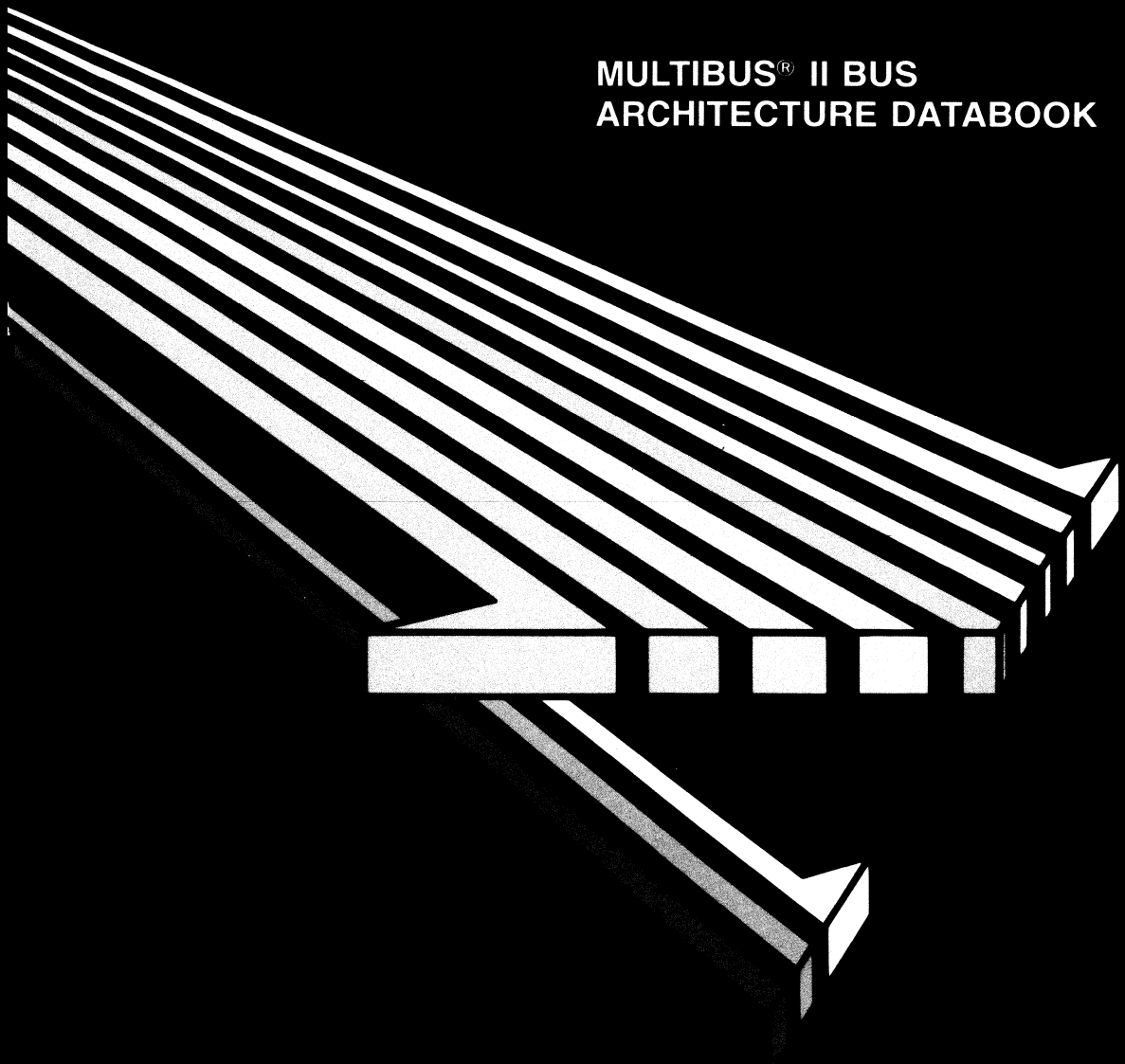
intel

# MULTIBUS® II BUS ARCHITECTURE DATABOOK



# Table of Contents

# INTRODUCTION TO THE MULTIBUS® II BUS ARCHITECTURE

## ARCHITECTURE OVERVIEW

The MULTIBUS® II Bus Architecture is an advanced, processor-independent, open system architecture suitable for a wide range of microprocessor-based designs. The multiple bus architecture includes three bus structures defined in this specification and compatibility with two existing MULTIBUS I/O busses. MULTIBUS II systems offer designers significant performance advantages and advanced features including a 32-bit parallel system bus with 40M byte/sec throughput, high-speed access to large amounts of off-board memory, a low-cost serial system bus, and effective multiprocessor support.

The MULTIBUS II Bus Architecture consists of the Parallel System (iPSB™) Bus, the Local Bus Extension (iLBX™ II Bus), the Serial System (iSSB™) Bus, and two buses carried over from the MULTIBUS I architecture — the iSBX™ I/O Expansion Bus and the MULTICHANNEL™ DMA (Direct Memory Access) I/O Bus (Figure 1-1). A common system interface which defines intermodule communication and data transfer protocols ties the buses together and allows designers to choose from several combinations of the five to meet specific application requirements.

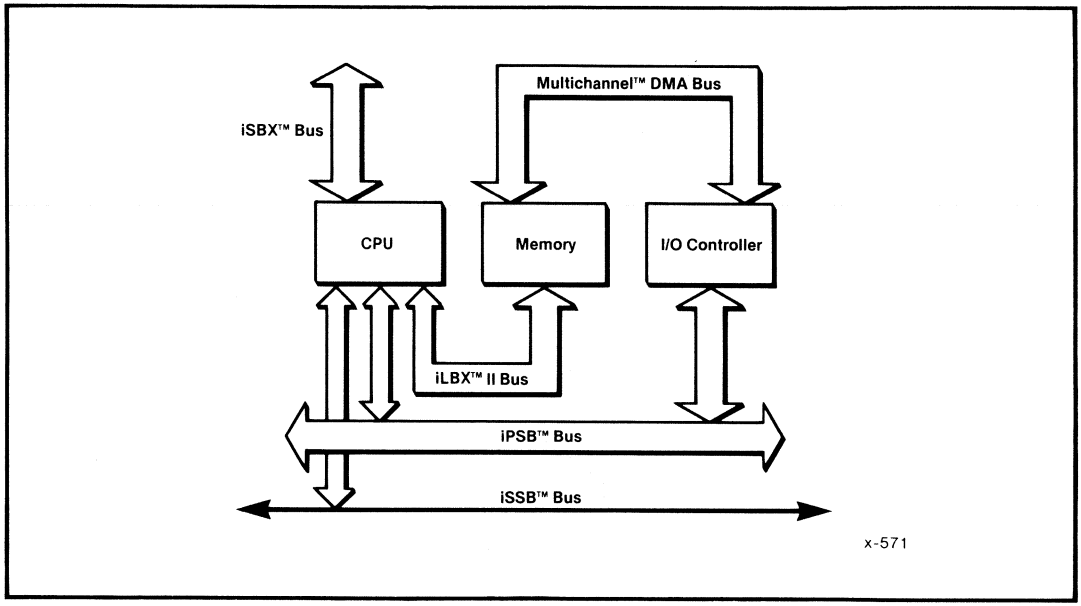



**Figure 1.1 MULTIBUS® II Bus Architecture (5 Buses)**

Because of its multiple bus structure and its ability to support 32-bit microprocessors, the MULTIBUS II Bus Architecture provides an evolutionary path to both future system expansion and future VLSI technology. The architectures supported by the MULTIBUS II system have migrated from the MULTIBUS I system.

### AN ANSWER TO OBSOLESCENCE

In the last decade, the avalanche of new microcomputer technology, especially VLSI, threatened to obsolete products almost before they went into production. To buffer user's from this onrush of technology, Intel helped develop standard interfaces. One of the most notable was the MULTIBUS I Bus Architecture.

MULTIBUS I became not only the industry standard, but it was designated the IEEE 796 standard as well. With the MULTIBUS I interface, user's could exploit the benefits of VLSI technology without having to pay a premium for new system design.

