

# HiRelPCI—An Extensible High Reliability Extended PCI Bus

**Draft 0.65**

**November 9, 1998**

Sponsor

**Microprocessor and Microcomputer Standards Subcommittee  
of the IEEE Computer Society**

**Abstract:** This HiRelPCI specification defines a set of signals and protocols for communicating between elements of a robust and redundant computer using PCI protocols extended to include packet mode operations. This standard is designed to support transportation, telecommunications and industrial process control system.

**Keywords:** HiRelPCI, High Reliability, system bus, personal computer, packet bus.

Copyright © 199X by the Institute of Electrical and Electronics Engineers, Inc.  
345 East 47th Street  
New York, NY 10017, USA  
All rights reserved.

This is an unapproved draft of a proposed IEEE Standard, subject to change. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities. If this document is to be submitted to ISO or IEC, notification shall be given to the IEEE Copyright Administrator. Permission is also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position. Other entities seeking permission to reproduce this document for standardization or other activities, or to reproduce portions of this document for these or other uses must contact the IEEE Standards Department for the appropriate license. Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department  
Copyright and Permissions  
445 Hoes Lane, P.O. Box 1331  
Piscataway, NJ 08855-1331 USA

## Background

This document is a preliminary draft, for consideration as the next generation High Reliability bus interconnect system for rugged environments. This document is intended to be a draft for an IEEE P1996 working group and contains contributions from Summit Computer System Inc., Centigram Communications Corp. and Intersection Development Corp.. Additional contribution to the P2100 sections were made by Apple Computer, Intel Corp. and Sun Microsystems Corp. This document is not stable; the working group is expected to revise and refine its contents. However, a relatively-complete initial draft and well-defined concepts are expected to expedite the standards development process.

For further knowledge on the technical content or status of this specification, contact the Chair or the Editor. A more complete list of interested individuals is located near the end of this draft.:

**CHAIRMAN**

Bob Davis

President

Summit Computer Systems, Inc.

22685 Summit Rd.

Los Gatos, CA 95030-9310

(408)353-2706

bob@scsi.com

**EDITOR**

Bob Davis

## TABLE OF CONTENTS

1.	Introduction to HiRelPCI.....	17
1.1	ELEMENTS OF HiRelPCI BUS.....	17
1.2	Bus Signal Definitions.....	18
1.3	Parallel Bus Operations.....	19
1.3.1	CSM Functions.....	19
1.3.2	Priority Bus Request Mechanism.....	19
1.4	PacketBus Operations.....	19
1.4.1	CSR Space Mapping.....	20
1.5	TDM Bus Operations.....	20
1.6	Maintenance Bus Operations.....	21
1.7	CSR Specifications and Configuration Operations.....	23
1.7.1	Node Addressing.....	23
1.7.2	P1996 CSR Register Definitions.....	25
1.8	Supporting structures.....	26
1.8.1	Serial Interconnect.....	26
1.8.2	Support Signals.....	27
1.9	Electrical Specifications.....	27
1.9.1	Connector Pin Assignments.....	27
1.9.2	Power Distribution.....	30
1.10	Mechanical Specifications.....	31
1.10.1	Mechanical Sizes.....	31
1.10.2	Environmental.....	32
1.11	IEEE PAR TITLE.....	32
1.12	IEEE PAR PURPOSE of P1996 HiRelPCI.....	33
1.13	IEEE PAR SCOPE of P1996 HiRelPCI.....	33
2.	References, definitions, and notation.....	35
2.1	References.....	35
2.2	Conformance levels.....	35
2.3	Glossary of terms.....	35
2.4	Bit and byte ordering.....	57
2.5	State machine notation.....	57
3.	Signal Definitions.....	60
3.1	Signal Type Definitions.....	61
3.2	HiRelPCI Interface types.....	61
3.2.1	6su Full configuration.....	61
3.2.2	12su Full Redundant Configuration.....	62
3.3	Pin Function Groups.....	66
3.3.1	System Pins.....	66
3.3.2	Interface Control Pins.....	67
3.3.3	Arbitration Pins.....	68
3.3.4	Interrupt Pins.....	68
3.3.5	Geographical Address Pins.....	68
3.3.6	Communications Signals.....	68
3.3.7	Maintenance BUS (MB).....	69
4.	Bus Operations.....	71

4.0.1	Command Definitions .....	71
4.0.2	Priority Bus Request Mechanism .....	72
4.0.3	Command Usage Rules .....	72
4.1	Fundamental HiRelPCI Protocols .....	72
4.1.1	Basic Transfer Control .....	73
4.1.2	Addressing .....	73
4.1.3	Byte Alignment .....	74
4.1.4	Bus Driving and Turnaround .....	74
4.2	Bus Transactions .....	74
4.2.1	PacketBus Write .....	75
4.2.2	Transaction Termination .....	76
4.3	Arbitration .....	76
4.3.1	Arbitration Parking .....	77
4.4	Redundant Operation .....	77
4.4.1	Operation With Full Dual Service Module, Dual Bus System .....	77
4.4.2	Redundant operation with single processor dual bus system. ....	77
4.4.3	Redundant processor operation .....	77
4.4.4	Redundant operation over multiple bus segments .....	78
4.4.5	Methods of redundant operation .....	78
4.4.6	Address mapping of redundant elements .....	78
4.4.7	Resource Map definition .....	78
4.4.8	Testing of redundant elements .....	79
4.4.9	Fault testing mechanisms .....	79
4.4.10	Switching of Redundant Elements for Reliability .....	80
4.4.11	Start-up of redundant elements .....	80
4.4.12	Resynchronization of Redundant Elements .....	80
5.	PacketBus Operation .....	81
5.1	Common packet components .....	83
5.1.1	Generic packet components .....	83
5.2	Split-response packet formats .....	83
5.2.1	Request-packet components .....	83
5.2.2	Response-packet components .....	86
5.2.3	Acknowledge packet components .....	87
5.3	Special packet formats .....	87
5.3.1	timeSync packet formats .....	87
5.3.2	Basic tCode values .....	88
5.3.3	Lock subcommand-code values .....	91
5.3.4	stat (standard status) values .....	93
5.4	Subaction data formats .....	95
5.4.1	Selected-byte write16/read16 formats .....	95
5.4.2	lock16 transaction formats .....	96
5.4.3	Selected-range transaction formats .....	97
5.5	Critical-word ordering .....	99
5.5.1	Critical-quadlet ordering .....	99
5.6	move and talk packet formats .....	102
5.6.1	move packet formats .....	102
5.6.2	mMove packet formats .....	102
5.6.3	talk packet formats .....	103
5.6.4	mTalk packet formats .....	103
5.6.5	Responseless-write tCode values .....	104
6.	TDM Operation .....	109

6.1	TDM Overview.....	109
6.2	TDM Bus .....	109
6.3	Normative references .....	109
6.4	General.....	109
6.5	Rates.....	110
6.5.1	STS-1/OC-1 rate .....	110
6.5.2	Synchronous hierarchical rates .....	110
6.5.3	Frame structure of the STS-1 .....	110
6.5.4	Transport overhead .....	111
6.6	Frame structure of the STS-12.....	112
7.	Maintenance Bus.....	115
7.1	Objectives .....	115
7.2	The interconnection of modules with MTM-Bus .....	116
7.3	Benefits of distributed approach: .....	116
7.4	<b>HiRelPCI P1996 Additional Function Codes .....</b>	<b>117</b>
7.4.1	Command Code 50h - Disconnect module HiRelPCI from Backplane Bus .....	117
7.4.2	Command Code 51h - Connect module HiRelPCI to Backplane Bus.....	117
7.4.3	Command Code 52h - Disable module on board main power converters .....	117
7.4.4	Command Code 53h - Enable module on board main power converters .....	117
7.4.5	Command Code 54h - Read Module Serial Number and NodeID .....	118
7.4.6	Command Code 55h - Read PCI Configuration Space.....	118
7.4.7	Command Code 56h - Write PCI Configuration Space.....	118
7.4.8	Command Code 57h - Read Power Status Block .....	118
7.4.9	Command Code 58h - Address Port and Mode (64 bit mode) .....	118
7.4.10	Command Code 59h - Data Port (64 bit mode) .....	118
7.4.11	Command Code 5Ah - TBD .....	118
7.4.12	Command Code 5Bh - TBD .....	118
7.4.13	Command Code 5Ch - Read 1149.1 JTAG Data/Program Data .....	118
7.4.14	Command Code 5Dh - Write 1149.1 JTAG Data/Program Data .....	119
7.4.15	Command Code 5Eh - Read Message ( Header + Data + CRC) .....	119
7.4.16	Command Code 5Fh - Write Message ( Header + Data + CRC ) .....	119
7.5	HiRelPCI P1996 Module Addressing.....	119
7.6	Data transfer timing .....	121
7.6.1	MTM-Bus Link Layer: Packet requirements .....	122
7.6.2	ACKNOWLEDGE packet requirements .....	123
7.6.3	Packet Count packet requirements.....	124
7.6.4	DATA packet requirements .....	124
7.6.5	NULL packet requirements .....	124
7.6.6	Formatting bit strings of more than 16 bits for transmission in DATA packets .....	125
7.7	Full MTM command set .....	126
8.	Control and Status Register and Configuration Operation .....	129
8.0.1	Addressing .....	129
8.0.2	Requirements .....	129
8.1	Addressing .....	131
8.1.1	64 Bit Fixed Addressing .....	131
8.1.2	P1996 CSR Register Definitions .....	132
8.2	Node Addressing.....	135
8.2.1	Node Addresses. ....	135
8.3	ROM Formats .....	135
8.3.1	First ROM Quadlet. ....	135

9.	Support Structures.....	137
10.	Electrical Specifications .....	139
10.1	Overview.....	139
10.1.1	Dynamic vs. Static Drive Specification.....	139
10.1.2	LVPECL DC Specifications .....	139
10.1.3	PCI AC Specifications.....	139
10.1.4	LVPECL AC Specifications .....	139
10.1.5	Maximum AC rating and device protection .....	140
10.2	Timing Specifications .....	140
10.2.1	TDM Timing Specification.....	140
10.2.2	PCI and TDM Clock Specifications .....	141
10.2.3	TDM Clock Specifications .....	141
10.3	Backplane Specifications.....	141
10.3.1	Backplane Signal Timing Budgets .....	141
10.4	Power .....	141
10.4.1	For power rail A these pins consist of: .....	142
10.4.2	Power Requirements .....	143
10.4.3	Backplane Impedance .....	143
10.5	Connectors .....	143
10.6	Non-CSM Pin Assignments for Main PCI connectors .....	145
10.7	CSM Pin Assignments.....	146
10.8	TDM module pin assignments.....	147
10.8.1	Non CSM TDM Pin Assignments .....	147
10.8.2	CSM TDM Pin Assignments .....	148
11.	Mechanical Specifications .....	149
11.1	Mechanical Philosophy.....	149
11.2	Sub Chassis Specifications .....	149
11.3	Board Formats.....	149
11.4	Connector Specifications .....	150
11.5	Module characteristics .....	151
11.6	Module compatibility considerations.....	151
11.7	12su Module size .....	151
11.8	6su Module size .....	152
11.9	Warpage, bowing, and deflection .....	153
11.10	Cooling.....	154
11.11	Optional Connector Configurations.....	158
11.11.1	I/O Fiber/RF Connectors .....	158
11.11.2	6su 32 Bit only .....	158
11.11.3	12su Single Bus .....	158
11.11.4	12su No TDM .....	158
11.11.5	18su system.....	158

## LIST OF FIGURES

Figure 1—Request packet formats .....	84
Figure 2—Response packet formats .....	86
Figure 3—timeSync packet format .....	87
Figure 4—write16 subaction formats .....	95
Figure 5—read16 subaction formats .....	95
Figure 6—lock16-quadlet subaction formats, accessing address 24 .....	96
Figure 7—lock16-octlet subaction formats, accessing address 16 .....	96
Figure 8—writeLo64/writeHi64 request subaction formats .....	97
Figure 9—readLo64/readHi64 subaction formats .....	98
Figure 10—read64 response formats, 4/8-byte alignment (accessing address 60) .....	100
Figure 11—read64 response formats, 16/32-byte alignment (accessing address 60) .....	100
Figure 12—read64 response format, 64-byte alignment (accessing address 60) .....	101
Figure 13—mMove packet formats .....	102
Figure 14—Basic talk-packet formats .....	103
Figure 15—mTalk-packet format .....	104
Figure 16—Hierarchy of test and maintenance buses .....	117
Figure 17—Backplane MTM-Bus signals .....	119
Figure 18—Sample MTM-Bus message format .....	120
Figure 19—Hold time requirement .....	122
Figure 20—HEADER packet format .....	123
Figure 21—ACKNOWLEDGE packet format .....	123
Figure 22—PACKET COUNT packet format .....	124
Figure 23—DATA packet format .....	124
Figure 24—NULL packet format .....	124
Figure 25—Formatting a bit string of length greater than 16 into multiple DATA packets. ....	125
Figure 26—Packet transfers defining the MTM-Bus message sequences .....	126
Figure 27—Fig 1-1 Byte and Quadlet Ordering???? .....	132



## LIST OF TABLES

Table 1—Maintenance Bus Signals .....	18
Table 2—Basic 64 Bit Fixed Address CSR Architecture .....	21
Table 3—Maintenance Bus Commands .....	22
Table 4—CSR Register Space Address Allocations .....	25
Table 5—Non-CSM Position Connector Pin Assignments .....	27
Table 6—CSM POSITION CONNECTOR Additions .....	29
Table 7—CSM Connector .....	29
Table 8—A Power Rails .....	31
Table 9—B Power Rails .....	31
Table 10—Power Pin Locations on connectors .....	31
Table 11—1 .....	71
Table 12—Data-transfer services .....	75
Table 13—Resource Map Table .....	78
Table 14—System Resource Map Entry .....	79
Table 15—Data-transfer services .....	82
Table 16—type (packet type) field values .....	85
Table 17—ph (retry phase) values .....	85
Table 18—tCode assignments, normal (mt=0) subactions .....	89
Table 19—lock4 subcommand values .....	91
Table 20—stat field values .....	93
Table 21—tCode assignments, responselss writes, mt=1 .....	105
Table 22—STS and OC data rates .....	110
Table 23—Maintenance Bus Commands .....	127
Table 24—Basic 64 Bit Fixed Address CSR Architecture .....	132
Table 25— .....	133
Table 26—LVPECL DC Parameters .....	139
Table 27—LVPECL I/O Buffers DC Parameters .....	140
Table 28—LVPECL AC Specifications .....	140
Table 29—TDM Clock Specifications .....	141
Table 30—Timing Budgets .....	141
Table 31—Pin Stagger .....	144
Table 32—Non CSM Pin Assignments .....	145
Table 33—CSM PIN Assignments .....	146
Table 34—NonCSM TDM Pin Assignments .....	147
Table 35—CSM TDM Pin Assignments .....	148



## 1. Introduction to HiReIPCI

This is an introduction to a IEEE P1996 to define a:

### Standard for an Extensible High Reliability Enhanced PCI Bus.

This bus came from a need to provide a High Reliability, High Availability system for transportation control and telecommunications systems. Transportation and traffic control have a need for very low system failure rate in difficult environments. These system must operate on the street corners from northern Alaska to the southern deserts of Arizona with the extremes of temperature and humidity prevalent to such locations. The telecommunications requirements are not quite as severe in the environmental area, but are similar in the need for R.A.M. (Reliability, Availability and Maintainability) plus the additional need of a Time Division Multiplexed bus for circuit switched data using elements of the international Synchronous Digital Hierarchy (SDH) at rates defined in ANSI T1.105-1991. Both areas of usage need redundant capabilities to allow continued operation with a single fault. Additional areas of interest for this project are Process Control, Communications, and embedded systems. Maintenance functions utilize the MTM bus structure as defined in IEEE Std 1149.5-1995 with compatible extensions.

These application areas need a method of extending beyond the local bus to other similar buses in a uniform, redundant manner. The solution to this need was to include a provision in the bus to handle the message packets of the IEEE Std 1394-1995, IEEE P2100 Serial Express and IEEE Std 1596-1992 SCI. This addition provides for 64K addressable nodes and a uniform 64 bit address space. In more demanding multiprocessor systems, the cache coherency methods of SCI can be employed as needed.

This bus is designed to satisfy these needs.

#### FEATURES OF THIS HiReIPCI BUS

The following set of features of the P1996 HiReIPCI Bus drive its utility:

- Reliability - Through passive backplane, Redundant Buses, Redundant power
- Availability - Through design for NO Single Point of Failure
- Maintainability - Through MTM bus, Hot Swap, Standby modules
- Scalability - Through PacketBus to P2100 and SCI, Circuit Switch I/O through STS-12
- Performance - PacketBus performance to 533 MByte/second with up to 95.5% efficiency
- Redundancy - no single point of failure on a normally configured system.
- 40C to +85C Industrial/Automotive temperature range operation
- Power Distribution at nominal 48V for highest efficiency
- Configuration Management through CSR Architecture, Serial Express and MTM busses
- Redundant connectivity through backplane Firewire and 10Base2 Ethernet

#### 1.1 ELEMENTS OF HiReIPCI BUS

This bus standard is a combination of elements taken from other elements and modified to provide the structure to meet the needs described above. The elements that make up this high reliability, high availability bus include:

1. A Parallel system bus using PCI style signaling for normal bus access
2. A PacketBus added on top of the Parallel system bus for packet switched connections
3. A Time Domain Multiplexed bus for circuit switched connections
4. A maintenance bus for control of system resources for hot swap operations
5. An Addressing structure that uses the standard 64 bit address space
6. Several supporting interconnection systems
7. A set of electrical specification for low voltage signaling, 3.3V PECL signaling and power

8. A set of mechanical specifications to provide a well defined enclosure system

The combination of these elements build a system that provides:

1. Extensibility to 64K nodes of 256 Terabytes each
2. IEEE 1212-1991 CSR Architecture for easy expansion via SCI, Serial Express, 1394-1995 etc.
3. Redundant functions to provide no single point of failure for continuous operation
4. Hot Plug of any board to replace faulty modules or update features.
5. Circuit Switched base for Audio, Telephony, Video, ATM for consistent external/internal model
6. Packet Switched connection model for shared memory operations via SCI/CSR internal/external

## 1.2 Bus Signal Definitions

Nominal bus signaling on the P1996 is based on the PCI Local Bus Specification 2.1 using 3.3V signal levels. This provides leverage on all the silicon built to support the PCI market. This protocol is flexible and provide performance from 0 - 533 megabytes per second with the addition of Packet Write burst transactions.

This bus has been extended to include a packet mode, called PacketBus, to support the packets sent over the SCI/Serial Express busses to extend transparent communications to other elements of a distributed, redundant system. Packet operations are transparent to the normal PCI transactions compliant to Rev 2.1. HiRelPCI bus can operate in PacketBus, normal PCI mode or a mix of both modes. PacketBus mode of operation can be used to expand the normal PCI functions and has been proposed to the PCI technical committee for inclusion in the future PCI Local Bus specifications.

Transparent operation is accomplished with the addition of a command mode on the PCI that is a write only mode. In this mode the packet is a write on the bus with the high order 16 bits being used as an address for one or more nodes to capture or indicate the failure to capture. Write Only operations are supported in both 32 and 64 bit wide operations that will deliver the bit serial equivalent of 4.266 gigabits per second.

Today's limitation on the driver technology and the physics of the backplane restrict the number of backplane segment positions to 7, or 8 when stretched. Extensibility is required to extend the operation to additional nodes in systems attached through SCI/Serial Express links.

Each component in the system is attached to the Maintenance Bus which is a bus defined by the IEEE Std 1149.5-1995 documents. Maintenance Bus signal lines are:

**Table 1—Maintenance Bus Signals**

Signal Name	Signal Function	Signal Direction
MCLK	MTM Clock	To all from CSM
MMD	Master Data	From Master
MSD	Slave Data	To Master
MPR	Master Pause Request	To Master
MCTL	Master Control	From Master

## 1.3 Parallel Bus Operations

In addition to the normal PCI style operation, additions and deletions have been made to improve the operation of the bus. Specific additions have been the prioritized bus request facilities, PacketBus, the MTM maintenance bus, 48VDC power, constant clock, backplane Ethernet and Firewire. Specific deletions are INTA#, INTB#, INTC# and INTD#, 5VDC and 3.3VDC power.

### 1.3.1 CSM Functions

Centralized resources needed for bus operations are supplied by the Central Services Module. There shall be one CSM for each bus segment. In most cases, a CSM will span two HiRelPCI buses in the 12su configurations and only one bus in the 6su configuration. This module provide all the clocking and arbitration for the parallel bus and the TDM bus when present.

The Central Services Module provides the following services for the bus:

- Clock drivers for each slot on the bus segment
- Bus Arbitration for each slot on the bus segment
- Repeater Hub for the Ethernet signals
- Repeater Hub for the 1394 Firewire serial bus
- Possible Bridge to IEEE P2100
- PCI to PCI Bridge if needed
- Possible TDM interface between outside world and backplane bus
- TDM to TDM Bridge and framing store if needed
- Clocks for the TDM slots when used
- System monitor functions
- Maintenance Bus Master

### 1.3.2 Priority Bus Request Mechanism

This provides a prioritized request in addition to the normal round robin Fairness priority system of the PCI bus. This priority scheme uses four pins REQ0# to REQ3# to define the priority level of each bus request as level 0 to 15. Bus request are issued at the rising edge of FRAME# by activating their REQ# and priority level on REQ[3:0]#. The priority level on the bus for this cycle is seen by all participants. Lower level requesters will retire and remove the REQ# if a higher level request is pending. The CSM arbiter determines the winner or winners and issues the bus grants to the boards with the highest level of priority on a round robin basis and then to the next lower level of priority on a round robin basis until there are no pending bus requests.

Prioritized bus requests may have 1 to 8 requests pending at any given level of request, the requests are granted in round robin priority until all are issued while building a queue at the same level to insure fairness within the bus request queueing level. This also guarantees priority among the bus request queueing levels.

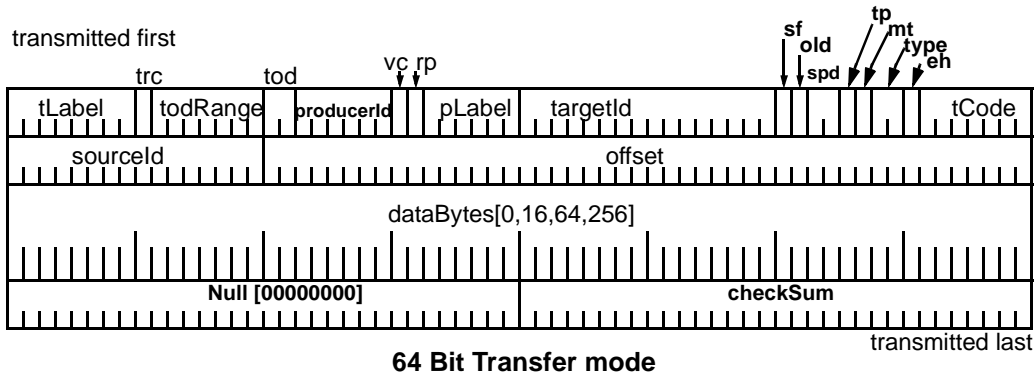
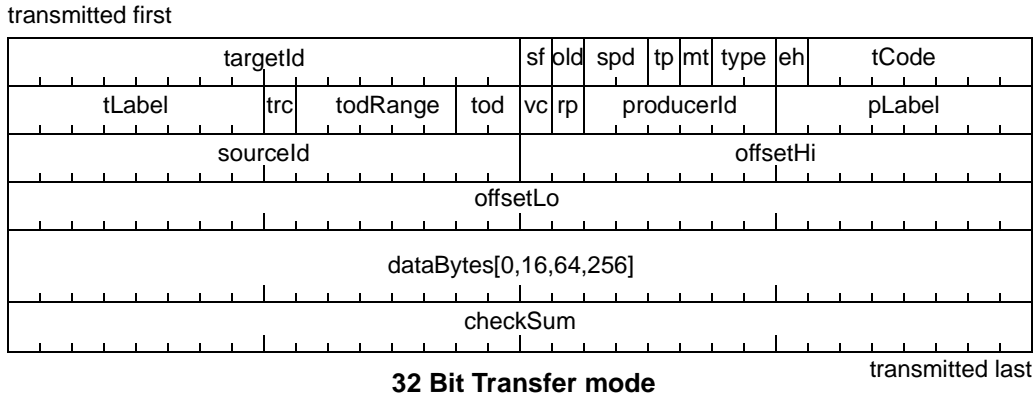
Boards using the PCI style single REQ# line will request at bus request level 0. Any higher level of request will block the lower level request. This mechanism is transparent to the single level request mechanism and will be seen as a possibly longer delay until a GNT# is received.

## 1.4 PacketBus Operations

PacketBus operations are added to the basic parallel bus structure to improve the performance of the system with a split response mode of operation. This mode is also known as a write only mode of bus operation in that each request and response is a write transaction. The basic function is to free the bus resource during the time needed to fetch or process the data at the target end of the transaction. Packet mode operation does impose additional requirements on the participants as buffers are needed to handle the packets to and from the bus.

PacketBus operation is optional for the HiRelPCI bus. This mode can be used to produce the highest bandwidth operations on the bus with greater than 4 gigabit throughput a reality.

PacketBus operations require the use of a new command mode in the PCI specification and the redefinition



of several of the PCI signal lines during this mode of operation. In all cases these changes are transparent to the existing PCI devices.

The preferred command mode is the "0101" mode which would be an unused WRITE transaction. Current devices are required by PCI Local Bus Specification Revision 2.1 to ignore this command code and hence no interference with the existing operations. These packets are of the general form shown the Figure above.

A command field identifies the different forms of send packets, as described in the chapter on the Packetbus.

### 1.4.1 CSR Space Mapping

This standard adheres to the IEEE Std 1212-1992 Control and Status Register Architecture standard using the 64bit fixed address model.

Within the node the address is initially defined in the CSR (IEEE Std 1212 - 1992) as shown in Table 2 below.

## 1.5 TDM Bus Operations

Dual TDM busses are included as an option for Telecommunications usage. These busses share a

**Table 2—Basic 64 Bit Fixed Address CSR Architecture**

<b>Begin Address nnnn = nodeId</b>	<b>End Address nnnn = nodeId</b>	<b>Function</b>
nnnn 0000 0000 0000	nnnn 0000 0FFF FFFF	Register Address Space
nnnn 0000 1000 0000	nnnn 0000 1FFF FFFF	Node Private Address Space (256M)
nnnn 0000 0000 0000	nnnn 0000 0FFF F7FF	Initial Units Address Space (256M-2K)
nnnn 0000 0FFF F800	nnnn 0000 0FFF FFFF	Initial Register Address Space (2K)
nnnn 0000 2000 0000	nnnn FFFF FFFF FFFF	Initial Memory Space (281 Tbytes- 2 <sup>40</sup> )

common clock and frame signal. Each TDM block consists of 6 ground pins, 1 clock pin, 1 frame pin and 2 sets of 8 data lines. The clock pin is routed individually to each slot to provide tight timing specification.

This bus will follow the structure of ANSI T1.105-1991 SONET (as revised) using the blocking factors used in the international standard. This standard provides a reference for multiple embedded carriers and provides an internationally accepted framework for circuit presentation and ATM integration if needed. Basic timing is provided by the CSM board position and consists of clocks with less than 1 ns of skew and a framing pulse to define the local chassis time slot count. Adjustment may be made to synchronize with one external source of framing information. The first 24 time slots are used for synchronization purposed and to move the relative frame time on different parts of the system to be in coordination with the backplane requirements though frame buffers and other necessary techniques.

The overhead portions of the ANSI SONET standard are not yet specified

Simulations indicate possible clock speeds to 77.76 MHz using 3.3V PECL logic to provide an upper limit of STS12 on each of the sets of data lines.

12su systems would have dual TDM blocks for reliable redundant operation. Each of these busses has 2 byte wide paths that may be used as input and output streams and are each readable and writable on each time period. Each cycle is 12.86 ns in duration which requires terminated 3.3V PECL bus to reduce reflections and lower the propagation delay to incident wave travel.

## 1.6 Maintenance Bus Operations

The Maintenance Bus Master issues sets of commands that are fully described in the IEEE Std 1149.5-1995. These commands are summarized in the following table.

In addition to the normal commands provided by the 1149.5 command set, the command extension for HiRelPCI are added as commands 50 to 5F.

The Maintenance Bus (abbreviated MTM in this document) is attached to all elements in a common chassis. This extends to all boards, power supplies and fans that support a physical backplane which may contain up to 2 segments. The IEEE Std 1149.5 MTM bus that uses 5 signal lines and has been defined for use in Avionics an other similar systems use by the same standards group the provided the IEEE 1149.1 JTAG testing functions.

Required functions:

- Read Board Unique Identifier
- Read Board Configuration ROM (or other device)
- Read Board Slot Identifier (SAD lines)
- Connect and disconnect Board from Bus
- Control Power on board

**Table 3—Maintenance Bus Commands**

Command Class	Command Code (Binary)	Command Code (HEX)	Command	Status
Core	0000000	00	Read Status	Required
	0000001	01	Abort	Required
	0000010	02	Reset Slave Status	Required
	0000011	03	Contend for Bus	Required
	00001XX	04-07	Multicast Group Select	Required
	0001000	08	Enable Idle Interrupts	Required
	0001001	09	Enable Pause Interrupts	Required
	0001010	0A	Disable Idle Interrupts	Required
	0001011	0B	Disable Pause Interrupts	Required
	0001100	0C	Enable Module Control	Required
	0001101	0D	Data Echo Test	Required
	0001110	0E	Verify BMR	Required
	0001111	0F	Initialize Application	Required
	0010000	10	Disable Module Control	Required
	0010001	11	Start	Required
	0010010-0011111	12-1F	Reserved	Reserved
	1111111	7F	Illegal Command	Required
Data Transfer	0100000	20	Read Data	Recommended
	0100001	21	Write Data	Recommended
	0100010	22	Read/Write Data	Recommended
	0100011-0101111	23-27	Reserved	Reserved
Module Initialization and Self-Test (MIST)	0101000	28	Reset Module with SBIT	Recommended
	0101001	29	Reset Module without SBIT	Recommended
	0101010	2A	Module IBIT	Recommended
	0101011-0101111	2B-2F	Reserved	Reserved
Module I/O Control and Test (MICT)	0110000	30	Disable Module I/O	Recommended
	0110001	31	Enable Module I/O	Recommended
	0110010	32	Force Module Outputs	Recommended
	0110011	33	Sample Module-No Change	Recommended
	0110100	34	Sample Module-Don't Care	Recommended
	0110101	35	Sample Module with Force	Recommended
	0110110	36	Release Module I/O	Recommended
	0110111-1001111	37-4F	Reserved	Reserved
Standard Extension	1010000-1011111	50-5F	Reserved for use of standards making bodies	Reserved
HiRelPCI Extensions	1010000	50	Disconnect module from HiRelPCI Bus	Required
	1010001	51	Connect Module to HiRelPCI Bus	Required
	1010010	52	Disable main power converters	Required
	1010011	53	Enable main power converters	Required
	1010100	54	Read Serial Number, NodeId	Required
	1010101	55	Read PCI configuration space	Required
	1010110	56	Write PCI configuration space	Required
1010111	57	Read Power Status Block	Required	

**Table 3—Maintenance Bus Commands**

Command Class	Command Code (Binary)	Command Code (HEX)	Command	Status
	1011000	58	Set Poke Address and mode (64 bits)	Required
	1011001	59	Read Poke Data (64 Bit mode)	Required
	1011010	5A	Write Poke Data (64 bit mode)	Required
	1011011	5B	TBD	Required
	1011100	5C	Read 1149.1 JTAG data	Required
	1011101	5D	Write 1149.1 JTAG Data/Program Data	Required
	1011110	5E	Read Message (Header + Data + CRC)	Required
	1011111	5F	Write Message (Header + Data + CRC)	Required
User-Defined	1100000-1111110	60-6E	User-Defined Commands	Reserved

- Perform Built In Self Test (BIST) functions
- Perform the JTAG Testing of the board
- Read/Write IEEE 1212-1991 Address space
- Control diagnostic systems testing such as induced failures
- Other functions as defined by the board designer

## 1.7 CSR Specifications and Configuration Operations

### 1.7.1 Node Addressing

Extending beyond a single PCI style bus is required for applications that need more or vastly more processors, memory and I/O boards. Redundant operations also require more than one processor, memory and I/O board. Connectivity beyond a single system is greatly enhanced by addressing a card slot on a bus segment as a node. Each slot has a node address consisting of 16 bits, split into 3 bits of slot-id and 13 bits of address assigned by a switch on the backplane or by a bridge to the SCI/P2100 node address. While this does allow 8192 bus segments of 8 slots/nodes each, the use of all slots within a backplane environment will define up to 16 slots/nodes for each chassis and limit the number of chassis to 4096. P2100 bus addressing which is defined around 1024 buses of 64 nodes each and makes specific allowances to support more busses of less nodes. All addressing for normal operation is directed to 16 bit nodeId which are routed to the proper bus and node.

These 16 pins are distributed along the connector and bypassed to ground, or connected to ground to provide a high quality signal return path.

### 1.7.2 P1996 CSR Register Definitions

The CSR registers are at fixed addresses offset relative to  $nnnn\ 0000\ 0FFF\ F800_{16}$  as shown in Table 4

**Table 4—CSR Register Space Address Allocations**

Offset into CSR REGS	Offset in Hex nnnn 0000 0FFF F800	Register Name	Required
	0	STATE_CLEAR	
	4	STATE_SET	
	8	NODE_IDS	Required
	C	RESET_START	Required
	10	INDIRECT_ADDRESS	
	14	INDIRECT_DATA	
	18	SPLIT_TIMEOUT_HI	
	1C	SPLIT_TIMEOUT_LO	
	20	ARGUMENT_HI	
	24	ARGUMENT_LO	
	28	TEST_START	
	2C	TEST_STATUS	
	30 - 4C	Reserved for Extended Memory Space	
	50	INTERRUPT_TARGET	
	54	INTERRUPT_MASK	
	58	CLOCK_VALUE_HI	
	5C	CLOCK_VALUE_MID	
	60	CLOCK_TICK_PERIOD_MID	
	64	CLOCK_TICK_PERIOD_LO	
	68	CLOCK_STROBE_ARRIVED_HI	
	6C	CLOCK_STROBE_ARRIVED_MID	
	70 -7C	CLOCK_INFO(0-3)	
	80 - BC	MESSAGE_REQUEST	
	C0 - FC	MESSAGE_RESPONSE	
	100 -180	Reserved	
	180 - 1FC	ERROR_LOG_BUFFER	
	200	PCI DeviceID VendorID Register 00h	Required
	204	PCI STATUS COMMAND Register 04h	Required
	208	PCI ClassCode RevisionID Register 08h	Required
	20C	PCI INFO Register 0Ch	Required
	210	PCI Base_Address_0 Register 10h	
	214	PCI Base_Address_1 Register 14h	
	218	PCI Base_Address_2 Register 18h	
	21C	PCI Base_Address_3 Register 1Ch	
	220	PCI Base_Address_4 Register 20h	
	224	PCI Base_Address_5 Register 24h	
	228	PCI CardBus CIS Pointer Register 28h	
	22C	PCI SubSystemID SubSystem VendorID 2Ch	
	230	PCI Expansion ROM Base Address Register 30h	
	234	PCI Reserved Register 34h	
	238	PCI Reserved Register 38h	
	23C	PCI Grant Interrupt Info 3c	
	240 - 2FC	PCI REGISTER EXPANSION	

**Table 4—CSR Register Space Address Allocations**

Offset into CSR REGS	Offset in Hex nnnn 0000 0FFF F800	Register Name	Required
	300 - 307	MTM Fault Logs	Required
	308 - 30F	MTM Test Data Storage	
	310	MTM Slave Status Register	Required
	314	MTM Bus Error Registers	Required
	318	MTM Module Status Register	Required
	31C	MTM Additional Status Registers	
	320	MTM Manufacturers ID Port	Required
	324	MTM Module Manufacturer Port	Required
	328 - 32F	MTM User Identification Ports	Required
	330 - 34F	MTM Access to IEEE Std 1149.1 bus	Required
	350	MTM Command Register	
	354 - 35B	MTM Command Data Pointer	
	380 - 3BF	Voltage/Current/Temperature Storage	
	3C0 - 3C7	Pointer to Primary Resource Table	
	3C8 - 3CF	Pointer to Secondary Resource Table	
	3D0 - 3D7	Pointer to Primary Configuration Manager	
	3D8 - 3DF	Pointer to Secondary Configuration Manager	
	3E0 - 3FF	Reserved	
	400 - 7FC	ROM_WINDOW	Required

where nnnn is the 16 bit address of the node.

IEEE 1212-1991 addressing is used with these 16 bits defining a slot as one of the 65536 in the address space. Each node has a sub address range of 48 bits for total address space of 64 bits.

## 1.8 Supporting structures

### 1.8.1 Serial Interconnect

In addition to the primary bus, there are additional communication systems on the backplane. These include a 10 Megabit Ethernet (10Base2), and lines reserved for IEEE 1394-1995 Firewire.

The Maintenance Bus also provides a very low speed interconnect between the boards and support equipment for the purpose of maintaining and testing the bus systems. This bus is an implementation of the IEEE 1149.5 MTM bus. This bus is described later in the chapter on the Maintenance Bus.

Ethernet provides communications between the boards in the system and through a hub located on the Central Service Module (CSM) to the outside world for a low cost connection between bus systems.

Firewire (IEEE 1394-1995) can provide read/write capability to the boards in the system when a redundant path is needed with a single main bus. Boards that use this system of redundancy would all need to support the 1394 interface. Firewire is a multimaster bus and can be used when more than one processor is available.

IEEE P2100 is added to the system through a bridge in one or more of the bus slots. As P2100 devices become available, the bridge may be added to the CSM function.

## 1.8.2 Support Signals

Support signals on the backplane include:

CCLK10M - 10.0000 MHz Constant Clock line (accuracy to be determined)  
 REQ0# - REQ3# - Bus Request Priority Request Pins.

CCLK10M, Constant clock line, has a frequency of 10.00 MHz and is used as a reference frequency. This frequency may also be used as the clock for the IEEE Std 1149.5 MTM maintenance bus. The bus clock in this system has a frequency range for 0 to 66.66Mhz and may use the 0 Hz for debugging, and power conservation functions, and can not be counted on as present for board level functions.

REQ0# - REQ3# are the request lines for the priority request system for the bus. These signals work in conjunction with the PCI REQ# signal and the FRAME and IRDY signals to establish 16 levels of bus request queing with round robin scheduling within each que and to prevent starvation at lower priority levels.

## 1.9 Electrical Specifications

Several different specification are used within this standard: 1) the LVTTTL 3.3V signaling levels of the main PCI style parallel structure; 2) 3.3V PECL signaling for the TDM bus; 3) Ethernet ECL signal levels for the ethernet and 4) Firewire signal levels for the IEEE 1394-1995 bus.

### 1.9.1 Connector Pin Assignments

The first pinout shown in Table 3-? is for the normal bus module that connects to any position on  
**Table 5—Non-CSM Position Connector Pin Assignments**

	a	b	c	d
1	P48VA	FGND	P48VA	P48VA
2	P48VA	FGND	N48VA	N48VA
3	N48VA	P48VAPRE	N48VA	ISHAREA
4	P48VB	N48VAPRE	P48VB	ISHAREB
5	P48VB	P48VBPRE	P48VB	N48VB
6	N48VB	N48VBPRE	N48VB	N48VB
7	ENET	GND	1394CLK	1394DATA
8	AD32	GND	AD33	SAD15
9	AD34	GND	AD35	AD36
10	AD37	GND	AD38	AD39
11	AD40	GND	AD41	SAD14
12	AD42	GND	AD43	AD44
13	AD45	GND	AD46	AD47
14	AD48	GND	AD49	SAD13
15	AD50	GND	AD51	AD52
16	AD53	GND	AD54	AD55
17	AD56	GND	AD57	SAD12
18	AD58	GND	AD59	AD60
19	AD61	GND	AD62	AD63
20	PAR64	GND	C/BE4#	SAD11
21	C/BE5#	GND	C/BE6#	C/BE7#
22	ACK64#	GND	REQ64#	CCLK10M
23	AD00	GND	AD01	SAD10
24	AD02	GND	AD03	AD04
25	AD05	GND	AD06	AD07
26	C/BE0#	GND	M66EN	SAD09
27	AD08	GND	AD09	AD10
28	AD11	GND	AD12	AD13
29	AD14	GND	AD15	SAD08

**Table 5—Non-CSM Position Connector Pin Assignments**

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
30	C/BE1#	GND	PAR	SERR#
31	SBO#	GND	SDONE	PERR#
32	LOCK#	GND	STOP#	SAD07
33	IRDY#	GND	TRDY#	DEVSEL#
34	FRAME#	GND	C/BE2#	AD16
35	AD17	GND	AD18	SAD06
36	AD19	GND	AD20	AD21
37	AD22	GND	AD23	IDSEL
38	C/BE3#	GND	AD24	SAD05
39	AD25	GND	AD26	AD27
40	AD28	GND	AD29	AD30
41	REQ#	GND	AD31	SAD04
42	SAD03	GND	TX+	RX+
43	GNT#	GND	TX-	RX-
44	SAD01	GND	MCLK	SAD02
45	CLK	GND	REQP0#	SAD00
46	REQP1#	GND	REQP2#	RST#
47	MMD	GND	MSD	REQP3#
48	MPR	GND	MCTL	Spare5#
49	user defined	user defined	user defined	user defined
50	user defined	user defined	user defined	user defined
51	user defined	user defined	user defined	user defined
52	user defined	user defined	user defined	user defined
53	user defined	user defined	user defined	user defined
54	user defined	user defined	user defined	user defined

the bus other than for the CSM module position, the use of rows 49 to 54 are define by the board and connect through on the backplane to provide user I/O with a flexible connection

Central Service modules require the additional pins to provide the clocks and arbitration for the bus

**Table 6—CSM POSITION CONNECTOR Additions**

	a	b	c	d
48				BPSN#
49	CLKS0	CLKS6	REQ0#	GNT0#
50	CLKS1	GNT6#	REQ1#	GNT1#
51	CLKS2	REQ6#	REQ2#	GNT2#
52	CLKS3	GND	REQ3#	GNT3#
53	CLKS4	PUPOWER	REQ4#	GNT4#
54	CLKS5	PUPOWER	REQ5#	GNT5#

segment of up to 8 slots. The above chart shows the changed definitions for a 7 slot backplane entity.

On 12su boards an additional set of 72 pins are located in the center of the board and are defined for the TDM bus and the service modules clocks for the TDM bus and the REDUNDANT TDM structure. The pinouts for this connector are:below.

