

# ***CompactPCI***<sup>®</sup>

---

PICMG<sup>®</sup> 2.14 R1.0

## **MultiComputing Short Form Specification**

**September 6, 2001**



***FOR INFORMATION ONLY; DO NOT ATTEMPT TO DESIGN  
FROM THIS DOCUMENT***

NOTE: This short form specification is a subset of the CompactPCI MultiComputing specification, PICMG 2.14 R 1.0. For complete guidelines on the design of CompactPCI MultiComputing implementations, the full specification is required.

For a full copy of the PICMG 2.14 specification, go to [www.picmg.org](http://www.picmg.org), or contact the PCI Industrial Computer Manufacturers Group at 401 Edgewater Place, Suite 500, Wakefield, Mass., 01880. Phone 781-246-9318, fax 781-224-1239, email [info@picmg.org](mailto:info@picmg.org).

©Copyright 1995, 1996, 1997, 2001 PCI Industrial Computer Manufacturers Group.

PICMG disclaims all warranties and liability for the use of this document and the information contained herein, and assumes no responsibility for any errors or omissions that may appear in this document, nor is PICMG responsible for any incidental or consequential damages resulting from the use of any data contained in this document, nor does PICMG assume any responsibility to update or correct any information in this publication.

*PICMG<sup>®</sup>, CompactPCI<sup>®</sup>, and the PICMG<sup>®</sup> and CompactPCI<sup>®</sup> logos are registered trademarks of the PCI Industrial Computer Manufacturers Group.*

*All other brand or product names may be trademarks or registered trademarks of their respective holders.*

**FOR INFORMATION ONLY;  
DO NOT ATTEMPT TO DESIGN FROM THIS DOCUMENT**

---

## **INTRODUCTION**

This short form specification is a subset of the full CompactPCI MultiComputing specification, Revision 1.0 as approved by PICMG. This document is meant as a guide for those considering CompactPCI MultiComputing applications, but is not a design document. Anyone intending to design a system employing CompactPCI MultiComputing should obtain the full specification from PICMG.

CompactPCI is an open specification family supported by PICMG, a consortium of companies involved in open architecture computing for demanding applications. PICMG develops, administers and maintains the specifications in this family.

### **Objectives**

The CompactPCI MultiComputing Network (MCNet) allows multiple Peripheral Nodes (PN) CPU boards and one System Node (SN) CPU board to perform inter-processor communications (IPC) over a CompactPCI backplane (or other shared PCI address space) as if they were discrete computers on a network. The software interface to perform IPC may be the protocol stack packaged with the operating system using the MCNet as a data link layer, or there may be a direct application programming interface (API) to the MCNet data link layer driver, bypassing the OS network protocol stack. The interface to the data link layer is not fully specified, allowing implementers to choose an interface suited to their application requirements.

An MCNet comprises one System Node and zero to  $N$  Peripheral Nodes. An MCNet operates within 1 to  $M$  PCI bus segments that are logically one PCI address space. The design will accommodate over 4,000 PNs per MCNet. An MCDevice may have a PN on one MCNet and an SN or PN on another, thereby facilitating packet routing between two or more distinct PCI busses.

All System and Peripheral Node service sharing is a function of the operating system's ability to provide shared network services via the protocol stacks (e.g. Windows file and print sharing capabilities, NFS, etc). The MCNet protocol specification accommodates Hot Swap and other fault management of MCNodes and MCDevices.

Packet buffering is performed in a non-contentious manner, thereby not requiring read-modify-write locked cycles.

### **Purpose**

Define packet-based communications between heterogeneous PCI agents (MultiComputing) within the CompactPCI system architecture.

Users of CompactPCI have expressed interest in seeing standardization in the methods for processor boards to communicate with each other when co-existing in a shared PCI address space.

### **Goals**

These are goals whose definition will allow multiple vendors with diverse architectures to interoperate.

- OSI Data Link Layer network model approach
- Discovery and Initialization of heterogeneous devices following system initialization or Hot Swap
- Buffer structures
- Messaging

**FOR INFORMATION ONLY;  
DO NOT ATTEMPT TO DESIGN FROM THIS DOCUMENT**

---

- Supported features (i.e. multi and broad cast, promiscuous mode, nodal quiescing for hot swap, reset, driver unload)
- Non-transparent bridging devices and resources to be utilized
- Byte ordering
- Hot Swap re-enumeration of bus and effect on system communications
- High Availability (compatible with the pending PICMG 2.13 specification for dual-host high availability)

## **PROTOCOL OVERVIEW**

The purpose of this protocol is to

- Support up to 4096 MCNodes communicating in a heterogeneous (single global PCI memory address space) PCI environment
- Occupy Layer 2 (data link) of the OSI reference model
- Allow for dynamic discovery and configuration of network elements
- Emulate broadcast and multicast capabilities to allow higher level network protocols to operate via this data link layer
- Accommodate Compact PCI Hot Swap capabilities
- Provide for fault detection, containment and notification in a High Availability environment
- Allow extensions in subsequent versions of this specification which take advantage of additional hardware features
- Ensure interoperability among all vendors complying with this specification

The information in this chapter is intended to be descriptive in nature, and is not an actual part of the specification. Subsequent chapters in the full CompactPCI MultiComputing specification specify the protocol in detail.