Other benefits from the use of standard interfaces in the MULTIBUS I architecture soon followed. As Intel's "Open Systems" design strategy gained wide acceptance, aftermarket support grew, providing multiple supply sources, wide product selections and competitive prices. Today, 200 companies manufacture over 1,200 MULTIBUS I compatible products.

The Open Systems approach was also demonstrated in Intel products which provide compatible products at three levels of integration: components, boards, and systems. This multilevel approach has enabled OEMs to adapt to new business environments and opportunities as VLSI technology expanded.

Standard interfaces for hardware and software combined with many MULTIBUS products (from both Intel and aftermarket suppliers) made it possible to configure new systems having unique requirements withminimal risk and investment.

## MULTIPLE BUS STRUCTURES

Microcomputer systems require many types of data movement: memory-to-CPU for instructions and data; CPU-to-CPU for interrupts and messages; I/O-to-memory for high speed data transfer; and CPU-to-I/O for direct control of I/O. In most systems, one general purpose bus can do all these three functions. However, for high performance systems, a general purpose bus lacks the total bandwidth required. And, in low-cost systems, the general purpose bus may be too costly.

A multiple bus structure provides specialized buses for specific critical functions. Four important advantages result.

1. The bandwidth of the general purpose bus is preserved, creating a "virtual bandwidth" for interprocessor communication and data movement.
2. The specialized bus does its function better than a general purpose bus.
3. Functions can be carried out in parallel on different busses.
4. Users can tailor their system performance and avoid unnecessary costs by selecting only those buses required for their application.

While the MULTIBUS I interface pioneered the multiple bus approach, the MULTIBUS II Bus Architecture refines it and extends its range. The new architecture offers five processor-independent buses which give system designers the ability to configure their systems for their needs today as well as meeting the future system requirements.

Each of the MULTIBUS II interfaces is fully compatible with the others. Thus, in general, it is simply a matter of choosing the appropriate bus or combination of buses to fit the exact needs of a particular application. Moreover, the standard interfaces mean designers can reconfigure the system as new requirements demand — or as VLSI technology provides improvements in microprocessor technology.

MULTIBUS I users can upgrade to the MULTIBUS II architecture as their needs expand to 32-bit capabilities, or as their 16-bit systems begin to require more sophisticated multiprocessing capability. Therefore, new users requiring high performance 16- or 32-bit data paths optimized for multiprocessing will find the multiple bus structure of the MULTIBUS II Bus Architecture ideal for advanced microprocessor design.

## MULTIPLE BUS STRUCTURES ENHANCE FUNCTIONAL PARTITIONING

Each multiple bus structure is tailored for a particular purpose. This is part of a design philosophy called "Functional Partitioning." Basically, functional partitioning is a modular design approach that requires breaking an overall problem into manageable pieces based on function. For example, typical microcomputer system functions are mass storage control, data processing, communications and graphics.

In typical functionally partitioned systems, data movement between modules is minimized. Requests for data movement are kept to a minimum, and critical real-time data should be kept in the local environment. Once the modules have been defined, they are implemented by optimizing each for its specific requirement. The MULTIBUS II Bus Architecture defines standard interfaces between each functional module and tailors each interface to its specific function.

For example, the Parallel System Bus (iPSB bus) is optimized for interprocessor communication and data movement. The Local Bus Extension, (iLBX II bus) is similarly optimized for very high speed execution. And the Serial System Bus (iSSB bus) is optimized for low-cost interprocessor communication.

Thus, the MULTIBUS II Bus Architecture provides the means to design a system optimized for performance with each bus serving a specialized function. Since each bus is also optimized for performance, functional partitioning of the modules is supported and enhanced.

## THE MULTIBUS® II BUSES

The MULTIBUS II Bus Architecture consists of the Parallel System Bus, the Local Bus Extension, the Serial System Bus, and two buses carried over from the MULTIBUS I architecture — the iSBX I/O Expansion Bus and the MULTICHANNEL DMA (Direct Memory Access) I/O Bus.

### Parallel System Bus (iPSB™ Bus)

The MULTIBUS II Parallel System Bus is a high-performance, general purpose bus that provides important data movement and inter-processor communication functions. Being general purpose in nature, the iPSB bus also supports arbitration, execution and I/O data movement and board configuration support.

The iPSB bus supports four address spaces: a 32-bit wide memory address space, a 16-bit I/O address space, a 16 or 32-bit message address space, and a 16-bit interconnect address space. Data is clocked at 10 MHz and can be up to 32 bits wide.

In addition, the iPSB bus incorporates features which:

- Provide high-performance data movement
- Enhance multiprocessor support
- Improve ease-of-use
- Increase system reliability

The following is a brief discussion of those features.

### High Performance

The Parallel System Bus has a burst transfer capability yielding a maximum sustained bandwidth of 40 megabytes/second. The burst is implemented as a single address cycle followed by multiple data transfers which maximize the bus bandwidth.

### Multiprocessor Support

Message Passing is another important attribute of the Parallel System Bus. This feature allows two bus agents (i.e., boards) to exchange information in blocks of data. The iPSB bus method of message passing provides a high performance facility for moving data from one functional module to another without having to worry about memory management or synchronization problems at the bus interface.

The iPSB bus supports multiple master processor agents. That is, the arbitration features can support up to 20 requests for the bus at the same time. This supports the functional partitioning approach and maximizes the effectiveness of each function.

### Ease of Use

The iPSB bus is a processor-independent general purpose system bus. As such, it supports 8-, 16-, and 32-bit processors, and is capable of transferring 8, 16, 24, or 32 bits of data. This attribute makes it extremely flexible.

General system-wide functions such as power-up/power-fail, bus time-out, system timing references, time-or-day clock, etc. are centralized into one function called the Central Services Module. In multiple agent systems, centralization of such functions reduces system cost, frees board area for other functions and is generally more efficient. The Central Services Module can be located on a unique board dedicated to that function or on any board in the system.

The Interconnect Address Space is provided for dynamic configuration of MULTIBUS II boards. This feature allows board options to be programmed or read from interconnect space which simplifies the configuration task and allows system resources to be identified. It also allows system-level diagnostics to access the results of board-level diagnostics for system confidence reports.

### Reliability

Since the Parallel System Bus is synchronous, the signals only have to be valid at specific intervals. Thus, noise-induced signals are not likely to be erroneously interpreted as data or control.

Reliability is also enhanced by such features as parity on transfers, DIN connectors and distributed ground pins. Byte parity is checked and generated for address, data and command lines. The pin and socket design of DIN connectors makes them extremely reliable. Also, the isolation of critical signals with adjacent ground pins reduces noise in nearby circuits on the backplane.

In summary, the Parallel System Bus is a full-functioning, general purpose system bus which enhances multiprocessor support by its unique message passing facility. It improves ease-of-use through its multiple data width facility. It improves ease-of-use through its multiple data width support and interconnect facilities. Finally, the iPSB bus increases system reliability by virtue of its synchronous nature, its parity on transfers, and its two-piece DIN connectors.

### Local Bus Extension (iLBX™ II Bus)

Multiple CPUs executing instructions in shared global memory can easily saturate the system bus, resulting in overall system performance degradation. An extension of the on-board processor bus provides the means to remove the processor execution functions from the general purpose system bus and extend local on-board processor bus provides the means to remove the processor execution functions from the general purpose system bus and extend local on-board performance capability to off-board memory resources.

The iLBX II Bus helps to maximize performance in MULTIBUS II systems by providing arbitration-free local memory expansion to 64 megabytes and a maximum bus clock rate of 12 MHz. It is a processor-independent bus that supports 8-, 16-, and 32-bit processors and up to six agents or boards.

The iLBX II Bus provides an alternate very high bandwidth path (48 megabyte/sec) to the processor's memory resources. In doing so, it reduces the processor's system bus bandwidth requirements by 60%-90%. Since it has been optimized for execution, it does not provide the functions of I/O or message passing.

A synchronous bus providing increased reliability, the iLBX II bus incorporates a number of advanced features which enhance system performance. For example, the iLBX II bus allows pipelining; the ability for the address portion of the following cycle to overlap on the data portion of the current cycle. Pipelining promotes the efficient use of the execution bus and helps make possible its higher bandwidth.