**Table 7—CSM Connector**

	a	b	c	d
1	ATDM0	GND	ATDM1	ATDM2
2	ATDM3	+1.3VTTA	ATDM4	ATDM5
3	ATDM6	GND	ATDM7	ATDM8
4	ATDM9	ATDM12	ATDM10	ATDM11
5	ATCLK6+(in)	ATCLK6+(out)	ATDM13	ATDM14
6	ATCLK6-(in)	ATCLK6-(out)	AFRAME	ATDM15
7	ATCLK0+(out)	ATCLK1+(out)	BTCLK0+(out)	BTCLK1+(out)
8	ATCLK0-(out)	ATCLK1-(out)	BTCLK0-(out)	BTCLK1-(out)
9	ATCLK2+(out)	ATCLK3+(out)	BTCLK2+(out)	BTCLK3+(out)
10	ATCLK2-(out)	ATCLK3-(out)	BTCLK2-(out)	BTCLK3-(out)
11	ATCLK4+(out)	ATCLK5+(out)	BTCLK4+(out)	BTCLK5+(out)
12	ATCLK4-(out)	ATCLK5-(out)	BTCLK4-(out)	BTCLK5-(out)
13	BTDM0	GND	BTDM1	BTDM2
14	BTDM3	+1.3VTTB	BTDM4	BTDM5
15	BTDM6	GND	BTDM7	BTDM8
16	BTDM9	BTDM12	BTDM10	BTDM11
17	BTCLK6+(in)	BTCLK6+(out)	BTDM13	BTDM14
18	BTCLK6-(in)	BTCLK6+(out)	BFRAME	BTDM15

The second set is for the normal board case of not being in the CSM position. This connector is

	a	b	c	d
1	ATDM0	GND	ATDM1	ATDM2
2	ATDM3	NC	ATDM4	ATDM5
3	ATDM6	GND	ATDM7	ATDM8
4	ATDM9	ATDM12	ATDM10	ATDM11
5	ATCLK6+(in)	NC	ATDM13	ATDM14
6	ATCLK6-(in)	NC	AFRAME	ATDM15
7	user-defined	user-defined	user-defined	user-defined
8	user-defined	user-defined	user-defined	user-defined

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
9	user-defined	user-defined	user-defined	user-defined
10	user-defined	user-defined	user-defined	user-defined
11	user-defined	user-defined	user-defined	user-defined
12	user-defined	user-defined	user-defined	user-defined
13	BTDM0	GND	BTDM1	BTDM2
14	BTDM3	NC	BTDM4	BTDM5
15	BTDM6	GND	BTDM7	BTDM8
16	BTDM9	BTDM12	BTDM10	BTDM11
17	BTCLK+(in)	NC	BTDM13	BTDM14
18	BTCLK-(in)	NC	BFRAME	BTDM15

shown below. Additional configuration options are under consideration that will allow for more rear I/O from the board. Within the 24pin blocks additional configurations are available the include single RF or Fiber I/O and up to 6 RF/Fiber connectors per block using smaller diameter coax/fiber. These configurations include:

1. 6su board with only 32 bits of address and data (opens up about 36 I/O lines)
2. 12su board with out redundant bus or TDM (frees middle connectors and lower connectors.
3. 6su and 12su boards with mechanical keying to support the use of Fiber and RF connectors.
4. 18su and 24su boards with additional sets of connectors.

Part of the effort here is to prevent these boards from plugging into destructive conditions. For this purpose the board that have RF or Optical connectors in place of the normal pin I/O will have connectors keyed to prevent the insertion of the normal pin I/O into the backplane positions with the RF/OPTICAL connectors

### 1.9.2 Power Distribution

The power paradigm for this system is the distribution of redundant wide tolerance main voltage with local voltage regulation as needed. The distribution power rails are DC isolated from the incoming power supply to meet isolation requirements of 3750V with required creepage and clearances distances. It is also expected that the onboard regulators will also be DC isolated from the power rails with the only common element from node to node being signal ground.

This power is distributed through dual power rails, each rail can deliver up to 4 Amps of power at a nominal voltage of 48V to each board slot. This nominal 48V supply has a range of 36V to 58V. This stays within the definitions of “Safety Extra Low Voltage” that will comply with IEC 950 and EN 60 950 for European requirements and UL1450 for US requirements. This voltage provides sufficient power while reducing the current density in the connector pins. In telecommunication system the voltage would be a nominal 52.8V to 55V with normal batteries under charging conditions.

Additional precharge lines are located on longer pins to bring the input circuits up to voltage before the main power pins connect. The precharge pins are current limited to 100 mA for safety on the long pins. The last pins to make contact on each set of power rails is the power-on pins to start the onboard power converters.

Power rail A contain 11 pin:

Power rail B contain 11 pin:

The remaining two pins in the first block of 24 pins are FGND = Frame Ground. This ground is used on a strip around the board and contacts the grounding clips on the card guides. Frame Ground is isolated from common signal ground. Overall the power block of 24 pins looks like this:

**Table 8—A Power Rails**

Positive Rail	Negative Rail	Voltage Difference	Maximum Current	Number of pins
P48VA	N48VA	36VDC to 58VDC	4 Amperes	4 pins per Rail
P48VAPRE	N48VAPRE	36VDC to 58VDC	0.1 Ampere	1 pin per Rail
ISHAREA	N48VA	0 to 15VDC	20 mA	1 pin

**Table 9—B Power Rails**

Positive Rail	Negative Rail	Voltage Difference	Maximum Current	Number of Pins
P48VB	N48VB	36VDC to 58VDC	4 Amperes	4 pins per Rail
P48VBPRE	N48VBPRE	36VDC to 58VDC	0.1 Ampere	1 pin per Rail
ISHAREB	N48VB	0 to 15VDC	20 mA	1 pin

**Table 10—Power Pin Locations on connectors**

Row\Column	a	b	c	d
1	P48VA	FGND	P48VA	P48VA
2	P48VA	FGND	N48VA	N48VA
3	N48VA	P48VAPRE	N48VA	ISHAREA
4	P48VB	N48VAPRE	P48VB	ISHAREB
5	P48VB	P48VBPRE	P48VB	N48VB
6	N48VB	N48VBPRE	N48VB	N48VB
<b>Length</b>	<b>6.5 mm</b>	<b>8 mm</b>	<b>6.5 mm</b>	<b>5.75 mm</b>

## 1.10 Mechanical Specifications

### 1.10.1 Mechanical Sizes

This standard uses the Hard Metric mechanical system with the following features:

- IEC 917-2-2/IEEE Std 1301-1992 standard dimensions

- Board top surface offset 10 mm from left reference on module (from the front panel)

- Board position allows front and back component placement (7.23mm back, 19mm front)

- 30 mm front panel allows 1 inch disk drives on boards

SEME Format for mobile applications:

- US DOD MilStd xxxx

- IEEE Std 1301.xx

#### 1.10.1.1 HiRelPCI Connectors

2mm Connectors - FutureBus style - Metral style

Bellcore, UL, CSA approved

- IEC 1076-4-OX (48B) compliant
- EIA 616 compliant
- Stackable connectors for I/O
- 8x24 short pin = Berg Part Number 70235-977 or equivalent
- 1x24 short pin = Berg Part Number 70232-977 or equivalent
- 1x24 Long pin for rear plug = Berg Part Number 70232-987 or equivalent
- Standard RF and Optical I/O available for I/O connector positions
- Keying available to prevent damage when mixing RF and pin I/O
- Staggered Pin Height to support Live Insertion
  - A Row = 6.5mm
  - B Row = 8.0mm
  - C Row = 6.5mm
  - D Row = 5.75mm
- Signal return path for every 2 signals.
- 6su boards contain
  - 216 pins - 4 column of 54 pins including 24 user defined I/O pins
- 12su boards contain
  - 504 pins - 4 columns of 126 pins including 72 user defined I/O pins and TDM blocks

Pin configurations for the SEME format TBD.

#### 1.10.1.2 HiRelPCI board formats

- 6su HiRelPCI
  - 6su = 115mm (4.53") x 213mm depth (8.39")
- 12su HiRelPCI
  - 12su = 265mm (10.43") x 288mm depth (11.34")
- SEME HiRelPCI
  - Board Format specified by MIL Std xxxx

#### 1.10.1.3 HiRelPCI board size comparison to VME style boards

- 6su HiRelPCI ~ 3u VME style
  - Board Size - 115mm x 213mm vs. 100mm x 160mm
  - Board Area - 245 sq. cm vs. 160 sq. cm raw size
  - Usable Area - 430 sq. cm vs. 140 sq. cm component area
- 12su HiRelPCI ~ 6u VME style
  - Board Size - 265mm x 288mm vs. 233.35mm x 160mm
  - Board Area - 763 sq. cm vs. 374 sq. cm raw size
  - Usable Area - 1416 sq. cm vs. 336 sq. cm component area

#### 1.10.2 Environmental

P1996 is intended to operate in the following environmental conditions:

- Temperature -40C to +85C
- Humidity 0 to 100%
- Altitude -100m to +15000m

Cooling

- Convection cooling for transportation and other severe environmental applications.
- Forced air cooling may be used in telecom and other less demanding environmental applications.

#### 1.11 IEEE PAR TITLE

Standard for an Extendible High Reliability Enhanced PCI Bus

### **1.12 IEEE PAR PURPOSE of P1996 HiReIPCI**

Purpose: The transportation, telecommunications, and process control industries need reliable high availability fault tolerant systems that support harsh environments and extended temperature ranges, with packet protocols that support serial interconnections to similar systems at speeds from 10 megabits/sec to >1 gigabit/second. There is no currently defined standard for such systems that can take advantage of the enormous (and still growing) industrial support for the PCI system by way of silicon, software, and tools. This standard is intended to fill that need.

### **1.13 IEEE PAR SCOPE of P1996 HiReIPCI**

Scope: This project will develop a Standard for an Extendible High Reliability Enhanced PCI Bus for transportation, telecommunications, and process control systems. This bus will define and use PCI-style protocol, extending it to include packet messages as defined in the SCI(IEEEstd 1596-1992) and Serial Express (P2100) projects. This bus will include redundancy and support hot swap of boards for fault tolerant, high availability operation. This project will include the physical and electrical implementation of this bus and define the interface to a high speed serial interconnect such as the Serial Express bus. For telecommunication applications, an optional Time Division Multiplexed bus using international signaling rates from the Synchronous Digital Hierarchy will be defined.



## 2. References, definitions, and notation

### 2.1 References

This standard shall be used in conjunction with the following publications. When they are superseded by an approved revision, the revision shall apply:

ANSI/ISO 9899-1990, Programming Language—C.<sup>1</sup>

IEEE Std 1596-1992, IEEE Standard for Scalable Coherent Interface (SCI) (ANSI).

IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus.

IEEE Std 1149.5-1995, IEEE Standard for Module Test and Maintenance Bus (MTM-Bus) Protocol.

IEEE P2100-xxxx, SerialExpress- A Scalable Gigabit Extension to Serial Bus.

IEEE Std 1212-1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Buses.

ANSI T1.105-1991, Digital Hierarchy - Optical Interface Rates and Format Specifications (SONET).

IEEE Std 1301-1991

### 2.2 Conformance levels

Several keywords are used to differentiate between different levels of requirements and optionality, as follows:

**2.2.1 expected:** A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

**2.2.2 may:** A keyword that indicates flexibility of choice with no implied preference.

**2.2.3 shall:** A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products conforming to this standard.

**2.2.4 should:** A keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

### 2.3 Glossary of terms

A large number of bus and interconnect-related technical terms are used in this document. These terms are defined below:

**2.3.1 64-bit extension:** IEEE 1212-1991, A group of PCI signals that support a 64-bit data path.

---

<sup>1</sup>Replaces ANSI X3.159-1989. ANSI/ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembe, CH-1211, Genève 20, Switzerland/Suisse. ANSI/ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

**2.3.2 acknowledge packet:** IEEE 1149.5-1995, The first packet returned by an individually addressed S-module that conveys to the M-module that the appropriate S-module is responding and indicates the current status of the responding S-module.

**2.3.3 active:** When associated with a logic level (e.g., in the word “active-low”), this term identifies the logic level to which a signal shall be set to cause a defined action to occur. When referring to an output driver (e.g., in the phrase “an active driver”), this describes the mode in which the driver is capable of determining the voltage of the network to which it is connected.

**2.3.4 active test:** IEEE 1149.5-1995, An on-going test that is invoked by a write to the *TEST START* register. The node is in the testing state (*STATE\_CLEAR.state* is equal to testing) while an active test is in progress.

**2.3.5 address spaces:** PCI Rev.2.1, A reference to the four separate physical address regions of PCI: Memory, I/O, Configuration, and Packet Bus.

**2.3.6 agent:** An active switch, switch-like component, or bridge, between the requester and responder. During normal system operation, the agent’s intervention is transparent to the requester and responder.

**2.3.7 allocation protocols:** The protocols used to allocate resources that are shared by multiple nodes. These include bandwidth allocation protocols and queue allocation protocols.

**2.3.8 amnesia address:** IEEE 1149.5-1995, The module address (‘FA’ HEX) to which a module will respond as though uniquely addressed if that module (1) implements the ability to detect when it cannot determine its address unambiguously and (2) detects that it cannot determine its address unambiguously (3.3).

**2.3.9 analog/RF modules:** Modules whose electronics are primarily analog or radio frequency (RF) rather than digital. These modules often utilize some digital logic for control and test, and may interface to digital buses and interconnects. Analog modules often require shielding from EMI produced by digital modules.

**2.3.10 application logic:** IEEE 1149.5-1995, That portion of a module that excludes the MTM-Bus interface logic.

**2.3.11 assert:** To change the value of a bus signal from logic 0 (released) to logic 1 (asserted) or ensure that such a signal remains at a logic 1.

**2.3.12 arbitration latency:** The time that the master waits after having asserted REQ# until it receives GNT# and the bus returns to the idle state after the previous master’s transaction.

**2.3.13 asserted:** Having a current value equal to logic 1 (said of any signal).

**2.3.14 auxiliary bus:** IEEE 1149.5-1995, A relatively low capacity narrow bus used for miscellaneous functions for which the primary bus is not suited, or for which an alternate module access is needed.

**2.3.15 availability:** The fraction of time that the system is actually capable of performing its mission.

**2.3.16 backplane:** A subassembly that holds the connectors into which one or more boards can be plugged. In addition to providing bus signal connections, the backplane usually provides power connections, power status information, and physical position information to the board.

**2.3.17 backplane bus:** A means in a backplane to connect corresponding signals of circuit modules using a standard set of electrical, timing, and logical rules.

**2.3.18 backplane interface standard:** A set of specifications, such as this document, that define physical and electrical attributes, and some functional and protocol properties, of electronic modules for interconnection to a common backplane interface.

**2.3.19 backplane slot:** The backplane connections devoted to a single module.

**2.3.20 benign failure:** Failure whose penalties are of the same order of magnitude as the benefit provided by correct service delivery.

**2.3.21 big addressian:** A term used to describe the physical location of data-byte addresses on a multiplexed address/data bus. On a big-addressian bus, the data byte with the largest address is multiplexed (in time or space) with the least-significant byte of the address.

**2.3.22 big endian:** A term used to describe the arithmetic significance of data-byte addresses within a multi-byte register. Within a big endian register or register set, the data byte with the largest address is the least significant.

**2.3.23 BIST register:** An optional register in the header region used for control and status of built-in self tests.

**2.3.24 board:** The physical component that is inserted into one of the backplane slots.

**2.3.25 box (electronic):** A protective enclosure to house modules, backplane(s), I/O connector assemblies, internal cables, and other electronic, mechanical, and thermal devices. Syn: **box; cabinet; rack.**

**2.3.26 bridge:** PCI Rev.2.1, A hardware adapter that forwards transactions between buses.

**2.3.27 bridge:** IEEE 1596-1992, A pair of communicating nodes, each of which selectively (based on target address) accepts certain packets for retransmission by the other. For example, a symmetric bridge may be used to connect two SCI ringlets. Such a bridge (the simplest kind of switch) acts as an agent, taking the place of the target on one ringlet and of the source on the other. It acts like a node that has many addresses.

**2.3.28 Broadcast/Multicast Received (BMR) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the last broadcast or multicast message was received without error.

**2.3.29 broadcast transaction:** A transaction that is distributed to all nodes on a bus.

**2.3.30 BSE:** Acronym for "Bus Error".

**2.3.31 BSY:** Acronym for "Slave Busy."

**2.3.32 buffered write:** A write transaction that appears to complete when the request is queued in the agent or responder. A buffered-write transaction returns an optimistic (`done_correct`) status before the responder's completion status (which could report an error) is available.

**2.3.33 burst transfer:** The basic bus transfer mechanism of PCI. A burst is comprised of an address phase and one or more data phases.

**2.3.34 bus commands:** Signals used to indicate to a target the type of transaction the master is requesting.

**2.3.35 bus device:** A bus device can be either a bus master or target:  
master: drives the address phase and transaction boundary (`FRAME#`). The master initiates a

transaction and drives data handshaking (IRDY#) with the target.

target: claims the transaction by asserting DEVSEL# and handshakes the transaction (TRDY#) with the initiator.

**2.3.36 Bus Error (BSE) bit:** A bit in the Slave Status register of every S-module that is set by the S-module when a Bus Error is recorded in the Bus Errors register.

**2.3.37 Bus Error register:** A status register that is required to be implemented in the MTM-Bus interface circuitry of every S-module. Bits in this register provide the S-module with the ability to record error conditions associated with message transmission. The register may be interrogated by the M-module. Some bits in the register are reserved for application-specific uses.

**2.3.38 bus-dependent:** A term used to describe parameters that may vary among different bus standards, but are defined by them. Although the CSR Architecture may constrain the definition of these fields, their detailed definition is provided by the appropriate bus standard.

**2.3.39 Bus Node:** Refers to an addressable unit on the bus.

**2.3.40 Bus Segment:** A set of nodes located on a single backplane.

**2.3.41 bus standard:** An abbreviated notation used throughout this document, rather than the more exact "bus standard document that claims conformance to this specification."

**2.3.42 busied:** A status indication returned in an echo packet that indicates to the sender that the send packet was not accepted (and was discarded), probably because there was no room in the destination queue. The sender should retransmit the packet later.

**2.3.43 byte:** Eight bits of data, used as a synonym for octect.

**2.3.44 cache line:** Often called simply a "line." The unit of data on which coherence checks are performed, and for which coherence tag information is maintained. In SCI, a line consists of 64 data bytes.

**2.3.45 central resources:** Bus support functions supplied by the host system, typically in a PCI compliant bridge or standard chipset.

**2.3.46 clean snoop:** A snoop that does not result in a cache providing modified data.

**2.3.47 cleanse instruction:** A cleanse (cache-control) instruction converts a line to the clean state (the data in cache and memory are the same).

**2.3.48 clear packet:** A packet used during initialization to empty linc buffers and initialize the linc. CSR state is unaffected; e.g., the node's address is unchanged by a "clear." Clear may be sent by any node that has lost synchronization in order to trigger reinitialization.

**2.3.49 clockStrobe signal:** A packet that causes a node to record its time-of-day registers (if any) when it is received, and to record the duration of the propagation of the packet within the node. Used for precisely synchronizing multiple time-of-day clocks within a system.

**2.3.50 coherent transaction:** A transaction (typically read or write) that provides protocols for checking and maintaining consistency with other caches. Coherent transactions are expected to address a cache-line. For example, tightly coupled multiprocessors are expected to use coherent transactions when accessing shared-memory resident data.

**2.3.51 command\_reset:** IEEE 1149.5-1995, An initialization event that is initiated by a write to the RESET\_START register.

**2.3.52 company\_id:** A 24-bit binary value used to identify a company within the context of the CSR Architecture. The company\_id values are expected to be uniquely assigned to each company through RAC.

**2.3.53 concurrent bus operation:** In dual bus systems, “concurrent bus operation capable” describes buses capable of conducting simultaneous unrelated bus transactions. “Non-concurrent bus operation capable” describes buses capable of conducting a transaction on only one bus at a time.

**2.3.54 configuration Address Space:** PCI Rev.2.1, A set of 64 registers (DWORDS) used for configuration, initialization, and catastrophic error handling. This address space consists of two regions: a header region and a device-dependent region.

**2.3.55 configuration cycle:** Bus cycles used for system initialization and configuration via the configuration address space.

**2.3.56 conflict\_error:** An error-status code that is returned when a transaction has been transmitted successfully, but a queue or usage conflict inhibits the transaction completion. A conflict\_error status is returned to the original requester, which is expected to retry the transaction. This is different than a bus-dependent delay (wait or busy status), which delays the forwarding of a transaction or subaction across the bus.

**2.3.57 CSR:** An abbreviation for control and status register. A CSR is a quadlet register that is accessed through read4 or write4 transactions and is used to observe a node's state or to control its operation.

**2.3.58 CSR Architecture:** IEEE Std 1212-1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Buses.

**2.3.59 CSR Architecture Bus:** A term that refers to the CSR standards document (IEEE 1212-1992).

**2.3.60 consumer:** The node on a ringlet that strips a send packet from the ringlet and created the echo packet that is returned to the producer.

**2.3.61 consumer:** The node that responds to a directed transaction

**2.3.62 control and status register (CSR):** A register, storage location, or address that is used to control buses, interconnects, and multiple processor systems.

**2.3.63 coverage:** Measure of the representative nature of situations to which a system is submitted during its validation compared to the actual situations it will be confronted with during its operational life.

**2.3.64 CPU:** Central processing unit of a computer.

**2.3.65 CRC:** The cyclic redundancy code used for error detection on each packet.

**2.3.66 D32 module:** A module whose data space is limited to a 32-bit width.

**2.3.67 D64 module:** A module whose data space is limited to a 64-bit width.

**2.3.68 Data Overrun Error (DOR) bit:** A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the module has received input data from the M-module when the S-module was not ready to receive it.

**2.3.69 dead state:** A node state that is reflected by the value of 3 in the STATE\_CLEAR.state field. A node enters the dead state when a fatal error has been detected and the node is connected but no longer operational. Note that the severest errors could leave the node in a broken state, with its registers undefined, rather than indicating a dead state.

**2.3.70 Delayed Transaction:** The process of a target latching a request and completing it after the master was terminated with Retry.

**2.3.71 detection, error:** The action of identifying that a system state is erroneous.

**2.3.72 detected error:** Error recognized as such by a detection algorithm or mechanism.

**2.3.73 device dependent region:** PCI Rev.2.1, The last 48 DWORDS of the PCI configuration space. The contents of this region are not described in this document.

**2.3.74 diagnosis, fault:** IEEE 1149.5-1995, The action of determining the cause of an error in location and nature.

**2.3.75 diagnostic test:** IEEE 1149.5-1995, A test, or collection of tests, that is invoked by writing to the TEST\_START register. There are four forms of diagnostic tests: initialization tests, extended tests, manual tests, and system tests.

**2.3.76 directed transaction:** A transaction that is processed by one and only one responder. The read, write, and lock transactions are always directed transactions.

**2.3.77 directory:** A contiguous collection of one or more entries, which is contained within the node's ROM.

**2.3.78 directory entry:** A ROM entry that specifies the address of another ROM directory.

**2.3.79 disconnected state:** A state in which the node no longer responds to bus transactions. Since the node no longer responds to bus transactions, a power\_reset is required to change to another node state.

**2.3.80 disruptive test:** A test that is invoked through a write to the TEST\_START register and disrupts the node's operation by temporarily moving the node to the testing state.

**2.3.81 DMA:** A direct memory access (or simply DMA) architecture is an optional capability of an DO controller. After being started by the processor, I/O controllers with DMA capabilities can access their commands, fetch data, and report status by accessing memory directly.

**2.3.82 done\_correct:** A-status code that is returned when a transaction is completed without errors. On many buses, the done\_correct status is implicitly assumed when no error-status codes are observed.

**2.3.83 DOR:** Acronym for "Data Overrun Error."

**2.3.84 double-sided board:** A board with components on both sides.

**2.3.85 doublet:** Two bytes (16 bits) of data.

**2.3.86 DWORD:** A 32-bit block of data. Two bytes of data.

**2.3.87 EEPROM:** Electrically erasable PROM. A variety of PROM whose contents can be erased by special means, so that new contents can be supplied.

**2.3.88 ECL:** Acronym for “emitter coupled logic.”

**2.3.89 edge-sensitive signal:** Signals whose leading and/or trailing edges are used to strobe information contained on level sensitive signals.

**2.3.90 electrical pitch:** The distance between two adjacent connections to the electrical backplane.

**2.3.91 EMC:** Acronym for “Enable Module Control.”

**2.3.92 emperor:** The processor that has the responsibility for initialization of an entire multiprocessor system.

**2.3.93 emperor processor:** The monarch processor that is selected to initialize and configure the system. On a single-bus system, the monarch and emperor processor are always the same. On a multiple-bus system, the single emperor processor is selected from the available monarch processors.

**2.3.94 entry:** (1) *See:* **ROM.** (2) A component of a directory, which is located within the node's ROM. An entry may contain information, or a pointer to another directory or leaf.

**2.3.95 error:** Manifestation of a failure in a system.

**2.3.96 error bit:** A bit in a status register of an S-module that is associated with detection of some error detected by that S-module. Such bits may be found in the Bus Error register, the optional Module Status register, or in an Additional Status register. Error bits of the optional Module Status register or of an Additional Status register are permitted to affect the value of either the BSE bit or EVO bit of the Slave Status register.

**2.3.97 Event Occurrence (EVO) bit:** A bit in the Slave Status register of every S-module that is set by the S-module when a module-application-related condition requiring an interrupt has occurred.

**2.3.98 expansion board:** See add-in board.

**2.3.99 extended test:** A test or collection of tests that shall perform bus transactions and use an external memory buffer. An extended test is invoked by writing to the TEST\_START register.

**2.3.100 falling edge:** The transition from a logic one to logic zero.

**2.3.101 fault tolerance:** The ability of a system or a component to continue normal operation despite the presence of hardware or software faults.

**2.3.102 Fail-Operate system:** A system that can operate in the presence of faults.

**2.3.103 Fail-Safe system:** A system whose failures can only be, or are to an acceptable extent, benign failures.

**2.3.104 Fail-Stop system:** A system whose failures can only be, or are to an acceptable extent, stopping failures.

**2.3.105 failure:** An event in which a system or system component does not perform a required function within specified limits. A failure may be produced when a fault is encountered.

**2.3.106 fault:** A defect in a hardware device or component; for example, a short circuit or broken wire.

**2.3.107 fixed addressing (64-bit):** An address model implemented by bus standards supporting only 64-bit addresses. The initial node space is large (258 Tbytes), is fixed in size, and extended spaces are not supported.

**2.3.108 flag:** A signal used to delimit packets in parallel signal transmission implementations. For example, in the 16-bit parallel implementation the flag is a 17th signal. In some serial implementations special symbols could be used in place of flag transitions.

**2.3.109 flexcircuit:** Flexible printed wiring board. A patterned arrangement of printed circuit and components utilizing flexible base materials with or without flexible cover layers.

**2.3.110 flush instruction:** A flush (cache-control) instruction changes a line to the uncached state. If the data are dirty, they are copied back to memory before the old cache line is invalidated.

**2.3.111 fsm:** Acronym for “finite state machine.”

**2.3.112 functional element:** A set of one or more modules that perform a particular function.

**2.3.113 general ROM format:** IEEE 1212-1991, format for the node-provided ROM. The general ROM format provides bus-dependent information and a root\_directory; the root\_directory directly provides additional ROM entries.

**2.3.114 global system time:** Nodes may maintain time-of-day clocks as described in the CSR Architecture. Software may adjust each of these clocks in order to make them consistent to high accuracy. If this is done, the system is said to implement global system time. Otherwise each clock runs independently, which is sufficient for local timeout purposes but is not sufficient to implement the optional packet "time of death" feature.

**2.3.115 green machine:** A system designed for minimum power consumption.

**2.3.116 HEADER packet:** IEEE 1149.5-1995, A packet originating in the MTM-Bus Master that is the first packet of an MTM- Bus message. The HEADER packet includes an address and a command field. The address identifies which S-module(s) are to interpret and act upon the command contained within the command field.

**2.3.117 host bus bridge:** A low latency path through which the processor may directly access PCI devices mapped anywhere in the memory, I/O, or configuration address spaces.

**2.3.118 IBIT:** Acronym for “initiated built-in test.”

**2.3.119 Idle Interrupts Enabled (IIE) bit:** A bit in the Slave Status register of every S-module that is set to indicate that the S-module may generate an interrupt during S-idle states.

**2.3.120 idle state:** IEEE 1149.5-1995, Any Link Layer Controller state the name of which begins with the uppercase letters “IDLE” Such states in the MTM-Bus Master Link Layer Controller are called M-idle states and in the MTM-Bus Slave Link Layer Controller are called S-idle states.

**2.3.121 Idle state:** PCI Rev.2.1, Any clock period that the bus is idle (FRAME # and IRDY# deasserted).

**2.3.122 IIE:** Acronym for “Idle Interrupts Enabled.”

**2.3.123 ILC:** Acronym for “Illegal Command.”

**2.3.124 Illegal Command (ILC) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the module has received an illegal command.

**2.3.125 immediate effect:** An effect of a transaction, which appears to occur between the time the request subaction is accepted and the response subaction is returned. If a bus standard allows CSR transactions to be split, and sufficient time is allowed between the acceptance of a request subaction and the return of a response subaction, an immediate effect can be emulated by a processor on the node.

**2.3.126 immediate entry:** A ROM entry that provides a 24-bit immediate data value.

**2.3.127 inactive:** When referring to an output driver (e.g., in the phrase “an inactive driver”), this term describes the mode in which the driver is not capable of determining the voltage of the network to which it is connected.

**2.3.128 Incorrect Packet Count (IPC) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that, with respect to a just completed message transfer, either the S-module has received a request for an ACKNOWLEDGE packet and was not given the opportunity to send it or, in the case of an S-module in which the packet counting option is implemented, that it received a different number of packets than was specified in the PACKET COUNT packet.

**2.3.129 independently powered:** An adjective used to describe a node on a bus, when the node's power supply may fail while other nodes remain powered and operational.

**2.3.130 initial memory space:** A portion of the initial node space, which provides a RAM-access window for a memory-controller unit architecture. Unit architectures can also be mapped to non-conflicting portions of the initial memory space. The initial memory space is only relevant to bus standards implementing 64-bit fixed addressing.

**2.3.131 initial node space:** The address space that is initially mapped to a node. The initial node space contains the initial register space (2 Kbytes) and initial units space. On buses implementing 64-bit fixed addressing, the initial node space is 256 Tbytes in size and also includes the node's initial memory and private spaces.

**2.3.132 initialization test:** IEEE 1149.5-1995, A test or collection of tests that does not require cooperation of any other node(s) on the bus. The default and vendor-dependent initialization tests are invoked by writing to the TEST\_START register.

**2.3.133 initializing state:** A node state that is reflected by the value of 1 in the STATE\_CLEAR.state field. The initializing state is an optional transient state which is entered immediately after a power\_reset or command\_reset event.

**2.3.134 Kbyte:** Kilobyte. Indicates 2<sup>10</sup> bytes.

**2.3.135 leaf:** A contiguous information field that is pointed to by a ROM-directory entry. A leaf contains a header (length and CRC specification) and other information fields.

**2.3.136 initial register space:** A 2-Kbyte portion of the initial node space that is adjacent to the initial units space. The registers defined by the CSR Architecture are located within the initial register space. The initial register space also provides addresses for defining busdependent registers.

**2.3.137 initial units space:** The portion of the initial node space that is adjacent to but above the initial register space. When its size is sufficient, unit architectures are expected to be located within this space.

**2.3.138 Institute of Electrical and Electronics Engineers (IEEE):** An organization that, among other functions, sponsors standards development.

**2.3.139 integrity:** The condition of being unimpaired.

**2.3.140 intermateability:** The capability for units of equipment to fit together mechanically but not necessarily work together electrically.

**2.3.141 intermittent fault:** A temporary or unpredictable fault in a component.

**2.3.142 interchangeable:** Said of two modules that, although possibly of different design, perform identical functions and have identical interface characteristics.

**2.3.143 interoperable:** IEEE 1149.5-1995, Said of two modules indicating that they may both be placed on the same physical MTM-Bus without causing errors of operation.

**2.3.144 interoperability:** The capability for units of equipment to work together to do useful functions.

**2.3.145 interoperable modules:** A set of modules that have the properties necessary to allow them to work together in a system to do useful functions. The necessary parameters include items such as electrical, protocol, mechanical, thermal, and I/O interfaces.

**2.3.146 Illegal Port Selected (IPS) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the module has received a command addressed to an unsupported port.

**2.3.147 IPC:** IEEE 1149.5-1995, Acronym for "Incorrect Packet Count."

**2.3.148 IPS:** IEEE 1149.5-1995, Acronym for "Illegal Port Selected."

**2.3.149 ISA:** Acronym for "Instruction Set Architecture". Industry Standard Architecture expansion bus built into the IBM PC AT computer.

**2.3.150 keepers:** Another name for pullup resistors that are only used to sustain a signal state.

**2.3.151 latency:** See arbitration latency, master data latency, target initial latency, target subsequent latency.

**2.3.152 Latency Timer:** A mechanism for ensuring that a bus master does not extend the access latency of other masters beyond a specified value.

**2.3.153 leaf entry:** A ROM entry that specifies the address of a leaf.

**2.3.154 least significant bit (lsb):** The bit having the smallest effect on the value of a binary numeral; usually the right-most bit in figures.

**2.3.155 least significant word (lsw):** In a multiword representation of a binary number, the word containing the lsb of that number.

**2.3.156 level-sensitive signal:** Signals whose high or low state is sampled based on the leading and/or trailing edges of edge-sensitive strobe signals.

**2.3.157 livelock:** A condition in which two or more operations require completion of another operation before they can complete.

**2.3.158 linc:** IEEE 1212-1991, The interface circuitry that attaches to an SCI ringlet. A linc typically contains control/status registers (including identification ROM and reset command registers) that are initially defined in a 4 Kbyte (minimum) ringlet-visible initial node address space.

**2.3.159 linc:** IEEE 1212-1991, The block of memory (sometimes called a sector) that is managed as a unit for coherence purposes; i.e., cache tags are maintained on a per-line basis. SCI directly supports only one line size, 64 bytes.

**2.3.160 little addressian:** A term used to describe the physical location of data-byte addresses on a multiplexed address/data bus. On a little addressian bus, the data byte with the smallest address is multiplexed (in time or space) with the least-significant byte of the address.

**2.3.161 little endian:** A term used to describe the arithmetic significance of data-byte addresses within a multibyte register. Within a little endian register or register set, the data byte with the smallest address is the least significant.

**2.3.162 local interconnect:** A special interconnect among a few boards. This interconnect might be used to create a multi-board functional element such as HiRelPCI.

**2.3.163 lock transaction:** A transaction that passes an address, subcommand, and data parameter(s) from the requester to the responder and returns a data value from the responder to the requester. The subcommand specifies which indivisible update is performed at the responder; the returned data value is the previous value of the updated data.

**2.3.164 logic 0 (logic zero):** The lowest voltage value of the two logic levels on an active-high signal and the highest voltage value of the two logic levels on an active-low signal.

**2.3.165 logic 1 (logic one):** The highest voltage value of the two logic levels on an active-high signal and the lowest voltage value of the two logic levels on an active-low signal.

**2.3.166 lsb:** Acronym for “least significant bit.”

**2.3.167 lsw:** Acronym for “least significant word.”

**2.3.168 manual test:** A test or collection of tests that requires an operator. The tests may involve power failure testing, on-line replacement testing, media testing, or cable connection tests (when loop-back cables are required). A manual test is invoked by writing to the TEST\_START register. Mbyte. Megabyte.

**2.3.169 masking, fault:** The result of applying error compensation systematically, even in the absence of error.

**2.3.170 master:** An agent that initiates a bus transaction.

**2.3.171 Master-Abort:** IEEE 1149.5-1995, A termination mechanism that allows a master to terminate a transaction when no target responds.

**2.3.172 Master-capable:** IEEE 1149.5-1995, Said of an MTM-Bus module that is an S-module at a given time, but contains appropriate circuitry so that it may be converted by system control to an M-module if required. IEEE 1149.5-1995, ed.

**2.3.173 Master Controller state:** IEEE 1149.5-1995, A state of the fsm required of M-modules that controls M-module Link Layer behavior with regard to message transmission.

**2.3.174 master data latency:** The number of PCI clocks until IRDY# is asserted from FRAME# being asserted for the first data phase, or from the end of the previous data phase.

**2.3.175 M-Controller:** IEEE 1149.5-1995, Abbreviation for “MTM-Bus Master Link Layer Controller.”

**2.3.176 MFS bit:** Acronym for “Module Fail Status.”

**2.3.177 message:** IEEE 1149.5-1995, A set of packets starting with a HEADER, consisting of the HEADER and all (ACKNOWLEDGE and DATA) packets transmitted as the immediate consequence of the command in that HEADER, and terminating when the M-module returns to the IDLE Master Controller state.

**2.3.178 messages, class of:** IEEE 1149.5-1995, A group of messages having in the Command fields of their respective HEADER packets command codes of commands belonging to a single class of commands. The name, C, of a class of messages is the same as the name of the class of commands that defines the class C.

**2.3.179 messages, species of:** IEEE 1149.5-1995, A group of messages having in the Command fields of their respective HEADER packets a common command code. The name, S, of a message species is the same as the name of the command that defines the message species.

**2.3.180 MICT:** IEEE 1149.5-1995, Acronym for “Module I/O Control and Test.”

**2.3.181 minimal ROM format:** IEEE 1149.5-1995, A format for the node-provided ROM. The minimal ROM format provides a 24-bit company\_id value; although additional ROM parameters can be provided, their format and meaning are vendor-dependent.

**2.3.182 MIST:** IEEE 1149.5-1995, Acronym for “Module Initialization and Self-Test.”

**2.3.183 M-module:** IEEE 1149.5-1995, Abbreviation for MTM-Bus Master module.

**2.3.184 module:** An addressable unit or interconnected set of units attached to the MTM-Bus and fully supporting the MTM-Bus protocols. The boundary of an MTM-Bus module may correspond to the physical partitioning of the system, but is not required to do so. For the purposes of this document, a module is comprised of an MTM-Bus interface and module application logic.

**2.3.185 module address:** IEEE 1149.5-1995, An eight-bit value uniquely identifying an MTM-Bus module generated from a backplane nodeId.

**2.3.186 Module Fail Status (MFS) bit:** IEEE 1149.5-1995, A bit in the Slave Status register of every S-module that is set by the S-module when the module’s built-in test has failed or is currently executing.

**2.3.187 module interface:** All aspects of the electrical, fiber optic, protocol, mechanical, and thermal interfaces of a module to associated modules, the backplane, I/O connections, module cage and cabinet mounting, conduction and/or convection cooling, and power and ground.

**2.3.188 Module Status register:** IEEE 1149.5-1995, A status register that is recommended to be implemented in the MTM-Bus interface circuitry of every S-module. The bits in this register are defined by the manufacturer of the MTM-Bus interface circuitry of an S-module. The bits of such a register may serve to record error-condition detection or the module’s application-related status.

**2.3.189 monarch:** A processor that has the responsibility for initializing a part of the system, such as a ringlet. If a system has multiple monarchs, they eventually defer to an emperor that coordinates the initialization process.

**2.3.190 monarch processor:** The processor that is selected to partially initialize the local-bus resources and fetch the initial boot code.

**2.3.191 most significant bit (msb):** The bit having the greatest effect on the value of a binary numeral; usually the left-most bit in figures.

**2.3.192 most significant word (msw):** In a multiword representation of a binary number, the word containing the msb of that number.

**2.3.193 move:** An abbreviation for move action.

**2.3.194 move action:** A request-only form of write transaction. A move may be accepted by an agent and forwarded to other nodes in the system. Since there is no response to confirm when the move action has completed; a distinct write or sync transaction is needed to confirm when a move has completed.

**2.3.195 msb:** Acronym for “most significant bit.”

**2.3.196 MBO; MSB1:** IEEE 1149.5-1995, Acronyms for “multicast select bit 0” and “multicast select bit 1.”

**2.3.197 MSD:** IEEE 1149.5-1995, Acronym for “MTM-Bus Slave Data.”

**2.3.198 MTM-Bus:** IEEE 1149.5-1995, A serial, backplane, test and maintenance bus, consisting of one or more logic boards, that can be used to integrate modules from different design teams or vendors into testable and maintainable subsystems, as specified in this Standard.

**2.3.199 MTM-Bus interface logic:** IEEE 1149.5-1995, The portion of a module that is designed for the purpose of MTM-Bus compliant communication and through which takes place all the communication between the given module and any other on a given MTM-Bus implementation. MTM-Bus interface logic need not be defined on the basis of physical package boundaries.

**2.3.200 MTM-Bus Master:** IEEE 1149.5-1995, The module in control of the MTM-Bus. This is the module that, at a given time, is sourcing MCTL and MMD.

**2.3.201 MTM-Bus mastership:** IEEE 1149.5-1995, Property of being the current MTM-Bus Master module.

**2.3.202 MTM-Bus Slave:** IEEE 1149.5-1995, An MTM\_Bus module that cannot command actions of other modules on the bus, but that may be selected by the MTM-Bus Master module to participate in a message.

**2.3.203 multicast:** IEEE 1149.5-1995, A mode of operation in which the M-module transmits data simultaneously (i.e., during a single message) to a predefined subset of the S-modules currently connected to the bus. Also, a message transmitted in this mode.

**2.3.204 multicast action:** A multicast form of the move action, which transfers data from a requester to one or more responders.

**2.3.205 multicast responder:** One or more nodes responding to the multicast-action initiated by a Multicast requester.

**2.3.206 multicast select bit 0; multicast select bit 1 (MSB); MSB1):** IEEE 1149.5-1995, Those bits in the Slave Status register of every S-module by means of which the S-module is programmed to be a member of one of the four possible multicast select groups.

**2.3.207 multicast select group:** IEEE 1149.5-1995, A group of S-modules that may be addressed simultaneously in a multicast. Four such groups are possible. Each has an address defined by this Standard. The multicast select group of an S-module is programmable.

**2.3.208 multidrop:** IEEE 1149.5-1995, Said of the configuration of a bus with a single shared medium segment that allows one or more of its module connectors to be unoccupied without disturbing bus operation.

**2.3.209 multi-port module:** A module that has a special pin assignment to connect to multiple other modules. Examples of multi-port modules are switch modules and multi-port memory modules.

**2.3.210 NMI:** Non-maskable interrupt.

**2.3.211 node:** The software-visible station on a bus. Each node is allocated a set of control register addresses (including identification-ROM and reset command registers) that are initially defined in a 4-Kbyte (minimum) initial node address space. Although multiple nodes may share one bus interface, each node can be reset independently (a reset of one node has no effect on other nodes).

**2.3.212 nodeId:** A 16-bit number that determines the node address space. After system initialization, unique nodeId values have been assigned to all nodes within a tightly coupled system. The nodeId is the part of the 64-bit address that is used for routing packets.

**2.3.213 nonconcurrent bus operation:** In dual bus systems “concurrent bus operation capable” describes buses capable of conducting simultaneous unrelated bus transactions. “Nonconcurrent bus operation capable” describes buses capable of conducting a transaction on only one bus at a time.

**2.3.214 noncoherent transaction:** A transaction (typically read or write) that is completed without checking for consistency with other caches on the local bus. For example, noncoherent 4-byte read and write transactions are used to access CSRs. Note that noncoherent transactions may be converted into coherent transaction sequences when passing through bus bridges.

**2.3.215 non-NULL DATA packet:** IEEE 1149.5-1995, A DATA packet other than a NULL packet.

**2.3.216 null DATA packet:** IEEE 1149.5-1995, A special type of DATA packet containing a data field entirely filled with logic zeros and a parity bit equal to logic one.