### **Protocol Model**

The general model of the MCNet protocol is that of a dedicated shared memory region for the exchange of messages. The entire protocol takes place within a single global PCI memory address space. Each MCNode of a pair of MCNodes desiring to communicate allocates regions of the global PCI memory space for communication and control structures dedicated to that single connection. A single System Node manages the discovery and setup of the remaining connection pairings on demand.

### **MultiComputing Devices**

The term MultiComputing Device (MCDevice) refers to actual PCI hardware that conforms to the PCI specification for a PCI device/function. The MCDevice is discovered and managed by the PCI enumeration software. Resources are allocated by the PCI system software via the PCI type 0 configuration header mechanism. For a PCI device/function to be an MCDevice, it must have at least a 4k byte PCI memory space allocation.

MCDevices are important to the 2.14 protocol because they form the basis for communications among hardware provided by multiple vendors. Additionally, MCDevices are treated as hot

**FOR INFORMATION ONLY;  
DO NOT ATTEMPT TO DESIGN FROM THIS DOCUMENT**

---

swappable entities and the MCNet system maintains associations between MCDevices and MultiComputing Nodes for purposes of dynamic configuration management and fault containment.

### **MultiComputing Nodes**

MultiComputing Nodes (MCNode) are the entities that actually communicate using the 2.14 protocol. An MCNode is the software abstraction that performs the protocol with a second MCNode. Each MCNode resides on an MCDevice. More than one MCNode may reside on a single MCDevice. Each MCNode is assumed to have a MultiComputing Node Interface (MCNI), which provides the abstract connection to the rest of the software running on an MCDevice. MCNet services are requested and received via the MCNI.

There are two types of MCNodes. Each MCNet must have one and only one active System Node (SN). All other MCNodes participating in the MCNet are Peripheral Nodes (PN). Both Peripheral Nodes and System Nodes have an MCNI via which services are provided to associated software. Additionally, the System Node has a System Node Interface (SNI), which provides the abstract software definition of additional services required to manage the MCNet.

### **Monikers and Moniker Assignments**

In the 2.14 protocol, every MCNode is known by a 64-bit value called a Moniker. The Monikers are similar to 64-bit Ethernet addresses. The complete MultiComputing specification describes the Moniker format in detail. Monikers are characterized as locally administered or globally administered, and, orthogonally, as individual or broadcast (multicast). An MCNode has an MCNet-unique individual Moniker.

### **Short Atomic Messages**

Short Atomic Messages (SAMs) are used to deal with the PCI bus mutual exclusion problem. While the PCI specification provides for bus locking mechanisms, bus locking is not widely or consistently implemented by PCI hardware. Therefore, the SAM mechanism is used in the one area requiring a shared resource.

### **The System Map and Moniker Resolution**

One of the main functions of the System Node is to maintain a system map of all of the known MCNodes participating in the MCNet. The system map is built by the SN during initialization of PNs, which are created by the MCDevice discovery process and other processes. Each call to the `sni_create_node` function, and certain control messages within the MCNet, cause the SN to allocate table space for the new PN's system map entry.

### **Broadcast and Multicast**

In this specification, the term Broadcast is used to represent both the true Broadcast, consisting of all MCNodes, and Multicast, which is an arbitrary subset of the MCNodes in voluntary association. True broadcast is used where the group consisting of all MCNodes is meant. Technically, there is no default true broadcast group, since all MCNet Broadcast Groups require that MCNodes explicitly join the group in order to receive data packets destined for a particular broadcast moniker.

Since the physical layer characteristics of the PCI bus only allow for point to point communications, broadcast and multicast functions must be emulated. While broadcast operations are not strictly required for operation of the MCNet protocol, those functions are provided to ease the use of this protocol as a data link layer for other network protocols, many of which rely heavily upon broadcast capabilities.

Emulation of the broadcast capability is provided by introducing an MCNode role called the Broadcast Group Broker, or simply, Broker. The Broker function is optional for all MCNodes

**FOR INFORMATION ONLY;  
DO NOT ATTEMPT TO DESIGN FROM THIS DOCUMENT**

---

except the System Node, which provides the default Broker functions for the true Broadcast Group. The Broker functions are used via the MCNI. All MCNodes provide the broker functions within the MCNI, but may deny requests for services requiring the Broker functions.

## **Alerts**

In the context of this specification, Alerts are mechanisms to indicate to an MCNode that a state change has been effected by another MCNode. The purpose of alerts is to eliminate or reduce the need for an MCNode to poll its shared data structures to detect state changes.

## **High Availability Considerations**

Considerations for planned removal and unplanned MCNode failure have been designed into this specification. Redundant MCNode roles have also been considered. However, the level of redundancy covered by this first revision is limited to cold replacement of MCNode roles.

Additionally, given the nature of the PCI bus, there can be no guarantee that an MCNode failure or unexpected removal will not result in an unrecoverable bus fault. Care has been taken to provide system freeze mechanisms that can be quickly propagated in the event of failures, but the propagation relies on the PCI bus's continued operation.