Another contributing factor to the higher bandwidth on the iLBX II bus is its ability to support block transfers. With one address phase, the bus can obtain multiple pieces of data, again a more efficient use of the execution bus.

As with the iLBX bus in the MULTIBUS I architecture, more than one iLBX II bus may exist in a MULTIBUS II system to optimize each processor's performance.

The iLBX II bus, then, is a high-speed, high-bandwidth execution bus which extends the on-board local bus to off-board memory resources. Its 48 megabytes/sec bandwidth, bus clock rate of 12 MHz and advanced features like pipelining and block transfers make it a high performance, low latency, reliable bus for today's multiprocessor architectures.

### Serial System Bus (iSSB™)

The Serial System Bus is very closely tied to some characteristics of the Parallel System Bus since it implements the message passing functions of the iPSB bus with a low-cost serial interconnect. That is, the iSSB bus is a low-cost alternative to the message interface on the iPSB bus.

Whereas the iPSB bus 32-bit wide parallel transfers and runs at 10 MHz, the iSSB bus is 1-bit wide and runs at 2 MHz. Thus, the performance is roughly 2 orders of magnitude less. However, the cost is also 2 orders of magnitude less.

There are two cost-reduction mechanisms involved. The first is the interconnect device. While the iPSB bus has 96 pins, the iSSB bus has only two.

The second consideration is packaging flexibility. The electrical requirements of the iPSB bus require it to be implemented in a very restrictive manner, with boards stacked and cables run to each device being controlled. On the other hand, the iSSB bus may be extended a distance of 10 meters, allowing boards to be scattered within a box or even be located in separate boxes. As VLSI devices become more capable, this will actually eliminate boards within a system by allowing the bus to be extended to the point of control. Thus, the iSSB bus has the ability to be physically distributed since it is no longer restricted to the backplane. However, it will likely remain

inside the physical packaging of the system, even though it may extend short distances to connect a modular package.

As VLSI technology continues to shrink more and more functions into a single piece of silicon, the printed circuit board area needed to implement a function will be decreased. In fact, today the interface to the iSSB bus can be implemented with a single chip. This level of integration will provide the lowest possible system cost. Further, the controller function found in contemporary systems will migrate to the printed circuit boards of the peripherals. The iSSB bus, then, allows reduced interconnect costs and physical distribution of system modules.

While taking advantage of the cost and distribution of a serial media,the iSSB bus allows functional modules to communicate with an identical software message passing interface to that of the Parallel System Bus. This means that modules that communicate via message passing on the iPSB bus can migrate easily to an iSSB bus with only minimal software changes.

Whereas the iPSB bus in combination with VLSI advances will offer increased capabilities at the same cost, the iSSB bus in combination with VLSI technology will offer the same capabilities at a lower cost.

### MULTICHANNEL™ DMA I/O Bus

Carried over from the MULTIBUS I Bus Architecture, the MULTICHANNEL I/O Bus solves the problem of high-speed I/O data to and from physically distributed custom peripherals such as mass storage devices or graphics display systems. The MULTICHANNEL bus allows high-speed (8 megabytes/sec) block transfers of data over the data path between such peripherals and single board computers.

The MULTICHANNEL bus provides a standardized I/O interface with full speed operation at up to 15 meters with a simple asynchronous protocol. It supports up to 16 devices, both 8- and 16-bit and provides 16 megabyte memory or register address space per device. Typical uses of this bus include computer graphics, specialized peripheral control, data acquisition and high-speed MULTIBUS system-to-system communication.

### iSBX™ I/O Expansion Bus

Also a carryover from the MULTIBUS I Bus Architecture, the iSBX I/O expansion bus allows incremental on-board system expansion through the use of small iSBX MULTIMODULE boards. Currently, iSBX boards add capability in the areas of parallel I/O, serial I/O graphics, bubbles and advanced mathematics functions. All iSBX boards afford system expansion without the additional cost of adding another full expansion board.

The expansion bus allows users to customize their single board computers to individual applications in response to latest VLSI technology. Since they are able to buy exactly the capabilities needed, both system cost and system size are kept to a minimum.

## BUILDING A COMPUTER SYSTEM FOR THE FUTURE

The advantages of the multiple bus approach of MULTIBUS II Bus Architecture is easily demonstrated by a simple system that evolves over time to a family of products. Suppose the basic requirement is for moderate CPU power, variable sizes of RAM depending on system usage, and a variety of simple I/O devices.

The initial system may contain a sixteen bit microprocessor like the 8086 which provides the CPU requirements (Figure 1-2). Considerable RAM can be placed on the same board as the CPU: up to one megabyte with 256K DRAMS and a RAM expansion MULTIMODULE which mechanically does not require a second slot. The CPU board could still accommodate extensive I/O facilities such as serial, parallel, or analog I/O, as well as iSBX connectors for low-cost I/O additions to the base board.

Additional RAM can be added with memory boards on the Parallel System Bus with capacities up to 2 megabytes with 256K DRAMS and features such as parity or ECC. Additional I/O can be added on boards accessed via the Parallel System Bus with software compatibility with the on-board I/O. In both cases, the I/O is addressed via the I/O space of the 8086 since the iPSB bus supports an equivalent I/O space.

Finally, diagnostic and system start-up code can be written which utilizes the interconnect space to dynamically determine the features of the boards in the system. Thus, customization and incremental enhancement of the computer for each customer is simplified, resulting in greater customer satisfaction.

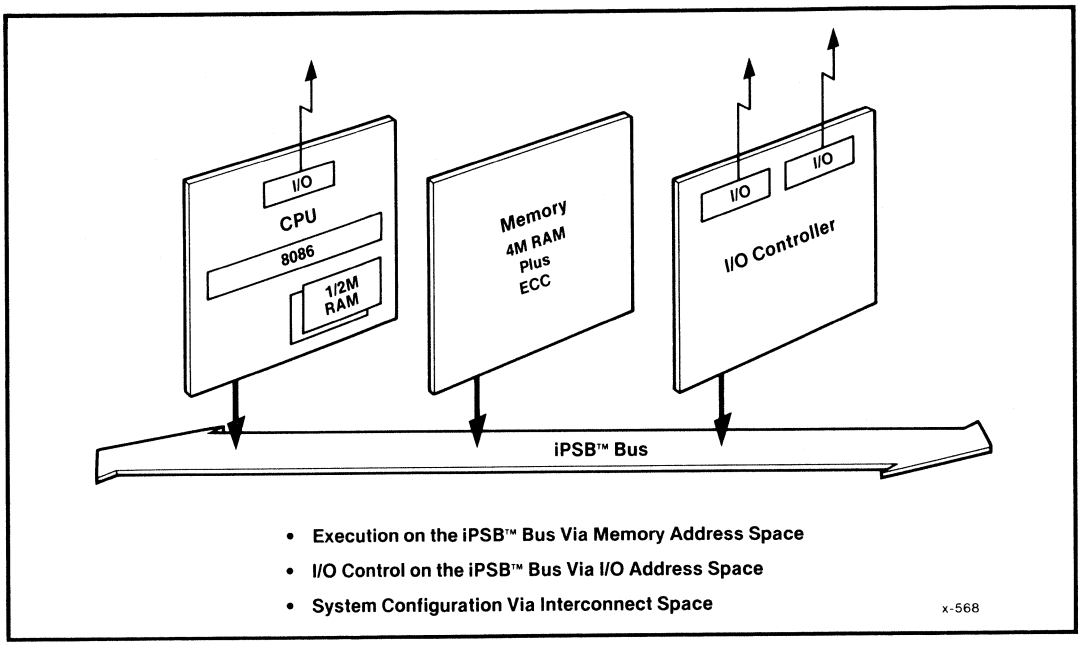



**Figure 1-2. System Providing Basic Requirements**

### Increased CPU Demand

In time the computer will be expected to perform increased workloads as competitive pressures demand productivity improvements from the user. The processor board could be designed to use a 80286 processor (Figure 1-3). Alternately, a second processor board could be added.