**2.3.217 nondisruptive test:** A test that is invoked through a write to the TEST\_START register and does not disrupt the node's operation (the node does not enter the testing state).

**2.3.218 nonvolatile memory (NVM):** Read/write storage that is preserved through losses of power.

**2.3.219 NVM (nonvolatile memory):** Memory that retains its contents even through power failures.

**2.3.220 octlet:** Eight bytes of data.

**2.3.221 off-line:** IEEE 1149.5-1995, Used to describe an MTM-Bus module when it is in a mode such that it will not respond to state transitions on MTM-Bus signals whether or not the module is connected to the bus. Also used to describe such a mode.

**2.3.222 offset entry:** A ROM entry that provides a 24-bit offset value. The offset value specifies the location of a CSR that provides a 32-bit parameter.

**2.3.223 open system hardware architecture:** An electronic system design that allows components, which are developed or built by multiple parties, to work together.

- 2.3.224 operation:** A logical sequence of transactions, e.g., Lock.
- 2.3.225 output driver:** PCI Rev.2.1, An electrical drive element (transistor) for a single signal on a PCI device.
- 2.3.226 packet:** IEEE 1149.5-1995, A collection of symbols **that contains addressing information** and is protected by a CRC. A subaction consists of two **packets, a send packet and an echo packet.**
- 2.3.227 packet.** A 17- bit unit of data consisting of a 16- bit word plus 1 parity bit.
- 2.3.228 PACKET COUNT packet:** IEEE 1149.5-1995, A packet by means of which an M-module conveys to addressed S-modules the number of packets to follow in the current message. S-modules may or may not include the ability to use the data in this packet.
- 2.3.229 packet latency:** During a send/receive message, the number of packet transfers by which the first non-NULL DATA packet returned by the addressed S-module lags the first non-null DATA packet transmitted by the M-module.
- 2.3.230 packet pair:** IEEE 1149.5-1995, Two packets, one transmitted by the M-module and one by an S-module , such that the last 14 bits of the M-module -originated packet are transmitted simultaneously with the first 14 bits of the S-module-originated packet.
- 2.3.231 packet symbol:** IEEE 1149.5-1995, A symbol contained within a packet and protected by the packet's CRC. (Exception: part of the second symbol in a packet contains flow control information that is not covered by the CRC, but the symbol as a whole is still considered to be within the packet.)
- 2.3.232 Parity Error (PRE) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the module has detected a Parity Error on a DATA packet received.
- 2.3.233 Pause Interrupt Enabled (PIE) bit:** IEEE 1149.5-1995, A bit in the Slave Status register of every S-module that is set to indicate that the S-module may generate an interrupt during the PAUSE Slave Controller state when the S-module is addressed.
- 2.3.234 PIE:** IEEE 1149.5-1995, Acronym for "Pause Interrupt Enabled."
- 2.3.235 PCI connector:** An expansion connector that conforms to the electrical and mechanical requirements of the PCI local bus standard.
- 2.3.236 PCI device:** A device (electrical component) conforms to the PCI specification for operation in a PCI local bus environment.
- 2.3.237 PGA:** Pin grid array component package.
- 2.3.238 phase:** One or more clocks in which a single unit of information is transferred, consisting of: an Address Phase (a single address transfer in one clock for a single address cycle and two clocks for a dual address cycle) A Data Phase (one transfer state plus zero or more wait states).
- 2.3.239 physical interface:** The circuitry that interfaces the module, board(s), and node(s) to the bus signals.
- 2.3.240 pitch:** The centerline to centerline spacing of modules or boards in a card cage or on a backplane.
- 2.3.241 port:** IEEE 1149.5-1995, A source or destination of data transferred by a Data Transfer class command into and/or out of an S-module (clause 21). A port may be an on-module memory, on-module inter-

face, a peripheral attached to a module, or some other mechanism to/from which data is passed. Within this Standard, a port is defined by a module address, a port ID meaningful to the MTM-Bus interface logic of that module, and the semantics and structure of packets by which data can be conveyed to and/or from that port. This latter often entails some description of the application to/from which data are passed. A port is selected/accessed/addressed via a Data Transfer class command.

**2.3.242 PORT ID packet:** IEEE 1149.5-1995, The first DATA packet transferred in a Data Transfer class message. This packet contains the identifier (Port Identifier) by which means a port is selected for the remainder of the message.

**2.3.243 Port Transfer Error:** IEEE 1149.5-1995, A port-specific error indicating some failure with relation to transmission of command or data to/from a currently selected port.

**2.3.244 Port Transfer Error (PTE) bit:** IEEE 1149.5-1995, A bit in the Bus Error register of all S-modules. An S-module sets this bit to indicate that the port selected by the command in the current message has reported a Data Transfer Port Error. Such errors will be found defined in the port documentation of specific S-modules. The acronym stands for "Port Transfer Error."

**2.3.245 PRE:** IEEE 1149.5-1995, Acronym for "Parity Error."

**2.3.246 PTE:** IEEE 1149.5-1995, Acronym for "Port Transfer Error."

**2.3.247 positive decoding:** A method of address decoding in which a device responds to accesses only within an assigned address range.

**2.3.248 POST:** Power-On Self Test. A series of diagnostic routines performed when a system is powered up.

**2.3.249 power\_reset:** An initialization event triggered by the restoration of primary power. On a backplane bus, a power\_reset event is generally triggered by one or several specialized signals driven by the shared power supply.

**2.3.250 primary bus:** The collection of signals that provides the system with the basic mechanism for exchanging data between boards.

**2.3.251 primary backplane bus:** A backplane bus that provides the main communications path among a cluster of modules.

**2.3.252 primary state:** The state on a node that is initialized by a power\_reset. For example, the CSRs defined by IEEE Std 1212-1991, CSR Architecture, are part of the node's primary state.

**2.3.253 producer:** The node on a ringlet bus that transmits a send packet to the consumer.

**2.3.254 PROM:** Programmable read-only memory. A form of nonvolatile memory that is supplied with null contents and is loaded with its contents in the laboratory or in the field. Once programmed, its contents cannot be changed.

**2.3.255 pullups:** Resistors used to insure that signals maintain stable values when no agent is actively driving the buss.

**2.3.256 purge instruction:** A purge (cache-control) instruction changes a line to the uncached state, invalidating the old cache line without copying dirty data back to memory.

**2.3.257 QOLB (queue on lock bit):** A mechanism for efficiently sequencing the access to resources that are not to be used by more than one process at a time.

**2.3.258 quadlet:** Four bytes of data.

**2.3.259 queue allocation protocols:** The protocols used to allocate queue space when several nodes are sending packets to a shared node. This involves rejecting packets (with a busy status), but reserving future queue space; the reserved queue space is eventually used during one of the packet's retransmissions.

**2.3.260 read transaction:** IEEE 1149.5-1995, A transaction that passes an address and size parameter from the requester to the responder and returns data values from the responder to the requester. The size parameter specifies the number of bytes that are transferred.

**2.3.261 recovery, error:** Form of error processing where an error-free state is substituted for an erroneous state.

**2.3.262 register:** IEEE 1212-1991, A term used to describe quadlet addresses that can be read or written by software. In the context of this document, a register does not imply a specific hardware implementation. If a bus standard allows transactions to be split, and sufficient time is allowed between the request and response subactions, the functionality of the register can be emulated by a processor on the module.

**2.3.263 release:** To cease to assert a logic 1 on a bus signal line. (One module's releasing a signal line produces a change in value of the signal line only if no module is asserting the signal.)

**2.3.264 released:** Having a value equal to logic 0 (said of any signal). Equivalently, in the case of an MTM-Bus signal line, not asserted by any module on the bus.

**2.3.265 reliability:** The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

**2.3.266 removal, fault:** Methods and techniques aimed at reducing the presence (number, seriousness) of faults.

**2.3.267 requester:** The node that initiates a transaction, by initiating a request subaction.

**2.3.268 request echo:** The echo packet generated by a responder or agent when it strips the request send packet.

**2.3.269 request send:** The packet generated by a requester to initiate an action in the responder, containing address, command, and, if appropriate, data. In a processor-to-memory read transaction, for example, the request send transfers the memory address and command from the processor to memory.

**2.3.270 request subaction:** A request send and its echo. Often called simply a "request."

**2.3.271 request subaction:** A subaction that is generated by a requester to initiate an action on the responder. For a processor-to-memory read transaction, for example, the request subaction transfers the memory address and command from the processor to memory.

**2.3.272 request subaction:** The first of two subactions within split-response directed-asynchronous transaction.

**2.3.273 reset packet:** A packet used during initialization to reset the node's CSR state, empty ring buffers, initialize the ring interface and establish that ring closure has been achieved.

**2.3.274 request packet:** An abbreviation for request-send packet.

**2.3.275 request-send packet:** The packet generated by a requester to initiate an action in the responder, containing address, command, and (if appropriate) data. In a processor-to-memory read transaction, for example, the request packet transfers the memory address and command from the processor to memory.

**2.3.276 resend:** The action of resending a transaction request; the transaction-request is resent after a response containing a CONFLICT status is returned.

**2.3.277 reset test:** IEEE 1149.5-1995, A test or collection of tests that is invoked by a command\_reset. Although a reset test is actually a form of initialization test, the term reset test is used to avoid confusing its functionality with the initialization tests that are invoked by writing to the TEST\_START register.

**2.3.278 responder:** The node that completes a transaction, by returning a response subaction.

**2.3.279 response:** In the context of message transmission, the set of packets sent by an S-module during a single message. In the context of the operation of S-modules, an S-module's action that is a direct consequence of the command most recently received by that S-module

**2.3.280 response-expected request:** The request subaction component of a response-expected transaction.

**2.3.281 response-expected transaction:** A transaction that normally consists of a request subaction and a response subaction. For example, the read, write, and lock transactions are all response-expected transactions.

**2.3.282 response packet:** An abbreviation for response-send packet.

**2.3.283 response-send packet:** The packet generated by a responder to complete a transaction initiated by a requester. In a processor's memory-read transaction, for example, the response packet returns the requested data and related status information from the memory to the processor.

**2.3.284 response subaction:** A subaction that is returned by a responder to complete a transaction initiated by a requester. In a processor-memory read transaction, for example, the response subaction returns the data and status from the memory to the processor.

**2.3.285 responseless request:** The request subaction component of a responseless transaction.

**2.3.286 responseless transaction:** A transaction that consists of only a request subaction (there is never any response subaction).

**2.3.287 response send:** The packet generated by a responder to complete a transaction initiated by a requester. In a processor's memory-read transaction, for example, the response send returns the requested data and related status information from the memory to the processor.

**2.3.288 response\_timeout:** An implied split-transaction-error status that is returned when the response subaction is not returned within an expected timeout interval.

**2.3.289 retried:** A term that refers to a subaction that has been previously busied and is being retransmitted.

**2.3.290 ringlet:** The closed path formed by the connection that provides feedback from the output link of a node to its input link. This connection may include other nodes or switch elements.

**2.3.291 rising edge:** A transition from a logic zero to a logic one.

**2.3.292 ROM (read-only memory):** The memory on a node that provides storage locations for normally read only data or code. The ROM data are maintained across losses of primary and secondary power. In some implementations ROM may be writable, using (normally disabled) vendor-specific protocols.

**2.3.293 SAC:** Single address cycle. A HiRelPCI transaction where a 32-bit address is transferred across a 32-bit data path in a single clock cycle.

**2.3.294 root\_directory:** A term used to describe the directory at the top level of the hierarchical ROM directory structure.

**2.3.295 running state:** A node state that is reflected by the value of 0 in the STATE\_CLEAR.state field. The running state is the normal operational state in which access to all of the node's CSRs are defined.

**2.3.296 S-Controller:** Abbreviation for "MTM-Bus Slave Link Layer Controller."

**2.3.297 S-module:** Abbreviation for "MTM-Bus Slave Module."

**2.3.298 SBIT:** Acronym for "Start-up Built-in Test" (performed at power-up or after resets of a module).

**2.3.299 SCI (Scalable Coherent Interface):** The name that refers to IEEE Std 1596-1992 [B9]. Though functionally behaving as a bus, the SCI's physical implementation is a collection of point-to-point unidirectional links (i.e., a ring).

**2.3.300 SCI standard:** Refers to IEEE 1596-1992.

**2.3.301 scrubber:** The node on a serial link that marks packets as they go past in a ringlet, and discards any previously marked packet. This prevents damaged or misaddressed packets from circulating indefinitely. The scrubber also performs other housekeeping tasks for the ringlet. There is always exactly one scrubber on a ringlet. Normal nodes may all have scrubber capability built in, but exactly one is enabled as scrubber per ringlet. Often the scrubber will take responsibility for initializing a ringlet, but this could be done by another (unique) node.

**2.3.302 SDF:** Acronym for "Slave Data Fault" in the Maintenance Bus.

**2.3.303 secondary bus:** A collection of signals that provides the system with an alternate mechanism for exchanging data between boards as a means to recover from faults in the primary bus.

**2.3.304 select:** IEEE 1149.5-1995, To identify and fix (for the duration of the current message) a port to which data of a Data Transfer class message are to be directed.

**2.3.305 send:** The first of two packets within a request or response subaction. The send packet contains a 16-byte header (containing command and status) and may optionally contain a data component (up to 256 bytes).

**2.3.306 sequence:** A set of bits, packets, or messages ordered in time and that are, or that are intended to be, transmitted consecutively without interruption.

**2.3.307 Serial Bus:** Refers to the P1394 standards project that is defining an inexpensive serial network that can be used as an alternate control or diagnostic path, as an I/O connection, or even in place of a parallel bus in some systems.

**2.3.308 set:** (used as a verb) To force the contents of one or more storage elements to a logic 1.

**2.3.309 simple message:** An MTM-Bus message that consists of no more than a HEADER and, optionally, an ACKNOWLEDGE/PACKET COUNT packet pair.

**2.3.310 singlecast:** A mode of operation in which the M-module transmits data to a single S-module. Also, a message transmitted in this mode.

**2.3.311 single point of failure:** With respect to a system, a failure that would result in the inability of that system to perform its intended function.

**2.3.312 Slave Busy (BSY) bit:** A bit in the Slave Status register of every S-module that is set by the S-module when the application logic of the S-module is executing a previously transmitted MTM-bus instruction or is executing its power-up sequence.

**2.3.313 Slave Controller state:** A state of the fsm required of S-modules that controls S-module Link Layer behavior during message transmission to the module.

**2.3.314 Slave Data Fault (SDF) bit:** A bit in the Bus Error register of all S-modules. An S-module sets this bit when it is transmitting on the MSD signal and detects a fault on that signal.

**2.3.315 Slave Status register:** A status register that is required to be implemented in the MTM-Bus interface logic of every S-module. Bits in this register are used to record such items of module status as interrupt enable status, whether an error condition has occurred, whether a module application-related error has occurred, whether the module has failed its Built-In Self Test, etc.

**2.3.316 slot:** A module-insertion position provided by the backplane and associated card cage.

**2.3.317 specialId:** A reserved nodeId value associated with special-send packets.

**2.3.318 special send:** A packet having one of a particular set of special addresses and a special format used for initialization, such as "reset" or ~clear."

**2.3.319 Special Cycle:** A message broadcast mechanism used for communicating processor status and/or (optionally) logical sideband signaling between PCI agents.

**2.3.320 split transaction:** A transaction that consists of separate request and response subactions. On a backplane bus, for example, a split transaction is one in which bus mastership is relinquished between the request and response subactions. Few buses permit split transactions. *See also:* **unified transaction.**

**2.3.321 SSE:** Acronym for "State Sequence Error."

**2.3.322 stale data:** Data in a cache-based system that is no longer valid and, therefore, must be discarded.

**2.3.323 standard bus:** An abbreviated notation used throughout this document, rather than the more exact "physical bus standard that claims conformance to this specification."

**2.3.324 State Sequence Error (SSE) bit:** A bit in the Bus Error register of all S-modules. An addressed S-module sets this bit to indicate that the S-module's Slave Link Layer Controller has entered the ERROR Slave Controller State.

**2.3.325 state, system:** The values assumed at a given instant by the variables that define the characteristics of a system, component, or simulation.

**2.3.326 status bit:** A bit used to indicate a non-error condition important to S-module operation. Status bits are located in an S-module's Slave Status register and Bus Error register (the BMR bit) and may be located

in the optional Module Status register or an Additional Status register. Status bits in the Module Status register or in an Additional Status register are permitted to affect the value of the EVO bit of the Slave Status register.

**2.3.327 status register:** A register in an S-module by means of which current operating conditions of the S-module (e.g., interrupt enabled, module pass/fail status, multicast address of the S-module, etc.) and event occurrence (e.g., detection of an error condition during transmission of a message) can be recorded either for later interrogation by the M-module or to record the necessity of particular S-module activity at a later time.

**2.3.328 stub length:** Stub length is measured from the point at which the connector assembly connects to the PC board to the pad where the transceiver lead is soldered to the PC board.

**2.3.329 system failure:** Malfunctions in the hardware and software that could compromise the security of the system (for example, non-security-related failures and design flaws are not considered). The malfunctions include both intentional and inadvertent design or implementation flaws (including malicious hardware and software) and component failures. For intentional attacks, this threat area assumes that an intruder has access to the design or implementation processes of the system or to the operational system in such a way as to be able to cause a failure in a component. For inadvertent attacks, there may not be a specific intruder.

**2.3.330 subaction:** A component of a transaction; a request or a response.

**2.3.331 subtractive decoding:** A method of address decoding in which a device accepts all accesses not positively decoded by another agents. See also positive decoding.

**2.3.332 successful test:** A completed test that is invoked by a write to the TEST\_START register, in which no errors are detected.

**2.3.333 supported transaction:** IEEE 1212-1991, A transaction whose returned data value and side effects are defined by the hardware architecture that is addressed. For example, a write4 transactions to the 4-byte STATE\_CLEAR register is supported.

**2.3.334 switch:** A device that connects ringlets and has queues. It can behave as a consumer (when accepting remote subactions) and as a producer (when forwarding the subaction to another ringlet). It may be visible as a node, with a nodeId, or be transparent, with no nodeId. A switch differs from a bridge in that a switch may connect more than two ringlets, but a bridge connects only two. A switch is generally assumed to connect multiple instances of the same bus standard, while a bridge may connect different bus standards.

**2.3.335 switched network:** A computer interconnect that uses switches to allow intermodule communications.

**2.3.336 system test:** A test, or collection of tests, that may use an external memory buffer and may require cooperation of other nodes. For example, identical unit architectures may collaborate to test cache coherence or to generate background "noise" traffic for other nodes being tested. A system test is invoked by writing to the TEST\_START register.

**2.3.337 Tbyte:** Terabyte. Indicates  $2^{40}$  bytes.

**2.3.338 target:** An agent that responds (with a positive acknowledgement by asserting DEVSEL#) to a bus transaction initiated by a master.

**2.3.339 Target-Abort:** A termination mechanism that allows a target to terminate a transaction in which a fatal error has occurred, or to which the target will never be able to respond.

**2.3.340 target initial latency:** The number of clocks that the target takes to assert TRDY# for the first data transfer.

**2.3.341 target subsequent latency:** The number of clocks that the target takes to assert TRDY# from the end of the previous data phase of a burst.

**2.3.342 termination:** A transaction termination brings bus transactions to an orderly and systematic conclusion. All transactions are concluded when FRAME# and IRDY# are deasserted (an idle cycle). Termination may be initiated by the master or the target.

**2.3.343 testing state:** A node state that is reflected by the value of 2 in the STATE\_CLEAR.state field. The testing state is an optional transient state that is entered immediately after a write to the TEST\_START register. The node remains in the testing state until the active test completes.

**2.3.344 time of death:** The term used to describe a field within a send packet that is used to determine when a send packet is stale and should be discarded.

**2.3.345 tolerance, faults:** Methods and techniques aimed at providing a service complying with the specification in spite of faults.

**2.3.346 transaction:** An address phase plus one or more data phases.

**2.3.347 transaction:** An information exchange between two components. A transaction consists of a request subaction and a response subaction. The request subaction transfers commands (and possibly data) between a requester and a responder. The response subaction returns status (and possibly data) from the responder to the requester.

**2.3.348 transfer:** IEEE 1149.5-1995, The successful movement of a bit or bits between an MTM-Bus Master module and one or more modules co-connected by the MTM-Bus.

**2.3.349 transfer state:** Any bus clock, during a data phase, in which data is transferred.

**2.3.350 transient:** A fault or error resulting from temporary environmental conditions.

**2.3.351 TTL:** Acronym for “transistor-transistor logic.”

**2.3.352 turnaround cycle:** A bus cycle used to prevent contention when one agent stops driving a signal and another agent begins driving it. A turnaround cycle must last one clock and is required on all signals that may be driven by more than one agent.

**2.3.353 type\_error:** A status code that is returned when the transaction is directed to an existing address, but the transaction command (for example, a read64 directed to a quadlet register) is not implemented.

**2.3.354 unified transaction:** A transaction in which the request and response subactions are completed in an indivisible sequence; i.e., no other subactions may be performed on the bus until this response subaction is complete. Most buses use unified transactions, but SCI, Serial Express, and HiRelPCI uses split transactions.

**2.3.355 uniquely addressed:** Said of an MTM-Bus S-module participating in a singlecast.

**2.3.356 unit:** The subcomponent of the node that provides a processing, memory, or I/O functionality. (After the node has been initialized (typically, by generic software), the unit provides the register interface, which is accessed by I/O driver software. The units normally operate independently of each other, and do

not affect the operation of the node upon which they reside. Note that one node could have multiple units (i.e., processor, memory, and SCSI controllers).

**2.3.357 unit architecture:** The specification document describing the format and function of the unit's software-visible registers.

**2.3.358 unit-dependent:** A term used to describe parameters that may vary between different unit architectures. Although the CSR Architecture may specify the size and location of these fields, their format and most of their definition is provided by the appropriate unit architecture specification.

**2.3.359 unsuccessful test:** A completed test that is invoked by a write to the TEST\_START register and detects one or more errors.

**2.3.360 unsupported transaction:** A transaction whose returned data value or side effects are not defined by the hardware architecture that is addressed. For example, a write64 transactions to the 4-byte STATE\_CLEAR register is unsupported.

**2.3.361 utility signals:** A set of discrete lines in the backplane that provide communications among modules for which a bus is not adequate because of latency or other reasons.

**2.3.362 validation:** Evaluating a system or component during or at the end of the development process to determine whether it satisfies specific requirements.

**2.3.363 vendor-dependent:** A term used to describe parameters that may vary between vendors supplying the same node or unit architectures. Although the CSR Architecture may constrain the definition of these fields, their format and definition is provided by the module vendor. Note that vendor-dependent fields may be standardized or left implementation-specific, depending on the vendor's needs.

**2.3.364 verification:** Evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

**2.3.365 word:** An ordered set of 16 bits operated on as a unit. The most significant bit is labeled bit 15 and the least significant bit is labeled bit 0.

**2.3.366 wait state:** A bus clock in which no transfer occurs.

**2.3.367 write transaction:** A transaction that-passes an address, size parameter, and data values from the requester to the responder. The size parameter specifies the number of bytes that are transferred.

## 2.4 Bit and byte ordering

The addressable unit in HiRelPCI is the byte. HiRelPCI is primarily defined in terms of packets, constructed from 4-byte (quadlet) on 8-byte (octlet) symbols, that contain a single data value or multiple items located in separate fields. For all packet headers, SCI defines the order and position of fields within the doublets. For data symbols in packets SCI defines the ordering of byte addresses within the symbols.

## 2.5 State machine notation

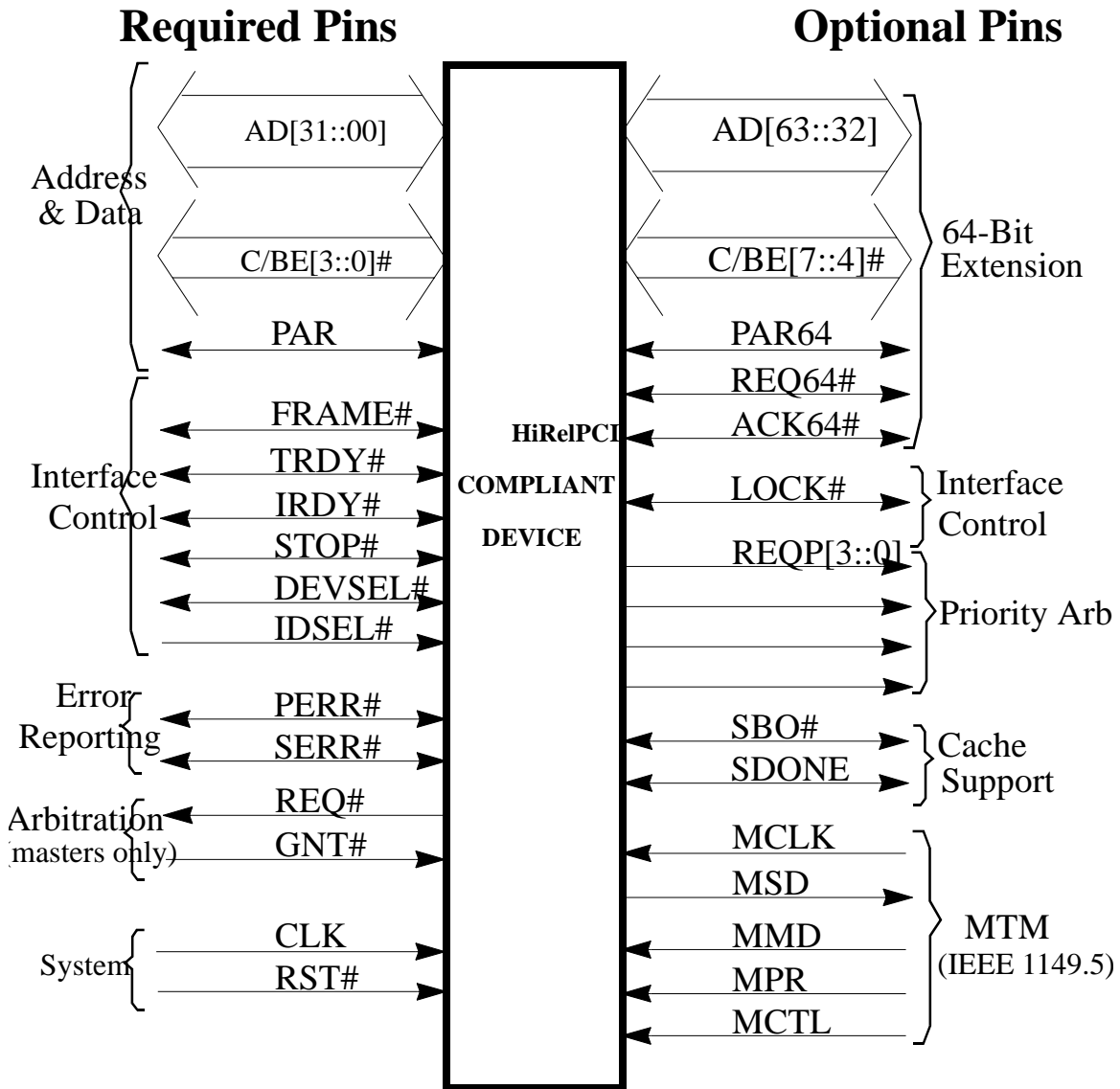
The state machines in this document use the style illustrated in . The states have descriptive names (stateA and stateB) as well as abbreviated names (s0 and s1). Transitions between states are performed when a con-

dition (listed above the state-transition arrow) is met and can cause actions (listed below the state-transition arrow).



### 3. Signal Definitions

The PCI interface requires a minimum of 47 pins for a target-only device and 49 pins for a master to handle data and addressing, interface control, arbitration, and system functions. Figure 2-1 shows the pins in functional groups, with required pins on the left side and optional pins on the right side. The direction indication on signals in Figure 2-1 assumes a combination master/target device.



**6su HiRelPCI Pin List**

### 3.1 Signal Type Definitions

The following signal type definitions are from the view point of all devices other than the arbiter or central resource module. For the arbiter, **REQ#** is an input, **GNT#** is an output, and other PCI signals for the arbiter have the same direction as a Master or Target. The central resource module is a "logical" device where all system type functions are located on the CSM position on each HiRelPCI bus segment.

*in* = *Input* is a standard input-only signal.

*out* = *Totem Pole Output* is a standard active driver.

*t/s* = *Tri-State* is a bi-directional, tri-state input/output pin.

*s/t/s* = *Sustained Tri-State* is an active low tri-state signal owned and driven by one and only one agent at a time. The agent that drives an stls pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a stls signal any sooner than one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

*od* = *Open Drain* allows multiple devices to share as a wire-OR. A pull-up is required to sustain the inactive state until another agent drives it, and must be provided by the central resource.

*enet* = *Ethernet* backplane signal which is 10Base2 equivalent. This signal is electrically equivalent to the signal in a coaxial ethernet but implemented in a 50 ohm backplane environment.

*LVPECL* = Low Voltage PECL level signals as defined in Motorola data sheets for 3.3V PECL devices

### 3.2 HiRelPCI Interface types

#### 3.2.1 6su Full configuration

This 6su configuration uses the full capabilities of the 6su physical format.

```
entity 6sufullslave is port(  
    AD:    inout std_logic_vector(63 downto 0);  
    C/BE:  inout std_logic_vector(7  downto 0);  
    PAR:   inout std_logic;  
    PAR64: inout std_logic;  
    FRAME: inout std_logic;  
    TRDY:  inout std_logic;  
    IRDY:  inout std_logic;  
    STOP:  inout std_logic;  
    DEVSEL:inout std_logic;  
    IDSEL: in  std_logic;  
    REQ:   out  std_logic;  
    REQ64: inout std_logic;  
    ACK64: inout std_logic;  
    REQP:  inout std_logic_vector(3  downto 0);  
    GNT:   in  std_logic;  
    PERR:  inout std_logic;  
    SERR:  inout std_logic;  
    CLK:   in  std_logic;  
    RST:   in  std_logic;  
    LOCK:  inout std_logic;  
    SBO:   inout std_logic;  
    SDONE: inout std_logic;  
    CCLK10M:in std_logic;  
    MCLK:  in  std_logic;
```

```
MMD:    in std_logic;
MSD:    out std_logic;
MPR:    inout std_logic;
MCTL:   in std_logic;
SAD:    in std_logic_vector(15 downto 0)
ENET:   inout std_logic;
1394DATA: inout std_logic;
1394CLK:inout std_logic;
IODATA:inout std_logic_vector(23 downto 0));
end 6sufullslave;

entity 6sufullCSM is port(
  AD:      inout std_logic_vector(63 downto 0);
  C/BE:    inout std_logic_vector(7 downto 0);
  PAR:     inout std_logic;
  PAR64:   inout std_logic;
  FRAME:   inout std_logic;
  TRDY:    inout std_logic;
  IRDY:    inout std_logic;
  STOP:    inout std_logic;
  DEVSEL:  inout std_logic;
  IDSEL:   in std_logic;
  REQ:     out std_logic;
  REQ64:   inout std_logic;
  ACK64:   inout std_logic;
  REQP:    inout std_logic_vector(3 downto 0);
  GNT:     in std_logic;
  PERR:    inout std_logic;
  SERR:    inout std_logic;
  CLK:     in std_logic
  RST:     in std_logic
  LOCK:    inout std_logic;
  SBO:     inout std_logic;
  SDONE:   inout std_logic;
  CCLK10M:in std_logic
  MCLK:    in std_logic;
  MMD:     in std_logic;
  MSD:     out std_logic;
  MPR:     inout std_logic;
  MCTL:    in std_logic;
  SAD:     in std_logic_vector(15 downto 0);
  ENET:    inout std_logic;
  1394DATA: inout std_logic;
  1394CLK:inout std_logic;
  BPSN:    inout std_logic);

-- Clock generator

  CLK0, CLK1, CLK2, CLK3, CLK4, CLK5, CLK6:out std_logic;
  REQ0, REQ1, REQ2, REQ3, REQ4, REQ5, REQ6:in std_logic;
  GNT0, GNT1, GNT2, GNT3, GNT4, GNT5, GNT6:out std_logic;

end 6sufullCSM;
```

### 3.2.2 12su Full Redundant Configuration

```
entity 12sufullslave is port(
  TAD:      inout std_logic_vector(63 downto 0);
  TC/BE:    inout std_logic_vector(7 downto 0);
  TPAR:     inout std_logic;
  TPAR64:   inout std_logic;
  TFRAME:   inout std_logic;
```

```
TTRDY: inout std_logic;
TIRDY: inout std_logic;
TSTOP: inout std_logic;
TDEVSEL:inout std_logic;
TIDSEL:in std_logic;
TREQ: out std_logic;
TREQ64:inout std_logic;
TACK64:inout std_logic;
TREQP: inout std_logic_vector(3 downto 0);
TGNT: in std_logic;
TPERR: inout std_logic;
TSERR: inout std_logic;
TCLK: in std_logic
TRST: in std_logic
TLOCK: inout std_logic;
TSBO: inout std_logic;
TSDONE:inout std_logic;
TCCLK10M:in std_logic
TMCLK: in std_logic;
TMMD: in std_logic;
TMSD: out std_logic;
TMPR: inout std_logic;
TMCTL: in std_logic;

-- TOP SAD lines define the address of the boards for 12su configuration

TSAD: in std_logic_vector(15 downto 0);

-- Top ethernet connection to backplane

TENET: inout std_logic;

-- Top backplane Firewire connection

T1394DATA: inout std_logic;
T1394CLK:inout std_logic;

-- Top User I/O May be replaced with Fiber/Coax connections

TIODATA:inout std_logic_vector(23 downto 0);

-- Top TDM Bus

TTDM: inout std_logic_vector(15 downto 0);
TTDMFRAME: in std_logic;
TTDMCLK+: in std_logic;
TTDMCLK-: in std_logic;

-- The bottom set of connectors with the second bus

BAD: inout std_logic_vector(63 downto 0);
BC/BE: inout std_logic_vector(7 downto 0);
BPAR: inout std_logic;
BPAR64:inout std_logic;
BFRAME:inout std_logic;
BTRDY: inout std_logic;
BIRDY: inout std_logic;
BSTOP: inout std_logic;
BDEVSEL:inout std_logic;
BIDSEL:in std_logic;
BREQ: out std_logic;
BREQ64:inout std_logic;
BACK64:inout std_logic;
BREQP: inout std_logic_vector(3 downto 0);
```

```
    BGNT: in std_logic;
    BPERR: inout std_logic;
    BSERR: inout std_logic;
    BCLK: in std_logic;
    BRST: in std_logic;
    BLOCK: inout std_logic;
    BSBO: inout std_logic;
    BSDONE: inout std_logic;
    BCCLK10M: in std_logic;
    BMCLK: in std_logic;
    BMMD: in std_logic;
    BMSD: out std_logic;
    BMPR: inout std_logic;
    BMCTL: in std_logic;

-- BSAD not used for all 12su boards - used for 6su additions
    BSAD: in std_logic_vector(15 downto 0);

-- Bottom ethernet
    BENET: inout std_logic;

-- Bottom Firewire connection
    B1394DATA: inout std_logic;
    B1394CLK: inout std_logic;

-- Bottom user I/O area - may be Fiber/Coax connections
    BIODATA: inout std_logic_vector(23 downto 0);

-- Bottom TDM bus
    BATDM: inout std_logic_vector(15 downto 0);
    BTDMFRAME: in std_logic_vector(7 downto 0);
    BTDMCLK+: in std_logic;
    BTDMCLK-: in std_logic;

end 12sufullslave;

entity 12sufullCSM is port(
    TAD: inout std_logic_vector(63 downto 0);
    TC/BE: inout std_logic_vector(7 downto 0);
    TPAR: inout std_logic;
    TPAR64: inout std_logic;
    TFRAME: inout std_logic;
    TTRDY: inout std_logic;
    TIRDY: inout std_logic;
    TSTOP: inout std_logic;
    TDEVSEL: inout std_logic;
    TIDSEL: in std_logic;
    TREQ: out std_logic;
    TREQ64: inout std_logic;
    TACK64: inout std_logic;
    TREQP: inout std_logic_vector(3 downto 0);
    TGNT: in std_logic;
    TPERR: inout std_logic;
    TSERR: inout std_logic;
    TCLK: in std_logic;
    TRST: in std_logic;
    TLOCK: inout std_logic;
    TSBO: inout std_logic;
```

```
    TSDONE: inout std_logic;
    TCCLK10M: out std_logic
    TMCLK: out std_logic;
    TMMD: out std_logic;
    TMSD: inout std_logic;
    TMPR: inout std_logic;
    TMCTL: out std_logic;

-- Backplane serial bus to two pin ID ROM and temperature sensor

    TBPSN: inout std_logic;

-- TOP SAD lines define the address of the boards for 12su configuration

    TSADF: in std_logic_vector(3 downto 0);
    TSADS: inout std_logic_vector(15 downto 4);

-- Top ethernet connection to backplane

    TENET: inout std_logic;

-- Top backplane Firewire connection

    T1394DATA: inout std_logic;
    T1394CLK: inout std_logic;

-- Top clock generator

    TCLK0, TCLK1, TCLK2, TCLK3, TCLK4, TCLK5, TCLK6: out std_logic;
    TREQ0, TREQ1, TREQ2, TREQ3, TREQ4, TREQ5, TREQ6: in std_logic;
    TGNT0, TGNT1, TGNT2, TGNT3, TGNT4, TGNT5, TGNT6: out std_logic;

-- TDM clocks and frame are Differential LVPECL signal levels

    TTCLK0+, TTCLK1+, TTCLK2+, TTCLK3+, TTCLK4+, TTCLK5+, TTCLK6+: out std_logic;
    TTCLK0-, TTCLK1-, TTCLK2-, TTCLK3-, TTCLK4-, TTCLK5-, TTCLK6-: out std_logic;
    TTDMFRAME: out std_logic;

-- Top TDM Bus - these signals are all LVPECL

    TATDM: inout std_logic_vector(15 downto 0);
    TTDMFRAME: out std_logic;
    TTDMCLK+: in std_logic;
    TTDMCLK-: in std_logic;

-- The bottom set of connectors with the second bus

    BAD: inout std_logic_vector(63 downto 0);
    BC/BE: inout std_logic_vector(7 downto 0);
    BPAR: inout std_logic;
    BPAR64: inout std_logic;
    BFRAME: inout std_logic;
    BTRDY: inout std_logic;
    BIRDY: inout std_logic;
    BSTOP: inout std_logic;
    BDEVSEL: inout std_logic;
    BIDSEL: in std_logic;
    BREQ: out std_logic;
    BREQ64: inout std_logic;
    BACK64: inout std_logic;
    BREQP: inout std_logic_vector(3 downto 0);
    BGNT: in std_logic;
    BPERR: inout std_logic;
    BSERR: inout std_logic;
```

```
BCLK: in std_logic
BRST: in std_logic
BLOCK: inout std_logic;
BSBO: inout std_logic;
BSDONE:inout std_logic;
BCCLK10M:out std_logic
BMCLK: out std_logic;
BMMD: out std_logic;
BMSD: inout std_logic;
BMPR: inout std_logic;
BMCTL: out std_logic;

__ Backplane serial bus to lower ID ROM and temperatures sensor

BBPSN: inout std_logic;

-- BSAD not used for all 12su boards - used for 6su additions

BSAD: out std_logic_vector(15 downto 0);

-- Bottom ethernet

BENET: inout std_logic;

-- Bottom Firewire connection

B1394DATA: inout std_logic;
B1394CLK:inout std_logic;

-- Bottom clock generator

BCLK0, BCLK1, BCLK2, BCLK3, BCLK4, BCLK5, BCLK6:out std_logic;
BREQ0, BREQ1, BREQ2, BREQ3, BREQ4, BREQ5, BREQ6:in std_logic;
BGNT0, BGNT1, BGNT2, BGNT3, BGNT4, BGNT5, BGNT6:out std_logic;

-- TDM clocks and frame are differential LVPECL signal levels
-- BTDMFRAME is a LVPECL signal

BTCLK0+, BTCLK1+, BTCLK2+, BTCLK3+, BTCLK4+, BTCLK5+, BTCLK6+:out std_logic;
BTCLK0-, BTCLK1-, BTCLK2-, BTCLK3-, BTCLK4-, BTCLK5-, BTCLK6-:out std_logic;
BTDMFRAME:out std_logic;

-- Bottom TDM bus - all LVPECL signal levels

BATDM: inout std_logic_vector(15 downto 0);
BTDMCLK+: in std_logic;
BTDMCLK-: in std_logic;

end 12sufullCSM;
```

### 3.3 Pin Function Groups

The PCI pin definitions are organized in the functional groups shown in Figure 2-1. A # symbol at the end of a signal name indicates that the active state occurs when the signal is at a low voltage.

#### 3.3.1 System Pins

Derive from Section 2.2.1 of the PCI Local Bus Specification Revision 2.1. Please refer to this specification for the definition of standard PCI pin functions. This standard will define the changes made in certain pin definitions to meet the PacketBus requirements.

**CLK** in

*Clock* provides timing for all transactions on HiRelPCI and is an input to every PCI device. All other signals, except **RST#**, are sampled on the rising edge of **CLK** and all other timing parameters are defined with respect to this edge. This bus operates from 0 to 66 MHz.

**CCLK10M** in

**CCLK10M** is a constant clock at a frequency of 10.00000 MHz with a tolerance of 1 ppm minimum with a recommended accuracy and stability of 0.1ppm for communications frequency generation and SONET Stratum 4 or better clocking. This clock is used as a reference clock for other function on boards that must have a clock either at 10.00000 MHz or a frequency derived from it.

### 3.3.2 Interface Control Pins

**FRAME#** s/t/s

*Cycle Frame* is driven by the current master to indicate the beginning and duration of an access. **FRAME#** is asserted to indicate a bus transaction is beginning. While **FRAME#** is asserted, data transfers continue. When **FRAME#** is deasserted, the transaction is in the final data phase or has completed.

**IRDY#** s/t/s

*Initiator Ready* indicates the initiating agent's (bus master's) ability to complete the current data phase of the transaction. **IRDY#** is used in conjunction with **TRDY#**. A data phase is completed on any clock both **IRDY#** and **TRDY#** are sampled asserted. During a write, **IRDY#** indicates that valid data is present on **AD[63::00]**. During a read, it indicates the master is prepared to accept data. Wait cycles are inserted until both **IRDY#** and **TRDY#** are asserted together.

During PACKET mode operation **IRDY#** is used to indicate the second cycle of the transfer and it is active through the last cycle of the transfer. When **IRDY#** is deasserted, the next packet operation or normal bus operation is initiated.

**TRDY#** s/t/s

This signal is used in two modes, one for PCI style transactions and one for PacketBus transactions.

*Target Ready* indicates the target agent's (selected device's) ability to complete the current data phase of the transaction. **TRDY#** is used in conjunction with **IRDY#**. A data phase is completed on any clock both **TRDY#** and **IRDY#** are sampled asserted. During a read, **TRDY#** indicates that valid data is present on **AD[31::00]**. During a write, it indicates the target is prepared to accept data. Wait cycles are inserted until both **IRDY#** and **TRDY#** are asserted together.

*Target Ready* is used in PacketBus in a different mode. The signal is used to indicate the target IS NOT ready for the transaction. If the **TRDY#** is active during the last clock when **FRAME#** is deasserted and **IRDY#** is still asserted, this indicates that the target, or targets in the case of a multicast or broadcast transaction, is not able to accept the packet and it must be resent. The reason for not being ready can be that the data is in 64 bit mode and the current target is a 32 bit target or that there is insufficient room in the receive FIFO for the data length being sent.