As a means of detecting "silent" failures, an MCNode heartbeat capability has been incorporated in the data structures of this protocol. Control messages for creating and specifying the properties of the heartbeat have been provided, as well as mechanisms for reporting the failure of an expected heartbeat.

All of the HA mechanisms serve to maximize the probability, but not guarantee, that a failure can be contained.

## **Command, Control Messages, and Data Packets**

Three types of information exchange are used by this protocol. The first two types apply to all connections, and allow for the establishment and management of connections, connection resources, and the expeditious control of an MCNode's activity on the PCI bus. These two mechanisms are referred to as Commands and Control Messages. Commands take the form of flags or short values that are written to the connection data structures to effect immediate state changes prior to an MCNode's continuation of ordinary operations. Control Messages exist in two priority classes, urgent and normal. There is a queue of up to 15 normal control messages and two urgent control messages. Urgent messages are processed before normal messages.

The third type of information exchange is called data packet exchange. Data packets are exchanged as a result of requests to send information to another MCNode. Data packet exchange is the normal in-band communication of this protocol layer.

## **Operational Overview of the Protocol**

The 2.14 MultiComputing Protocol begins following initialization of the PCI system software. The SN and SNI are created by the system initialization software in a manner determined by the supplier or integrator of the system software. Once the SNI is available, the PCI system software, or adjunct to the system software, calls the SNI with the MCDevices discovered during PCI bus enumeration.

During or following the process of PCI bus enumeration and driver configuration, MCDevices are discovered, either as a direct result of the PCI enumeration, or subsequently by action of a particular PCI device driver. For each MCDevice discovered, the SNI is called with the parameters for the MCNode that will be created on that device. The SNI receives the MCNode parameters and performs a rendezvous with the Peripheral Node at the location specified by the SNI MCNode initialization request.

**FOR INFORMATION ONLY;  
DO NOT ATTEMPT TO DESIGN FROM THIS DOCUMENT**

---

The Rendezvous of the SN and PN results in a System Map entry being created in the SN, and a control/command connection between the PN and SN being created. This control/command connection will remain in existence so long as the SN and the PN remain a part of the MCNet.

During the initial SN/PN connection establishment, certain key parameters are exchanged relating to the alert mechanism and addressing capabilities of the PN, as well as the operational mode of the connection. Moreover, the Moniker for the PN is established, either as requested by the SNI service, or as supplied by the PN by local (to the PN) means.

A connection is established by creation of a Connection Control Block and Bi-directional Information Block on both MCNodes party to the connection. These structures remain so long as the connection exists. Data packet exchange is created by control messages requesting the establishment of a data packet capability, and results in the creation of an additional data structure, the Packet Information Block (PIB), which contains Transmitter and Receiver information.

Command and Control Message exchanges take place in the CCB/BIB structures. The transmitting MCNode finds a free resource in the receiving MCNode's data structure, writes the message or command, updates the receiver's queue indices or flags, and then executes the control message alert mechanism on the receiver. The receiver then processes the command or message(s) and updates the queue or flags in the transmitting MCNode's data structures.

If data packet exchange is established between two MCNodes, then data packet exchange takes place in the PIB data structures. An MCNode wishing to transmit requests allocation of free block(s) from the receiver. The receiver makes the free block available to the transmitter by writing the free block information into the transmitter's PIB. Then the transmitter uses the free block and writes the data packet into the free block. After the data packet has been written, the transmitter updates the packet descriptor in the receiver's PIB and updates the respective packet descriptor indices. Finally, the transmitter executes the Data Packet alert mechanism on the receiver.

The receiver of a data packet processes the data packet and passes it up through the MCNI to upper layer software. Following data processing, the packet descriptor queue indices are updated in the transmitter's PIB. If the buffer block is now free, the receiver may choose to return the buffer to the transmitter's free block queue. If the free block list was empty, or the packet descriptor queue was full, then the receiver will execute the Resource alert mechanism on the transmitter.

### **Operational Overview of Broadcast and Multicast**

In order to receive broadcast data packets, an MCNode must join the broadcast group. It does this by sending a broadcast Moniker Resolution Request to the SN. The SN responds with the Moniker of the Broker for that broadcast group. The MCNode then connects with the Broker and makes a Join Broadcast Group request via the control message queue. The Broker then updates the broadcast policies so that future data packets to the broadcast moniker will also be sent to the newly joined MCNode.

When an MCNode wishes to send a data packet to a broadcast group, it contacts the broker for the group to receive a policy. The policy is the list of MCNodes to which the MCNode must connect and send the data packet in order to propagate the data to all of the group members. The policy is requested and received via the control connection to the broker.

While a policy may simply be the list of all MCNodes in the group, optimizations are allowed in this protocol specification. When an MCNode joins a broadcast group, the MCNode specifies if it can act as a proxy for the broadcast group. If so the broker may place several MCNodes into what is called a proxy group. In this case, the broker will provide a policy that includes only some of the MCNodes in the group. Each broadcast data packet that is sent to a proxy MCNode will be replicated by the proxy and passed on to the other members of the proxy group.