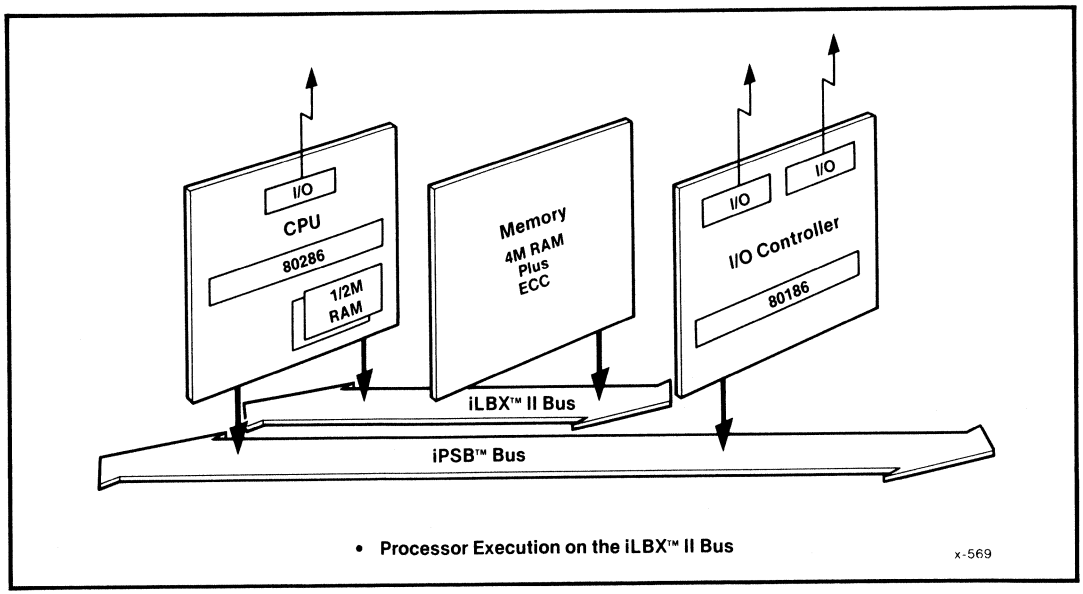



**Figure 1-3. System With Increased CPU Requirements**

To save system bus bandwidth and expand performance with large memory sizes, the new processor board and memory could be designed with iLBX II bus interfaces. The two board set, connected via the iLBX II bus, will function as a virtual single board, independent of any other sets of iLBX II connected boards.

### Increased Data Movement

With an increased demand for services of the I/O, preprocessing of the data will be required to reduce the information flow to only the relevant data. This can be accomplished by adding processor power to the I/O interfaces. As standard interfaces are desirable to preserve the software investment, the simple I/O address mapping of the example would be replaced with a higher level procedural interface. This higher level would be provided by software for on-board I/O. The off-board I/O, however, is more complicated.

The data could be moved between processors via shared memory, but this would require the addition of a memory board or a dual port facility for shared memory. The MULTIBUS II Message Passing Facility is intended to simplify the design and configuration of such systems while maintaining an equivalent performance of the more tightly coupled design (Figure 1-4).

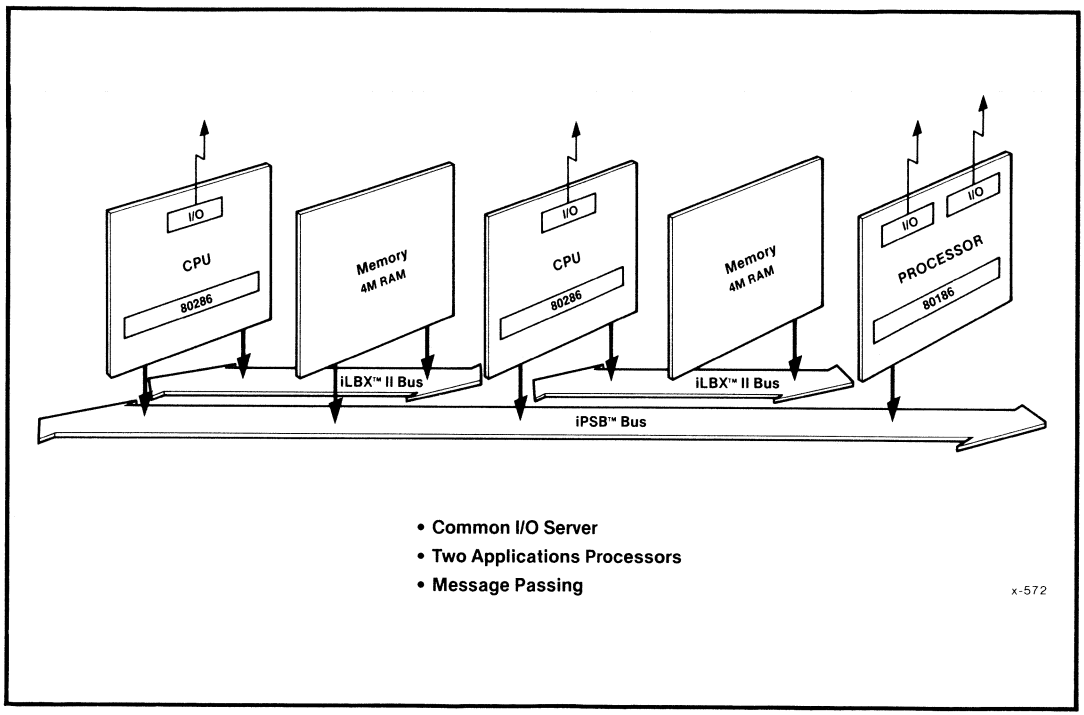



**Figure 1-4. Enhanced Support of Multiple Processors**

### Lower Cost requirements

With a successful system, competition will attempt to undermine the leader's market share with lower cost. Furthermore, a low-cost system design is often required to achieve design wins that lead to future product sales of more costly systems. The MULTIBUS II Serial System Bus allows a low-cost implementation of the message-passing facility of the Parallel System Bus with complete software compatibility. Thus, the software investment is protected and leveraged in low-cost designs based on highly integrated processors such as the 80186 (Figure 1-5).

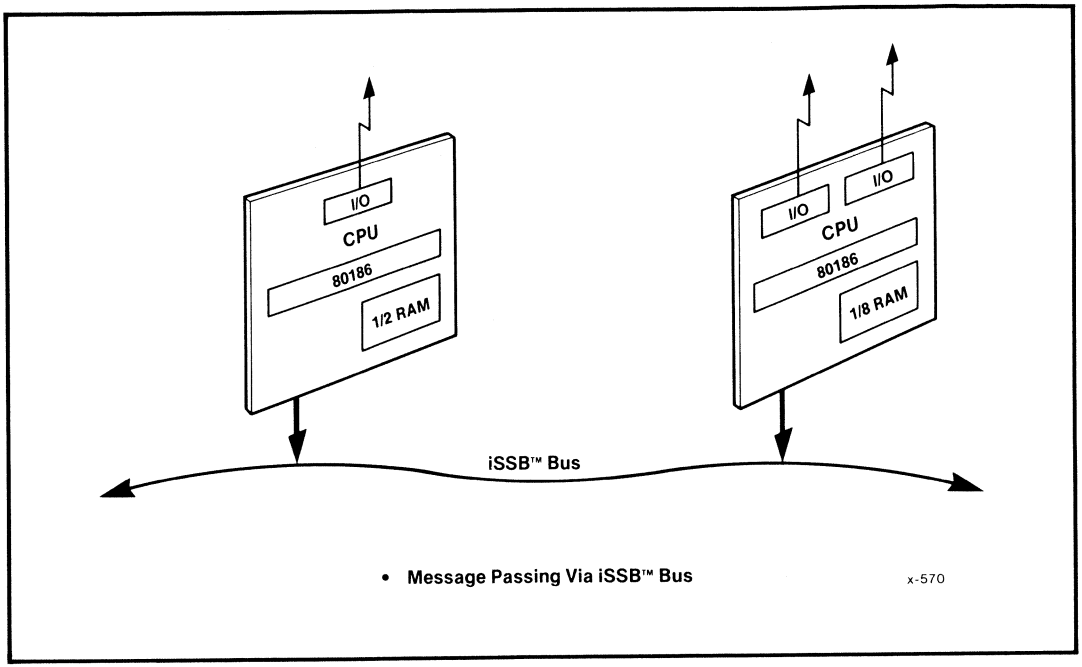



**Figure 1-5. Low-Cost System Design**

## SUMMARY

This overview has attempted to present the characteristics of the MULTIBUS II Bus Architecture, an advanced open system multiple bus architecture. Its intent has also been to demonstrate the attributes of the MULTIBUS II buses: the 32-bit Parallel System Bus with its 40 megabytes/sec throughput and effective support of multiple processors; the iLBX II bus with its high-speed access to large amounts of off-board memory; and the Serial System Bus, a low-cost alternative to the message-passing facilities of the Parallel System Bus.