**STOP#** s/t/s

*Stop* indicates the current target is requesting the master to stop the current transaction in PCI mode.

*Stop* is used in PacketBus to indicate that a 32 bit target has been selected and the initiator is sending a 64 bit wide packet. The initiator is required to complete the current packet transmission and then to repeat the packet in narrow 32 bit mode.

**DEVSEL#**        *s/t/s*

*Device Select* is used to indicate that a target has been selected in PCI mode. In PacketBus mode the *Device Select* is used to indicate a target has been selected and has acknowledged the packet. This is only valid in directed transaction modes, Multicast and Broadcast modes assume completion if there is **STOP#** or **TRDY#** is not activated.

### 3.3.3 Arbitration Pins

**REQ#**            *t/s*

*Request* indicates to the arbiter that this agent desires use of the bus. This is a point to point signal. Every master has its own **REQ#** which must be tri-stated while **RST#** is asserted.

**GNT#**            *t/s*

*Grant* indicates to the agent that access to the bus has been granted. This is a point to point signal. Every master has its own **GNT#** which must be ignored while **RST#** is asserted.

**REQP[3::0]#** *o/d*

Request Priority lines are asserted on the rising edge of the **FRAME#**. They are used to specify the request priority of the device making the request for bus access..

### 3.3.4 Interrupt Pins

Interrupts on PCI are optional and have been removed from the HiRelPCI specification.

### 3.3.5 Geographical Address Pins

**SAD[15::00]**. *o/d and Fixed*

The **SAD[4::0]** are fixed in the backplane. **SAD[2::0]** uniquely identify each of the slots in a HiRelPCI bus segment while **SAD3** differentiates between each of the HiRelPCI buses that may exist on a common backplane in 6su mode (either left or right half of backplane). **SAD4** can alternately indicate the upper or lower bus in a 12su configuration. **SAD4** is should be pulled up and can be jumpered to indicate the different address between top and bottoms busses if needed. **SAD[15::5]** are connected to pullups and may be driven independently on the top and bottom busses by an attached bridge to indicate a negotiated set of node identifiers.

### 3.3.6 Communications Signals

**ENET0**            *ethernet*

Ethernet0 is the alternate maintenance bus and an alternate method of communications within the backplane environment. This ethernet line is a single wire on the backplane that extends over both bus segments of a 6su implementation and complements the Maintenance Bus (MB) described in the next section. This line is terminated with a 50 ohm resistor at each end of the chassis. This network may be connected to the outside world through a bridge on the CPU or the CSM to protect the integrity of the line.

**1394DATA**        *o/d*

IEEE 1394-1995 defines a backplane implementation of the standard on two lines. This set of connections is used for a low cost interface between the boards in the system and through a bridge to the attachments of the Firewire environment. In some environments this will be used as an alternate for the maintenance bus.

**1394CLK**      *o/d*

This is the second wire of the Firewire IEEE 1394-1995 interface.

### **3.3.7 Maintenance BUS (MB)**

This section differs materially from the PCI Specification 2.1 in that the IEEE Standard 1149.1 can not be implemented directly to each board in the system as the testing of the parts of the board would disturb the functionality of the bus. The IEEE Standard 1149.5, *Module Test and Maintenance Bus (MTM)* solves this problem by forming a test bus that interfaces to control processors on each board and causes that processor to test the internal function while not interfering with the functioning of the main bus. This bus consist of 4 lines plus a clock line.

**MCLK**      *t/s*

*Master Clock* is output from the master module and input of all Slave Modules.

**MMD**      *t/s*

*Master Data* is output from the master module and input of all Slave Modules.

**MSD**      *t/s*

*Slave Data* is output from each slave and input to the master module.

**MPR**      *t/s*

*Pause Request* is output from each slave module and input to the master module.

**MCTL**      *t/s*

*Control* is output from the current master module and input to the Slave modules.

A more complete description of this sub bus is in the Maintenance Bus Chapter.



## 4. Bus Operations

Bus command indicate to the target the type of transaction that the master is requesting. This information is transmitted on C/BE[3::0]# lines during the address phase of the transaction.

### 4.0.1 Command Definitions

The commands used on the HiRelPCI bus are shown in the table below. With the exception of the Packet Write command, the commands are the same as those defined in the PCI Local Bus Specification Revision 2.1 and it should be used to define these commands. The command encodings are true logic while the byte lane enables are negative true logic.

**Table 11—1**

C/BE[3::0]	Command Type
0000	(Interrupt Acknowledge)
0001	Special Cycle
0010	I/O READ
0011	I/O Write
0100	Reserved
0101	Packet Write
0110	Memory Read
0111	Memory Write
1000	Reserved
1001	Reserved
1010	Configuration Read
1011	Configuration Write
1100	Memory Read Multiple
1101	Dual Address Cycle
1110	Memory Read Line
1111	Memory Write and Invalidate

In this standard we will define the following extentions and modifications of the PCI Local Bus Specification:

1. Packet Write operation
2. Prioritized Arbitration
3. Removal of Interrupt Acknowledge operation
4. Redefinition of signal lines during Packet Write operations.

All other definitions should be referenced to the PCI Local Bus Specification Revision 2.1.

The *Interrupt Acknowledge* command is a read command directed to the system interrupt controller and has no logical target on a multiprocessor passive backplane. This command should be treated as a reserved command on the HiRelPCI bus.

The *Special Cycle* command provides a simple message broadcast mechanism on HiRelPCI. It is designed to be used as an event notification. This command is retained to be compliant with the PCI specification.

The *Packet Write* command is a write only command which is part of a split cycle response transaction. This command is added to the definition to provide packet message delivery throughout the larger system. Target addresses are decoded from the nodeId address of a specific board. Each segment may have multiple targets or no targets for these transfers. A complete description of these transactions will be found in Clause x.x. PacketBus operations (using the Packet Write command) are inherently a 64 bit address mode with the nodeId being the most significant 16 bits of the address with the lower 48 bits included in the header as the offset. This mode sends packets with 16 to 256 bytes nominally although other lengths and additional header information is available.

#### 4.0.2 Priority Bus Request Mechanism

This provides a prioritized request in addition to the normal round robin Fairness priority system of the PCI bus. This priority scheme uses four pins REQ0# to REQ3# to define the priority level of each bus request as level 0 to 15. Bus request are issued at the rising edge of FRAME# by activating thier REQ# and priority level on REQ[3:0]#. The priority level on the bus for this cycle is seen by all participants. Lower level requesters will retire and remove the REQ# if a higher level request is pending. The CSM arbitor determines the winner or winners and issues the bus grants to the boards with the highest level of priority on a round robin basis and then to the next lower level of priority on a round robin basis until there are no pending bus requests.

Priorized bus request may have 1 to 8 requests pending at any given level of request, the requests are granted in round robin priority until all are issued while building a queue at the same level to insure fairness within the bus request queueing level. This also guarentees priority among the bus request queueing levels.

Boards using the PCI style single REQ# line will request at bus request level 0. Any higher level of request will block the lower level request. This mechanism is transparent to the single level request mechanism and will be seen as possible a longer delay until a GNT# is received.

This bus request mechanism required 5 bus clocks to operate and will be will be ready to issue a grant with no delay for any bus operation that requires 5 clocks to complete and delays shorter bus operations by not issueing the grant until the 5th bus clock. For PacketBus operation there is no bus delays as required for Isochanous operation.

Bus GNT# is issued before the rising edge of FRAME# to the board that will drive FRAME# following the deactivation of IRDY# if the current bus operation is 5 bus clocks long. If the current bus operation is shorter than 5 bus cycles the GNT# will delay the bus for up to 3 bus cycles for event operations.

Each board shall have a mechanism to remove the REQ# assertion if a higher level of bus request is present and reassert it, with or without REQ0# to REQ3#, at the next rising edge of FRAME# or any clock edge that has both FRAME# and IRDY# and TRDY# inactive indicating an idle bus.

#### 4.0.3 Command Usage Rules

All HiRelPCI devices (except host bus bridges) are required to respond as a target to configuration (read and write) commands. All other commands are optional. Definitions of these commands with the exception of the Packet Write command are found in the PCI Local Bus Specification Revision 2.1.

### 4.1 Fundamental HiRelPCI Protocols

The basic bus transfer mechanism on HiRelPCI is a burst. The most efficient mode of data transfer in the HiRelPCI is the PACKET Write mode of operation with the split response cycles. A burst is composed of an

address phase and one or more data phases. Burst mode operation is enhanced through the HiRelPCI supports bursts in both memory and I/O Address Spaces.

All signals are sampled on the rising edge of the clock (except for **RST#**). Each signal has a setup and hold window with respect to the rising clock edge. Outside this window, signal values or transitions have no significance. This window occurs only on "qualified" rising clock edges for **AD[31::00]**, **AD[63::32]**, **PAR**, **PAR64**, and **IDSEL** signals and on every rising clock edge for **LOCK#**, **IRDY#**, **TRDY#**, **FRAME#**, **DEVSEL#**, **STOP#**, **REQ#**, **GNT#**, **REQ64#**, **ACK64#**, **SBO#**, **SDONE**, **SERR#** (on the falling edge of **SERR#** only), and **PERR#**. **C/BE[3::0]#**, **C/BE[7::4]#**. **RST#** is not synchronous. **SAD[15::0]** are fixed address and are not changed during a normal cycle. **SAD[4::0]** are hardwired on the backplane to provide unique addresses to each board position.

#### 4.1.1 Basic Transfer Control

Two modes of data transfer are used in HiRelPCI, the PACKET mode and the Normal mode.

Packet mode transactions are controlled using the **FRAME#**, **IRDY#**, **DEVSEL#**, **TRDY#** and **STOP#** signals. **FRAME#** and **IRDY#** are used in a manner similar to normal PCI while **DEVSEL#**, **TRDY#** and **STOP#** are redefined for PACKET mode operation (see Figure 5-1).

The standard PCI transaction of all HiRelPCI data transfers are controlled with three signals ( see Figure 5-2).

**FRAME#** is driven by the master to indicate the beginning and end of a transaction.

**IRDY#** is driven by the master to indicate that it is ready to transfer data.

**TRDY#** is driven by the target to indicate that it is ready to transfer data.

#### 4.1.2 Addressing

HiRelPCI defines Four physical address spaces. The *Memory* and *I/O Address Spaces* are customary. The *Configuration Address Space* has been defined to support HiRelPCI hardware configuration. The PacketBus Write addresses the 64 bit defined *Node Address Space*. Configuration Space has a limited Accesses as described in section 5.7.4.

HiRelPCI targets (except host bus bridges) are required to implement Base Address register(s) and CSR registers to access internal registers or functions. The Base Address registers are located in CSR register space and are addressable from both the Maintenance processor and through the main bus. The configuration software uses the Base Address register to determine how much space a device requires in a given address space and then assigns (if possible) where in that space the device will reside.

Node address space is defined in the CSR section and consists of a 16 bit node address which is the most significant address portion of the linear 64 bit address space. The remaining 48 bits of address are provided in the packets.

CSR space and Configurations space are mapped to each other to provide access using either packet operations or Configuration Cycle operation.

While the PCI specification specifies the use of the I/O address space, HiRelPCI does not recommend its use.

HiRelPCI supports two styles of address decoding: positive and subtractive. *Positive decoding* is faster since each device is looking for accesses in the address range(s) that it has been assigned. *Subtractive decoding* can be implemented by only one device on the bus since it accepts all accesses not positively decoded by some other agent. This decode mechanism is slower since it must give all other bus agents a "first right of refusal" on the access.

In PacketBus mode the address on AD[31::16] are compared with the soft nodeId stored in the configuration softId register or the hardId determined by access to the SAD[15::0] pins on the backplane. SAD[4::0] are determined by the physical position of the board (node) in the backplane to guarantee the uniqueness of the address. SAD[15::5] are determined by the bridge or are set into the backplane configuration switches or are left floating and read FFFxh where x is determined by the position of the board.

Addressing within each packet is defined by the 48 bit offset address. This is the lower 48 bits of the 64 bit address that defines a byte address. In the definitions of Packet mode, a more complete definition will be found for transfer of a single byte to 256 bytes in one transfer. A single node may contain 256 Terabytes (48 bit address). Each bus segment has up to 8 nodes with the bus bridges decoding and becoming agents for addresses not located on a given bus segment.

#### 4.1.3 Byte Alignment

Furthermore, HiRelPCI supports automatic bus sizing only in the PACKET mode operation where an operation may be repeated if a target is not capable of supporting a requested a QWORD (64 bit) transfer size.

In PACKET mode all transfers are aligned for 8 byte increments and all byte lanes are used for each transfers. Wide mode uses all 8 byte lanes, while narrow transfers use 4 byte lanes. All packets are multiples of 8 bytes.

#### 4.1.4 Bus Driving and Turnaround

A turnaround cycle is required on all signals that may be driven by more than one agent to avoid contention when driving the signals. This is indicated on the timing diagrams as two arrows pointing at each others' tail. This turnaround cycle occurs at different times for different signals. For instance, **IRDY#**, **TRDY#**, **DEVSEL#**, **STOP#**, and **ACK64#** use the address phase as their turnaround-cycle. **FRAME#**, **REQ64#**, **C/BE[3::0]#**, **C/BE[7::4]#**, **AD[31::00]**, and **AD[63::32]** use the Idle state between transactions as their turnaround cycle. The turnaround cycle for **LOCK#** occurs one clock after the current owner releases it. **PERR#** has a turnaround cycle on the fourth clock after the last data phase, which is three clocks after the turnaround-cycle for the AD lines. An Idle state is when both **FRAME#** and **IRDY#** are Reasserted (e.g., clock 9 in Figure 3-1).

All AD lines must be driven to stable values during 64-bit transfers every address and data phase. Parity must be calculated on all bytes regardless of the byte enables.

## 4.2 Bus Transactions

This section will be derived from Section 3.3 of the PCI Local Bus Specification Revision 2.1. Please see the Appendix for timing diagrams of transaction types.

### 4.2.1 PacketBus Write

PacketBus is a write only operation on HiRelPCI and is the preferred mode of high speed operation between nodes. This operation is also watched by any bus bridge for node addresses that are not on the current bus segment and may be forwarded via the bridge to the larger set of bus nodes. This does implement the 64 bit fixed address space defined in the IEEE 1212-1991 CSR standard.

This mode of operation is the equivalent of the IEEE P2100 Serial Express and IEEE 1596-1992 SCI in packet construction.

Several of the control lines are redefined during this PacketBus operation:

1. DEVSEL# is defined such that it is valid during the last cycle of the transfer to acknowledge the receipt of the packet. It may become active following the first address cycle and must be active during the last cycle (the checkword cycle). This signal is must drive the line high before releasing the drive.

2. TRDY# is redefined as an open collector signal during this mode of operation and the sense is reversed to indicate a NOT READY if activated to allow multiple devices to indicate the not ready conditions. If the TRDY# is active (low) at the during the checkword cycle a device on the bus is NOT READY which could indicate full buffers, wrong data width, or other reason for not receiving the packet and indication that the packet must be retransmitted. This line is used in conjunction with the STOP# line to indicate that packet must be retransmitted in narrow (32 Bit wide) mode.

3. STOP# is used by the 32 bit wide elements on the bus upon decoding the wide bit in the first transferred address/command cycle to indicate the reason for the TRDY# activation to be the data is in the width for the current target. This line is only activated if the targetId matches the board softId or the transaction is a broadcast/multicast and the channelId is required and not available on the 32 bit bus.

Data transfers fall into four general classifications:

**Table 12—Data-transfer services**

	<b>Directed</b>	<b>Multicast</b>
<b>Asynchronous</b>	directed transaction (read, write, lock, move)	multicast action (mMove)
<b>Isochronous</b>	directed stream (talk)	multicast stream (mTalk)

In all cases the packet is sent to completion any time one is started. The status of the packet is checked during the last transfer which is defined by the FRAME# being high (inactive) and the IRDY# being low (active). At that time the status conditions are checked as indicated above.

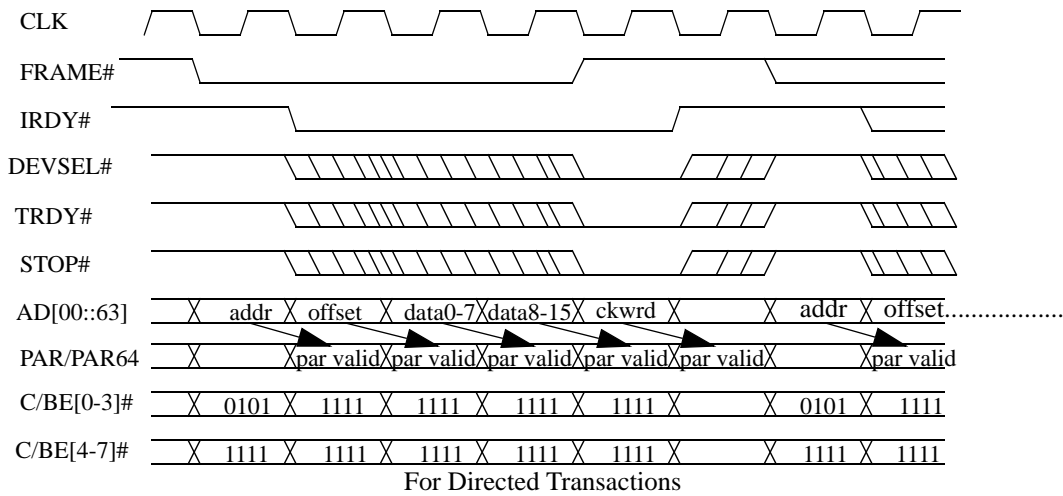
For Directed transactions all of the above listed signals are used. Each transaction on the PacketBus has a destination that will acknowledge the receipt of the information.

Multicast and broadcast operations do not use the DEVSEL# line as there may be more than one recipient for each transaction. The retry mechanisms for wrong width and device busy, STOP# and TRDY# are used to notify of changes needed in width and available buffer space. Each potential target is responsible to do

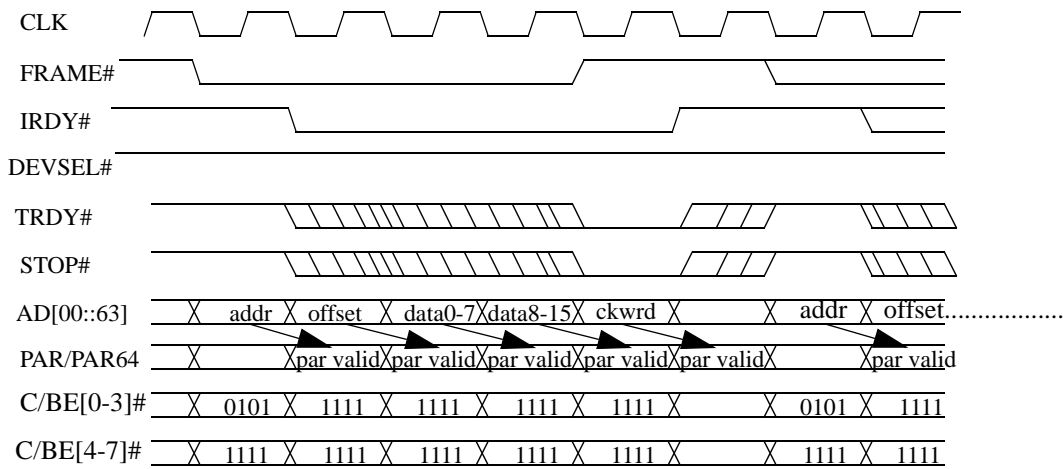
such decoding as is needed to capture the packet or cause the packet to be retransmitted until the data is captured on the board.

### 4.2.1.1 General Operation

Start a 64 bit transaction by placing the first octlet (8 byte) on the bus and lowering the FRAME# signal to indicate its validity. With each clock the following 8 byte frame elements are sent until the checksum is sent with the FRAME# inactive and IRDY# signal still active indicating the last cycle of the packet.



For Directed Transactions



For Multicast and Broadcast Operations

### 4.2.2 Transaction Termination

This section will be derived from Section 3.3.3 of the PCI Local Bus Specification Revision 2.1

### 4.3 Arbitration

Arbitration on the HiRelPCI bus is a combination of the PCI Local Bus Specification and the priority arbitration described in this document.

### **4.3.1 Arbitration Parking**

Arbitration parking is part of the PCI Local Bus Specification and modified to include the priority mechanism in a compatible manner.

## **4.4 Redundant Operation**

The heart of the HiRelPCI system is the redundant operation. There is no Single Point Of Failure in the system.

### **4.4.1 Operation With Full Dual Service Module, Dual Bus System**

This is the starting point of all operations with two fully functional CSM's (Central Services Modules) that provide redundant clocking to all boards. Each of the I/O boards has access to two PCI buses and two maintenance busses and to dual backplane ethernet and 1394 busses. The I/O through the high speed serial links is also duplicated.

With all functions in full operation there is twice the bandwidth available in each of the subsystems. This is in comparison to the TMR (Triple Modular Redundant) systems that use three processors in a voting configuration to determine the correct output of each operation. TMR processors do the same task and hence do not add to the bandwidth of the system.

### **4.4.2 Redundant operation with single processor dual bus system.**

In a single bus system with only one PCI structure the redundancy is provided by the backplane Ethernet connection and the backplane Maintenance Bus structure. These alternate connections allow all data to be reached in the event of failure of the main bus.

This single processor bus system also uses the slot adjacent to the CSM position as an alternate CSM position with all clocks wired through the second card. The clocks and other bus function can be transferred to the alternate board in event of failure of the first board. This restriction does preclude some of the available I/O positions as the I/O space is needed by the additional clock and arbitration outputs.

This system can continue operation with the loss of the main PCI bus but the system performance is limited by the bandwidth of the serial connections. The redundancy of the second processor is still required to avoid a single point of failure.

### **4.4.3 Redundant processor operation**

Dual processors are assumed for proper operation in the event of a processor card failure. Fault tolerance requires that the processors maintain all critical table in memory that will not all fail with a single processor. Duplicate tables are required for a minimum of:

- a. Task Tables
- b. Queing structures
- c. Check point tables
- d. Identification and initialization tables

A locking mechanism is needed to actively modify these tables. A procedure is needed to define the recovery from a partial set of tables that were in the process of modification at the time of failure. This may be a pointer to the previous table set as the backup. TBD

#### 4.4.4 Redundant operation over multiple bus segments

Addressing across multiple bus segments requires the same operation as normal redundant processor operation with the added complexity of working across a link between the segments. This is necessarily complicated by the requirement for redundant links between the segments.

#### 4.4.5 Methods of redundant operation

One assumption is operation of processors in pairs with each processor checking on the other member of the pair for proper checkpoint operation.

Other methods are to be determined.

#### 4.4.6 Address mapping of redundant elements

A control structure must be maintained that gives the EUI and node address of attached nodes. This map is maintained in duplicate and is signed by the control processor. This structure, with its encrypted signature, prevents aggressive intervention by an uncontrolled or hostile nodes.

#### 4.4.7 Resource Map definition

Overall structure of this map is shown in Table 11. This resource map defines the association between the

**Table 13—Resource Map Table**

Offset address	Length	Description
0	4	Table length in bytes
4	2	Map Name
6	2	Map Configuration node
8	2	Offset to 1st entry
A	6	Reserved
10	16 - 32	Map authentication
length - 4	4	Map checksum

assigned resourceId and the EUI and other address used to access this single element/node. The location of the map is stored in the CSR register set with an offset of 3C0 and is 8 bytes long. This will point to any location within the available memory of the system. The secondary pointer is located at 3C8 and is used if the primary map is corrupted. Individual resource contents of the map are shown in Table 12.

Entries in the overall map shown in Table 11 are:

1. Map overall length in bytes: This length is used to determine the space required for the table.

**Table 14—System Resource Map Entry**

Offset	length	Description
0	2 bytes	SoftId
2	2	Entry length and map
4	4	reserved.
8	8	Extended Unique Identifier
10	2	Configuration Node
12	2	Configuration signature
	n	Multicast/Broadcast pools
	16 - 32	Resv'd node public key
	4	checksum

2. Name of Table: This is used to distinguish between the several tables that are kept for redundancy purposes. The specific method of naming the table is TBD.
3. Configuration: This element defines the configuration of the table from basic 2 entry table to full table that contains the extended information about the node and the security for the node.
4. First entry offset: This is the offset from the beginning of the table to the location of the first resource entry.
5. Authentication: This element of the table is the authentication of the table. This is defined by the last modification of the table and the nodeId of the configuration node and possibly a signature of the configuration node.
6. Checksum; Over all 32 bit checksum for all entries in the table.

Entries for Resource map shown in Table 12 are;

1. SoftId: Assigned softId for the node. This is used to index into the table.
2. Entry length and map: Length of this entry and map number.
3. Reserved: Reserved for alignment reasons for the EUI to start on an 8 byte boundary.
4. EUI: Extended Unique Identifier for Node from the EUI attached to board.
5. Configuration Node: Node number of the node that configured this board.
6. Configuration Signature: Signature of the configuring node
7. Multicast/Broadcast Pool: This list is 2 byte (16bit) softId identifiers used in multicast and broadcast operations on this resource. This may contain 0 or more entries.
8. Reserved for node publickey signature: For more secure environments, This public key plus the private key maintained on the board, authenticate operation and changes to the setup of the board.
9. Entry Checksum: Checksum for this entry in the table.

#### 4.4.8 Testing of redundant elements

To Be Added

#### 4.4.9 Fault testing mechanisms

To Be Added

#### **4.4.10 Switching of Redundant Elements for Reliability**

To Be Added.

#### **4.4.11 Start-up of redundant elements**

To Be Added

#### **4.4.12 Resynchronization of Redundant Elements**

Reestablish connection base on the tables above and determine sync.

## 5. PacketBus Operation

PacketBus operations are added to the basic parallel bus structure to improve the performance of the system with a split response mode of operation. This mode is also known as a write only mode of bus operation in that each request and response is a write transaction. The basic function is to free the bus resource during the time needed to fetch or process the data at the target end of the transaction. Packet mode operation does impose additional requirements on the participants as buffers are needed to handle the packets to and from the bus.

PacketBus operation is optional for the HiRelPCI bus. This mode can be used to produce the highest bandwidth operations on the bus with greater than 4 gigabit throughput a reality.

Note that transaction codes used here are taken from the P2100 draft and may vary as that standard is approved. Direct mapping of the transaction codes to the IEEE 1596 - 1992 SCI will be assigned.

PacketBus operations require the use of a new command mode in the PCI specification and the redefinition of several of the PCI signal lines during this mode of operation. In all cases these changes are transparent to the existing PCI devices.

The preferred command mode is the "0101" mode which would be an unused WRITE transaction. Current devices are required by PCI Local Bus Specification Revision 2.1 to ignore this command code and hence no interference with the existing operations.

Several of the control lines are redefined during this PacketBus operation:

1. DEVSEL# is defined such that it is valid during the last cycle of the transfer to acknowledge the receipt of the packet. It may become active following the first address cycle and must be active during the last cycle (the checkword cycle). This signal is must drive the line high before releasing the drive.

2. TRDY# is redefined as an open collector signal during this mode of operation and the sense is reversed to indicate a NOT READY if activated to allow multiple devices to indicate the not ready conditions. If the TRDY# is active (low) at the during the checkword cycle a device on the bus is NOT READY which could indicate full buffers, wrong data width, or other reason for not receiving the packet and indication that the packet must be retransmitted. This line is used in conjunction with the STOP# line to indicate that packet must be retransmitted in narrow (32 Bit wide) mode.

3. STOP# is used by the 32 bit wide elements on the bus upon decoding the wide bit in the first transferred address/command cycle to indicate the reason for the TRDY# activation to be the data is in the width for the current target. This line is only activated if the targetId matches the board softId or the transaction is a broadcast/multicast and the channelId is required and not available on the 32 bit bus.

Priority encoding as used in Serial Express is not used in P1996 as a 16 level priority system is available in HiRelPCI and replaces the encoding in P2100.

Data transfer in PacketBus mode of operation fall into four general classifications as shown in Table 19.

In all cases the packet is sent to completion any time one is started. The status of the packet is checked during the last transfer which is defined by the FRAME# being high (inactive) and the IRDY# being low (active). At that time the status conditions are checked as indicated above.

For Directed transactions all of the above listed signals are used. Each transaction on the PacketBus has a destination that will acknowledge the receipt of the information.

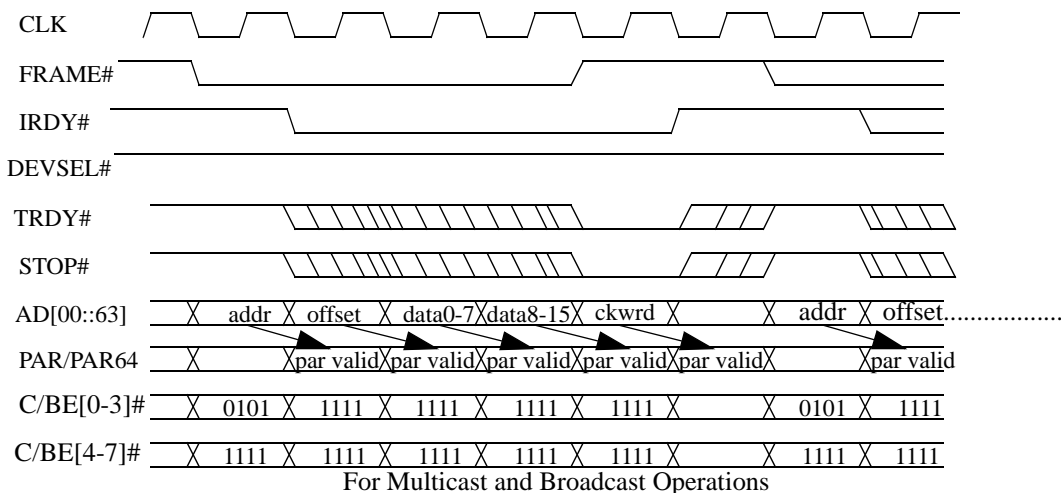
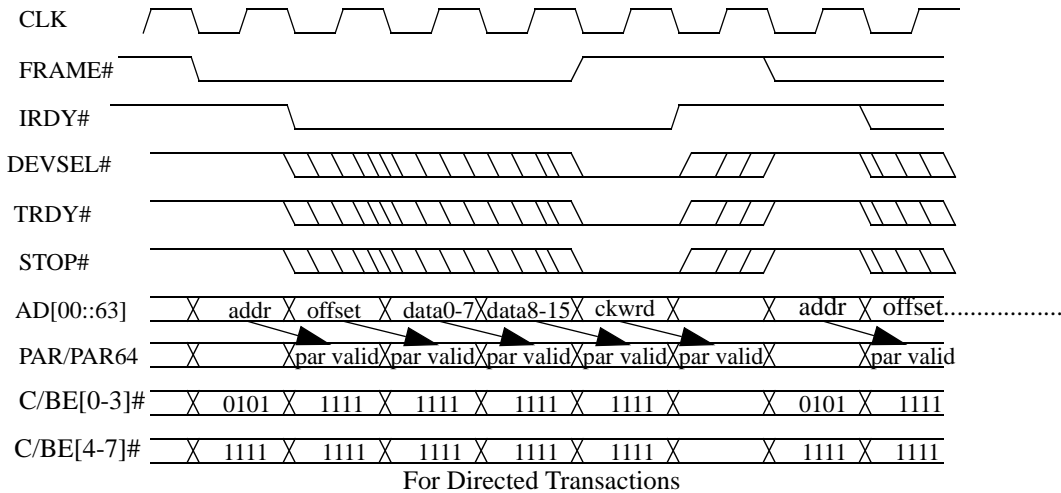
**Table 15—Data-transfer services**

	Directed	Multicast
Asynchronous	directed transaction (read, write, lock, move)	multicast action (mMove)
Isochronous	directed stream (talk)	multicast stream (mTalk)

Multicast and broadcast operations do not use the DEVSEL# line as there may be more than one recipient for each transaction. The retry mechanisms for wrong width and device busy, STOP# and TRDY# are used to notify of changes needed in width and available buffer space. Each potential target is responsible to do such decoding as is needed to capture the packet or cause the packet to be retransmitted until the data is captured on the board.

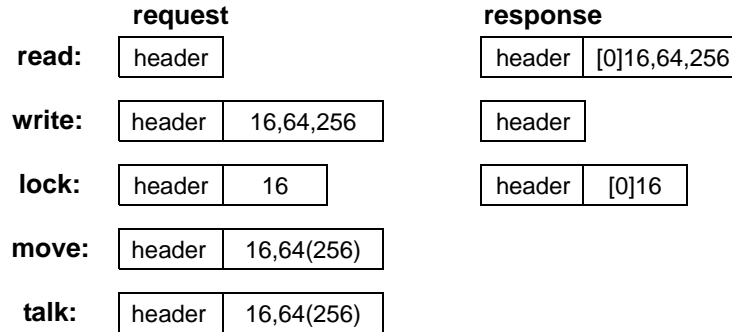
**5.0.1 General Operation**

Start a 64 bit transaction by placing the first octlet (8 byte) on the bus and lowering the FRAME# signal to indicate its validity. With each clock the following 8 byte frame elements are sent until the checksum is sent with the FRAME# inactive and IRDY# signal still active indicating the last cycle of the packet.



Packets of 32 bit width follow a similar procedure with the exception of the width bit is set to 0 and the checkword is a 32bit checkword. Drawings to follow.

The following transfer packets at to be used:



## 5.1 Common packet components

P2100 Packets are transferred to P1996 through a bridge. During the transfer process the packets are rearranged to meet the byte ordering requirement of the HiRelPCI bus and to replace serial bus specific data with information more relevant to the parallel bus environment. Specific mappings are shown where data order is required.

The operations in packet mode must be compatible with transparent operations of boards using normal interface IC's that are compatible with the PCI Local Bus Specification Revision 2.1. Implementations that could be made on normal PCI boards using the signals available to the Revision 2.1 definition must be available.

Packet transfers are designed for optimum operation for 64 bit parallel operation. Operation on a 32 bit wide data path is accomplished by performing 2 write cycles for each 64 bit word. All packets are aligned to present the required initial decoding targetId and operations code on the AD[31::0] for easy decoding.

Serial Express has 3 level of priority, normal, higher and highest. The equivalent of the normal priority is through the PCI style request/grant mechanism which is a fairness algorithm. The highest priority access is obtained via the LOCK# line that permits the current occupant to hold the bus in an unfair mechanism. The intermediate priority, 'higher', can be accomplished, a priori, by controlling the Latency Timer, offset 0Dh in the Configuration Space Header. Real time implementation of the 'higher' priority is under study.

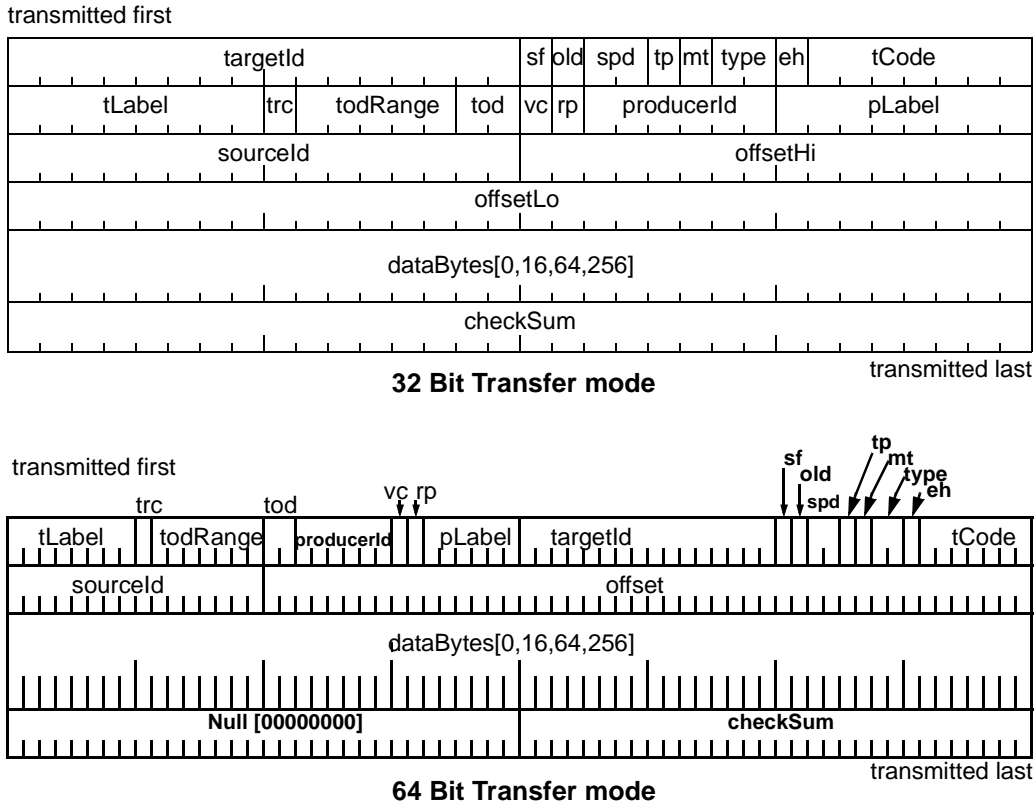
### 5.1.1 Generic packet components

The initial and final portions of all PacketBus packets have the same format, as illustrated in . Within this figure, the common fields are white and the packet-type-dependent fields are shaded gray. These packets can be sent as either 64 bit packets or as 32 bit packets depending on the width acceptable on a given bus path. If the path width is set to 1 a 64 bit path width is attempted. If the width bit is 0 the packet is transferred as a 32 bit operation.

## 5.2 Split-response packet formats

### 5.2.1 Request-packet components

Request-packet formats are illustrated in figure 1.



**Figure 1—Request packet formats**

The 16-bit **targetId** (target identifier) field specifies the address where this packet is to be stripped.

The **sf** (special format) is 1 for the packet formats defined in this P1996 standard.

The **old** (old packet) bit has no meaning in P1996 standard.

The 2-bit **spd** (speed of stripper) field is set by the sender when the packet is created and identifies the speed capabilities of the sending node. For PacketBus the encodings currently defined are:

00 - Indicates 32 bit operation requested on the backplane

01 - Indicates 64 bit operation requested on the backplane

The **tp** (transaction priority) bit is not used in P1996, it is replaced with the priority arbitration mechanism.

The **mt** (move or talk) bit is 0 for basic packet types; a 1 value identifies move-send and talk-send packets.

The 2-bit *type* (packet-type code) field specifies the basic packet type, distinguishing between send packets and acknowledge packets, as specified in table 16.

**Table 16—*type* (packet type) field values**

Value	Name	Description
0	ACKNOWLEDGE	Acknowledge packet
1	SEND_QUICK	Unorderable send packet
2	SEND_ORDER	Orderable send packet
3	SEND_FENCE	Ordered send packet, within orderable stream

The SEND\_QUICK packets have no ordering requirements and may be routed through primary and alternate paths. The SEND\_ORDER packets have ordering constraints and shall be routed through the primary path. The SEND\_FENCE packet is a form of SEND\_ORDER packet, whose delivery shall be delayed until all previously issued SEND\_ORDER packets have been delivered. Prespecifying the send-packet ordering constraints allows selective application of ordering constraints (as required), to improve the efficiency or ordered delivery on a variety of implementation technologies.

The *eh* (extended header) bit identifies packets with an extended header, whose size is therefore 8 bytes larger than normal.

Within send packets, the 7-bit *tCode* field (transaction code) specifies which form of data transfer is to be performed; see 5.3.2 and 5.6.5 for details.

The 8-bit *tLabel* field specifies an 8-bit sequence-identifier value that, when concatenated with the request's *targetId* and *sourceId* fields (or the response's *checkId* and *targetId* fields), uniquely identifies outstanding transactions. In theory, this allows each node to have  $2^{24}$  outstanding transactions ( $2^8$  *tLabel* values for each of  $2^{16}$  possible targets). Simple nodes are expected to have only one set of *tLabel*-allocation hardware, their *sourceId* and *tCode* values uniquely identify concurrently active transactions, and at most  $2^8$  transactions can be concurrently active.

The *trc* bit, the 5-bit *todRange* field, and the 2-bit *tod* field contain diagnostic tracing and time-of-death information; see xx for details. These fields are not required for P1996 operation and will be ignored.

The *vc* (virtual circuit) bit indicates which of the virtual-circuit queues shall be used: when 0, the primary virtual-circuit queues shall be used; when 1, the alternate virtual circuit queues (if any) shall be used.

The *rp* (retry phase) bit provides queue-reservation phase information, as specified in table 17. All subactions start using the NOTRY phase, the oldest request and the oldest response inherit the DOTRY state after a BUSY1 status is returned.

**Table 17—*ph* (retry phase) values**

Value	Name	Description
0	NOTRY	Reservationless send
1	DOTRY	Reservation-assigned send

The 6-bit *producerId* field is the 6 least-significant bits of the producer’s hardId identifier, to which the acknowledge packet shall be addressed. The 8-bit *pLabel* field is a ringlet-local send-packet identifier that is included in any acknowledge packets that are generated; its detailed meaning is producer-technology dependent.

The 16-bit *sourceId* field contains the nodeId where the request packet was created and (for split-response transactions) specifies the address to which the response should be returned.

The concatenation of the 16-bit *offsetHi* and 32-bit *offsetLo* fields provides address-offset information to the responder. A small portion of the *offset* locations is standardized, for node identification and configuration purposes; see the clause on Control and Status Register and Configuration space for definitions of the required registers and their configuration.

The *dataBytes* field provides data to the responder; it shall be 0, 16, 64, or 256 bytes.

The 32-bit *checksum* (cyclic-redundancy-check, 32 bits) field provides CRC coverage for the packet, using the CRC-32 algorithm specified in xx.

### 5.2.2 Response-packet components

Response-packet format is similar to the request-packet format, but the *offsetHi/offsetLo* fields are replaced with responder status information, as illustrated in figure 2.

