernet interface does not support 1000Mbits/s operation.

A.5.9 Processor Debug Port (J6) Pin-outs

The processor debug port, which is supported by a number of emulator devices, is accessible via an Intel specified 30-way receptacle connector with the following pin-out.

Pin No.	Signal Name
1	GND
2	CPU Reset
3	GND
4	Debug Reset
5	GND
6	CPU TCK
7	CPU TDI
8	CPU TMS
9	CPU TDO
10	Pull Up
11	CPU TRST
12	NC
13	NC
14	GND
15	CPU REQ
16	GND
17	CPU RDY
18	GND
19	NC
20	GND
21	Pull Up
22	GND
23	NC
24	GND
25	Pull Up
26	NC
27	NC
28	GND
29	Pull Up
30	GND

Table A-12 30-way Debug Connector Pin-outs

A.5.10 Port 80 (J10) Pin-outs

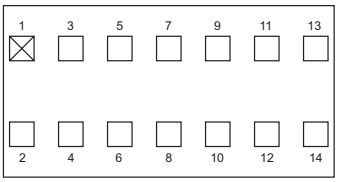


Figure A-5 Port 80 Connector

Pin No.	Signal Name
1	GND
2	NC
3	Port 80 Write #
4	NC
5	D3
6	D7
7	D2
8	D6
9	D1
10	D5
11	D0
12	D4
13	+5 Volts
14	GND

# Denotes active low

Table A-13 Port 80 Connector Pin-outs