Individually, the buses represent significant advances in bus design. Together, they represent an evolutionary path to future VLSI technology.

intel®

# MULTIBUS® II GLOSSARY

**Agent:**

A physical unit which has an interface to the system bus. There are typically one agent per board.

**Module:**

A basic functional unit. An agent on the MULTIBUS® II interface may be comprised of one or more modules, each of which perform a particular function.

**Bus cycle:**

The basic unit of activity on the bus whereby digital signals effect the transfer of data across the interface by means of a sequence of control signals and an integral number of bus clock cycles.

**Arbitration cycle:**

The bus cycle in which an agent attempts to gain exclusive control of the bus. The winner of the arbitration is called the bus owner.

Resolution phase:

The initial phase of an arbitration cycle in which all agents requesting access to the bus drive an arbitration identity onto the arbitration group signals. Agents mutually resolve requests and allow the winning agent to gain access to the bus.

Acquisition phase:

One agent at a time (the winner of the resolution phase) enters the acquisition phase. The agent receiving access to the bus is referred to as the bus owner or the requesting agent and immediately begins conducting transfer cycles.

**Transfer cycle:**

The bus cycle in which the bus owner transfers data on the bus. During the transfer cycle, the bus owner is called the requesting agent. The agents addressed by the requesting agent are called replying agents.

Request phase:

The initial phase of a transfer cycle in which the bus owner requests a data transfer by placing command and address information on the bus.

Reply phase:

The final phase of a transfer cycle in which another agent or module replies to the request from the requesting agent. The phase consists of one or more consecutive data and/or status transfers on the bus.

**Exception cycle:**

The bus cycle in which an agent signals a bus error condition. Exception cycles terminate all bus activity and allow for the bus to stablize before more bus cycles begin.

Signal phase:

The initial phase of the exception cycle in which an agent places an exception error indication on the bus. This causes termination of all arbitration and transfer cycles currently in progress. All agents receive the error indication.

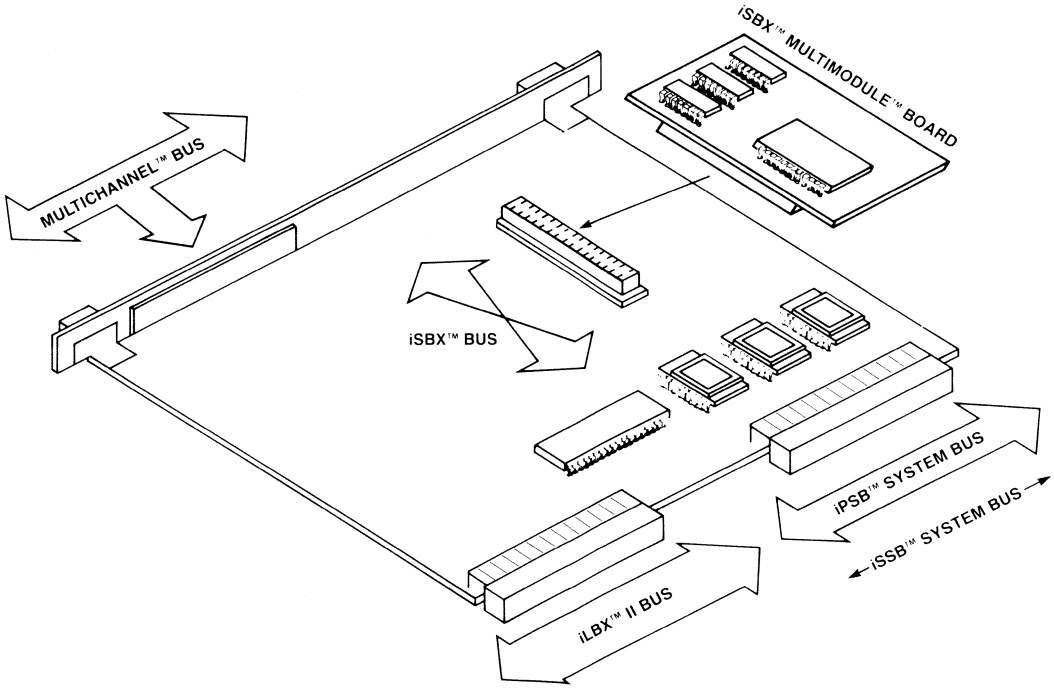
Recovery phase:

The final phase of an exception cycle in which the bus is allowed to sit idle for a defined amount of time. The idle time allows the bus to stabilize before more bus cycles begin.

# MULTIBUS® II
# iPSB™ Parallel System Bus

- **Very high bandwidth —**
  - **— 40 megabytes/sec using burst transfers**
  - **— 20 megabytes/sec with single cycles**
- **4 gigabyte (32-bit) addressing**
- **8-, 16-, 24-, and 32-bit data transfers over a 32-bit path**
- **Pin-efficient multiplexed structure**
- **Reliable synchronous clocking at 10 megahertz with full handshaking for data**
- **Distributed arbitration with up to 20 bus masters**
- **Full parity protection for data transfer integrity**
- **Message passing facility for inter-module communication**
- **Interconnect facility for software identification and configuration of boards**
- **Industry standard Eurocard form factors — 233mm x 220mm and 100mm x 220mm**

The MULTIBUS® II iPSB™ Parallel System Bus is the foundation of the MULTIBUS II Bus Architecture. It is a general-purpose, processor independent structure which fully supports 8-, 16-, and 32-bit microprocessors. This very high bandwidth structure is defined on a single 96-pin IEC 603-2 (DIN) connector. All data movement functions required in a microcomputer system are defined including such advanced functions as an integrated message passing protocol and an interconnect facility which allows software to address a board by its slot position for software-based board identification and configuration.




**MULTIBUS® II Physical Diagram**

## FUNCTIONAL DESCRIPTION

### Architectural Overview

The MULTIBUS II iPSB Parallel System Bus is the foundation of the MULTIBUS II bus architecture (see Figure 1). As a system bus, it is a very high bandwidth (40 megabytes/sec) bus optimized for inter-module communication, however, it also defines the complete set of basic bus functions required in a microcomputer system: memory accesses for execution or data, accesses to I/O for control of I/O functions, plus inter-module signalling. These basic functions are supplemented with additional functions supporting geographical (by-slot) addressing and an integral message passing protocol.

Geographical addressing allows addressing of individual boards via their physical position in the backplane. Software can determine what boards are being used and configure itself appropriately. Software also can configure the hardware characteristics of the board (eg. the starting address of a memory board). This can substantially reduce or even eliminate hardware jumper options and DIP switches for board configuration. Geographical addressing is a function of the interconnect address space.

MULTIBUS II's integral message passing protocol defines a standard and uniform way for modules to communicate over either the iPSB or iSSB™ buses. Integrating the protocol at the bus structure level lets the designer provide hardware support to increase system inter-module communication performance and opens the door for VLSI solutions. Standardizing the interface ensures a uniform software interface so that users can take advantage of new advances in technology without having to rewrite software.

### Structural Features

#### Overview

The iPSB bus structure is a processor-independent general-purpose bus designed to support 8-, 16-, and 32-bit processors. It is designed to operate at a maximum bandwidth of 40 megabytes/sec while using off-the-shelf components.

Special attention has been given to how the bus structure, both electrically and mechanically, impacts system reliability. Synchronous sampling of all bus signal lines assures good immunity from crosstalk and noise. Full byte parity generation and checking protects all transfers on the bus to ensure that any bus error is detected. And a signal quality on the bus is excellent due to the large number of interlaced ground lines. Mechanically, the iPSB bus is defined on a two-piece 96-pin IEC 603-2 to ensure good connector reliability.