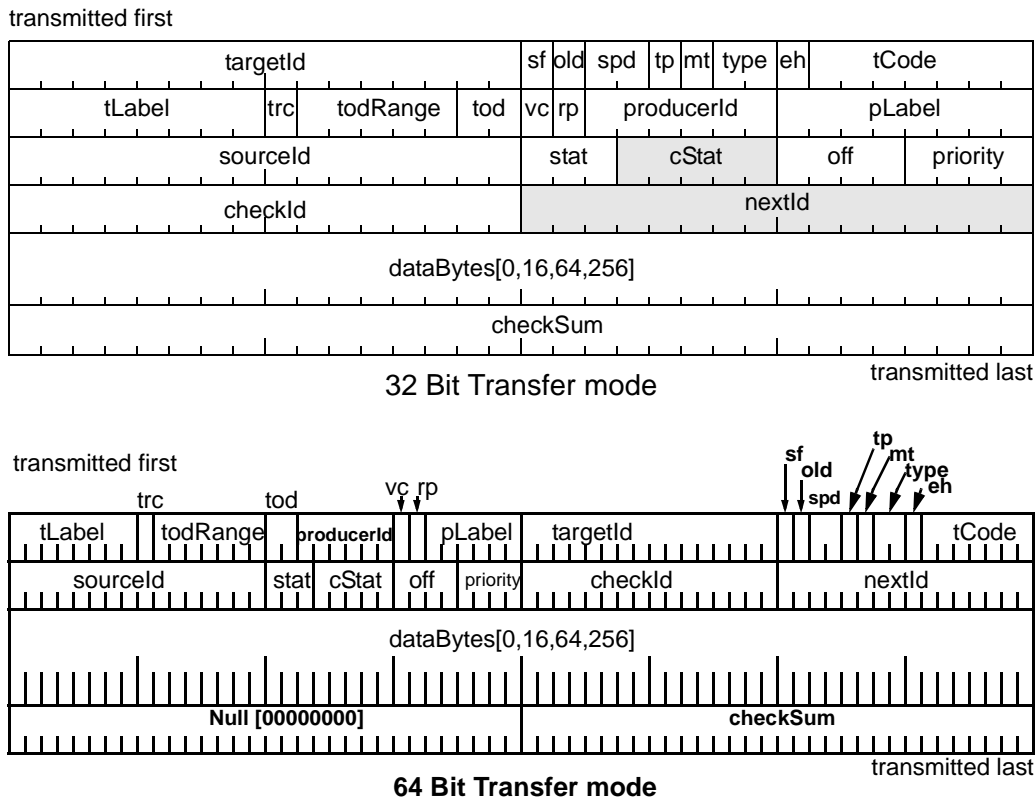


Figure 2—Response packet formats

The 16-bit *targetId* (target identifier) field, the *sf* (standard format), the *old* (old packet) bit, the 2-bit *spd* (speed of stripper) field, the *tp* (transaction priority) bit, the *mt* (move or talk) bit, the 2-bit *type* (packet-type

code) field, the *eh* (extended header) bit, and the 7-bit *tCode* (transaction code) field have the same definitions for request and response packets; see 5.2.1 for details.

The 8-bit *tLabel* field specify the packet’s priority and uniquely label its outstanding transactions (see xx). The *trc* bit, the 5-bit *todRange* field, and the 2-bit *tod* field contain diagnostic tracing and time-of-death information . The *tLabel*, *trc*, *todRange* and *tod* fields shall be created at the responder by copying the like-named fields from the affiliated request packet.

The *vc* (virtual circuit) bit, the *rp* (retry phase), the 6-bit *producerId* field, and the 8-bit *pLabel* field have the same definitions for request and response packets; see 5.2.1 for details.

The 16-bit *sourceId* field provides the nodeId where the actual responder. This is normally (but not always, see xx) the same as the targetId within the affiliated request packet.

The 4-bit *stat* field specifies the standard-transaction status; see 5.3.4 for details. The 5-bit *cStat* (complementary status) is reserved for extensions to this standard; see for details. The 4-bit *off* field (first response data-word offset) is reserved for critical-word-first extensions to this standard; see 5.5.1 for details. The 4-bit *priority* field specifies the transaction priority.

The 16-bit *checkId* field provides the nodeId where the intended responder: a copy of the targetId from the creating request.

The 16-bit *nextId* field is reserved for coherent extensions to this standard: it shall be set to zero when the response is created and shall be ignored when the response packet is processed.

The 32-bit *checkSum* field is defined in 5.2.1.

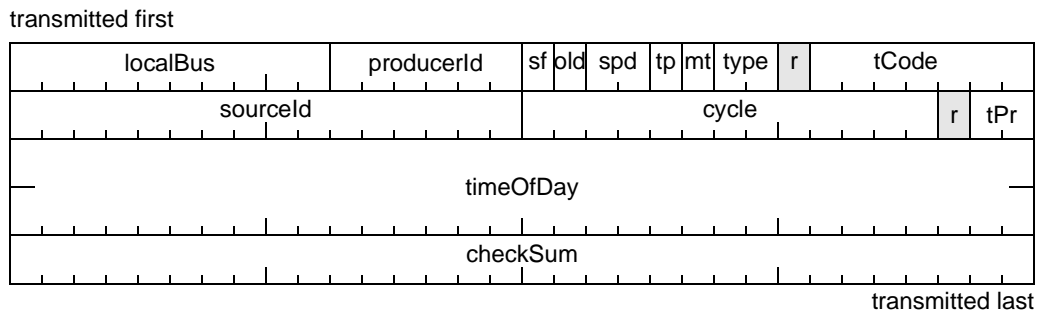
### 5.2.3 Acknowledge packet components

Within the backplane environment the acknowledgement is accomplished with the use of DEVSEL#.

## 5.3 Special packet formats

### 5.3.1 timeSync packet formats

The timeSync packets contain information for synchronizing nodes’ time-of-day clocks, as illustrated in .



**Figure 3—timeSync packet format**  
 Copyright 1998 IEEE. All rights reserved

The 10-bit *localBus* and 6-bit *producerId* field provide the 16-bit hardId address of the producer, to which the timeSync packet returns. The *localBus* field is a constant all-ones value; the *producerId* field is the least-significant bits of the producer's hardId.

The *sf* (special format), the *old* (old packet) bit, the 2-bit *spd* (speed of consumers) field, and the *tp* (transaction priority) bit the same definitions for request and timeSync packets; see 5.2.1 for details. The *mt* (move or talk) bit shall be zero.

The 2-bit *type* (packet-type code) field has the same definitions for request and timeSync packets; see 5.2.1 for details. The *r* bit shall be reserved. A distinct 7-bit *tCode* value identifies the timeSync packet; see 5.3.2 for details.

The 16-bit *sourceId* field is the softId of the requester, which originated the timeSync operations, and is expected to be used for timeSync routing purposes.

The 13-bit *cycle* field identifies the isochronous cycle when the timeSync packet was generated by the ringlet-local producer. The *r* bit shall be reserved.

The two-bit *tPr* (time precedence) field specifies the precedence level of the timeSync packet: 0-to-3 values correspond to the lowest-to-highest precedence respectively.

The 64-bit *timeOfDay* field identifies the time that this packet passed through the previous node and is used for time-of-day clock synchronization purposes; see xx for details.

The 32-bit *checksum* field is defined in 5.2.1.

### 5.3.2 Basic *tCode* values

For directed-transaction requests, the *tCode* value specifies how the request packet's data shall be used, the operation that shall be performed at the responder, and what form of response shall be returned. For directed-transaction responses, the *tCode* values provide status and data values. These *tCode* values shall be as specified within table 18.

*Constrained: Within some (register-only) spaces, only 4-byte transfers to aligned addresses are supported. If the transfer size or address alignment are unsupported, a type\_error-response shall be returned.*

The **read16** request initiates a 16-byte read. In the absence of an error, 16 bytes of data shall be returned. For the convenience of existing memory-mapped peripherals (whose side-effects depend on which bytes are read) a 0-through-16 byte address range specifies which bytes shall be valid (other returned values shall be ignored, see 5.4.1).

*Constrained: Within some (register-only) spaces, only 4-byte transfers to aligned addresses are supported. If the transfer size or address alignment are unsupported, a type\_error-response shall be returned.*

The **write16** request initiates a 16-byte write; a 0-byte response shall be returned. Although the request always contain 16 bytes of data, a 0-through-16 byte address range selects which bytes shall be written (other data values shall be ignored, see 5.4.1).

*Optional: Shall be implemented on memory, optional within simple (register-only) address spaces. If the tCode is not supported, a type\_error-response shall be returned.*

The **lock** request initiates an indivisible read-modify-write operation, updating the target address based on *arg0* and *arg1* values contained within the request. The two least-significant bits (and for some commands, three least-significant bits) of the address shall be zero, but shall be ignored when the request is processed. The update operation is specified by the *s* bits (one of 16 subcommands) within the *tCode* field (see 5.3.3 for details).

**Table 18—tCode assignments, normal (mt=0) subactions**

tCode		packet payload	response payload	Name	Description
msbs	lsbs				
000	eeee	16	0	write16	write data, 0-16 bytes
001	ssss	16	16	lock4,lock8	indivisible read/modify/write operations
010	eeee	0	16	read16	read data, 0-16 bytes
011	0000	"	-na-	reset	reset packet, for initialization purposes
	0001	"	-na-	timeSync	time synchronization information
	0010	"	-na-	leaderA	leader for following packet
	0011	"	-na-	leaderB	leader for following packet, fault retry
	01xx	"	-na-	countSync	scrubber's count-maintenance
	1000-1111	"	—	—	reserved
100	0000	0	64	readLo64	read low-range data, 1-64 bytes
	0001	"	"	readHi64	read high-range data, 1-64 bytes
	0010-1101	"	"	read64	64-byte read, critical-word ordering
	1110	"	"	cRead64	reserved (see )
	1111	"	-na-	response00	response, 64 bytes (or less)
100	0000	0	256	readLo256	read low-range data, 1-256 bytes
	0001	"	"	readHi256	read high-range data, 1-256 bytes
	0010-1101	"	0	mark64	reserved (see )
	1110	"	"	cMark64	mark, cache-to-cache
	1111	16	-na-	response16	response, 16 bytes
110	0000	64	0	writeLo64	write low-range data, 1-64 bytes
	0001	"	0	writeHi64	write high-range data, 1-64 bytes
	0010-1101	"	0	write64	reserved (see )
	1110	"	0	cWrite64	reserved (see )
	1111	"	-na-	response64	response, 64 bytes (or less)
111	0000	256	0	writeLo256	write low-range data, 1-256 bytes
	0001	"	0	writeHi256	write high-range data, 1-256 bytes
	0010-1101	"	0	—	reserved
	1111	"	-na-	response256	response, 256 bytes

Bit value names:

eeee — least-significant bits of the ending-byte address, for selected-byte transactions

-sss — lock transaction subcommand codes

----e — distinguished between Lo and Hi variants of the selected-range transactions

*Optional: Not expected to be supported within simple (register-only) address spaces.*

*If this tCode is not supported by the responder, a type\_error-response shall be returned.*

The **readLo64** and **readHi64** requests initiate 64-byte reads. In the absence of an error, a 64-byte block containing the selected data shall be returned. The data transfer ends-with or starts-with the byte-aligned address, for the readLo64 and readHi64 requests respectively (see 5.4.3).

*Optional: Only expected to be supported on the highest-performance asynchronous nodes.*

*If this tCode is not supported by the responder, a type\_error-response shall be returned.*

The **readLo256** and **readHi256** requests initiate 256-byte reads. In the absence of an error, a 256-byte block containing the selected data shall be returned. The data transfer ends-with or starts-with the byte-aligned address, for the readLo256 and readHi256 requests respectively (see 5.4.3).

*Optional: Only coherent caches and memories are expected to support these commands.*

*If not supported, shall be aliased to read64, ignoring the 6 least-significant address bits.*

An **read64** request initiates a 64-byte critical-word-first read. In the absence of an error, a 64-byte block containing the requested data shall be returned in critical-word-first order. Larger values of the *cccc* field specify the coherence update being performed (if any).

*Optional: Not expected to be supported within simple (register-only) address spaces.*

*If this tCode is not supported by the responder, a type\_error-response shall be returned.*

The **writeLo64** and **writeHi64** requests initiate 64-byte writes. In the absence of an error, the selected bytes within the request's 64-byte block shall be written into the specified responder locations; a 0-data-byte response shall be returned. The data transfer ends-with or starts-with the byte-aligned address, for the writeLo64 and writeHi64 requests respectively (see 5.4.3).

*Optional: Only expected to be supported on the highest-performance asynchronous nodes.*

*If this tCode is not supported by the responder, a type\_error-response shall be returned.*

The **writeLo256** and **writeHi256** requests initiate 256-byte writes. In the absence of an error, the selected bytes within the request's 256-byte block shall be written into the specified responder locations; a 0-data-byte response shall be returned. The data transfer ends-with or starts-with the byte-aligned address, for the writeLo256 and writeHi256 requests respectively (see 5.4.3).

*Optional: Only coherent caches and memories are expected to support these commands.*

*If not supported, shall be aliased to read64, ignoring the 6 least-significant address bits.*

An **write64** request initiates a 64-byte write into (memory or cache) storage. For cache-to-cache write transactions, an extended header is expected to provide additional cache-indexing and update information.

The **mPrepend** leader identifies an 8-byte ringlet-local packet prepend, where the prepend is provided to ensure delivery of the trailing packet to the producer-specified consumer.

The **resp00** response is a 0-data-byte second phase of a split-response transaction.

The **resp16** response is a 16-data-byte second phase of a split-response transaction.

*Optional: The simplest nodes only support 16-byte accesses, so resp64 is never returned.*

*If not expected, the response packet shall be discarded and an error indication shall be set.*

The **resp64** response is the 64-data-byte second phase of a split-response transaction.

*Optional: Most nodes only support 64-byte accesses, so resp256 is never returned.*

*If not supported, the response packet shall be discarded and an error indication shall be set.*

The **resp256** response is a 256-data-byte second phase of a split-response transaction.

### 5.3.3 Lock subcommand-code values

For lock subcommands, the LSBs of the *tCode* value specify which lock operation is performed, as specified in table 19.

**Table 19—lock4 subcommand values**

Value	Name	Update action
0	FETCH_ADDLO4	*ptr= AddLoToHi(old, arg0); return(old);
1	FETCH_ADDHI4	*ptr= AddHiToLo(old, arg0); return(old);
2	COMPARE_SWAP4	if (old==arg0) *ptr= arg1; return(old);
3	MASK_SWAP4	new= (arg0)   (old & ~arg1); return(old);
4	EXTRA_ADDLO8	Extended FETCH_ADDLO8, see xx
5	EXTRA_ADDHI8	Extended FETCH_ADDHI8, see xx
6-7	—	reserved
8	FETCH_ADDLO8	*ptr= AddLoToHi(old, arg0); return(old);
9	FETCH_ADDHI8	*ptr= AddHiToLo(old, arg0); return(old);
10	COMPARE_SWAP8	if (old==arg0) *ptr= arg1; return(old);
11	MASK_SWAP8	new= (arg0)   (old & ~arg1); return(old);
12	BLIND_ADDLO8	*ptr= AddLoToHi(old, arg0); return(final ? old : VOID);
13	BLIND_ADDHI8	*ptr= AddHiToLo(old, arg0); return(final ? old : VOID);
14	NULL_SWAP8	if (old==arg0) *ptr= arg1; return(first ? old : VOID);
15	MASK_STORE8	new= (arg0)   (old & ~arg1); return(first ? old : VOID);

Notes:

AddHiToLo() is addition; bytes with the lower addresses are the more significant.  
 AddLoToHi() is addition; bytes with the lower addresses are the less significant.

The FETCH\_ADDLO4 adds the *arg0*-specified argument to the 4-byte target address. When the addition is performed, the bytes with the lowest-through-highest addresses are assumed to be the least-through-most significant. The previous (unmodified) *old* data value shall be returned.

The FETCH\_ADDHI4 adds the *arg0*-specified argument to the target address. When the addition is performed, the bytes with the highest-through-lowest addresses are assumed to be the least-through-most significant. The previous (unmodified) *old* data value shall be returned.

The COMPARE\_SWAP4 performs a bit-wise comparison of the *arg0* and *old* values. If all bits are the same, the addressed 4-byte location is set to the *arg1* value. The computation of the result is not affected by the arithmetic significance of the data values. The previous (unmodified) *old* data value shall be returned.

The MASK\_SWAP4 computes a *result* value by OR'ing the *arg0* value with selected bits of the *old* data value; the addressed 4-byte location is set to this *result* value. The computation of the result is not affected by the arithmetic significance of the data values. The previous (unmodified) *old* data value shall be returned.

The EXTRA\_ADDLO8 is a variant of FETCH\_ADDLO8, with extended performance capabilities; see xx for details. The EXTRA\_ADDHI8 is a variant of FETCH\_ADDHI8, with extended performance capabilities; see xx for details.

The FETCH\_ADDLO8 adds the *arg0*-specified argument to the 8-byte target location. When the addition is performed, the bytes with the lowest-through-highest addresses are assumed to be the least-through-most significant. The previous (unmodified) *old* data value shall be returned.

The FETCH\_ADDHI8 adds the *arg0*-specified argument to the 8-byte target location. When the addition is performed, the bytes with the highest-through-lowest addresses are assumed to be the least-through-most significant. The previous (unmodified) *old* data value shall be returned.

The COMPARE\_SWAP8 performs a bit-wise comparison of the *arg0* and *old* values. If all bits are the same, the target 8-byte location is set to the *arg1* value. The computation of the result is not affected by the arithmetic significance of the data values. The previous (unmodified) *old* data value shall be returned.

The MASK\_SWAP8 computes a *result* value by OR'ing the *arg0* value with selected bits of the *old* data value; the addressed 8-byte location is set to this *result* value. The computation of the result is not affected by the arithmetic significance of the data values. The previous (unmodified) *old* data value shall be returned.

The BLIND\_ADDLO8 adds the *arg0*-specified argument to the 8-byte target location. When the addition is performed, the bytes with the lowest-through-highest addresses are assumed to be the least-through-most significant. When multiple requests combine, all but the final request (that updates memory with the cumulative result) may have a VOID value returned.

The BLIND\_ADDHI8 adds the *arg0*-specified argument to the 8-byte target location. When the addition is performed, the bytes with the highest-through-lowest addresses are assumed to be the least-through-most significant. When multiple requests combine, all but the final request (that updates memory with the cumulative result) may have a VOID value returned.

The NULL\_SWAP8 performs a bit-wise comparison of the *arg0* and *old* values. If all bits are the same, the target 8-byte location is set to the *arg1* value. The computation of the result is not affected by the arithmetic significance of the data values. When multiple requests combine, all but the first request (that forces memory to a non-NULL value) may have a VOID value returned.

The MASK\_STORE8 computes a *result* value by OR'ing the *arg0* value with selected bits of the *old* data value; the addressed 8-byte location is set to this *result* value. The computation of the result is not affected by the arithmetic significance of the data values. When multiple requests combine, all but the final request (that updates memory with the cumulative result) may have a VOID value returned.

NOTE:

The FORCE\_ADD\_LO8, FORCE\_ADD\_HI8, NULL\_SWAP8, and MASK\_STORE8 operations are (potentially) more efficient versions of the FETCH\_ADD\_LO8, FETCH\_ADD\_HI8, COMPARE\_SWAP8, and MASK\_SWAP8 lock operations. By restricting the argument value or allowing a VOID value to be returned (rather than the previous addressed value), these operations can be combined within switches. With the coherent version of NULL\_SWAP, sequential-use-lock thrashing can be minimized by delaying cache-line-ownership transfers until the requested data becomes NULL.

### 5.3.4 *stat* (standard status) values

The *stat* field is returned from the response packet, as summarized in 5.2.2; the functional meanings of this field's values are specified in table 20.

**Table 20—*stat* field values**

Value	Name	Description
0	NORMAL	Completion successful
1	—	Reserved, for normal-operation status
2	SIZE_ERROR	Reserved for other error indications
3	LOSS_ERROR	Addressing error, possibly unpowered segment related
4	CONFLICT	Queue conflict (end-to-end retry)
5	DATA_ERROR	unrecoverable data-transfer error
6	TYPE_ERROR	unsupported command or length
7	ADDR_ERROR	addressing error

The *stat* status value within the response is generated when the request is consumed by the responder (or an intermediate agent, acting on its behalf). The use of these status codes is further detailed below:

- a) NORMAL. Nodes shall respond with NORMAL in the circumstances described below (this is not an exhaustive list, but includes some examples of circumstances for which there might be confusion with other response codes):
  - 1) A write request is received for a writeable address that contains read-only bits or fields. The transaction completes successfully and the write effects on the read-only bits are as specified in xx or a unit architecture document. Generally an address is not considered writeable if all bits are read-only; see the discussion of TYPE status below.
- b) SIZE\_ERROR. Responder nodes shall respond with SIZE\_ERROR in the circumstances described below:
  - 1) A request is received, the *tCode* value is recognized and the checksum is valid, but the size of the data packet doesn't match its expected size, typically because the packet was truncated when passing through an unsupportive bridged. See xx for details.
- c) LOSS\_ERROR. Responder nodes shall respond with LOSS\_ERROR when an ADDR\_ERROR would have been reported, if the addressing error could be due to accessing an unpowered segment.
- d) CONFLICT. Nodes shall respond with CONFLICT in the circumstances described below:
  - 1) A request is received but the resources required to act upon the request are not available. The requester may reasonably expect a resent request to succeed in the future when the resources are available. Returning a CONFLICT status, rather than busying a request, is appropriate when an end-to-end retry is necessary to avoid a potential deadlock. For example, a SerialExpress-to-PCI bridge is expected to terminate reads with a CONFLICT status if a posted PCI-to-SerialExpress write has to be completed before additional read-requests can be processed. See xx for details.
- e) DATA\_ERROR. Nodes shall respond with DATA\_ERROR in the circumstances described below:
  - 1) For read requests, an otherwise valid packet is received but a hardware error at the node prevents the return of the requested data. For example, an uncorrectable ECC error reported by the underlying medium shall be reported as resp\_data\_error.

- 2) For write or lock requests, an otherwise valid packet is received but a hardware error at the node prevents the updates from being performed.
- f) TYPE\_ERROR. Nodes shall respond with TYPE\_ERROR status in the circumstances described below:
- 1) Reserved *tCode*. A request packet is received with a *tCode* field (transaction *tCode*) value that is reserved by this SerialExpress standard.
  - 2) Invalid *tCode*. A request is received with a valid *tCode* value, but the referenced address does not support the specified *tCode* value. An example of this is a write request to an address that is entirely read-only (note that this is distinct from a write request that references an otherwise writeable location that contains read-only bits or fields). Another example is a request whose *tCode* specifies a lock operation but the destination address supports only read and write operations.
  - 3) Illegal alignment. A request packet is addressed to a valid *destination\_ID*, the *destination\_offset* references an address implemented by the node but the alignment of the destination offset does not match the node's alignment requirements. For example, a quadlet register is implemented but cannot respond to a one byte read or write request.
- g) ADDR\_ERROR. Nodes shall respond with ADDR\_ERROR in the circumstances described below:
- 1) Fully unsupported. A request packet is addressed to a valid node, but the *offset* address references a location that is not supported by the node.
  - 2) Partially unsupported. A request packet is addressed to a valid node but a portion of the accessed data-byte locations is not supported by the node.

## 5.4 Subaction data formats

### 5.4.1 Selected-byte write16/read16 formats

Rather than providing byte-lane enables or special-length (1, 2, 4, and 8-byte data-field) packets, the **read16** and **write16** transactions transfer 16 bytes, although only selected bytes are actually accessed. Using selected-byte transfers simplifies packet processing (all data fields are a multiple of 16 bytes in length) while efficiently supporting specialized unaligned data transfers (as occur on the end of DMA data transfers).

Within a selected-byte write request, the address specifies the first data-byte address, the least-significant bits within the *tCode*-field specifies the least-significant bits of the last data-byte, and data bytes are arranged as though an aligned 16-byte block were being accessed.

For the write16 transaction, unselected values shall be zeroed by the requester and shall be ignored by the responder. For example, the first of several DMA transfers may modify the data[7]-through-data[15] bytes within an aligned 16-byte block, as illustrated in figure 4 (zeroed data bytes are shaded grey).

targetId=resId		codes	tCode	
tLabel	stamp	ph	prold	pLabel
sourceId=reqId		offsetHi		
offsetLo				
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
checksum				

targetId=reqId		codes	tCode	
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
checksum				

**Figure 4—write16 subaction formats**

Similarly, the response to an read16 transaction contains the selected data bytes within the 16 data bytes that are returned; the unselected data bytes shall be zero. For example, a 4-byte read of a control register at byte-address 24 generates the selected-byte read response-send packet illustrated in figure 5 (zeroed data bytes are shaded grey).

targetId=resId		codes	tCode	
tLabel	stamp	ph	prold	pLabel
sourceId=reqId		offsetHi		
offsetLo				
checksum				

targetId=reqId		codes	tCode	
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
checksum				

**Figure 5—read16 subaction formats**

No data bytes are accessed if the four least-significant bits of the *tCode* field are less than the four least-significant bits of the address field. For such writes, data shall not be modified; for such reads, zero data values shall be returned.

### 5.4.2 lock16 transaction formats

A small set of indivisible updates are supported, called **lock16**. Rather than locking the bus, locking memory, or locking the cache line (all of which have their disadvantages), specialized lock transactions are provided. These lock transactions allow the indivisible update to be performed at the responder, without locking other unaffected resources.

Within the lock16-quadlet transaction, two 4-byte data values (*arg0* and *arg1*) are provided. For the convenience of the responder, the arguments have fixed offset locations within the request packet. For uniformity, the returned data values in the response (that reflect the contents of memory before the lock operation started) are returned in their naturally-aligned positions. Thus, a lock4 transaction with address 24 returns four non-zero data bytes, as illustrated in figure 6 (zeroed data bytes are shaded grey).

lock16-quadlet: request-send					lock16-quadlet: response-send				
targetId=resId		codes		tCode	targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel	tLabel	stamp	ph	prold	pLabel
sourceId=reqId		offsetHi			sourceId=resId		status		
offsetLo					checkId				
data[0]	data[1]	data[2]	data[3]		data[0]	data[1]	data[2]	data[3]	nextId
test32					data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]		data[8]	data[9]	data[10]	data[11]	
next32					data[12]	data[13]	data[14]	data[15]	
checkSum					checkSum				

Figure 6—lock16-quadlet subaction formats, accessing address 24

Within an lock16-octlet transaction, two 8-byte data values (*arg0* and *arg1*) are provided. For the convenience of the responder, the arguments have fixed offset locations within the request packet. For uniformity, the returned data values in the response (that reflect the contents of memory before the lock operation started) are returned in their naturally-aligned positions. Thus, a lock8 transaction with address 16 returns eight non-zero data bytes, as illustrated in figure 7 (zeroed data bytes are shaded grey).

lock16-octlet: request-send					lock16-octlet: response-send				
targetId=resId		codes		tCode	targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel	tLabel	stamp	ph	prold	pLabel
sourceId=reqId		offsetHi			sourceId=resId		status		
offsetLo					checkId				
test64					data[0]	data[1]	data[2]	data[3]	nextId
next64					data[4]	data[5]	data[6]	data[7]	
					data[8]	data[9]	data[10]	data[11]	
					data[12]	data[13]	data[14]	data[15]	
checkSum					checkSum				

Figure 7—lock16-octlet subaction formats, accessing address 16

For consistency with other data-byte orderings, the first and second words of the test64 (or next64) value are the most- and least-significant halves of this value respectively.

For lock16 subcommands, the least-significant bits of the *tCode* field specify which lock operation is performed. Supported lock primitives include variations of the MASK\_SWAP (swap, with bit-mask enables), COMPARE\_SWAP (conditional swap), FETCH\_ADDLO (add, carry propagates from lower-to higher byte address), and FETCH\_ADDHI (add, carry propagates from higher-to-lower byte addresses) operations.

### 5.4.3 Selected-range transaction formats

Rather than supporting a wide range of uncached burst-transfer sizes and alignments (to transport 32-byte processor reads, 48-byte ATM-packet transfers, or initial/final components of unaligned DMA transfers), the larger (64-byte) read response and write requests allow the use of partial transfers. Partial transfers simplify packet processing while efficiently supporting specialized 16-byte-aligned transfers.

Within selected-range writeLo64 and writeHi64 requests, the byte-aligned address specifies the range of addresses that are used. For the writeLo64 request, the first portion of the 64-byte aligned data block, including data[0]-through-data[a] (where a is the six least-significant bits of the address) are written, as illustrated in the left half of figure 8.

writeLo64: request-send				writeHi64: response-send			
targetId=resId		codes	tCode	targetId=resId		codes	tCode
tLabel	stamp	ph	prold	pLabel	tLabel	stamp	ph
sourceId=reqId		offsetHi		sourceId=reqId		offsetHi	
offsetLo				offsetLo			
data[0]	data[1]	data[2]	data[3]	data[0]	data[1]	data[2]	data[3]
data[4]	data[5]	data[6]	data[7]	data[4]	data[5]	data[6]	data[7]
data[8]	data[9]	data[10]	data[11]	data[8]	data[9]	data[10]	data[11]
data[12]	data[13]	data[14]	data[15]	data[12]	data[13]	data[14]	data[15]
data[16]	data[17]	data[18]	data[19]	data[16]	data[17]	data[18]	data[19]
data[20]	data[21]	data[22]	data[23]	data[20]	data[21]	data[22]	data[23]
data[24]	data[25]	data[26]	data[27]	data[24]	data[25]	data[26]	data[27]
data[28]	data[29]	data[30]	data[31]	data[28]	data[29]	data[30]	data[31]
data[32]	data[33]	data[34]	data[35]	data[32]	data[33]	data[34]	data[35]
data[36]	data[37]	data[38]	data[39]	data[36]	data[37]	data[38]	data[39]
data[40]	data[41]	data[42]	data[43]	data[40]	data[41]	data[42]	data[43]
data[44]	data[45]	data[46]	data[47]	data[44]	data[45]	data[46]	data[47]
data[48]	data[49]	data[50]	data[51]	data[48]	data[49]	data[50]	data[51]
data[52]	data[53]	data[54]	data[55]	data[52]	data[53]	data[54]	data[55]
data[56]	data[57]	data[58]	data[59]	data[56]	data[57]	data[58]	data[59]
data[60]	data[61]	data[62]	data[63]	data[60]	data[61]	data[62]	data[63]
checksum				checksum			

Figure 8—writeLo64/writeHi64 request subaction formats

For the writeHi64 request, the final portion of the data block is written, including data[a]-through-data[63] (where a is the six least-significant bits of the address) are written, as illustrated in the right half of figure 8.

For example, a 64-byte unaligned write to addresses 49 would generate two transactions: the writeHi64 transaction illustrated in the right half of figure 8 and the writeLo64 transaction illustrated in the left of figure 8.

In a similar fashion, a 64-byte unaligned read of addresses 49 would generate two transactions; the data is returned in the readHi64 and readLo64 transaction responses illustrated in the right and left halves of figure 9 respectively.

targetId=reqId				codes		tCode	
tLabel	stamp	ph	prold	pLabel			
sourceId=resId				status			
checkId				nextId			
data[0]	data[1]	data[2]	data[3]				
data[4]	data[5]	data[6]	data[7]				
data[8]	data[9]	data[10]	data[11]				
data[12]	data[13]	data[14]	data[15]				
data[16]	data[17]	data[18]	data[19]				
data[20]	data[21]	data[22]	data[23]				
data[24]	data[25]	data[26]	data[27]				
data[28]	data[29]	data[30]	data[31]				
data[32]	data[33]	data[34]	data[35]				
data[36]	data[37]	data[38]	data[39]				
data[40]	data[41]	data[42]	data[43]				
data[44]	data[45]	data[46]	data[47]				
data[48]	data[49]	data[50]	data[51]				
data[52]	data[53]	data[54]	data[55]				
data[56]	data[57]	data[58]	data[59]				
data[60]	data[61]	data[62]	data[63]				
checkSum							

targetId=reqId				codes		tCode	
tLabel	stamp	ph	prold	pLabel			
sourceId=resId				status			
checkId				nextId			
data[0]	data[1]	data[2]	data[3]				
data[4]	data[5]	data[6]	data[7]				
data[8]	data[9]	data[10]	data[11]				
data[12]	data[13]	data[14]	data[15]				
data[16]	data[17]	data[18]	data[19]				
data[20]	data[21]	data[22]	data[23]				
data[24]	data[25]	data[26]	data[27]				
data[28]	data[29]	data[30]	data[31]				
data[32]	data[33]	data[34]	data[35]				
data[36]	data[37]	data[38]	data[39]				
data[40]	data[41]	data[42]	data[43]				
data[44]	data[45]	data[46]	data[47]				
data[48]	data[49]	data[50]	data[51]				
data[52]	data[53]	data[54]	data[55]				
data[56]	data[57]	data[58]	data[59]				
data[60]	data[61]	data[62]	data[63]				
checkSum							

**Figure 9—readLo64/readHi64 subaction formats**

The read256, write256, move256, and talk256 transactions allow the specification of the starting/ending addresses in an analogous fashion, with the block length being 256 bytes (rather than 64 bytes).

## 5.5 Critical-word ordering

The addressing within read64 (and reserved cRead64) transactions supports critical-word-first ordering within response packets. When the data is returned, the info field within the response packet provides the *off* (quadlet within the cache line) address for the first of the returned data bytes; the remaining data bytes shall be returned in a sequentially-increasing (modulo the line size) order.

For read64 commands, this allows the responder to select a convenient word size for the critical-word-first ordering, where the requested address is contained within the first word that is returned. For simplicity, only the 8-byte, 16-byte, 32-byte, and 64-byte word sizes are supported. For the write64 and cWrite64 commands, the least-significant bits of the address have no effect on the order in which the data is provided.

### 5.5.1 Critical-quadlet ordering

Assume that a read64 request contains address 30 (and the cache-line size is 64). If the responder assumes a word size of 4 bytes, the 4-byte-aligned block containing the addressed location comes back first, as illustrated in the left half of figure 10. If the responder assumes a word size of 8 bytes, the 8-byte-aligned block containing the addressed location comes back first, as illustrated in the right half of figure 10.

For critical-word transfers, the word size is determined by the responder, not the requester. The intent is to simplify the design of memory controllers and bridges without compromising system performance. As examples, a burst-mode DRAM array may only be able to transfer data in aligned 8-blocks. Forcing the data to be returned to 32-bit processors in their preferred order would not improve the critical-word timing in the response, but would complicate the memory controller (which swaps every pair of 32-bit words) and unnecessarily delay the access times to the remaining words.

Depending on the burst-size and data-path widths, a high-end memory controller may assume a 'word' size of 16 bytes; the 16-byte-aligned block containing the addressed location comes back first, as illustrated in the left half of figure 10. Similarly a bridge to ARBus, with an assumed cache-line size of 32 bytes, would assume a word size of 32 bytes, allowing the 32-byte-aligned block containing the addressed location to come back first, as illustrated in the right half of figure 11.

targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
data[60]	data[61]	data[62]	data[63]	
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
data[16]	data[17]	data[18]	data[19]	
data[20]	data[21]	data[22]	data[23]	
data[24]	data[25]	data[26]	data[27]	
data[28]	data[29]	data[30]	data[31]	
data[32]	data[33]	data[34]	data[35]	
data[36]	data[37]	data[38]	data[39]	
data[40]	data[41]	data[42]	data[43]	
data[44]	data[45]	data[46]	data[47]	
data[48]	data[49]	data[50]	data[51]	
data[52]	data[53]	data[54]	data[55]	
data[56]	data[57]	data[58]	data[59]	
checkSum				

read64 response, 4-byte alignment

targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
data[56]	data[57]	data[58]	data[59]	
data[60]	data[61]	data[62]	data[63]	
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
data[16]	data[17]	data[18]	data[19]	
data[20]	data[21]	data[22]	data[23]	
data[24]	data[25]	data[26]	data[27]	
data[28]	data[29]	data[30]	data[31]	
data[32]	data[33]	data[34]	data[35]	
data[36]	data[37]	data[38]	data[39]	
data[40]	data[41]	data[42]	data[43]	
data[44]	data[45]	data[46]	data[47]	
data[48]	data[49]	data[50]	data[51]	
data[52]	data[53]	data[54]	data[55]	
checkSum				

read64 response, 8-byte alignment

Figure 10—read64 response formats, 4/8-byte alignment (accessing address 60)

targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
data[48]	data[49]	data[50]	data[51]	
data[52]	data[53]	data[54]	data[55]	
data[56]	data[57]	data[58]	data[59]	
data[60]	data[61]	data[62]	data[63]	
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
data[16]	data[17]	data[18]	data[19]	
data[20]	data[21]	data[22]	data[23]	
data[24]	data[25]	data[26]	data[27]	
data[28]	data[29]	data[30]	data[31]	
data[32]	data[33]	data[34]	data[35]	
data[36]	data[37]	data[38]	data[39]	
data[40]	data[41]	data[42]	data[43]	
data[44]	data[45]	data[46]	data[47]	
checkSum				

read64 response, 16-byte alignment

targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel
sourceId=resId		status		
checkId		nextId		
data[32]	data[33]	data[34]	data[35]	
data[36]	data[37]	data[38]	data[39]	
data[40]	data[41]	data[42]	data[43]	
data[44]	data[45]	data[46]	data[47]	
data[48]	data[49]	data[50]	data[51]	
data[52]	data[53]	data[54]	data[55]	
data[56]	data[57]	data[58]	data[59]	
data[60]	data[61]	data[62]	data[63]	
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
data[16]	data[17]	data[18]	data[19]	
data[20]	data[21]	data[22]	data[23]	
data[24]	data[25]	data[26]	data[27]	
data[28]	data[29]	data[30]	data[31]	
checkSum				

read64 response, 32-byte alignment

Figure 11—read64 response formats, 16/32-byte alignment (accessing address 60)

Bridges to other strictly-sequential interconnects, such as Serrial Bus or PCI, are not expected to rearrange data returned from the remote bus. Nodes on these remote buses return response-subaction data in a lowest-through-highest byte-address order. In the absence of bridge reordering, this lowest-through-highest byte-address order is reflected in the response returned by the bridge, as illustrated in figure 12.

targetId=reqId		codes		tCode
tLabel	stamp	ph	prold	pLabel
sourceId=resId			status	
checkId		nextId		
data[0]	data[1]	data[2]	data[3]	
data[4]	data[5]	data[6]	data[7]	
data[8]	data[9]	data[10]	data[11]	
data[12]	data[13]	data[14]	data[15]	
data[16]	data[17]	data[18]	data[19]	
data[20]	data[21]	data[22]	data[23]	
data[24]	data[25]	data[26]	data[27]	
data[28]	data[29]	data[30]	data[31]	
data[32]	data[33]	data[34]	data[35]	
data[36]	data[37]	data[38]	data[39]	
data[40]	data[41]	data[42]	data[43]	
data[44]	data[45]	data[46]	data[47]	
data[48]	data[49]	data[50]	data[51]	
data[52]	data[53]	data[54]	data[55]	
data[56]	data[57]	data[58]	data[59]	
data[60]	data[61]	data[62]	data[63]	
checkSum				

**read64 response, 64-byte alignment**

**Figure 12—read64 response format, 64-byte alignment (accessing address 60)**

## 5.6 move and talk packet formats

### 5.6.1 move packet formats

The move16, moveLo64, and moveHi64 packets (directed-asynchronous responseless writes) have the format of other request-send packets; see 5.2.1 for details.

### 5.6.2 mMove packet formats

The format of mMove16, moveLo64, and moveHi64 packets (multicast-asynchronous responseless writes) is illustrated in figure 13.

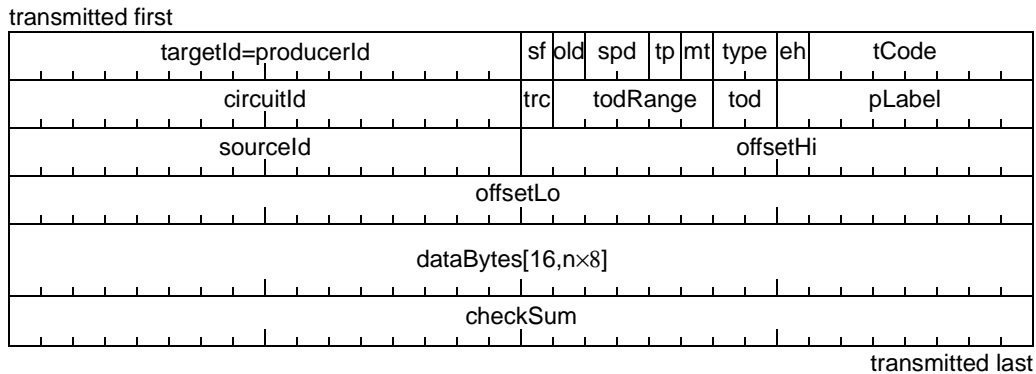


Figure 13—mMove packet formats

The 16-bit **targetId** field contains the ringlet-local hardId of the producer, allowing the packet to be stripped when it returns to the producer. This also provides the targetId address for use within the acknowledge that is returned by a responder with insufficient request-queue space.

The **sf** bit, **old** bit, 2-bit **spd** field, **tp** bit, 2-bit **type** field, and the **eh** bit, are defined in 5.2.1; the **mt** bit shall be one. Distinct 7-bit **tCode** (transaction command code) values identify mMove transactions, as defined in 5.6.5.

The 16-bit **circuitId** field specifies a multicast-circuit address. Although multicast transactions are broadcast to all, the **circuitId** field allows packets to be selectively discarded by uninvolved nodes.

The **trc** bit, the 5-bit **todRange** field, and the 2-bit **tod** field contain diagnostic tracing and time-of-death information (see xx).

The 8-bit **pLabel** field provides a ringlet-local send-packet identifier, as defined in 5.2.1.

The 16-bit **sourceId** field contains the nodeId where the request packet was created. This field is provided for packet routing as well as verification/protection purposes.

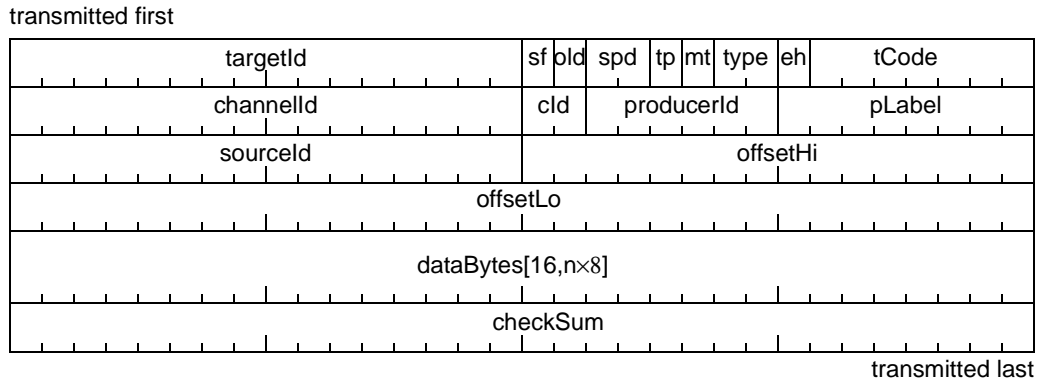
The concatenation of the 16-bit **offsetHi** and 32-bit **offsetLo** fields provides address-offset information to the responder (see xx).

The **dataBytes** field provides data to the responder; it shall be 0, 16, 64, or 256 bytes in size.

The 32-bit **checkSum** field is defined in 5.2.1.

### 5.6.3 talk packet formats

Directed packet-stream packets are called talk packets. The basic talk-packet formats are illustrated in figure 14.



**Figure 14—Basic talk-packet formats**

The 16-bit *targetId* field, *sf* bit, *old* bit, 2-bit *spd* field, *tp* bit, 2-bit *type* field, and the *eh* bit are defined in 5.2.1; the *mt* bit shall be 1. Distinct 7-bit *tCode* (transaction command code) values identify talk actions, as defined in 5.6.5.

The 16-bit *channelId* field is a multicast virtual-circuit address. This field has no effect on the packet routing, but may be used within the responder to route the data to the selected stream processor.

The 2-bit *clD* field (cycle identifier) provides the talker’s least-significant bits of the isochronous-cycle counter. The 6-bit *producerId* field provides the 6 least-significant bits of the (ringlet-local) talker’s hardId identifier. The 8-bit *pLabel* field is a ringlet-local send-packet identifier that is included in any acknowledge packets that are generated; its detailed meaning is producer-technology dependent.

The 16-bit *sourceId* field contains the nodeId where the talk packet was created and may be used for security-check purposes.

The concatenation of the 16-bit *offsetHi* and 32-bit *offsetLo* fields provides address-offset information to the responder (see xx). A small portion of the *offset* locations is standardized, for node identification and configuration purposes; the contents of the remaining locations are beyond the scope of this standard.

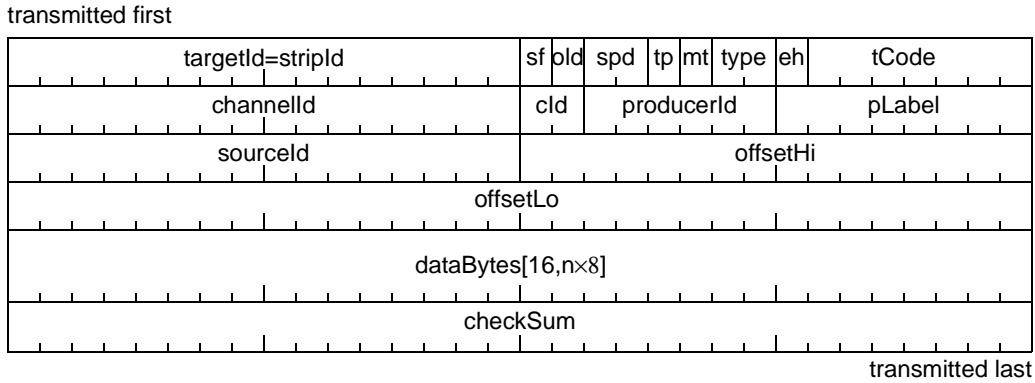
The *dataBytes* field provides data to the listener; by default, it shall be 16 or 64 bytes in size, for the talk16 and talk64 actions respectively.

The 32-bit *checksum* field is defined in 5.2.1.

### 5.6.4 mTalk packet formats

Multicast circuit-stream packets are called mTalk packets. The mTalk-packet formats are illustrated in figure 15.

The 16-bit *targetId* field provides the address of the ringlet-local talker, allowing it to strip the packet after it has circulated once around the ringlet. The *sf* bit, *old* bit, 2-bit *spd* field, *tp* bit, 2-bit *type* field, and the *eh* bit are defined in 5.2.1; the *mt* bit shall be one. Distinct 7-bit *tCode* (transaction command code) values identify talk actions, as defined in 5.6.5.



**Figure 15—mTalk-packet format**

The 16-bit *channelId* field is a multicast virtual-circuit address. This field has no affect on the packet routing, but may be used within the responder to route the data to the selected stream processor.

The 2-bit *cId* (cycle identifier) and the 6-bit *producerId* field are the same for multicast-stream and directed-stream packets, as defined in 5.6.3. The 8-bit *pLabel* field uniquely labels a producer’s outstanding transactions, as defined in 5.2.1.

The other *dataBytes* fields contain responder/circuitId-dependent data. For the mTalk16 and mTalkHi64 variants of *tCode* values, the full-packet size (64 bytes) shall be transferred. For the mTalkLo64 variant of *tCode* values, fewer data bytes may be transferred. However, the number of data bytes within partial packets (as well as filled packets) shall always be a multiple of 8 bytes.

The 32-bit *checkSum* field is defined in 5.2.1.

### 5.6.5 Responseless-write *tCode* values

The responseless write actions (move, mMove, talk, and mTalk) have distinct *tCode* values, as specified in table 21.

*Constrained: Within some (register-only) spaces, only 4-byte transfers to aligned addresses are supported. If the transfer size or address alignment are unsupported, the packet shall be discarded and an advisory error should be logged.*

The **move16** request initiates a 16-byte directed-asynchronous responseless write. Although the request always contains 16 bytes of data, a 0-through-16 byte address range selects which bytes shall be written (other data values shall be ignored, see 5.4.1).

*Optional: Not expected to be supported within simple (register-only) address spaces. If this tCode is not supported, the packet shall be discarded and an advisory error should be logged.*

The **moveLo64** and **moveHi64** requests initiate 64-byte directed-asynchronous responseless writes. In the absence of an error, the selected bytes within the request’s 64-byte block shall be written into the specified acceptor locations; a response shall not be returned. The data transfer ends-with or starts-with the byte-aligned address, for the moveLo64 and moveHi64 requests respectively (see 5.4.3).

*Optional: Only expected to be supported on the highest-performance asynchronous nodes. If this tCode is not supported, the packet shall be discarded and an advisory error should be logged.*

The **moveLo256** and **moveHi256** requests initiate 256-byte directed-asynchronous responseless writes. In the absence of an error, the selected bytes within the request’s 256-byte block shall be written into the

**Table 21— *tCode* assignments, responses writes, *mt=1***

<i>tCode</i>		packet payload	Name	Description
msbs	lsbs			
000	eeee	16	move16	directed asynchronous move, 0-16 bytes
001	eeee	16	mMove16	multicast asynchronous move, 0-16 bytes
010	eeee	16	talk16	directed isochronous talk, 0-16 bytes
011	eeee	16	mTalk16	multicast isochronous talk, 0-16 bytes
100-101	—	16	—	reserved
110	0000	64	moveLo64	directed asynchronous move low-range write, 1-64 bytes
	0001	"	moveHi64	directed asynchronous move high-range write, 1-64 bytes
	001x	"	—	directed asynchronous move, reserved acceptor actions
	0000	64	mMoveLo64	multicast asynchronous move low-range write, 1-64 bytes
	0001	"	mMoveHi64	multicast asynchronous move high-range write, 1-64 bytes
	001x	"	—	multicast asynchronous move, reserved acceptor actions
	0000	64	talkLo64	directed isochronous move low-range write, 1-64 bytes
	0001	"	talkHi64	directed isochronous move high-range write, 1-64 bytes
	001x	"	—	directed isochronous move, reserved listener actions
	0000	64	mTalkLo64	multicast isochronous move low-range write, 1-64 bytes
	0001	"	mTalkHi64	multicast isochronous move high-range write, 1-64 bytes
	001x	"	—	multicast isochronous move, reserved listener actions
111	0000	256	moveLo256	directed asynchronous move low-range write, 1-256 bytes
	0001	"	moveHi256	directed asynchronous move high-range write, 1-256 bytes
	001x	"	—	directed asynchronous move, reserved acceptor actions
	0000	256	mMoveLo256	multicast asynchronous move low-range write, 1-256 bytes
	0001	"	mMoveHi256	multicast asynchronous move high-range write, 1-256 bytes
	001x	"	—	multicast asynchronous move, reserved acceptor actions
	0000	256	talkLo256	directed isochronous move low-range write, 1-256 bytes
	0001	"	talkHi256	directed isochronous move high-range write, 1-256 bytes
	001x	"	—	directed isochronous move, reserved listener actions
	0000	256	mTalkLo256	multicast isochronous move low-range write, 1-256 bytes
	0001	"	mTalkHi256	multicast isochronous move high-range write, 1-256 bytes
	001x	"	—	multicast isochronous move, reserved listener actions

specified acceptor locations; a response shall not be returned. The data transfer ends-with or starts-with the byte-aligned address, for the moveLo256 and moveHi256 requests respectively (see 5.4.3).

*Optional: Shall be supported if multicast transactions are supported.*  
*If the tCode is unsupported, the packet shall be discarded and an advisory error should be logged.*  
The **mMove16** request initiate a 16-byte multicast-asynchronous responseless write. Although the request always contains 16 bytes of data, a 0-through-16 byte address range selects which bytes shall be written (other data values shall be ignored, see 5.4.1).

*Optional: Should be supported if multicast transactions are supported.*  
*If the tCode is unsupported, the packet shall be discarded and an advisory error should be logged.*  
The **mMoveLo64** and **mMoveHi64** requests initiate 64-byte multicast-asynchronous responseless writes. In the absence of an error, the selected bytes within the request's 64-byte block shall be written into the specified acceptor locations; a response shall not be returned. The data transfer ends-with or starts-with the byte-aligned address, for the mMoveLo64 and mMoveHi64 requests respectively (see 5.4.3).

*Optional: May be supported for improved data-transfer efficiency.*  
*If t unsupported, the packet shall be discarded and an advisory error should be logged at the producer.*  
The **mMove256** and **mMoveHi256** requests initiate 256-byte multicast-asynchronous responseless writes. In the absence of an error, the selected bytes within the request's 256-byte block shall be written into the specified acceptor locations; a response shall not be returned. The data transfer ends-with or starts-with the byte-aligned address, for the mMoveLo256 and mMoveHi256 requests respectively (see 5.4.3).

*Optional: Shall be supported if mTalk16 is supported.*  
*If unsupported, the packet shall be discarded and an advisory error should be logged.*  
The **talk16** request initiates a 16-byte directed-isochronous responseless write. A 16-byte block is transported; a 0-through-16 byte address range selects which bytes shall be written (other data values shall be ignored, see 5.4.1).

*Optional: Shall be supported if mTalk64 is supported.*  
*If not supported, the packet shall be discarded and an advisory error should be logged.*  
The **talkLo64** and **talkHi64** requests initiate 64-byte directed-isochronous responseless writes. A 64-byte block is transported; up-to 64 bytes of data shall be updated from the addressed 64-byte-aligned block. The data transfer ends-with or starts-with the byte-aligned address, for the talkLo64 and talkHi64 requests respectively (see 5.4.3).

*Optional: May be supported for more-efficient transfers.*  
*If not supported, the packet shall be discarded and an advisory error should be logged at the producer.*  
The **talkLo256** and **talkHi256** request initiate 256-byte directed-isochronous responseless writes. A 256-byte block is transported; up-to 256 bytes of data shall be updated from the addressed 256-byte-aligned block. The data transfer ends-with or starts-with the byte-aligned address, for the talkLo256 and talkHi256 requests respectively (see 5.4.3).

*Optional: For non-sequential delivery of data with isochronous timing constraints.*  
*If not supported, the packet shall be discarded and an advisory error should be logged at the consumer.*  
The **mTalk16** request initiates a 16-byte multicast-isochronous responseless write. A 16-byte block is transported; a 0-through-16 byte address range selects which bytes shall be written (other data values shall be ignored, see 5.4.1).

*Optional: For sequential delivery of data with isochronous timing constraints.*  
*If not supported, the packet shall be discarded and an advisory error should be logged at the consumer.*  
The **mTalkLo64** and **mTalkHi64** requests initiate 64-byte multicast-isochronous responseless writes. A 64-byte block is transported; up-to 64 bytes of data shall be updated from the addressed 64-byte-aligned block. The data transfer ends-with or starts-with the byte-aligned address, for the mTalkLo64 and mTalkHi64 requests respectively (see 5.4.3).

*Optional: For mTalk64 applications, when more efficient transfers are required.*

*If not supported, the packet shall be discarded and an advisory error should be logged at the consumer.*

The **mTalkLo256** and **mTalkHi256** requests initiate 256-byte multicast-isochronous responseless writes. A 256-byte block is transported; up-to 256 bytes of data shall be updated from the addressed 256-byte-aligned block. The data transfer ends-with or starts-with the byte-aligned address, for the mTalkLo256 and mTalkHi256 requests respectively (see 5.4.3)



## 6. TDM Operation

### 6.1 TDM Overview.

The TDM bus in the HiRelPCI system is a parallel byte wide electrical wire implementation of the OC-12 level Synchronous Optical Network (SONET) as described in ANSI Std T1.105-1991. The basic speed of the OC-12 (STS-12) is a serial bit rate of 622.08 Megabits per second. In byte wide mode the clock rate is 77.76 MegaBytes per seconds. The SONET framing allows for 9288 timeslots of 64K bit/second basic communications channels plus and overhead set of bytes.

SONET frames are transmitted 8000 times per second with each frame lasting 125us. Each frame consist of a total of 9720 bytes including 9288 in the payload. SONET links are bidirectional with incomming and outgoing data streams. The backplane uses two sets of 8 bits each to allow for full usage of the bandwidth to and from the interface to the backplane.

On each TDM connector is a dual 8 bit TDM bus and a clock and Frame line to determine position of the datat in the frame. All lines are duplicated on a second connector to provide a redundant TDM lines for critical lines.

High speed communications systems often require a synchronous operation of communications on a circuit switched basis.

### 6.2 TDM Bus

This chapter describes a base rate and format along with a multiplexing scheme that will result in a modular family of rates and formats available for use in optical interfaces generally referred to as SONET. Other characteristics described in this standard are: layering of overhead, definitions of function and position of overhead, frequency justification at terminals, scrambling, and standardized loading of different payloads into the transport frame. It does not include the specification of optical interface parameters (see *American National Standard for Telecommunications - Digital hierarchy - Optical interface specifications (single modeJ*, ANSI T1.106). It is always assumed that when the signal described in this standard is transmitted, it is directly converted to eight bit parallel, byte serial PECL signaling.

### 6.3 Normative references

All standards and publications are subject to revision, and parties to agreements based on this American National Standard are encouraged to investigate the possibility of applying the most recent editions of the standards and publications listed below.

ANSI T1.101 - 1987, *Telecommunications -Synchronization interface standards for digital networks*  
CCITT Recommendation G.707, *Synchronous digital hierarchy bit rates*  
CCITT Recommendation G.708, *Network node interface for the synchronous digital hierarchy*  
CCITT Recommendation G.709, *Synchronous multiplexing structure*

### 6.4 General

A primary goal in creating this standard for the rates and formats for backplane GTL+ interfaces is to define a synchronous digital hierarchy with sufficient flexibility to carry many different capacity signals. This has been accomplished by defining a basic signal of 51.840 Mbit/s, 6.48 Mbytes/s, and a byte interleaved multiplex scheme that results in a family of standard rates and formats defined at a rate of  $N$  times 51.840Mbit/s, where  $N$  is an integer. Presently, the maximum value of  $N$  is 255 for normal SONET specification, but is

limited to  $N = 12$  in this standard due to signaling methods used.

## 6.5 Rates

### 6.5.1 STS-1/OC-1 rate

The basic modular signal is called the Synchronous Transport Signal level 1 (STS-1). The rate is 51.840 Mbit/s. The optical counterpart of the STS-1 is the Optical Carrier level 1 signal (OC-1), which is the result of a direct optical conversion of the STS-1 after frame synchronous scrambling. Within this backplane implementation of the synchronous scrambling is not implemented as each byte may be individually driven.

Many different elements of overhead are defined in the SONET documents, including overhead for maintenance, user channels, frequency justification, orderwire, channel identification, and growth channels. A layered approach to overhead has been established whereby overhead bandwidth has been allocated to a layer based on the function addressed by that particular channel.

### 6.5.2 Synchronous hierarchical rates

The definition of the first level (STS-1, OC-1) defines the entire hierarchy of synchronous signaling, since the higher-level signals are obtained by synchronously multiplexing lower-level signals. The higher-level signals are denoted by STS-N and OC-N where N is an integer. For this standard implementation N is 12.

**Table 22—STS and OC data rates**

OC/STS Level	Line Rate Mbits/sec	Bus Byte/Rate- Mbytes/sec
1	51.840	6.48
3	155.200	19.44
9	466.560	58.32
<b>12</b>	<b>622.080</b>	<b>77.76</b>
18	933.120	116.64
24	1244.160	155.52
36	1866.240	233.28
48	2488.320	311.04
96	4976.640	622.08
192	9953.280	1244.16
256	13271.040	1658.88

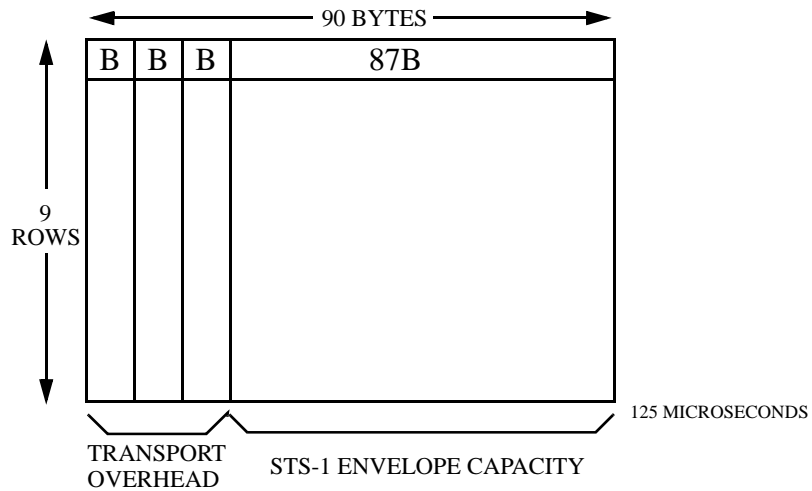
This table lists standard Optical Carrier (OC) rates from OC1 51.84 Mbit/s through OC256 13271.04 Mbit/s.

### 6.5.3 Frame structure of the STS-1

The STS-1 frame depicted in figure ? consists of 90 columns and 9 rows of 8-bit bytes, for a total of 810 bytes (6480 bits). With a frame length of 125 microseconds (i.e., 8000 frames per second), the STS-1 has a bit rate of 51.840 Mbit/s or 6.48 MBytes/s. The order of transmission of bytes is row by row, from left to right. Diagrams of the STS frame structure are included for information purposes, please reference the ANSI

-101 standard for difinitive documentation.

### STS-1 frame

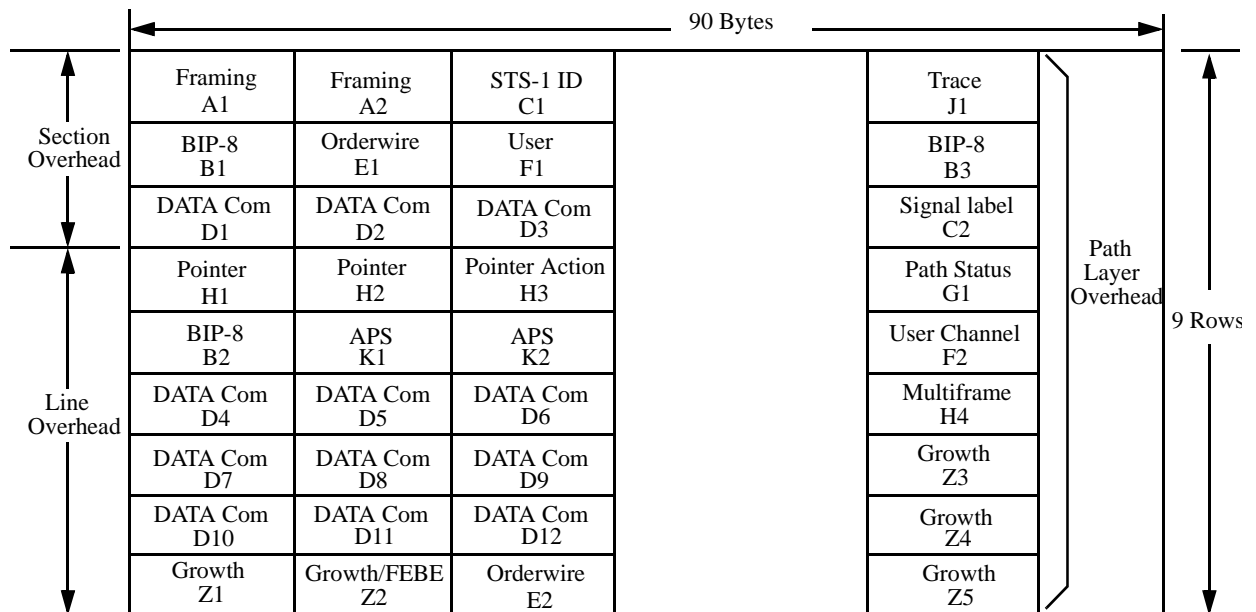


B denotes an 8-bit byte.

#### 6.5.4 Transport overhead

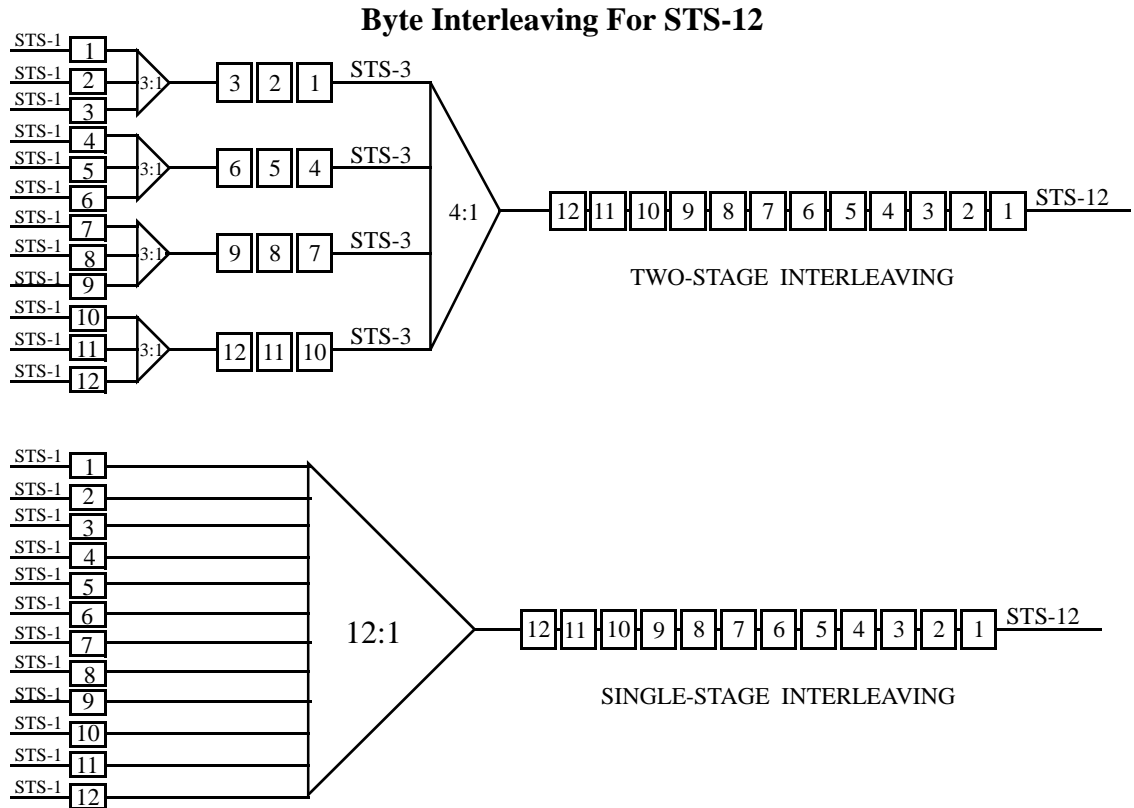
Referring to Following figure, the first three columns are the Transport Overhead, which contains overhead bytes for Section and Line layers. Twenty-seven bytes have been assigned, with nine bytes for Section Overhead and eighteen bytes for Line Overhead. Details of these overhead allocations are described in ANSI T1-101 as revised.

**Figure aaa**

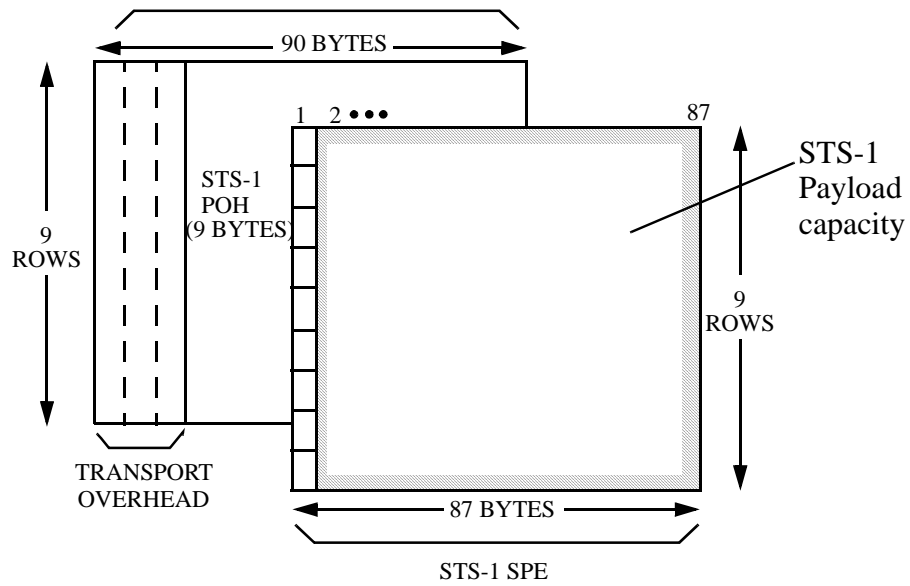


## 6.6 Frame structure of the STS-12

The STS-12 signal is formed by byte interleaving 12 STS-1 signals. The STS-12 frame structure is depicted in figure ?. The transport overhead channels of the individual STS-1 signals must be frame aligned before interleaving. The associated STS SPEs are not required to be aligned because each STS-1 will have a unique payload pointer to indicate the location of the SPE. This standard supports STS - 3C and STS-12C or STM1 and STM4.



### STS-1 synchronous payload envelope with STS-1 POH and STS-1 payload capacity illustrated





## 7. Maintenance Bus

For a complete description of the IEEE Std 1149.5-1995 MTM bus, please see the latest version of the standard available from IEEE.

### 7.1 Objectives

The MTM-Bus is intended for use in the testing, diagnosis, and maintenance of electronics subsystems and modules. Every module within the P1996 system has dual access with two MTM-Busses for redundancy, each uses a separate address to prevent conflict within a module. The MTM-Bus may be used to support the following:

- *Module test.* Testing of modules during manufacturing to provide fault isolation of defects to individual components.

- *Subsystem test.* Off-line testing of modules while contained within a subsystem, and the testing of interconnections between modules within a subsystem.

- *Subsystem diagnostics.* On-line diagnostics within a system to support logging of detected errors, initiation of self-test, reconfiguration of subsystem resources, and other diagnostic functions.

- *Software/hardware development.* Access to a module or subsystem state by use of low-level observability/controllability techniques (e.g., scan, boundary scan, direct control, etc.). Use of these techniques may allow for reduced hardware/software development costs and decreased time to market.

- *Alternate connectivity.* If main bus failure is detected, communications to a module is still available through the MTM bus. Critical Data can be transferred through the maintenance processor to or from the module.

- *Connectivity across chassis boundaries.* As the maintenance bus MTM can be extended through the maintenance processor Ethernet connectivity, MTM connectivity is extended to the 32K modules.

To support these applications, the MTM-Bus is used as a serial backplane bus with a multidrop topology. As the multidrop bus signals are common between all modules, a board may be removed from the backplane without breaking the communication link between other modules in the backplane. With the appropriate Physical Layer design, the protocol supports access between a single MTM-Bus Master module and up to 250 individually addressable MTM-Bus Slave modules. Each of the MTM-Busses has a default Master in one of the Central Service Modules but will fault over to the other CSM if the primary CSM for the assigned bus fails. Addressing is defined so that the MTM-Bus Master may communicate with one, some, or all of the Slaves at one time.

Because the MTM-Bus may be used to transmit a significant amount of data, such as serial test patterns, the protocol supports full duplex data transfer operations. Additional fault tolerance is available through redundant MTM-Bus.

To support the requirements described previously, the protocol identifies commands to provide for the following functions:

- Initialization of module/subsystem
- Accessing lower-level test buses
- Testing of module interconnections

- Interrupt control and handling
- Accessing module identification
- Accessing module fault logs
- Forcing modules off- and on-line
- Accessing module built-in test features
- Accessing P1996 additional control features
- Private and public user extensions

## 7.2 The interconnection of modules with MTM-Bus

As designs have become more complex and difficult to test, design-for-test features (e.g., internal scan, IEEE Std 1149.1 boundary-scan, and built-in test) are being integrated into component designs to aid in component, board, subsystem, and system test and maintenance. Since these techniques do not use "functional" methods to test the circuitry, alternate paths are needed to access on-chip design-for-test features. These alternate paths have to be kept both simple and dedicated to test and diagnostic functions so that bootstrapping to complete module testing from a testable kernel is feasible.

When components are assembled on boards, boards into subsystems, and subsystems into systems, a hierarchy of test buses is needed to retain access to the design-for-test features in the final product. Such a hierarchy of test buses is depicted in figure 1-1. The MTM-Bus has been developed to meet the requirements for a backplane bus in such a bus hierarchy. As shown in figure 1-1, the MTM-Bus provides subsystem test control (e.g., maintenance module, service console, etc.) access or external test equipment access to test features on modules within a subsystem.

The use of the MTM-Bus is a hierarchy extending from the system area network through the Central Service Modules to the Upper or Lower busses to each module within the system. Each module contain a maintenance processor of module to monitor the health of the module and control the module as needed.

## 7.3 Benefits of distributed approach:

*-Interoperability.* In a system where modules are provided by a number of different design teams, a distributed approach allows backplane integration to proceed more smoothly.

*-Reduction in test time.* It is possible that test time at the system level can be reduced if two or more modules can be tested concurrently.

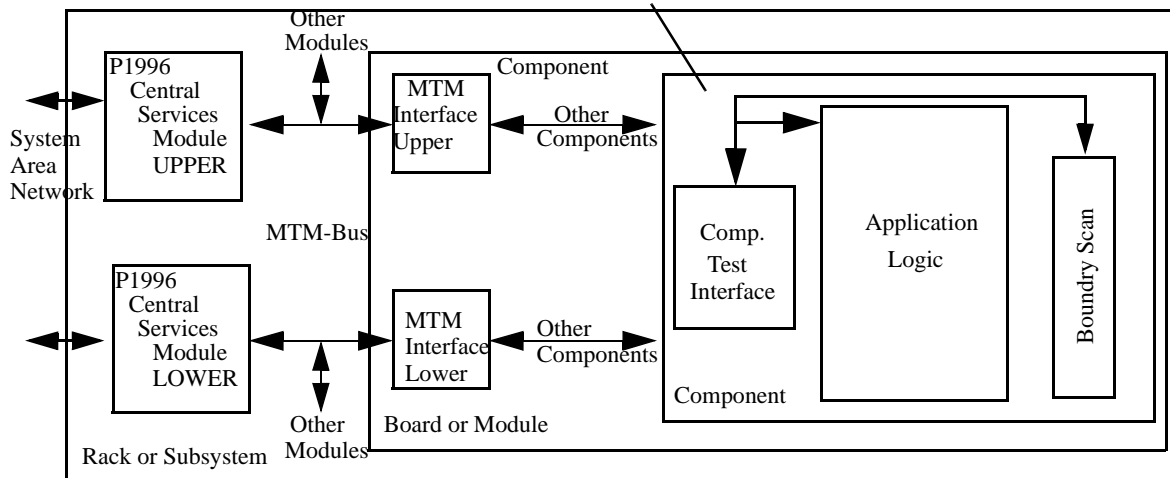


Figure 16—Hierarchy of test and maintenance buses

#### 7.4 HiRelPCI P1996 Additional Function Codes

The following function codes are added to the normal IEEE Std 1149 1995 to support the additional requirements of the HiRelPCI bus operation. These codes are drawn from the group of codes reserved in the original MTM Bus document for use by other Standards Bodies. These codes consist of Hexidecimal Codes 50 through 5F.

##### 7.4.1 Command Code 50h - Disconnect module HiRelPCI from Backplane Bus

This command controls the disconnection of the module from the PCI bus structure. All Modules must contain a method of disconnection from the PCI bus by switch or other means that will not load the PCI bus and induce a failure. When this command is issued the module is immediately disconnected from the PCI bus.

##### 7.4.2 Command Code 51h - Connect module HiRelPCI to Backplane Bus

This command controls the connection of the module to the PCI bus structure. All Modules must contain a method of disconnection from the PCI bus by switch or other means that will not load the PCI bus and induce a failure. When this command is issued the module is connected through the switches or other connection method to the PCI bus.

##### 7.4.3 Command Code 52h - Disable module on board main power converters

This command removes the enable for the main power converters for the module. The only remaining power on the board is the power to the maintenance processor to communicate to the MTM bus.

##### 7.4.4 Command Code 53h - Enable module on board main power converters

This command from the CSU commands the module to turn on the main power converters. This will allow the testing of the functionality of the board.

#### **7.4.5 Command Code 54h - Read Module Serial Number and NodeID**

This command reads the Eprom or Serial memory module to get the serial number of the Extended Unique Identifier and the nodeID from the Geographical address pins. This information is sent to the System master. This function shall not require the activation of main board power.

#### **7.4.6 Command Code 55h - Read PCI Configuration Space**

This command maps the PCI configuration registers and transfers the data to the maintenance bus master. This operation can only be performed after main power has been enabled and verified through the Maintenance processor. This function is performed through the Maintenance processor test mode to the connection through the switches used to isolate and test the board. This is a read function.

#### **7.4.7 Command Code 56h - Write PCI Configuration Space**

This command maps the PCI configuration registers and transfers the data from the maintenance bus master. This operation can only be performed after main power has been enabled and verified through the Maintenance processor. This function is performed through the Maintenance processor test mode to the connection through the switches used to isolate and test the board. This is a write function that allows full configuration of the PCI master prior to connection to a already working PCI bus.

#### **7.4.8 Command Code 57h - Read Power Status Block**

Transferred with this command to the Maintenance bus master and CSU is the input information on the voltages of all available sources, the current drawn from each source and the temperature of various parts of the board. This is a safety systems check.

#### **7.4.9 Command Code 58h - Address Port and Mode (64 bit mode)**

This address port is used to set the address for a data operation. This port is always in autoincrement mode such that following any operation the address is incremented by eight bytes (one octlet) and is ready for the next operation.

#### **7.4.10 Command Code 59h - Data Port (64 bit mode)**

This is a Data command transfers an octlet of data starting at the address written into the Address register with the Write Address command.

#### **7.4.11 Command Code 5Ah - TBD**

To be determined a a future time

#### **7.4.12 Command Code 5Bh - TBD**

To be determined at a future time

#### **7.4.13 Command Code 5Ch - Read 1149.1 JTAG Data/Program Data**

This command will upload current JTAG data being used in the board to the host. The format of the command is to be determined.

#### 7.4.14 Command Code 5Dh - Write 1149.1 JTAG Data/Program Data

This command will download new JTAG data to be used in testing the board and possibly loading the initial boot flash memory. Format is to be determined by this committee.

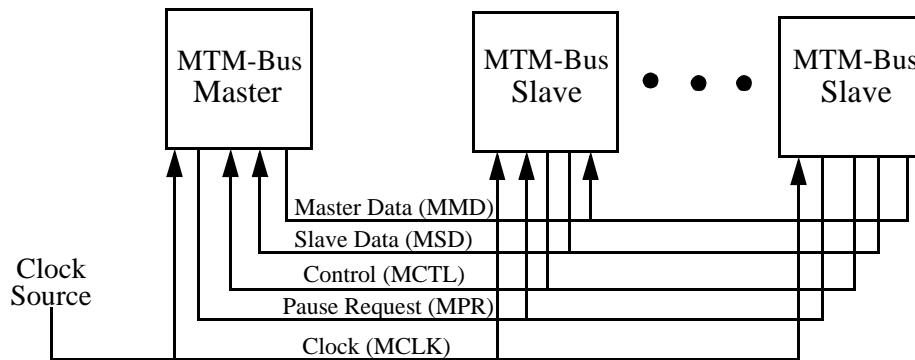
#### 7.4.15 Command Code 5Eh - Read Message ( Header + Data + CRC)

This command supports a general packet transfer to the master CSU board for use or remote addressable module.

#### 7.4.16 Command Code 5Fh - Write Message ( Header + Data + CRC )

This command supports a general packet transfer (possibly SCI format) from a master to the module.

### 7.5 HiRelPCI P1996 Module Addressing

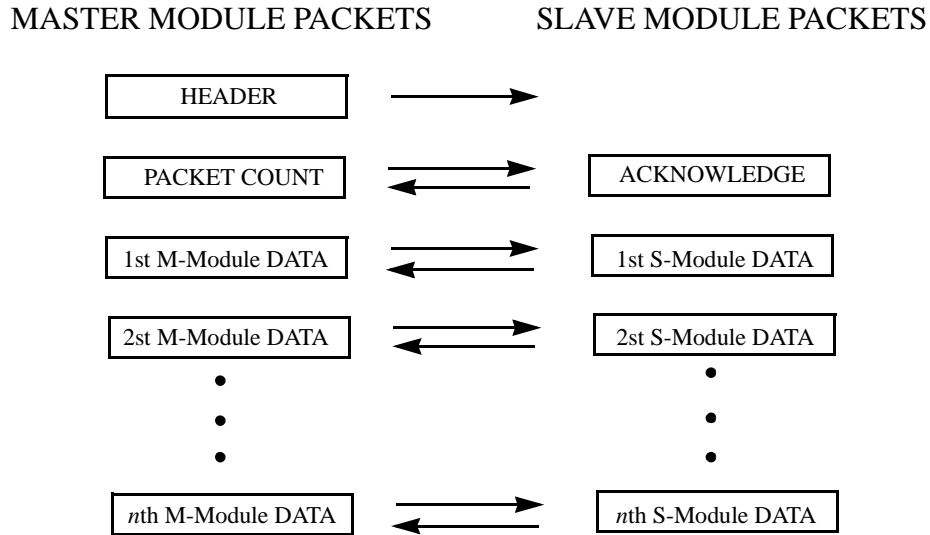


**Figure 17—Backplane MTM-Bus signals**

The MTM-Bus Master communicates with MTM-Bus Slaves using messages that consist of a series of packet transfers. A message consists of a HEADER packet, an optional ACKNOWLEDGE/PACKET COUNT packet, and a variable number of DATA packets.

Figure 1-4 depicts a standard format for an MTM-Bus message. To start a message, the MTM-Bus Master transmits a HEADER packet that includes the addresses of the MTM-Bus Slaves who are to participate in the message sequence along with a command. If a single MTM-Bus Slave is addressed and the HEADER packet includes a request for an ACKNOWLEDGE packet, the MTM-Bus Master sends the PACKET COUNT packet while the addressed MTM-Bus Slave is returning the ACKNOWLEDGE packet. This PACKET COUNT packet identifies how many packets the MTM-Bus Master expects to transfer during the message exclusive of HEADER and PACKET COUNT packets. The message may then include the transfer of DATA packets between the MTM-Bus Master and the single addressed MTM-Bus Slave module depending upon the command. If multiple MTM-Bus Slaves are addressed (i.e., via broadcast or multicast addressing), no ACKNOWLEDGE packet is sent and any transfer of DATA packets is only from the MTM-Bus Master to the MTM-Bus Slaves. In the broadcast and multicast cases, the MTM-Bus Master receive a con-

stant logic The value of the Slave Data line (figure 1-3) when it is not actively driven by any MTM-Bus Slave. The number of DATA packets to be transferred is dependent upon the type of command contained within the HEADER packet.



**Figure 18—Sample MTM-Bus message format**

All packet transfers occur under the control of the MTM-Bus Master module. Addressed MTM-Bus Slave modules may request that the MTM-Bus Master insert additional Pause states between data packet transfers by asserting the Pause Request (MPR) signal. This feature may be used to accommodate slow MTM-Bus Slaves or to simplify data flow control across asynchronous interfaces.

MTM-Bus Slave modules may also request the attention of the MTM-Bus Master by sending an interrupt, multiplexed on the Slave Data (MSD) signal wire, at any time between packet transfers. As there is only one MSD input to the MTM-Bus Master module, the MTM-Bus Master may need to identify the interrupting module via polling or by the Contend for Bus command..

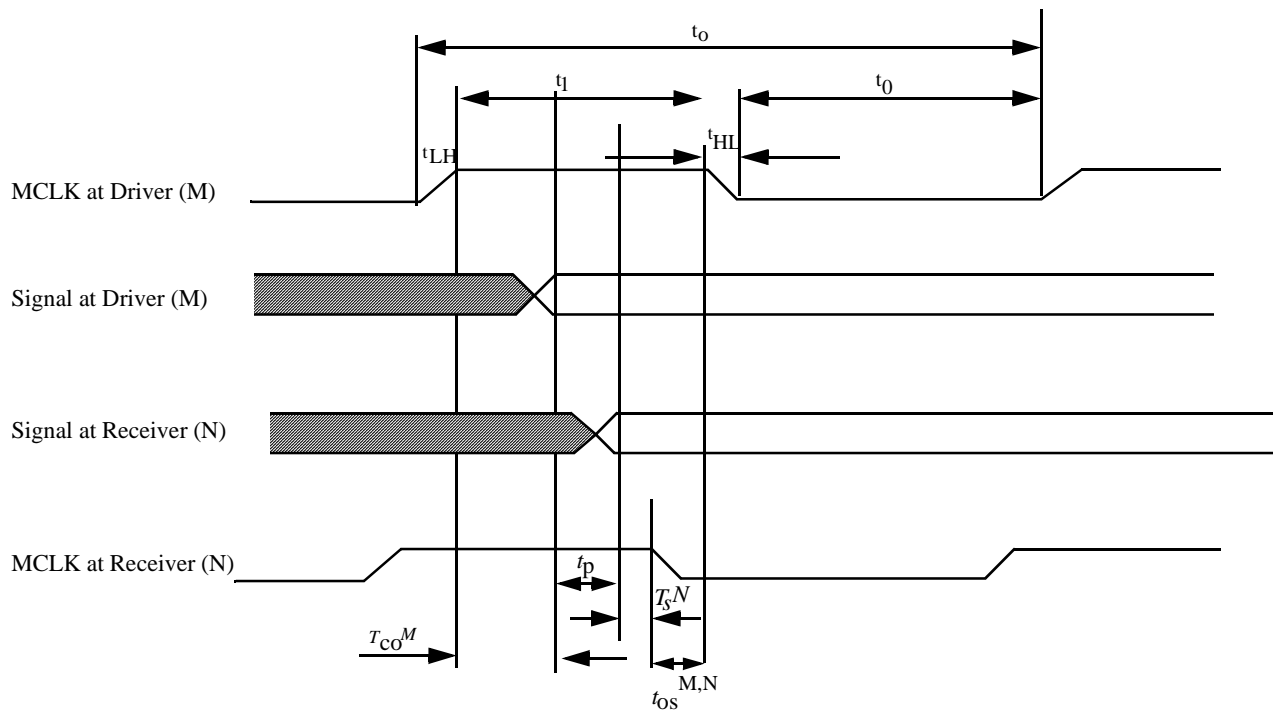
### 7.5.0.1 Description

Each S-module is to be assigned an eight-bit address, in the P1996 environment only 22 of them are uses at maximum configuration. The use of eight bits allows for a set of 256 addresses including 250 single module addresses ('0' through 'F9' HEX), one amnesia address ('FA' HEX) for use by an S-module that has lost access to its module address, one broadcast address ('FB' HEX), and four multi-cast addresses ('FC', 'FD', 'FE', and 'FF' HEX). An S-module is considered to be addressed (and thus allowed to execute the current MTM-Bus command) if it has received (in the most recently transmitted HEADER packet) its module address (or the amnesia address when appropriate), the broadcast address, or the multicast address that matches its current multicast select group. An S-module is considered uniquely addressed if it has received its module address or the amnesia address in the most recently transmitted HEADER packet. Whether an S-module is "addressed" or "uniquely addressed" influences the manner in which it responds during an MTM-Bus message.