#### Multiplexing

The iPSB bus is highly multiplexed. The 32-bit address and data paths are multiplexed and the nine system control lines have different uses depending upon the phase of the transfer cycle. The six arbitration lines also serve dual purposes between system initialization and normal operation.

This multiplexed structure has several benefits. First of all, the entire 32-bit iPSB bus is defined on a single connector. This allows a full 32-bit iPSB bus interface on even the smaller, single connector, form factor board. This opens the possibility of low cost 32-bit systems. Multiplexing also reduces by half the number of high current drivers required for the interface which significantly reduces a board's current requirements. The routing of signal lines between the bus interface and connector is simplified.

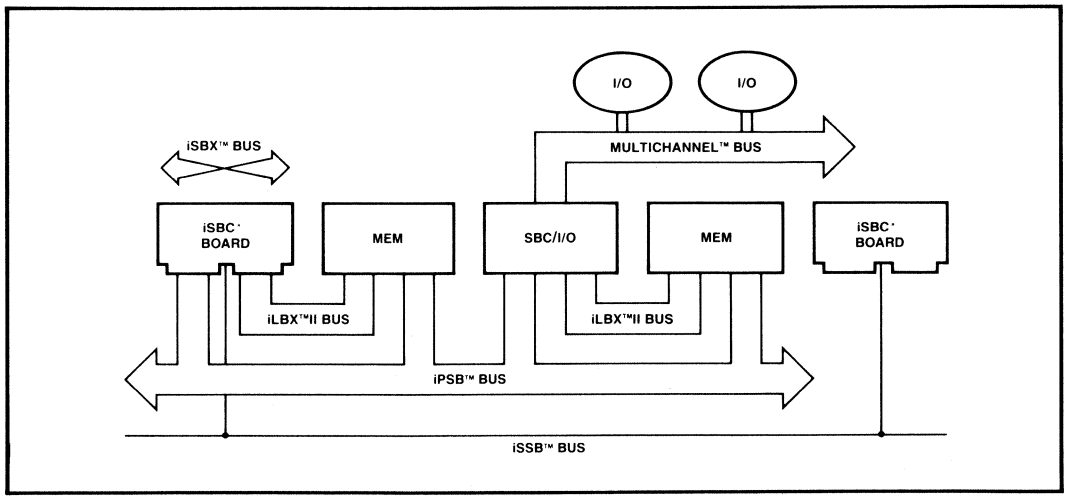



**Figure 1. MULTIBUS® II Bus Architecture**

### Bus Errors and Exceptions

The iPSB bus defines a complete set of bus error reporting mechanisms. Serious errors such as a parity error or the failure of a module to complete the data handshake, are flagged on unique bus signal lines and are seen by all modules on the bus. These errors induce a recovery time in which the bus is allowed to stabilize before further transfer cycles may begin.

The iPSB bus also provides mechanisms for signaling less serious operational errors. Operational errors, such as attempting to perform a 32-bit access to a 8-bit device or writing to read-only memory, are signaled as agent exceptions. These exceptions may induce retry operations by an intelligent bus interface or may be passed to the offending processor as errors.

### Interconnect Address Space

The ability to address a board by its physical position in the backplane is also supported in the iPSB bus. This facility allows board manufacturers to code such items as their vendor number, board type, board revision number, and serial number on the board. This information is available to the system software. This facility is defined in the iPSB bus interconnect address space.

Aside from this read-only information, the interconnect space allows write operations to support board configuration and diagnostics under software control. This facility can help reduce or eliminate hardware-based jumper options and DIP switches.

### Interrupts

The iPSB bus supports up to 255 distinct interrupt sources and 255 interrupt destinations. Rather than the use of the traditional method of dedicated interrupt signal lines on the bus, the iPSB bus defines a special bus cycle to convey interrupt information. This special bus cycle (actually part of the message passing protocol discussed below) redefines the meaning of the address; instead of a byte location in memory for example, 16 of the 32 lines encode 8 bits for the source module generating the interrupt and 8 bits for the destination module to service the interrupt.

This technique overcomes the significant problem of interrupt configuration found in traditional buses. Dedicated lines usually imply that only one particular destination can service one particular interrupt source. If an interrupt source wishes to target some interrupts to one destination and some to a different destination, separate bus interrupt lines are required for each destination. This can quickly consume all dedicated interrupt lines in even a moderate size system.

Using interrupt bus cycles with embedded source and destination module addressing removes the need for dedicated interrupt lines at the same time it allows any interrupt source to signal any interrupt destination.

### Message Passing

With the trend in microcomputer systems toward multiprocessing, it is important to provide the facilities and mechanisms to lend support for inter-module communication. The iPSB bus includes such mechanisms and defines the protocol for greatly enhanced performance in inter-module communication. This protocol is called MULTIBUS II Message Passing.

Most multiprocessor systems use either a "pass by reference" or a "pass by value" protocol for intermodule communication. In the "pass by reference" case, the two modules share a common memory resource and pass pointers or tokens to extend addressability of a desired data structure to the other module. In "pass by value", the modules exchange a copy of the desired data structure. Each of these protocols has a set of advantages and disadvantages associated with performance, data security, extensibility to additional modules, and ease of use.

MULTIBUS II Message Passing takes the best of both methods and lends hardware support. Message passing uses a hardware "pass by value" interface that gives the performance of a "pass by reference" system. It replaces the software module used by the "pass by value" method with a specialized message passing interface. The processor "passes by reference" the reference to the data structure to the message passing interface. This interface communicates with the destination module's message passing interface to transfer the data without processor intervention. This data transfer is performed in the message address space. This is illustrated in Figure 2. (In many ways, it is helpful to think of the two communication message passing interfaces as a distributed, smart, DMA controller.)

There are several significant benefits to this approach. First of all, the message passing interfaces can take advantage of the full capabilities of the bus (ie, 32-bit data and burst transfer) independent of the type or nature of the controlling processor. Even 8-bit processor or I/O boards can take full advantage of the bus. This means significantly higher inter-module communication performance over a completely software-base method. Another benefit is the elimination of any shared memory. Dual-ported memory structures are no longer needed nor are global memory boards. The other primary benefit is that MULTIBUS II message passing presents a uniform software interface for all modules. Modules can be replaced with new modules containing newer technology (e.g. moving from a single density to a double density disk controller) without any software changes required in the controlling module. This makes it easy for users to integrate new technology without the problem of completely rewriting the driver software.

### Central Services Module

The iPSB bus specification defines the central system functions as the Central Services Module (CSM). The minimal set of functions are: clock generation, power-down and reset, time-out, and assignment of slot IDs. Collecting these functions in a single module improves overall board area utilization, since the functions are not duplicated on every board and then only used on one. The system designer is free to implement the CSM on a separate board or to include the functions as just one of several modules on another board.

### Bus Cycle Overview

The iPSB bus defines three types of bus cycles: arbitration, transfer, and exception cycles. Each cycle is made up of one or more phases. Figure 3 illustrates the relationship among these cycles and phases.