If an S-module is unable to access its module address (e.g., due to a module address parity error, etc.), the S-module in question can respond only to commands sent to the amnesia address ('FA' HEX) or to any command sent to the broadcast address or the address of the S-module's current multicast group. The amnesia

address provides the possibility of limited system diagnosis based on data recovered from S-modules unable to access their unique module addresses. Some data can be recovered from such modules using the amnesia address because the modules will act as though uniquely addressed. Diagnosis might begin by directing the M-module to read the module status of the module with address 'FA' HEX. If there is an "all zero" response, then no S-module is responding and, hence, unless more serious failures have occurred, no S-module has self-diagnosed an address fault. If multiple S-modules respond to 'FA' HEX (i.e., become addressed by a given message) and an ACKNOWLEDGE packet is required, the collision-error handling of the S-modules can (at least sometimes) reduce the number of modules that remain addressed after ACKNOWLEDGE packet transmission requirements a transfer

A timing diagram showing the MCLK and another MTM-Bus signal waveform at both a transmitter and a receiver are illustrated in figure 4-4.



NOTE—This timing diagram shows the transitions of MCLK and another MTM-Bus signal at both the signal driver (on MTM-Bus module M) and signal receiver (on MTM-Bus module N) in the worst-case propagation situation—where the clock at the receiver leads the clock at the source. In every bidirectional data transfer, operation in one of the directions will fit the worst case just described.

## 7.6 Data transfer timing

For a given MTM-Bus implementation in a given system, the MCLK waveform will have to be adjusted so that worst-case timing requirements for the connected MTM-Bus interface devices [equation (1)], the worst-case system propagation budget [equation (2)], and the worst-case module hold time requirement [equation (3)] are not violated.

tc 2 max (TCX: X is a module or interface device connected to the bus)

to  $2 \max (TCOX + tp + TsY + tCsx$  Y: X is a signal output and Y is a connected signal input)

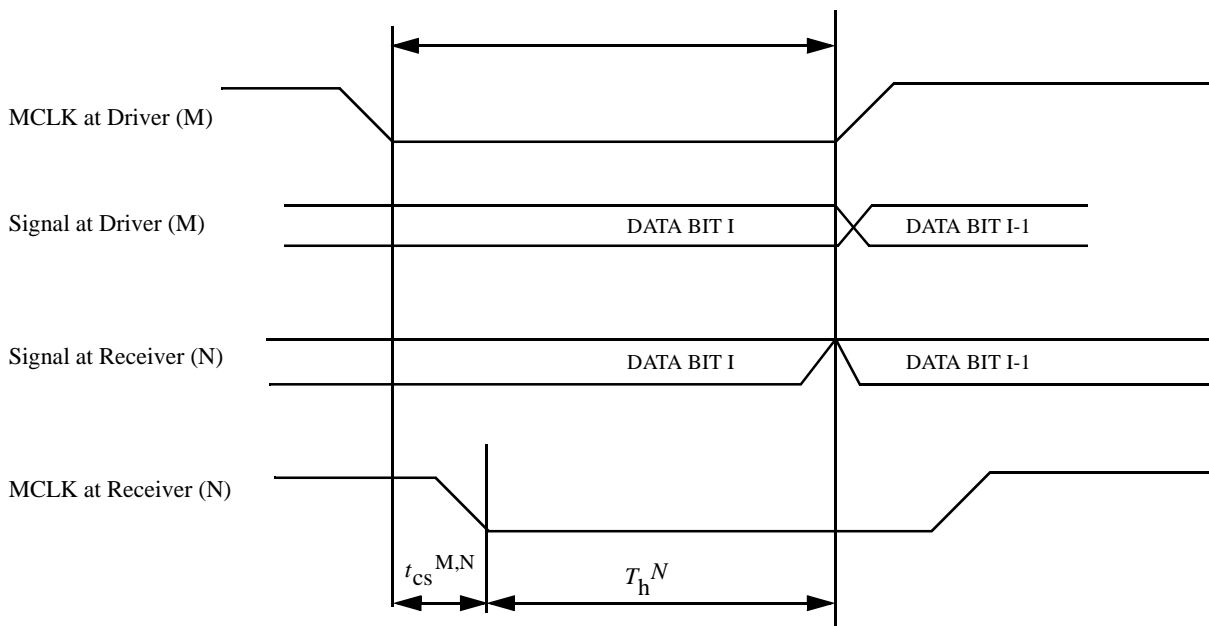
to  $2 \max (ThY + tCsXY$ : X is a signal output and Y is a connected signal input)

In addition, the system timing parameters cannot violate the worst-case module timing parameters.

$t_l$   $2 \max (TIM$ : M is a module connected to the bus) to  $2 \max (ToM$ : M is a module connected to the bus) to

$2 \max (TCM$  M is a module connected to the bus)

As long as the system MCLK waveform parameters  $t_l$ ,  $t_o$ , and  $t_c$  meet the relationships described in the present clause, system timing requirements will be met. This method for specifying the MCLK waveform requires the bus interface to operate over a range of MCLK frequencies from as low as may be desired by a user to a documented maximum.



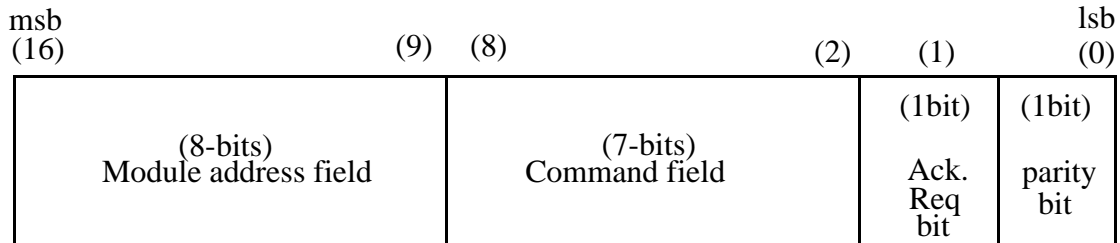
NOTE—This MTM-Bus timing diagram shows the transitions of the clock and another MTM-Bus signal at both the signal driver (M) and signal receiver (N) in the worst-case hold time situation. In this case, the clock at the receiver trails the clock at the source, propagation delay is zero, and clock to output delay for M is zero. The time to has to equal or exceed the sum of the clock skew and the hold time of M

**Figure 19—Hold time requirement**

*Performance requirements:* Although minimum performance requirements for an MTM-Bus module, M, are not specified, it is recommended that TIM and TOM be minimized to allow greater bandwidth if required by a system. An MCLK waveform has to be observed on the system backplane to ensure that module TIM and TOM requirements are met.

### 7.6.1 MTM-Bus Link Layer: Packet requirements

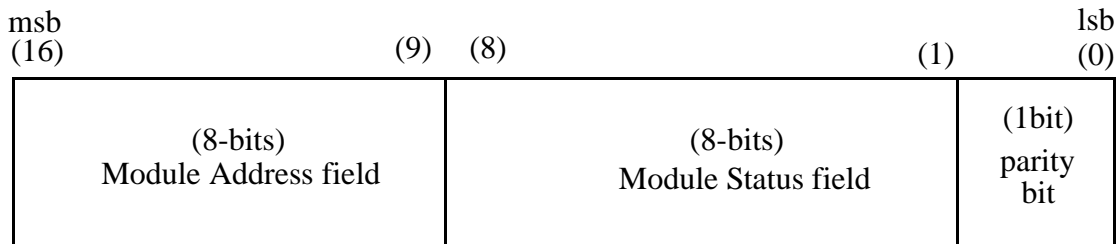
All MTM-Bus messages consist of the transfers of 17-bit packets. The packet types include HEADER packet, ACKNOWLEDGE packet, PACKET COUNT packet, and DATA packet.



**Figure 20—HEADER packet format**

Precisely one HEADER packet occurs in each MTM-Bus message and is the first packet of the message. From the information in a HEADER packet, each S-module connected to an MTM-Bus can determine if it is addressed, what command is to be executed, and whether the system application controlling the M-module is requesting an ACKNOWLEDGE packet from the S-module as part of the message transmission.

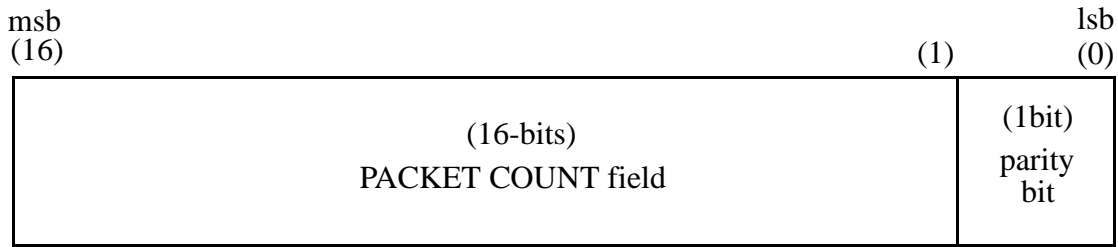
**7.6.2 ACKNOWLEDGE packet requirements**



**Figure 21—ACKNOWLEDGE packet format**

An ACKNOWLEDGE packet is transmitted by an S-module upon request and is both a convenient way to recover internal status information from an S-module and is designed to support the Contend for Bus sequence.

### 7.6.3 Packet Count packet requirements

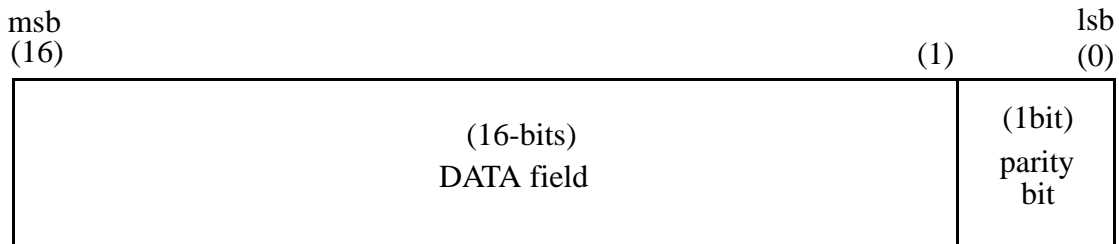


**Figure 22—PACKET COUNT packet format**

#### 7.6.3.1 Description

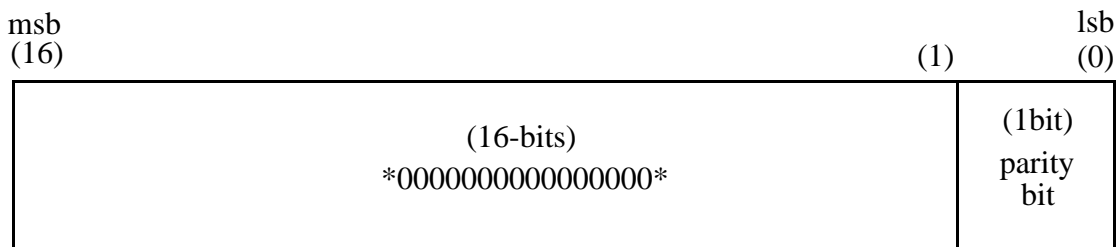
If the Acknowledge Request bit is set in the HEADER packet of a message, the M-module will transmit a PACKET COUNT packet as the packet immediately following the HEADER packet. A PACKET COUNT packet conveys either the known number of packets still to be transmitted from M-module to S-module in the current message or the fact that the number of such packets has not been, or cannot be, predicted.

### 7.6.4 DATA packet requirements



**Figure 23—DATA packet format**

### 7.6.5 NULL packet requirements



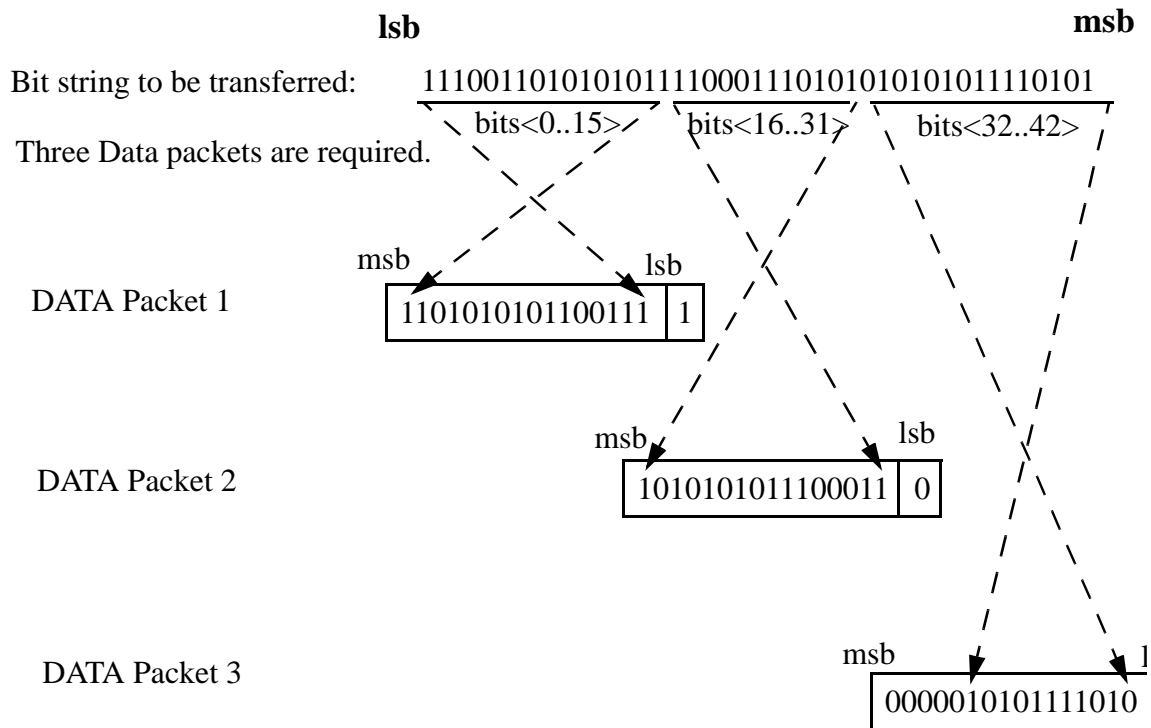
**Figure 24—NULL packet format**

### 7.6.5.1 Description

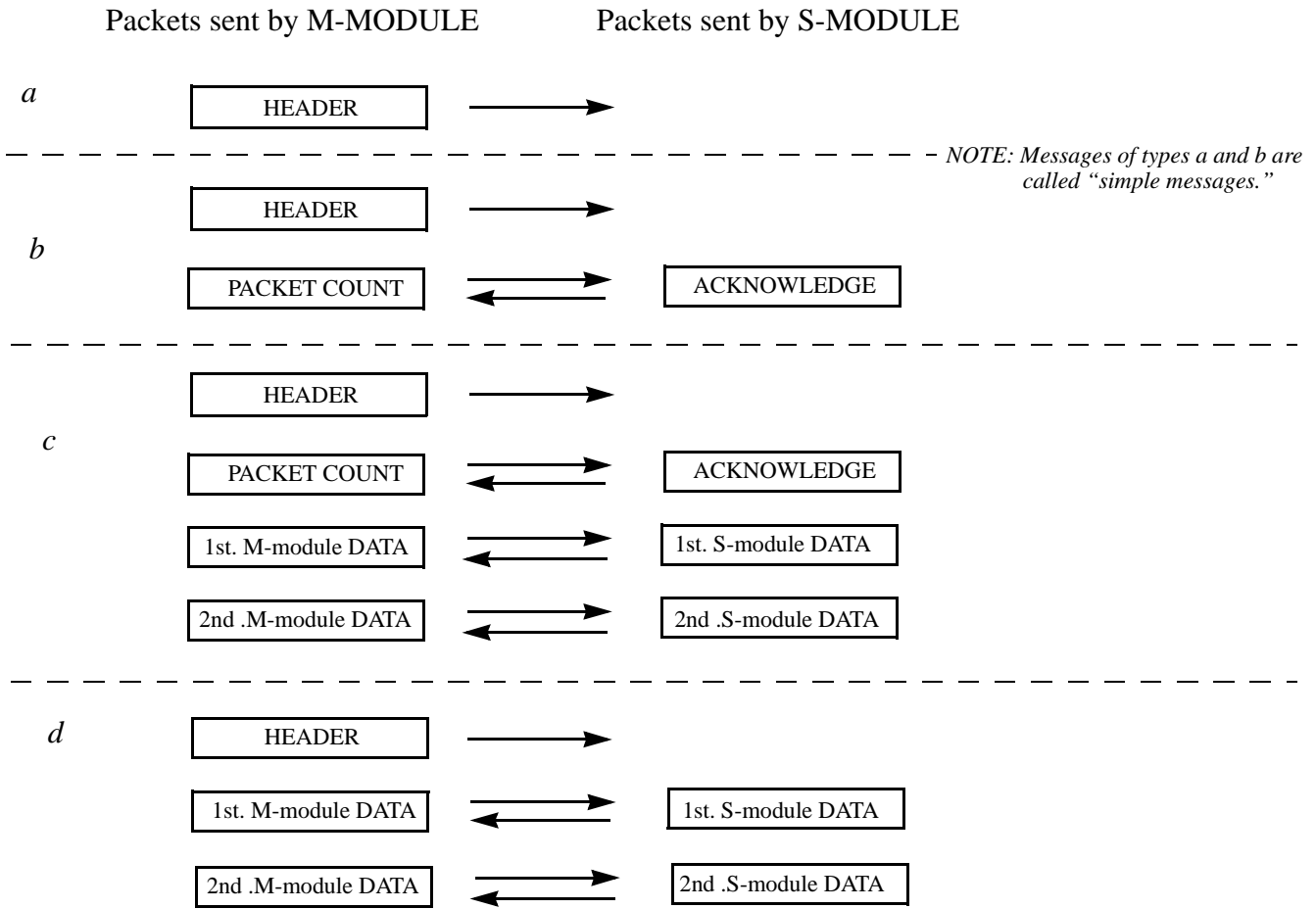
One use of a NULL packet is its transmission by the M-module during periods when one or more packets of data are being transmitted by an S-module while there is nothing to be transmitted in the reverse direction. The reverse case also will occur. It is important to note that, after transmission of a HEADER packet, during every period of a packet transfer during a singlecast, the MSD and MMD signals are being interpreted (by the M-module and addressed S-module, respectively) as conveying bits of a packet. Therefore, failure to transmit a packet is never an option for an S-module addressed by a singlecast message; in such a situation, the transmission of a NULL packet with correct parity is necessary to operation of the MTM-Bus.

### 7.6.6 Formatting bit strings of more than 16 bits for transmission in DATA packets

Figure 5-6 provides an illustration demonstrating how a bit string of more than 16 bits is divided so that it can be appropriately transmitted in a series of DATA packets.



**Figure 25—Formatting a bit string of length greater than 16 into multiple DATA packets.**



**Figure 26—Packet transfers defining the MTM-Bus message sequences**

NOTE—Messages may consist of a single HEADER packet or include transfer of ACKNOWLEDGE/PACKET COUNT and DATA packets. Messages with message sequence of types a and b are called *simple messages*. Only two DATA packet pairs are illustrated for formats c and d; however, there is no limit to the number of DATA packets that can be sent in an MTM-Bus message.

### 7.7 Full MTM command set

In addition to the normal commands provided by the 1149.5 command set, the command extension

**Table 23—Maintenance Bus Commands**

Command Class	Command Code (Binary)	Command Code (HEX)	Command	Status
Core	0000000	00	Read Status	Required
	0000001	01	Abort	Required
	0000010	02	Reset Slave Status	Required
	0000011	03	Contend for Bus	Required
	00001XX	04-07	Multicast Group Select	Required
	0001000	08	Enable Idle Interrupts	Required
	0001001	09	Enable Pause Interrupts	Required
	0001010	0A	Disable Idle Interrupts	Required
	0001011	0B	Disable Pause Interrupts	Required
	0001100	0C	Enable Module Control	Required
	0001101	0D	Data Echo Test	Required
	0001110	0E	Verify BMR	Required
	0001111	0F	Initialize Application	Required
	0010000	10	Disable Module Control	Required
	0010001	11	Start	Required
	0010010-0011111	12-1F	Reserved	Reserved
	1111111	7F	Illegal Command	Required
Data Transfer	0100000	20	Read Data	Recommended
	0100001	21	Write Data	Recommended
	0100010	22	Read/Write Data	Recommended
	0100011-0101111	23-27	Reserved	Reserved
Module Initialization and Self-Test(MIST)	0101000	28	Reset Module with SBIT	Recommended
	0101001	29	Reset Module without SBIT	Recommended
	0101010	2A	Module IBIT	Recommended
	0101011-0101111	2B-2F	Reserved	Reserved
Module I/O Control and Test (MICT)	0110000	30	Disable Module I/O	Recommended
	0110001	31	Enable Module I/O	Recommended
	0110010	32	Force Module Outputs	Recommended
	0110011	33	Sample Module-No Change	Recommended
	0110100	34	Sample Module-Don't Care	Recommended
	0110101	35	Sample Module with Force	Recommended
	0110110	36	Release Module I/O	Recommended
	0110111-1001111	37-4F	Reserved	Reserved
HiRelPCI Extensions	1010000	50	Disconnect module from HiRelPCI Bus	Required
	1010001	51	Connect Module to HiRelPCI Bus	Required
	1010010	52	Disable main power converters	Required
	1010011	53	Enable main power converters	Required
	1010100	54	Read Serial Number, NodeId	Required
	1010101	55	Failure Insertion Control (simulation)	Required
	1010110	56	TBD	
	1010111	57	Read Power Status Block	Required

**Table 23—Maintenance Bus Commands**

Command Class	Command Code (Binary)	Command Code (HEX)	Command	Status
	1011000	58	Set Poke Address and mode (64 bits)	Required
	1011001	59	Read Poke Data (64 Bit mode)	Required
	1011010	5A	Write Poke Data (64 bit mode)	Required
	1011011	5B	TBD	
	1011100	5C	Read 1149.1 JTAG data	Required
	1011101	5D	Write 1149.1 JTAG Data/Program Data	Required
	1011110	5E	Read Message (Header + Data + CRC)	Required
	1011111	5F	Write Message (Header + Data + CRC)	Required
User-Defined	1100000-1111110	60-6E	User-Defined Commands	Reserved

## 8. Control and Status Register and Configuration Operation

Configuration of the system will require the configuration of one or more modules located on one or more bus segments

Mechanisms available for Configuration of the system include:

IEEE 1212-1991 Configuration Space that should be mapped to use the nodeId space addressing.

Remote configuration from another bus segment through the 1394.2 bridge or switch.

Remotely through the Ethernet connection.

How do you configure 64K nodes?

A lot of work to be added.

### 8.0.1 Addressing

The addressing used in this standard must consider the ability to control and use a large set of nodes as well as working within each chassis and its nodes. This requires the merging of several address domains into a single cohesive one. This is accomplished through the CSR Architecture defined in IEEE Std 1212 - 1991 and using the required control and status register address allocation with mappings for CSR, MTM, and PCI to assure the ability of any node on the system to access the conditions in any other node, software permitting.

### 8.0.2 Requirements

#### 8.0.2.1 System Configurations

Small System 2 - 8 nodes 1 level interconnect

Serial Express Interconnect

PacketBus Interconnect

SCI Interconnect

Medium System 4 - 64 nodes 2 level interconnect

Serial Express Interconnect 3 - 4 Busses

PacketBus / Serial Express Interconnect

SE/PacketBus/SCI Interconnect

Large System 20 - 200 Crates 20 - 64K nodes

Serial Express Interconnect 3 - 100+ Busses

PacketBus / Serial Express Interconnect

SE/PacketBus/SCI Interconnect

## Medium System 4 - 64 nodes Redundant Links

Serial Express Interconnect 3 - 8 Busses

PacketBus / Serial Express Interconnect

SE/PacketBus/SCI Interconnect

Multiple Servers - Multiple Disk arrays

## Large System 20 - 200 Crates 20 - 64K nodes Multiple Redundant Links

Serial Express Interconnect 3 - 100+ Busses

PacketBus / Serial Express Interconnects

SE/PacketBus/SCI Interconnect

2 to 4 links / PacketBus

Different types of links

## NonStop Systems - Multiple Nodes Multiple Crates

Traffic Control

Air Traffic control

Telephony

(medical)

Security

Safety systems

NO SYSTEM WIDE RESETS

## Extensible

Over all system must be able to grow uniformly from a small system of two of three nodes to a full grown system that approaches the maximum possible configuration of 64K nodes on 4 to 8K small busses.

Reset and RestartReset needs to defined on the smallest segment space possible. This is to prevent undesirable reset structuree.

## Startup

The system must be able to startup cleanly in small (4 node) to largest (64k nodes) and in reasonable time. Startup and reconfiguration shall be completed by any node in the system. This imply and requires that the MTM bus must be addressable through the external link from within or outside of the current set of nodes once a link has been established to other nodes.

## Uniqueness

After Startup and system configuration, the elements of the system must have unique identifier. This identifier may be available for identification and correlation to the software nodeId.

Hot Swap Support each node/system must be able to unplug a broken board and replace with a new board without complete reconfiguration of the system.

Hot Standby Support - Redundant equipment in standby partial power mode.

Multiple Links on Nodes - Each PacketBus may/must have multiple links for redundancy - How are addresses assigned and what happens when another link is added.

EUI - Extended Unique Identifier - A 64 bit number that is unique as an identifier of this particular component.

Company ID - A 24 bit numerical identifier for a company that is assigned by the IEEE Registration Authority Committee. This number identifies the company that made the equipment.

NodeId - The 16 bit address that is the upper 16 bits of the 64 bit address. This address determines the target of a packet or the strip address of a broadcast packet in a ring structure.

Soft addresses - This is the address assigned to a node or set of nodes following the configuration of the extended

Hard addresses - Initial address before Links are working and have negotiated a unique set of addresses.

Multicast Addresses - A packet address directed to group of functions.

Broadcast Addresses - A packet address that targets all / group / of targets

Domain - Addresses within a 14-16 slot chassis

Resource Map - A Map of EUI to Softaddresses: softId and should contain - SoftId, EUI, Configured By (token? / EUI), Authentication (Verification signature 128 or more bits pgp/IEEE1363)

Pool Resource Map - A Map of Redundant resources to soft addresses.

Structures - Single Ring, Multiple Rings, Redundant Rings, Hub/Router

## **8.1 Addressing**

### **8.1.1 64 Bit Fixed Addressing**

This standard uses a 64 bit address space for the purposes of access and for messages. This 64 Bit address is called Fixed addressing in the CSR standard (IEEEstd 1212-1991). The basis of this address is a 16 bit node address and a 48 bit address in each node. There are definitions for a specific set of address within the 48 Bit address window to define the control registers for a given node. Each node has the same registers in the same address location which permits the management of large system.

Within HiRelPCI with hot pluggable board, many of which will be of similar design, geographical addressing is used to guarantee that each node in the system has a unique node address. Different geometries will have different needs depending on the type of chassis and the extent of subsystems within the chassis. It is the responsibility of the system designer to insure that each board that is plugged into a backplane will have a unique address for the A(63::48) nodeId address segment.

For descriptive purposes we will assume that there are up to 8 nodes on a given backplane bus segment and that 2 segments are joined in a chassis for a possible total of 16 addressable elements in a given chassis.

Of the 64 Bit address A(63::48) address a specific NodeId that shall be unique to the board. Other nodeIds may be assigned by software to form multicast and broadcast addresses. For our example A(51::48) Provide the Geographical Address in bus to guarantee uniqueness on the bus in startup hardId address and unit addresses if not reconfigured by external software agents.

If a bus were to include four PCI segments then A52 is used to differentiate top bus from bottom bus.

Configuration software, after initialization, will use A(63::52) to determine chassis identity and then A(47::0) will determine the Addressing within node.

Within the node the address is initially defined in the CSR (IEEE Std 1212 - 1992) as follows:

**Table 24—Basic 64 Bit Fixed Address CSR Architecture**

<b>Begin Address nnnn = nodeId</b>	<b>End Address nnnn = nodeId</b>	<b>Function</b>
nnnn 0000 0000 0000	nnnn 0000 0FFF FFFF	Register Address Space
nnnn 0000 1000 0000	nnnn 0000 1FFF FFFF	Node Private Address Space (256M)
nnnn 0000 0000 0000	nnnn 0000 0FFF F7FF	Initial Units Address Space (256M-2K)
nnnn 0000 0FFF F800	nnnn 0000 0FFF FFFF	Initial Register Address Space (2K)
nnnn 0000 2000 0000	nnnn FFFF FFFF FFFF	Initial Memory Space (281 Tbytes- 2 <sup>40</sup> )

Medium of Transfers may include P1996, SCI, P2100 1394-1995, P1394A/B, Ethernet, FDDI, SONET/ATM,

### 8.1.2 P1996 CSR Register Definitions

The CSR registers are at fixed addresses offset relative to nnnn 0000 0FFF F800 as follows

**1.5Bit, Byte, and Quadlet Ordering.** The CSR Architecture defines registers that are 4 bytes (orlarger) in size. To ensure interoperability across bus standards, the ordering of the bytes within these registers is defined by their relative addresses, not their physical position on the bus. Bus bridges are similarly expected to route data bytes from one bus to another based on their addresses, not their physical position on a bus. The routing of data bytes based on their address is called address-invariance. To support the address-invariance model, bus standards shall specify the mapping of their physical byte-lanes to relative data-byte addresses. The CSR Architecture specifies the mapping of data-byte addresses to bytes within the multibyte CSRS. These two specifications indirectly specify the mapping between a CSR and physical byte-lanes on the bus. For a quadlet CSR access, the data byte with the smallest address is the most significant, as illustrated in Fig 1-1.

**Figure 27— Fig 1-1 Byte and Quadlet Ordering????**

**Table 25—**

Offset Address from nnnn 0000 0FFF F800	Offset in Hex nnnn 0000 0FFF F800	Register Name	Required
	0	STATE_CLEAR	
	4	STATE_SET	
	8	NODE_IDS	Required
	C	RESET_START	Required
	10	INDIRECT_ADDRESS	
	14	INDIRECT_DATA	
	18	SPLIT_TIMEOUT_HI	
	1C	SPLIT_TIMEOUT_LO	
	20	ARGUMENT_HI	
	24	ARGUMENT_LO	
	28	TEST_START	
	2C	TEST_STATUS	
	30 - 4C	Reserved for Extended Memory Space	
	50	INTERRUPT_TARGET	
	54	INTERRUPT_MASK	
	58	CLOCK_VALUE_HI	
	5C	CLOCK_VALUE_MID	
	60	CLOCK_TICK_PERIOD_MID	
	64	CLOCK_TICK_PERIOD_LO	
	68	CLOCK_STROBE_ARRIVED_HI	
	6C	CLOCK_STROBE_ARRIVED_MID	
	70 -7C	CLOCK_INFO(0-3)	
	80 - BC	MESSAGE_REQUEST	
	C0 - FC	MESSAGE_RESPONSE	
	100 -180	Reserved	
	180 - 1FC	ERROR_LOG_BUFFER	
	200	PCI DeviceID VendorID Register 00h	Required
	204	PCI STATUS COMMAND Register 04h	Required
	208	PCI ClassCode RevisionID Register 08h	Required
	20C	PCI INFO Register 0Ch	Required
	210	PCI Base_Address_0 Register 10h	
	214	PCI Base_Address_1 Register 14h	

**Table 25—**

<b>Offset Address from nnnn 0000 0FFF F800</b>	<b>Offset in Hex nnnn 0000 0FFF F800</b>	<b>Register Name</b>	<b>Required</b>
	218	PCI Base_Address_2 Register 18h	
	21C	PCI Base_Address_3 Register 1Ch	
	220	PCI Base_Address_4 Register 20h	
	224	PCI Base_Address_5 Register 24h	
	228	PCI CardBus CIS Pointer Register 28h	
	22C	PCI SubSystemID SubSystem VendorID 2Ch	
	230	PCI Expansion ROM Base Address Reg- ister 30h	
	234	PCI Reserved Register 34h	
	238	PCI Reserved Register 38h	
	23C	PCI Grant Interrupt Info 3c	
	240 - 2FC	PCI REGISTER EXPANSION	
	300 - 307	MTM Fault Logs	Required
	308 - 30F	MTM Test Data Storage	
	310	MTM Slave Status Register	Required
	314	MTM Bus Error Registers	Required
	318	MTM Module Status Register	Required
	31C	MTM Additional Status Registers	
	320	MTM Manufacturers ID Port	Required
	324	MTM Module Manufacturer Port	Required
	328 - 32F	MTM User Identification Ports	Required
	330 - 34F	MTM Access to IEEE Std 1149.1 bus	Required
	350	MTM Command Register	
	354 - 35B	MTM Command Data Pointer	
	380 - 3BF	Voltage/Current/Temperature Storage	
	3C0 - 3C7	Pointer to Primary Resource Table	Required
	3C8 - 3CF	Pointer to Secondary Resource Table	Required
	3D0 - 3D7	Pointer to Primary Configuration Table	Required
	3D8 - 3DF	Pointer to Secondary Configuration Table	Required
	3E0 - 3FF	Reserved	
	400 - 7FC	ROM_WINDOW	Required

## 8.2 Node Addressing

### 8.2.1 Node Addresses.

The CSR Architecture defines compatible address-space mappings, as follows:

- (1)*Fixed.* The **fixed** address-space model supports only 64-bit addresses. The entire 64bit address space is split into 64K equal initial node spaces, which are 256 Tbytes in size. A node space contains initial memory, private, register, and initial units spaces.

Since some of the bus addresses and node offset addresses are reserved for special uses, this restricts the total number of realizable buses and nodes to a slightly smaller number.

The same number of node addresses (64K) are supported in both address-space mappings, to simplify the address conversions through bus bridges. See the following sections for details.

## 8.3 ROM Formats

### 8.3.1 First ROM Quadlet.

The first (most-significant) byte of the first ROM quadlet is used to specify which ROM format is implemented. After a node is initialized, ROM access may be unavailable while the node is being tested, the quadlet shall be zero. When the ROM access is enabled, the first byte is 1 for the minimal ROM implementation option, and is greater than one for the general ROM option, as illustrated in Fig ?.

#### Fetching ROM From a Desired Offset Address

```
/* Routine for accessing ROM, based on base address of the node's CSRs

* carBase - base address of node being accessed

* offsetOfValue - byte offset of quadlet being accessed */ #define
INDIRECT_ADDRESS 16 #define INDIRECT_DATA 20 ^;~ Quadlet FetchRomQua-
dlet(csrBase,offsetOfValue) Quadlet *csrBase, offsetOfValue; I Quadlet
*carAddress, *regAddress, *regData;

assert((offsetOfValue%4)==0); /* Offsets is multiple of 4 */
if (offsetOfValue<1024) |
/* Internal address is mapped directly (but offset from csrBase) */
carAddress= (Quadlet *)((char *)carBase+1024+offsetOfValue);
return(*cerAddress);
else |
/* Otherwise, address is mapped indirectly */
```

```
regAddress= (Quadlet *)((char *)csrBase+INDIRECT_ADDRESS);  
regData= (Quadlet *)((char *)csrBase+INDIRECT_DATA);  
*regAddress= offsetOfValue;  
return(*regData);
```

## **9. Support Structures**

Is more needed to be added at this point? Else erase this chapter.



## 10. Electrical Specifications

### 10.1 Overview

This standard uses the 3.3V variation of the PCI style boards. While the onboard logic of any attached boards will use the logic voltages appropriate to the board, all PCI interfaces to the passive backplane environment will be at the 3.3V level. Please note that all references to the PCI 3.3 volt operation should be referenced to the PCI Local Bus Specification 2.1 or later.

The TDM bus operates at LVPECL levels and must be closely controlled. This chapter contains the information on this interface. An active termination voltage on the backplane for the LVPECL lines is maintained at a nominal 1.3V. All LVPECL lines are terminated at each end with a 56 ohm resistor to this voltage.

#### 10.1.1 Dynamic vs. Static Drive Specification

HiRelPCI is a low voltage swing CMOS level bus. The drive requirements are for a DC static drive level and a AC drive used during the voltage transitions. In addition to the drive characteristics of the components, the specification of the voltage swings on the backplane will be limited to reduce the overvoltage conditions that can be present and to limit the noise permitted on the backplane.

#### 10.1.2 LVPECL DC Specifications

The TDM part of the backplane requires the use of LVPECL signaling to achieve the clock periods to support the STS-12 frame structure used. This bus is clocking at 77.76 MHz with a period of 12.86008 ns. The CSM board is required to provide the clock and frame signal to each node for the top and bottom busses.

The DC Parameters for the GTL signaling are as follows:

**Table 26—LVPECL DC Parameters**

Symbol	Parameter	Value	Tolerance	Notes
$V_{TT}$	Termination Voltage	1.3V	+/- 2%	
$V_{REF}$	Input Reference Voltage	1.98 V	+/- 3%	
$R_T$	Termination Resistance	$Z_{EFF}$ (nominal)		$Z_{EFF} = Z_0(nominal) / (1 + C_d/C_o)^{1/2}$
$Z_{EFF}$	Effective (loaded) Network Impedance	45 - 65 ohms		Under all conditions

GTL DC I/O Buffer Specifications for TDM Signaling are shown in TABLE 26.

#### 10.1.3 PCI AC Specifications

Please refer to the PCI Local Bus Specification Revision 2.1 or later for current information.

#### 10.1.4 LVPECL AC Specifications

This section is derived from the specification for LVPECL as defined by Motorola and Synergy.

**Table 27—LVPECL I/O Buffers DC Parameters**

Symbol	Parameter	Min	Max	Notes
V <sub>OL</sub>	Driver/Output Low Voltage	1.47V	1.745 V	
V <sub>IH</sub>	Receiver Input High Voltage	2.135V	2.420	
V <sub>CUTOFF</sub>	Cutoff Voltage on Bus	1.20V	1.26V	
V <sub>IL</sub>	Receiver Input Low Voltage	1.490V	1.825V	
I <sub>ILC</sub>	Input Leakage Current		150 uA	
V <sub>BB</sub>	Reference Voltage	1.92V	2.04V	
V <sub>CC</sub>	Nominal Power supply	3.2V	3.4V	
C <sub>IN</sub> /C <sub>O</sub>	Total Input/Output Capacitance		10 pF	

LVPECL AC Bus Buffer Specification for the TDM bus are as follows:

**Table 28—LVPECL AC Specifications**

Symbol	Parameter	Min	Max	Units	Figure	Notes
Tr	Output Bus Rise time	500	1000	ps		20 - 80%
Tf	Output Bus Fall time	300	800	ps		20 - 80%
T <sub>CO</sub>	Output Clock to Data	500	1000	ps		
T <sub>CBUS</sub>	Output Clock to Bus	825	1800	ps		
T <sub>SU</sub>	Input setup time	150		ps		
T <sub>HOLD</sub>	Input hold time	450		ps		

### 10.1.5 Maximum AC rating and device protection

In the implementation of this standard a series resistor shall be inserted between the driver and the bus connector. The value of this resistor shall be determined to prevent the overshoot and undershoot at the load connector positions in excess of 5.0V peak and -1 Volt negative peak. A typical value for this resistor is 10 ohms.

## 10.2 Timing Specifications

### 10.2.1 .TDM Timing Specification

All timing references for the TDM bus are take with respect to the Differential Clock input pins at the board edge.

This section to be added to support the 77.76MHz timing for the SONET STS12 Byte parallel bus.

## 10.2.2 PCI and TDM Clock Specifications

**Table 29—TDM Clock Specifications**

Symbol	Parameter	Min	Max	Units	Notes
t <sub>cy</sub>	CLK Cycle Time	12.86008	12.86008	ns	0.1ppm
t <sub>high</sub>	CLK High Time	5	7	ns	
t <sub>low</sub>	CLK Low Time	5	7	ns	
t <sub>rise</sub> t <sub>fall</sub>	CLK Slew Rate	200	600	ps	Differential Signaling
Clock Skew	CLK Skew	0	500	ps	Clock-Clock

## 10.2.3 TDM Clock Specifications

Timing specifications for the TDM GTL clocks to be added.

## 10.3 Backplane Specifications

### 10.3.1 Backplane Signal Timing Budgets

The relevant timing budget can be expressed by the equation:

$$T_{cyc} \geq T_{val} + T_{prop} + T_{su} + T_{skew}$$

The following table compares PCI and LVPECL timing budgets at various speeds.

**Table 30—Timing Budgets**

Timing Element	33 MHz	66 MHz	77.76 MHz	Units	Notes
T <sub>cy</sub>	30	15	12.860082	ns	
T <sub>val</sub>	11	6	2.0	ns	
T <sub>prop</sub>	10	5	8	ns	
T <sub>su</sub>	7	3	0.15	ns	
T <sub>hold</sub>			0.45	ns	
T <sub>jitter</sub>			0.5	ns	
T <sub>skew</sub>	2	1	1.0	ns	

## 10.4 Power

Power for the bus segment is distributed at a nominal voltage of 48V and is regulated as needed on the board where the power is consumed. Each 6su bus slot connects to two power rails with 11 pins each, while the

12su boards connect to duplicate set of power rails on the lower connector. While the nominal power is 48V d.c. the limits of the supply are 36V d.c. to 58V d.c. This voltage is within the Safety Extra Low Voltage directive and meets this requirement defined in IEC 950, EN 60 950 and UL1450. The isolation is accomplished in the primary supply and that must meet the 3750V isolation requirement as well as creepage and clearance requirements. The SELV limits are 60V d.c. in the US and 75V d.c. in ISO/IEC areas although the limit is decreasing to 60V d.c.

This main power distribution voltage is derived from the use of 4 “12V” lead acid batteries in series. The nominal voltage of a single “12V” battery under charging conditions is 13.75V, as in any automotive system, which is 55V for 4 batteries in series. When not being charged, the float voltage is about 52.8V.

The SCI standard IEEE 1596-1992, and IEEE Std. 896.5 1995 (ISO/IEC 14536: 1995) uses this power system as do telephony and industrial systems. Common use of this voltage range is the telephone system with normal line power being the “Battery” voltage of 52.8V to 55V. New computer system intended for internet service and database servers are also using “48V” power. Traffic intersection control with unreliable power are choosing a 48V system as the main/standby power for the traffic lights. DC power for large routers and switches are now 48V powered. This wide voltage range also allows for voltage sag when a board is hot plugged into the system.

Each set of boards has a dual power rail for 6SU sized boards with a limit of about 200 Watts per board per power rail. It could be possible to draw 200 Watts from the A rail and draw 200 Watts from the B rail. On a 12SU system, each board has four power rails to draw power from.

Power is brought to the backplane either through direct connection on the backplane or through plug in power supply modules.

#### **10.4.1 For power rail A these pins consist of:**

P48VA - 4 pins - This is the Positive Side of the Nominal 48 Volt supply. The four pins set the maximum current draw for this board on this rail to 4 amps or about 200 Watts at a current density of 1 amp per pin. This pin is floating relative to the chassis and only is referenced to the N48VA pin below and the other pins in the power rail A group.

N48VA - 4 pins - This is the Negative Side of the Nominal 48 Volt supply. The four pins set the maximum current draw for this board on this rail to 4 amps or about 200 Watts at a current density of 1 amp per pin. This pin is floating relative to the chassis and only is referenced to the P48VA pin above and the other pins in the power rail A group.