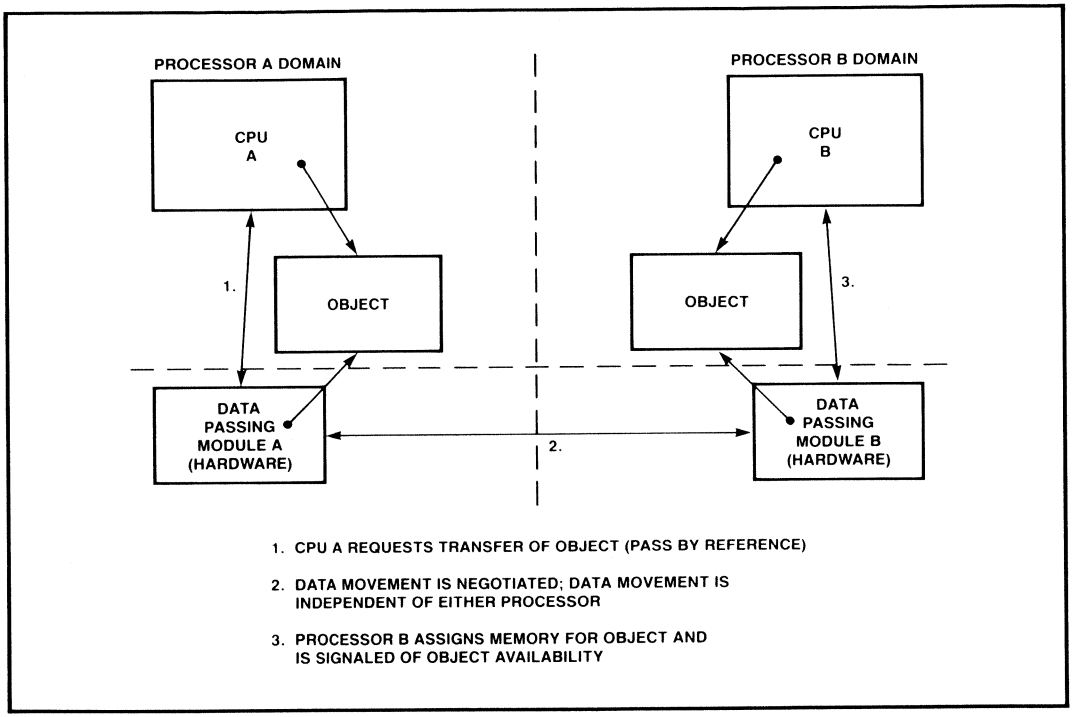



**Figure 2. MULTIBUS® II Message Passing**

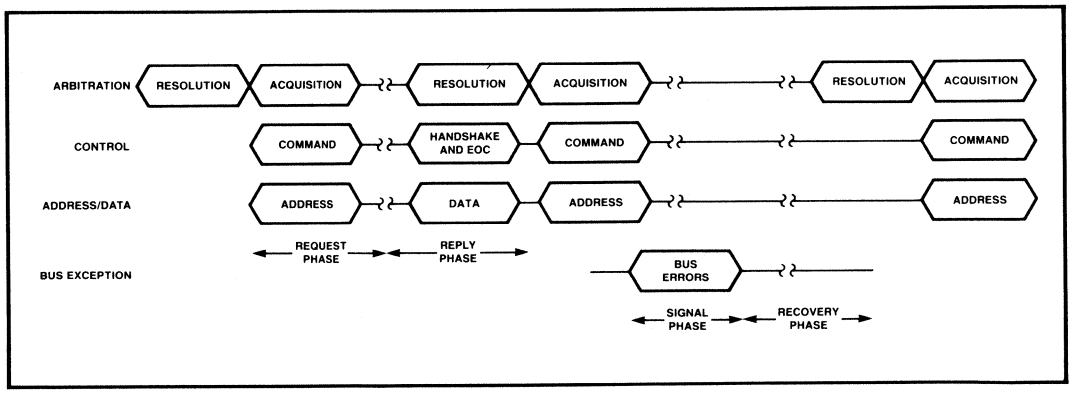



**Figure 3. Bus Cycle Relationships**

### Arbitration Cycle

The arbitration cycle is made up of a resolution phase and an acquisition phase. The resolution phase is the time-period in which all agents collectively arbitrate for access rights to the bus. Depending on the arbitration algorithm, the agents decide among themselves which of them is going to control the bus after the current bus owner is done. This arbitration method is referred to as self-selecting since the agents decide ownership among themselves.

The agent that wins the arbitration and obtains access rights to the bus begins the acquisition phase; that agent becomes the bus owner. This agent begins its transfer cycle and holds the arbitration logic in the resolution phase (resolving for the next access rights) until the transfer cycle is completed.

### Transfer Cycle

Starting the transfer cycle is the request phase. In this phase, the bus owner (requesting agent) places address and command information on the bus. This information defines the replying agent(s), the type of operation, and the type of address space. The request phase lasts one bus clock cycle.

The reply phase starts immediately after the request phase, in which the replying agent(s) satisfy the request. During this phase, the requesting and replying agents engage in a handshake that synchronizes the data transfer sequence. The reply phase can contain one or more data cycles. The final data transfer is signaled by the requesting agent. During this final transfer, the requesting agent releases ownership of the bus allowing the new bus owner to use the bus immediately. Note how the transfer cycle overlaps the resolution phase of the arbitration cycle to minimize bus dead time.

### Exception Cycle

If an agent detects an error during a transfer cycle, it immediately begins an exception cycle. The exception cycle terminates any arbitration cycles and transfer cycles in progress. The exception cycle starts with the signal phase in which the detecting agent activates one of the bus' error lines. This notifies all agents of the problem causing them to terminate any arbitration or transfer cycles. Next the recovery phase begins. During this phase, all agents idle; this allows the bus a fixed amount of idle-time to stabilize before resuming normal operation.

## Signal Groups

### Overview

The iPSB bus contains five groups of signals, Figure 4, over which the requesting and replying agents can enact the protocol. An asterisk following the signal name indicates that that particular signal or group of signals are active when at their electrical low.

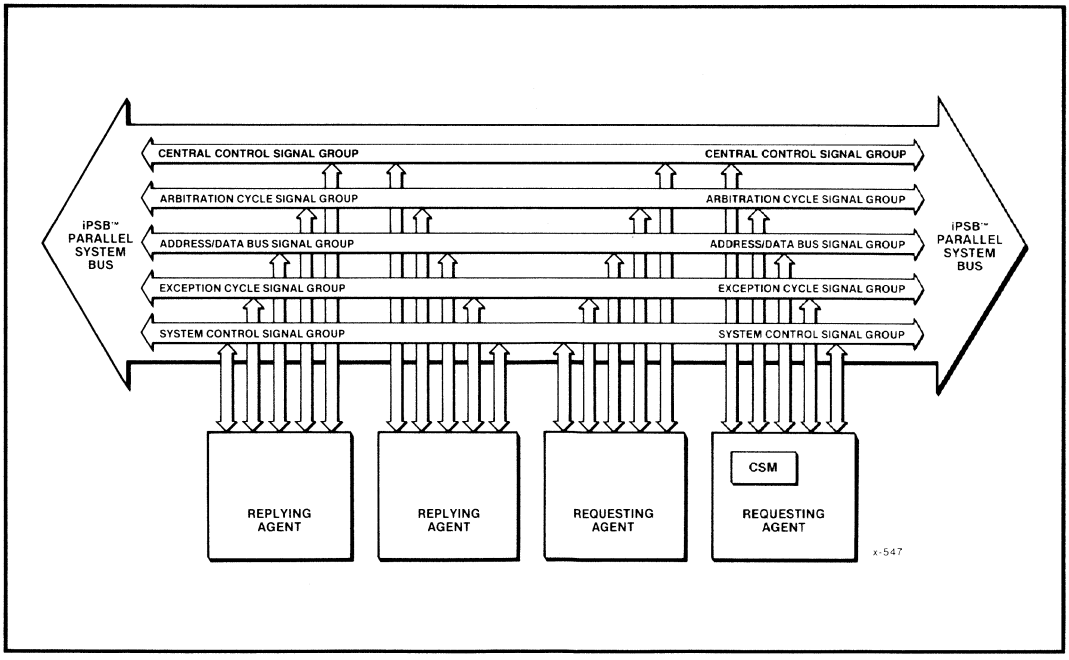



**Figure 4. iPSB™ Bus Signal Groups**

### Arbitration Group

The arbitration signals on the iPSB bus determine which agent gains exclusive access to the bus (which agent is the bus owner). All requesting agents that require access to the bus resources must arbitrate for use of the bus. On being granted bus ownership, an agent begins using the address/data lines to perform a transfer cycle. There are seven signals in the arbitration group: BREQ* and ARB5* through ARB0*.

**BREQ*** (Bus Request) is an OR-tied signal which is bussed on the backplane. All agents that require access to the bus assert the BREQ* signal.

A particular agent's arbitration ID number is coded on lines **ARB4* through ARB0*** (Arbitration). An agent requiring use of the iPSB bus asserts BREQ* and drives its arbitration ID onto the OR-tied ARB lines. The ARB5* line selects one of two arbitration algorithms: fairness or high priority.

### Address/Data Bus Group

This signal group contains the lines used to transfer the address and data information plus their respective byte parity lines. The **AD31* through AD0*** (Address/Data) lines are multiplexed and serve a dual purpose depending upon the phase of the transfer cycle.