P48VAPRE - 1 pin - This pin is derived from P48VA but is current limited to 100mA. This pin is used to precharge the input capacitors and to power up the initial power converter on the board, but not to supply power to the board. Input capacitance on each board is limited to 30uF maximum. This line should be diode isolated from the P48VA line to prevent any back feeding of power to the P48VA rail. The precharge lines are on the longest connector pins (8.0mm) to make first on insertion to perform their function prior to the connection of the main power pins and before the power converters are enabled by the PRIONA pin which is on the shortest row (5.75mm)

N48VAPRE - 1 pin - This pin is derived from N48VA but is current limited to 100mA. This pin is used with P48VAPRE to precharge the input capacitors and to power up the initial power converter on the board, but not to supply power to the board. This line should be diode isolated from the N48VA line to prevent any back feeding of power to the N48VA rail.

ISHAREA - 1 pin - This pin is located on the D column of the connector that has 5.75mm pin lengths and makes connection after all other power pins have made contact. This pin is used to allow the var-

ious power supplies on this power bus to share the percentage of power provide. for the A rail powered supplies.

For power rail B these pins consist of:

P48VB - 4 pins - This is the B Rail Positive Side of the Nominal 48 Volt supply. The four pins set the maximum current draw for this board on this rail to 4 amps or about 200 Watts at a current density of 1 amp per pin.

N48VB - 4 pins - This is the Negative Side of the Nominal 48 Volt B rail power supply. The four pins set the maximum current draw for this board on this rail to 4 amps or about 200 Watts at a current density of 1 amp per pin. This pin is floating relative to the chassis and only is referenced to the P48VB pin above and the other pins in the power rail B group.

P48VBPRES - 1 pin - This pin is derived from P48VB but is current limited to 100mA. This pin is used to precharge the input capacitors and to power up the initial power converter on the board, but not to supply power to the board.

N48VBPRES - 1 pin - This pin is derived from N48VA but is current limited to 100mA. This pin is used with P48VBPRES to precharge the input capacitors and to power up the initial power converter on the board, but not to supply power to the board.

ISHAREB - 1 pin - This pin is located on the D column of the connector that has 5.75mm pin lengths and makes connection after all other power pins have made contact. This pin is used to allow the various power supplies on this power bus to share the percentage of power provide. for the B rail powered supplies.

#### **10.4.2 Power Requirements**

Power requirement depend on the application. Normal PCI board designs are limited to about 25 Watts per board. Cooling requirements dictate the maximum power used.

This system allows for a plug in power supply in one of the card positions with a total supplied power of a maximum of 4 amps per module. Redundant operation allows for a redundant B rail power supply in the event that the primary supply fails.

12SU boards duplicate the A and B power rails on the lower PCI bus for additional power redundancy.

#### **10.4.3 Backplane Impedance**

The nominal impedance of the backplane is 50 to 80 ohms as measured on a test Coupon on the board. Nominal impedance for the TDM section of the bus will be 45 to 55 ohms.

### **10.5 Connectors**

For the HiRelPCI standard the connectors used are the 2mm type known as the Metral or later the Future-Bus+ style of connector. These connectors are Bellcore, UL and CSA approved and compliant with IEC 1076-4-OX(48B) and EIA SP3179.

Backplane connectors shall be MALE signal connectors and shall have pin lengths as follows:

These connectors are built in block of 24 pins or multiple of 24 pins. A set of 24 pin in a single mode plastic shroud is called a single block connector, while the 8 block connector has 8x24 pins or 192 pins. Connectors that meet this requirement are available from Berg Electronic as:

**Table 31—Pin Stagger**

<b>ROW</b>	<b>Length in mm</b>
a	6.50
b	8.00
c	6.50
d	5.75

Block of 8x24 pins pressfit with tail lengths for up to 4.3mm is part number 70235-977

Block of 1x24 pins pressfit with tail lengths for up to 4.3mm is part number 70232-977

Block of 1x24 with pressfit with tail length of 13.6mm for rear panel connection is 70232-987.

The longer tails allow for a header shroud to be place over the pins on the back of the backplane and then be connected with an I/O connector.

## 10.6 Non-CSM Pin Assignments for Main PCI connectors

Pin assignments for non CSM's have user I/O defined in the rows 49 to 54 of the main block pin assignments as follows:

**Table 32—Non CSM Pin Assignments**

	a	b	c	d
1	P48VA	FGND	P48VA	P48VA
2	P48VA	FGND	N48VA	N48VA
3	N48VA	P48VAPRE	N48VA	ISHAREA
4	P48VB	N48VAPRE	P48VB	ISHAREB
5	P48VB	P48VBPRE	P48VB	N48VB
6	N48VB	N48VBPRE	N48VB	N48VB
7	ENET	GND	1394CLK	1394DATA
8	AD32	GND	AD33	SAD15
9	AD34	GND	AD35	AD36
10	AD37	GND	AD38	AD39
11	AD40	GND	AD41	SAD14
12	AD42	GND	AD43	AD44
13	AD45	GND	AD46	AD47
14	AD48	GND	AD49	SAD13
15	AD50	GND	AD51	AD52
16	AD53	GND	AD54	AD55
17	AD56	GND	AD57	SAD12
18	AD58	GND	AD59	AD60
19	AD61	GND	AD62	AD63
20	PAR64	GND	C/BE4#	SAD11
21	C/BE5#	GND	C/BE6#	C/BE7#
22	ACK64#	GND	REQ64#	CCLK10M
23	AD00	GND	AD01	SAD10
24	AD02	GND	AD03	AD04
25	AD05	GND	AD06	AD07
26	C/BE0#	GND	M66EN	SAD09
27	AD08	GND	AD09	AD10
28	AD11	GND	AD12	AD13
29	AD14	GND	AD15	SAD08
30	C/BE1#	GND	PAR	SERR#
31	SBO#	GND	SDONE	PERR#
32	LOCK#	GND	STOP#	SAD07
33	IRDY#	GND	TRDY#	DEVSEL#
34	FRAME#	GND	C/BE2#	AD16
35	AD17	GND	AD18	SAD06
36	AD19	GND	AD20	AD21
37	AD22	GND	AD23	IDSEL
38	C/BE3#	GND	AD24	SAD05
39	AD25	GND	AD26	AD27
40	AD28	GND	AD29	AD30
41	REQ#	GND	AD31	SAD04
42	SAD03	GND	TX+	RX+
43	GNT#	GND	TX-	RX-
44	SAD01	GND	MCLK	SAD02
45	CLK	GND	REQP0#	SAD00
46	REQP1#	GND	REQP2#	RST#
47	MMD	GND	MSD	REQP3#
48	MPR	GND	MCTL	Spare5#
49	user defined	user defined	user defined	user defined
50	user defined	user defined	user defined	user defined
51	user defined	user defined	user defined	user defined
52	user defined	user defined	user defined	user defined
53	user defined	user defined	user defined	user defined
54	user defined	user defined	user defined	user defined

## 10.7 CSM Pin Assignments

**Table 33—CSM PIN Assignments**

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
1	P48VA	FGND	P48VA	P48VA
2	P48VA	FGND	N48VA	N48VA
3	N48VA	P48VAPRE	N48VA	ISHAREA
4	P48VB	N48VAPRE	P48VB	ISHAREB
5	P48VB	P48VBPRE	P48VB	N48VB
6	N48VB	N48VBPRE	N48VB	N48VB
7	ENET	GND	1394CLK	1394DATA
8	AD32	GND	AD33	SAD15
9	AD34	GND	AD35	AD36
10	AD37	GND	AD38	AD39
11	AD40	GND	AD41	SAD14
12	AD42	GND	AD43	AD44
13	AD45	GND	AD46	AD47
14	AD48	GND	AD49	SAD13
15	AD50	GND	AD51	AD52
16	AD53	GND	AD54	AD55
17	AD56	GND	AD57	SAD12
18	AD58	GND	AD59	AD60
19	AD61	GND	AD62	AD63
20	PAR64	GND	C/BE4#	SAD11
21	C/BE5#	GND	C/BE6#	C/BE7#
22	ACK64#	GND	REQ64#	CCLK10M
23	AD00	GND	AD01	SAD10
24	AD02	GND	AD03	AD04
25	AD05	GND	AD06	AD07
26	C/BE0#	GND	M66EN	SAD09
27	AD08	GND	AD09	AD10
28	AD11	GND	AD12	AD13
29	AD14	GND	AD15	SAD08
30	C/BE1#	GND	PAR	SERR#
31	SBO#	GND	SDONE	PERR#
32	LOCK#	GND	STOP#	SAD07
33	IRDY#	GND	TRDY#	DEVSEL#
34	FRAME#	GND	C/BE2#	AD16
35	AD17	GND	AD18	SAD06
36	AD19	GND	AD20	AD21
37	AD22	GND	AD23	IDSEL
38	C/BE3#	GND	AD24	SAD05
39	AD25	GND	AD26	AD27
40	AD28	GND	AD29	AD30
41	REQ#	GND	AD31	SAD04
42	SAD03	GND	TX+	RX+
43	GNT#	GND	TX-	RX-
44	SAD01	GND	MCLK	SAD02
45	CLK	GND	REQP0#	SAD00
46	REQP1#	GND	REQP2#	RST#
47	MMD	GND	MSD	REQP3#
48	MPR	GND	MCTL	BPSN#
49	CLKS0	CLKS6	REQ0#	GNT0#
50	CLKS1	GNT6#	REQ1#	GNT1#
51	CLKS2	REQ6#	REQ2#	GNT2#
52	CLKS3	GND	REQ3#	GNT3#
53	CLKS4	PUPOWER	REQ4#	GNT4#
54	CLKS5	PUPOWER	REQ5#	GNT5#

This pinout shown in Table 32 is for the Central Services Module that provides the clocks and the arbitration for the bus segment. Rows 49 through 54 are used to provide these signals. In some instances the CPU board will serve this function and will reside in the slot in the backplane that has this configuration.

For 12SU boards in redundant mode the pinouts are repeated for the lower set of connectors. The CSM for a 12SU redundant chassis must provide independent arbitration and clocking for each serviced bus.

## 10.8 TDM module pin assignments

### 10.8.1 Non CSM TDM Pin Assignments

**Table 34—NonCSM TDM Pin Assignments**

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
1	ATDM0	GND	ATDM1	ATDM2
2	ATDM3	NC	ATDM4	ATDM5
3	ATDM6	GND	ATDM7	ATDM8
4	ATDM9	ATDM12	ATDM10	ATDM11
5	ATCLK6+(in)	NC	ATDM13	ATDM14
6	ATCLK6-(in)	NC	AFRAME	ATDM15
7	user-defined	user-defined	user-defined	user-defined
8	user-defined	user-defined	user-defined	user-defined
9	user-defined	user-defined	user-defined	user-defined
10	user-defined	user-defined	user-defined	user-defined
11	user-defined	user-defined	user-defined	user-defined
12	user-defined	user-defined	user-defined	user-defined
13	BTDM0	GND	BTDM1	BTDM2
14	BTDM3	NC	BTDM4	BTDM5
15	BTDM6	GND	BTDM7	BTDM8
16	BTDM9	BTDM12	BTDM10	BTDM11
17	BTCLK+(in)	NC	BTDM13	BTDM14
18	BTCLK-(in)	NC	BFRAME	BTDM15

**10.8.2 CSM TDM Pin Assignments****Table 35—CSM TDM Pin Assignments**

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
1	ATDM0	GND	ATDM1	ATDM2
2	ATDM3	+1.3VTTA	ATDM4	ATDM5
3	ATDM6	GND	ATDM7	ATDM8
4	ATDM9	ATDM12	ATDM10	ATDM11
5	ATCLK6+(in)	ATCLK6+(out)	ATDM13	ATDM14
6	ATCLK6-(in)	ATCLK6-(out)	AFRAME	ATDM15
7	ATCLK0+(out)	ATCLK1+(out)	BTCLK0+(out)	BTCLK1+(out)
8	ATCLK0-(out)	ATCLK1-(out)	BTCLK0-(out)	BTCLK1-(out)
9	ATCLK2+(out)	ATCLK3+(out)	BTCLK2+(out)	BTCLK3+(out)
10	ATCLK2-(out)	ATCLK3-(out)	BTCLK2-(out)	BTCLK3-(out)
11	ATCLK4+(out)	ATCLK5+(out)	BTCLK4+(out)	BTCLK5+(out)
12	ATCLK4-(out)	ATCLK5-(out)	BTCLK4-(out)	BTCLK5-(out)
13	BTDM0	GND	BTDM1	BTDM2
14	BTDM3	+1.3VTTB	BTDM4	BTDM5
15	BTDM6	GND	BTDM7	BTDM8
16	BTDM9	BTDM12	BTDM10	BTDM11
17	BTCLK6+(in)	BTCLK6+(out)	BTDM13	BTDM14
18	BTCLK6-(in)	BTCLK6+(out)	BFRAME	BTDM15

## 11. Mechanical Specifications

This chapter contains the Mechanical Specifications for the P1996 HiRelPCI system.

### 11.1 Mechanical Philosophy

The basic guiding philosophy for the mechanical portion of this specification is to adopt the work of others and use what they have worked out. The theory continues here with the adoption of the IEEE Std1301-1991 mechanical specification and the IEC 917-2-2 specification. The chassis and board formats are found in those document and are Metric Standards.

### 11.2 Sub Chassis Specifications

Initial designs use the 6su and 12su height chassis with 30mm front panel spacing. This size is similar to the IEEE 1596-1992 board formats and spacing. This format allows 14 modules to be used in a nominal electronic racking system.

Note that the specification for the opening is specified at 425mm from inside to inside of the opening. 14 30mm modules use 420mm of space. This is adjusted by displacing the left side of the opening an additional 5mm to the right. Note that the opening is offset 7.5mm from the left wall and 2.5mm from the right wall. The backplanes shown in figure \_\_\_ for 6su and 12 su assume this offset.

This system of sheet metal and components is available, off the shelf, from several vendors as shown in the appendix to this standard.

### 11.3 Board Formats

Two board formats are defined in the initial version of this standard, they are the 6su and the 12su formats. Nominal board formats for the 6su system is 115mm x 213mm and includes a connector with 9 blocks of 24 pins each for a total of 216 pins including 24 pins of user defined I/O.

The nominal board format for the 12su style board is 300mm by 300mm with a printed circuit board of 265mm x 288mm and include three sets of connectors:

2 sets of normal bus interfaces of 216 pins each (9 blocks of 24 pins each or 1 block of 216 pins)

1 additional set of TDM bus interfaces including 72 pins in three blocks of 24 pins in the center of the board edge. In CSM modules all three blocks are used while in non-CSM modules the middle block of 24 pins may be used for board dependent I/O, including up to 6 fiber optic or RF connectors. This set of pins is not available to the 6su boards.

The nominal board format for the 6su style board is 150mm by 225mm with a printed circuit board of 115mm x 213mm and include one sets of connectors:

normal bus interface of 216 pins each (9 blocks of 24 pins each or 1 block of 216 pins)

Nominal dimensions for these boards are shown in Figure \_\_\_\_\_. Any differences in the dimensions on this drawing and IEEE Std 1301-1991, with the singular exception of the vertical position of the connectors, is unintentional and the dimension stated in IEEE Std 1301-1991 shall prevail.

An adapter board is used to add two 6su cards into a 12su environment to provide the required support.

Reasons for choice:

International Standards

Available from multiple vendors

format allows high component density both sides of the board.

Can be enclosed for RFI elimination.

Good mechanical size.

## 11.4 Connector Specifications

The connectors used on the backplane and the boards are known as the 2mm Metral connector system. This same connector is also available as the 2mm Future Bus connector. The pins are located in headers on the backplane and the sockets are located on the board. These connectors are available from about 10 vendors with the Berg part numbers shown for example.

8x24 short pin no rear plug in = Berg Part Number 70235-977

1x24 short pin no rear plug in = Berg Part Number 70232-977

1x24 Long pin for rear plug = Berg Part Number 70232-987

All connector positions have staggered pin heights to support Live Insertion and lower insertion force

A Row = 6.5mm

B Row = 8.0mm

C Row = 6.5mm

D Row = 5.75mm

Reasons for choice of these connectors:

Approved by Many agencies

Available from multiple vendors

Cost effective

Reliable

Long performance history

When modular construction is appropriate to a system application, the user should seriously consider using standard modules rather than a custom design. Many designers have underestimated the subtleties of mechanical packaging, discovering too late that the compromises needed in order to use a standard package would have been much less costly than the effort for debugging a custom design. The HiRelPCI Module is based on the IEEE 1301 family of standards, which were produced by a large body of industry experts. This work is shared by (at least) HiRelPCI, SCI and Futurebus+, increasing the volume of production of standard mechanical components and thus reducing their cost.



### 6.1 Type 1 module

The HiRelPCI 6su and 12su modules is intended for applications that need interchangeable modular components. The details needed for compatible module and subrack construction have been worked out by IEEE Std 1301-1991 and IEEE Std 1301.1-1991, so HiRelPCI only needs to choose from the many options provided by those standards. Those standards govern in case of discrepancy with numbers shown here except for a few specific, identified, items.

## 11.5 Module characteristics

The 6su and 12su module specification defines a mechanical, electrical, and thermal environment. This module is optimized for the needs of high performance systems. No backplane is detailed for HiRelPCI, except for connector locations and pinout. The wiring of the backplane signals is beyond the scope of the standard and is the responsibility of the system implementors. There are, however, some recommendations about systems in the HiRelPCI overview (section 1).

## 11.6 Module compatibility considerations

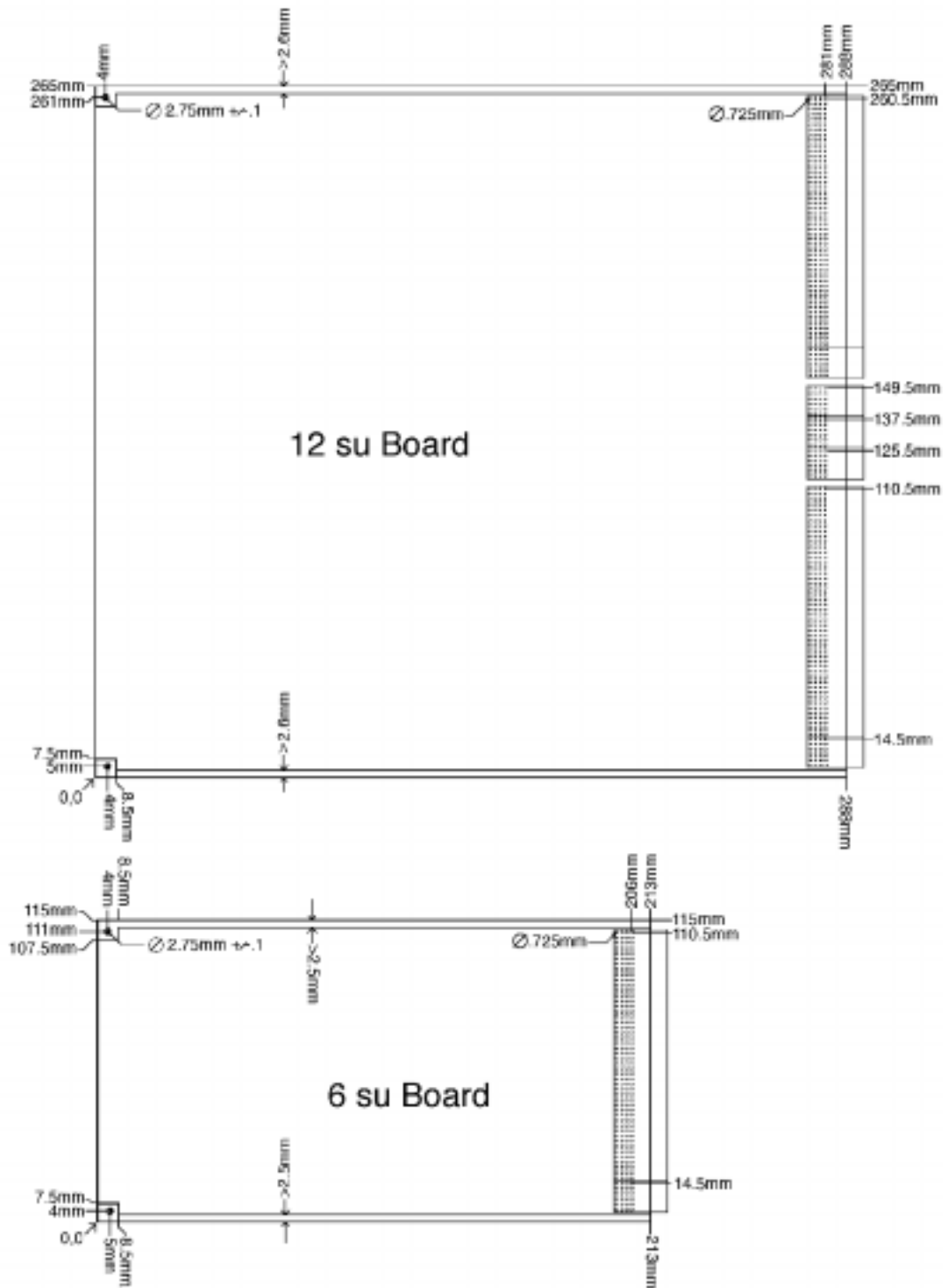
The 12su module size is the same as used by IEEE Std 1596-1991, IEEE Std 896.5 Futurebus Military profile 12su format. to get the economies of scale due to the larger combined market for the mechanical components.

HiRelPCI specifies the ESD etch to be on side 1 to avoid static discharge problems. However, one must be careful that the subrack provides a discharge clip on side 1. Some vendors plan to put discharge clips on both sides of the guide rail near the module entry end, which solves the problem nicely, but others may choose not to. The hazard is reduced by HiRelPCI's connector design, which also incorporates an ESD discharge feature, but during insertion discharges from tall components to neighboring modules remain a possibility if the side 1 discharge clip is missing. If the HiRelPCI board does not need milling to reduce its thickness in that area, it would be good practice to include the optional ESD discharge etch on side 2 as well.

## 11.7 12su Module size

The 12su module size is nominally 300 mm high (12 SU) by 300 mm deep by 30 mm wide, with all associated dimensions defined in IEEE Std 1301.1-1991. The dimensions of the circuit board in such a module are 265.0 (+0, -0.3) mm high by 288 (+0, -0.2) mm deep. The module may use in the common side-mount connector (board mounted connector solder-to-board) as shown in the figures in IEEE Std 1301.1-1991. The module board dimensions are shown in figure 6-2, including the connector position. All dimensions are in millimeters. Tolerance, if not shown, is specified in IEEE Std 1301.1-1991.

### 11.8 6su Module size

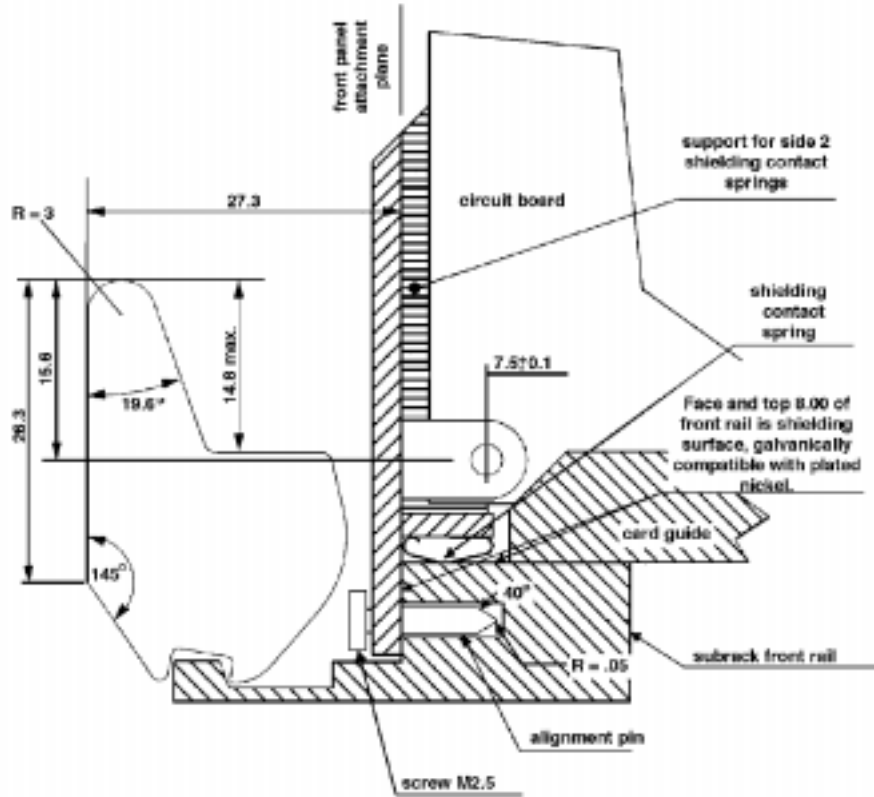


The 6su module size is nominally 150 mm high (6SU) by 225 mm deep by 30 mm wide, with all associated dimensions defined in IEEE Std 1301.1-1991. The dimensions of the circuit board in such a module are 115.0 (+0, -0.3) mm high by 213 (+0, -0.2) mm deep. The module may use in the common side-mount connector (board mounted connector solder to board) as shown in the figures in IEEE Std 1301.1-1991. The module board dimensions are shown in figure 6-2, including the connector position. All dimensions are in millimeters. Tolerance, if not shown, is specified in IEEE Std

1301.1-1991.

The injector/ejector and top and bottom shielding arrangement is shown in more detail in figure 6-3. The module retention screws should always be tightened to ensure reliable mating of the connectors. If the screws are not tightened, it is possible for the module to rotate slightly in the gap between the card guides and the module board, and cumulative tolerance effects including the bow of the backplane could cause the connector to have less than the specified insertion distance.

The relationship between the module circuit board and the front panel and its shielding is shown in figure 6-4, for the side-mount connector.



**Module injector/ejector and top and bottom shielding**

I

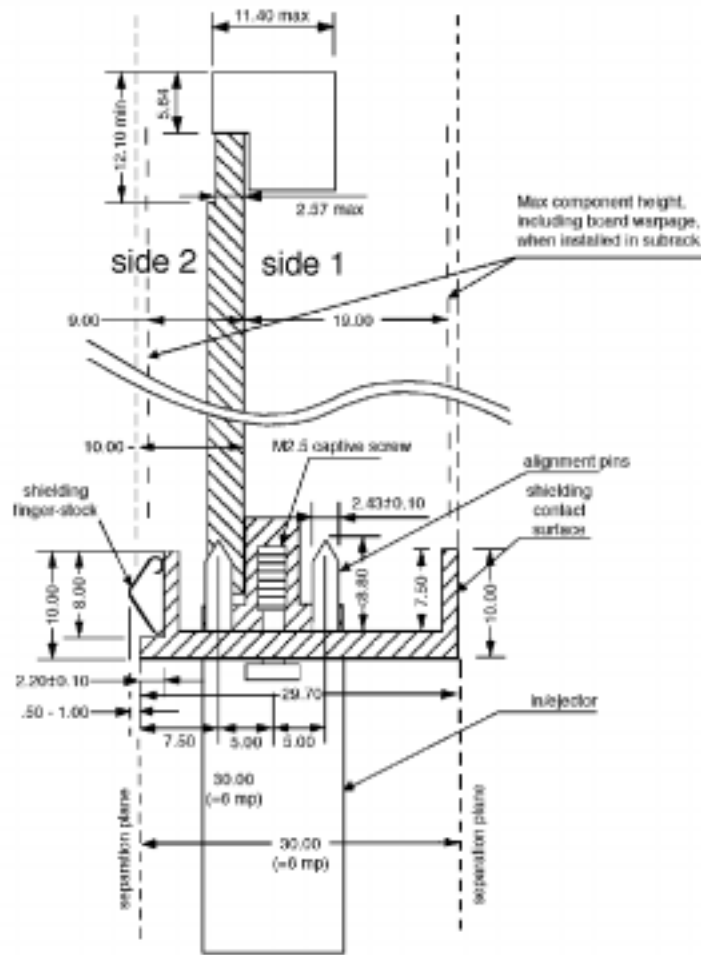
Front panels may use a single alignment pin at the top and another at the bottom, positioned to the right or to the left of the M2.5 mounting screw, however all four pins should be used where possible. On the subrack front attachment plane a configuration of ABA shall be provided to guarantee interchangeability between different suppliers. Unlabeled positions may be implemented in the same pattern.

The ESD contact in the lower guide rail is intended to contact side 1 of the module board as soon as possible during module insertion. An optional additional ESD contact may be on the other side of the guide rail to contact side 2 of the module board.

Figure 6-7 shows the backplane connector location and guide rail position.

## 11.9 Warpage, bowing, and deflection

To assure reliable mating of module and backplane connectors, deviation of the module and backplane from their ideal planar form has to be limited. Module bow also reduces the maximum component



Front panel arrangement, module shielding and clearances

height that can be used without intruding into another module's space during insertion. Care is required throughout the manufacturing process to keep bow and warpage within the following limits:

**Backplane:** When mounted in the subrack assembly, the backplane shall have total warpage plus bowing of 0.6 mm maximum. Total static plus dynamic deflection shall be less than 0.6 mm.

**Module:** Bowing along the connector edge shall be less than 1.325 mm. Dynamic deflection shall be less than 0.6 mm. Note that the module board could be under compression if the module retention screws are overtightened under worst case tolerance conditions.

## 11.10 Cooling

Cooling in a system with high power dissipation VLSI chips is not trivial. It is probably impossible to define a standard cooling capability that will avoid all problems (for example, if a hot chip on the back side of one board is next to a hot chip on the front side of its neighbor, there probably is going to be a cooling problem). Nevertheless, it seems useful to establish guidelines that result in some common assumptions, then leave it to vendors and customers to note exceptional situations and resolve them. Thus, this document recommends an airflow and temperature specification, and suggests certain construction and layout practices.

The cooling air provided by the subrack environment for the HiRelPCI 12su module should flow from bottom to top at not less than 3.5 m/s when the pressure drop is 12.5 Pa (1.27 mm or 0.05 in H<sub>2</sub>O) or less, entering at an inlet temperature

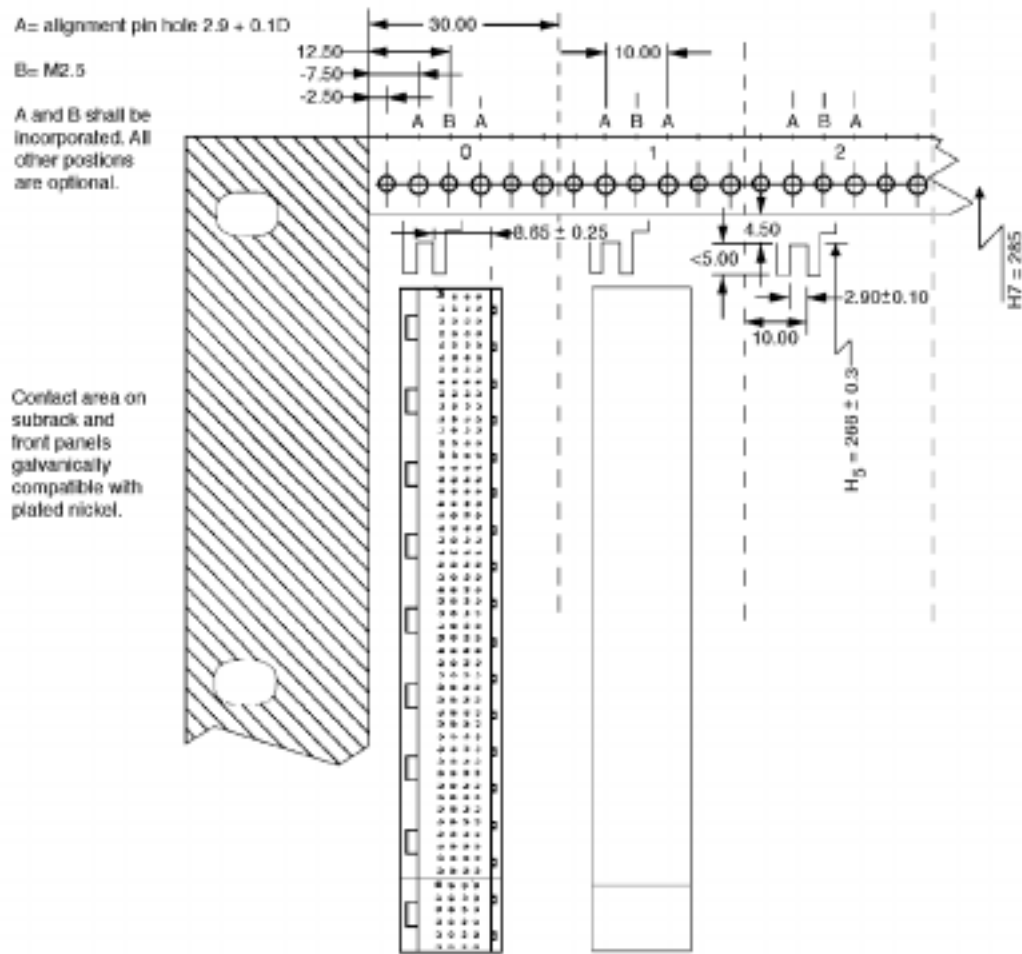
of 25 °C or less (noncondensing). If the module requires more cooling or can tolerate significantly less cooling than this, its requirements should be clearly documented for the user.

Air gains little in cooling capability at velocities above 3.5 m/s. Large systems using enclosed racks (recirculating air past multiple subracks, power supplies and heat exchangers) have been built that are so quiet when operating at this air velocity that their operation cannot be heard in a room with normal computer equipment in use. However, noise is likely to be a problem for nonenclosed systems.

In many of the systems that will use this standard, reliability will be of prime concern and the requirement for convection cooled only systems will dictate a strategy of less power and efficient use of heat dissipaters without the use of fans.

Care must be taken to prevent air from channeling away from hot chips into unobstructed free space between heat-sinks on modules. Baffles can be useful for this. Turbulent, not laminar, flow is desired in high powered systems. Modules should be baffled (possibly on both sides) in such a way that a subrack full of adjacent identical modules creates a pressure drop of 12.5 Pa at 3.5 m/s airflow. Empty module positions in a subrack should be filled with air resistors to prevent free air flow, again producing a pressure drop of 12.5 Pa. Front panels should be used to prevent air leakage. Because the baffles on adjacent sides of two modules interact, designs with critical cooling requirements may find it advantageous to use a cover plate on side 1 and perhaps also on side 2 to eliminate the unpredictable effects of neighboring modules.

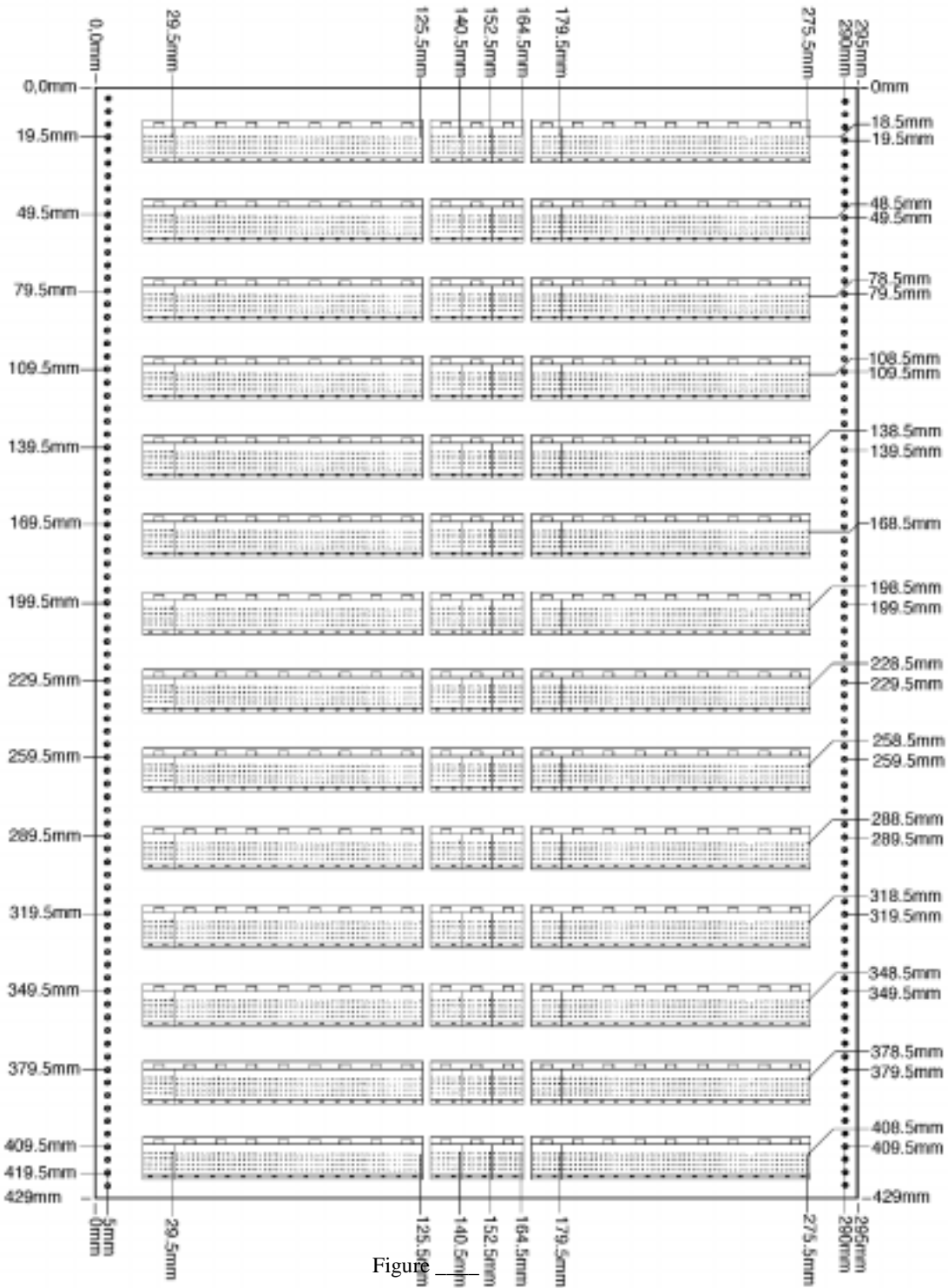
The hottest components should be on the connector side (side 1) of the board, with cooler components (if any) on the back (side 2). Power converters should be located along the top of the module because they can tolerate the higher temperatures there better than VLSI logic circuits can.



Front view of subrack, top left detail

Heat sinks should be designed to keep junction temperatures as low as possible for greatest reliability. 60 °C is a desirable goal, but difficult to achieve. Heat sinks tend to occupy a certain volume per watt, because thickness is needed to

conduct heat to fin extremities. Often a heavy flat plate to spread the heat over a large area will be as effective as a tall



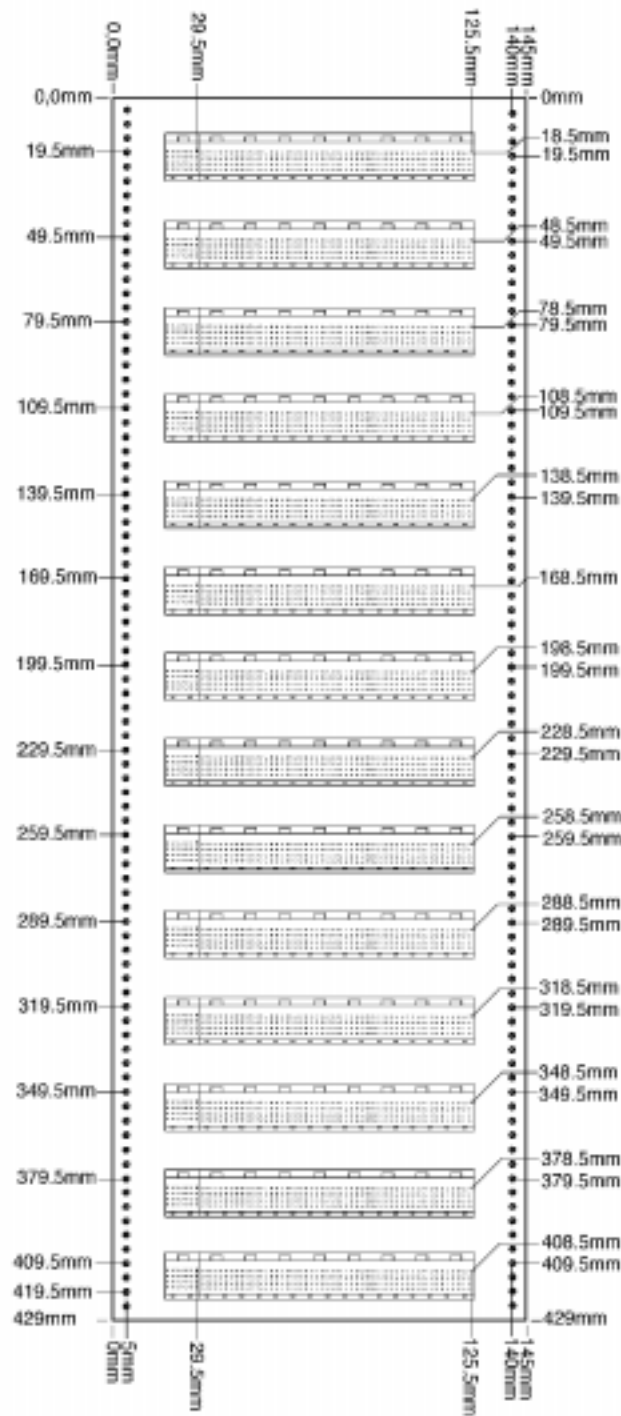


Figure \_\_\_\_\_

elaborately finned structure of similar volume. Significant radiant transfer occurs, so shiny reflective surfaces should be avoided on heat sinks and nearby cool surfaces that heat might radiate toward.

## 11.11 Optional Connector Configurations

Optional Configurations are available to meet different needs for I/O. Some users will not need the full 64 bit implementations or the Dual Bus configuration, or desire Optical Fiber or RF I/O through the rear backplane of the system. When using a connector scheme that differs from the dual redundant system specified in the main part of this standard, care is needed to prevent a non-compatible connector being plugged into a backplane not configured for it. A set of connector keying shall be used to prevent the damage that could be caused by forcing dissimilar connector together.

### 11.11.1 I/O Fiber/RF Connectors

Optional configuration of the I/O connector positions are specified for through backplane connections of Optical Fiber and RF connectors. Connectors within the 2mm "Metral" series are available with configurations that will either one larger diameter fiber/RF connector within the 24 pin block spacing or 6 smaller fiber/RF connectors of the mini-coax size.

If the fiber/RF connector configuration is used the connector blocking shall be used to prevent the installation of boards with standard 2mm connector pins in that location. This keying shall be done at the connector positions where the fiber/RF connectors have been inserted.

Keying pattern to be determined.

### 11.11.2 6su 32 Bit only

This configuration of the 6su format uses only the component of the 32 bit system bus and frees the upper bit of the 64 bit system for use as I/O pins.

Keying pattern to be determined.

### 11.11.3 12su Single Bus

This configuration of the 12su format uses only the TOP bus as a system bus and defines only the ground and power pins for the lower 9 block of connector. Power still takes 1 block and grounds still take 42 pins of the B row. Other pins may be used for I/O.

Connector using this configuration shall be keyed to prevent the insertion of dual bus configured boards.

Keying pattern to be determined.

### 11.11.4 12su No TDM

For non telecommunications environment, the additional I/O space that is being used can be made available by using the keying system that indicates the TDM bus pins are being used for user I/O.

Keying pattern to be determined.

### 11.11.5 18su system

This extension is being considered for more extensive backplane space needed for bus switching applications.