During the request phase, they contain the address for the ensuing transfer. This address refers to the byte location for memory and I/O spaces, a processing agent module in message space, and a board slot location in interconnect space. The requesting agent drives these lines during the request phase.

During the reply phase, they contain either eight, sixteen, twenty-four, or thirty-two bits of data. They are driven by the requesting agent for write transfers and by the replying agent for read transfers.

The **PAR3* through PAR0*** (Parity) lines are the byte parity lines associated with the respective bytes of the AD lines. They form even parity with their respective address/data byte.

### System Control Signal

The transfer signal group consists of ten signals, SC9* through SC0* (System Control). Agents use these signals to define commands or to report status, depending on the phase of the transfer cycle.

During the request phase, the requesting agent drives SC9* through SC0*. The SC lines provide command information to the replying agent(s). During the reply phase, the requesting agent drives SC9* and SC3* through SC0* with its handshake and additional control information. The replying agent drives the remainder with its handshake and status. Table 1 lists the request and reply phase functions for this group.

### Exception Signal Group

The iPSB bus provides a group of two signals for passing indications of exception errors to all agents: **BUSERR*** (Bus Error), and **TIMOUT*** (Time-out).

An agent activates BUSERR* to indicate its detection of a data integrity problem during a transfer. Parity errors on the AD or SC lines and the detection of uncorrectable errors in the memory subsystem are typical of errors signaled on BUSERR. Any agent detecting such errors must signal BUSERR* and all agents must receive BUSERR*.

TIMOUT* is signaled by the CSM whenever it detects the failure of a module to complete a handshake. TIMOUT* is received by all agents on the bus.

### Central Control Group

The system control group provides status concerning the operating state of the entire iPSB bus environment.

**Table 1. System Control Definition**

| Signal | Function | |
|---|---|---|
| | Request Phase | Reply Phase |
| SC0 | Request Phase | Request Phase |
| SC1 | Lock | Lock |
| SC2 | Data Width 0 | End-of-Cycle |
| SC3 | Data Width 1 | Requesting Agent Ready |
| SC4 | Address Space 0 | Replying Agent Ready |
| SC5 | Address Space 1 | Agent Error 0 |
| SC6 | Read/Write | Agent Error 1 |
| SC7 | Reserved | Agent Error 2 |
| SC8 | Parity (SC7-4) | Parity (SC7-4) |
| SC9 | Parity (SC3-0) | Parity (SC3-0) |

It consists of seven signals plus the power and ground lines.

The **RST*** (Reset) signal is a system-level initialization signal sent to all agents by the CSM.

The **RSTNC*** (Reset Not Complete) signal is an OR-tied line driven by any agent whose internal initialization sequence is longer than that provided by the RST* signal itself. Due to its OR-tying, RSTNC* remains active until every agent has completed its initialization sequence. Agents cannot perform bus transfer cycles until RSTNC* is inactive.

The CSM provides a **DCLOW** (DC Power Low) signal to all agents as a warning of an imminent loss of DC power. DCLOW is typically generated from a signal supplied by the system power supply on the loss of AC power. Any agent needing to preserve state information in battery backed-up resources should do so upon receiving an active DCLOW.

Accompanying DCLOW for power-down sequencing is the **PROT*** (Protect) signal. The CSM drives PROT* active a short time after it activates DCLOW to inform all bus interfaces to ignore any transitions on the bus as power is lost.

The **BCLK*** (Bus Clock) and **CCLK*** (Constant Clock) signals are supplied by the CSM to all agents. Agents use the BCLK to drive the arbitration and timing state machines on the iPSB bus. The active going edge of BCLK* provides all system timing references. The CCLK* is an auxiliary clock at twice the frequency of BCLK.

An agent uses its **LACHn*** (ID Latch) signal to save the slot ID it receives from the CSM at RST time via the ARB4* through ARB0* lines. The ID latch signal is called LACHn* where the "n" is the card slot to which the ID is assigned. At each card slot, the LACHn* signal is connected to the AD line of the same number. As an example, card slot 7 has a LACH7* signal that is connected to AD7*.

When RST is active, the CSM sends successive slot ID's (0 through 19) on the ARB4* through ARB0* lines while activating the corresponding AD line. Agents know when the ARB lines contain the correct slot number when they see their LACHn line go active.

### Power

System power supplied in the iPSB connector includes +5 volts, +12 volts, −12 volts, and facilities for +5 volt battery back-up. Also defined are numerous ground lines some of which are interlaced throughout the connector. The iPSB bus power arrangement conforms to the proposed IEC standard for power railing in IEC 603-2 connectors.

## iPSB Bus Protocol

### Overview

In the MULTIBUS II specification, both timing diagrams and state-flow diagrams describe the iPSB bus protocol. The state-flow diagrams present the lowest-level and most rigorous definition while the timing diagrams help conceptual understanding. For the purposes of this data book, only the timing diagram description is used.

### Arbitration Cycle

An agent that wishes to transfer data on the iPSB bus must begin by performing an arbitration cycle. The cycle performs two functions: first, it gives all agents the opportunity to be granted access to the bus, and second, it eliminates the possibility of more than one agent trying to transfer data on the bus at any one instant. In the case where more than one agent requests access to the bus at the same instant, the arbitration cycle grants access to the agents based upon one of two arbitration algorithms: normal or high priority.

Normal priority mode provides "fairness" or "no starvation", which each agent has an equal opportunity to grant access to the bus. For example, assume all agents request the bus at the same instant. In the normal priority mode, each agent is granted the bus, one by one, until all requests have been serviced. If an already serviced agent desires to use the bus again before all of the original agents are serviced, it will wait until all of original requesting agents have their requests granted. This "round-robin" granting of access ensures that any agent requesting the bus will eventually get it.

The high priority mode allows an agent with high priority to force its way into the arbitration and grant the bus before agents with lesser priority. This means that a high priority agent gets access to the bus quickly, however it can also consume much of the bus that agents with lesser priority never gain access; they will "starve".

At reset, the CSM supplies each agent with its slot ID and its arbitration ID. An agent making a normal priority request activates BREQ*, holds ARB5* inactive, and drives its arbitration ID onto ARB4* through ARB0*. If the ARB lines hold its ID after a specified time (3 bus clocks), this agent won the arbitration and can use the bus once any ongoing transfer completes. However, if the ARB lines do not match its ID (after all, other agents might be also requesting the bus and driving the ARB lines), another agent won the arbitration. The losing agent removes its ID and waits for the next resolution phase before trying again.

An agent makes a high priority request by activating BREQ*, holding ARB5* active (ARB5* selects the arbitration mode), and driving its arbitration ID onto the ARB lines. The high priority algorithm requires that

when a high priority request enters during an arbitration cycle, the request immediately enters the next resolution phase rather than waiting for the next arbitration cycle as do normal priority requests. ARB5* being active causes the other requesting agents to remove their requests guaranteeing the high priority agent access to the bus before any simultaneous normal priority requests. When more than one agent simultaneously makes a high priority request, the agent with the higher priority (lower numerical value) arbitration ID will go first. Figure 5 illustrates the logic required to implement the iPSB bus arbitration. With either priority mode, once an agent owns the bus, it can perform any number of transfer cycles until forced off by arbitration. This characteristic of the arbitration algorithms is called "bus parking".

### Transfer Cycle

Transfer cycles consist of two phases: request and reply. For illustration, an example of a access read cycle is shown in Figure 6. During the request phase, the bus owner (requesting agent) uses the transfer cycle signal group (SC lines) to notify the replying agent of the address space (memory, I/O, interconnect, or message), the data width (8-, 16-, 24- or 32-bit), and whether the cycle is read or write. The AD lines contain the desired address for the selected address space. Replying agents know the SC lines contain this request information by the requesting agent activating SC0* (Request Phase). The request phase lasts one clock cycle. All potential replying agents use the request phase to determine whether they contain the addressed resource.

The reply phase starts immediately following the request phase. During this phase the agent with the addressed resource (replying agent) and the requesting agent exchange data and status. Both the requesting and replying agent must agree that the data on the AD lines and the status on the appropriate SC lines are valid via the RQRDY (Requesting agent ready — SC3*)

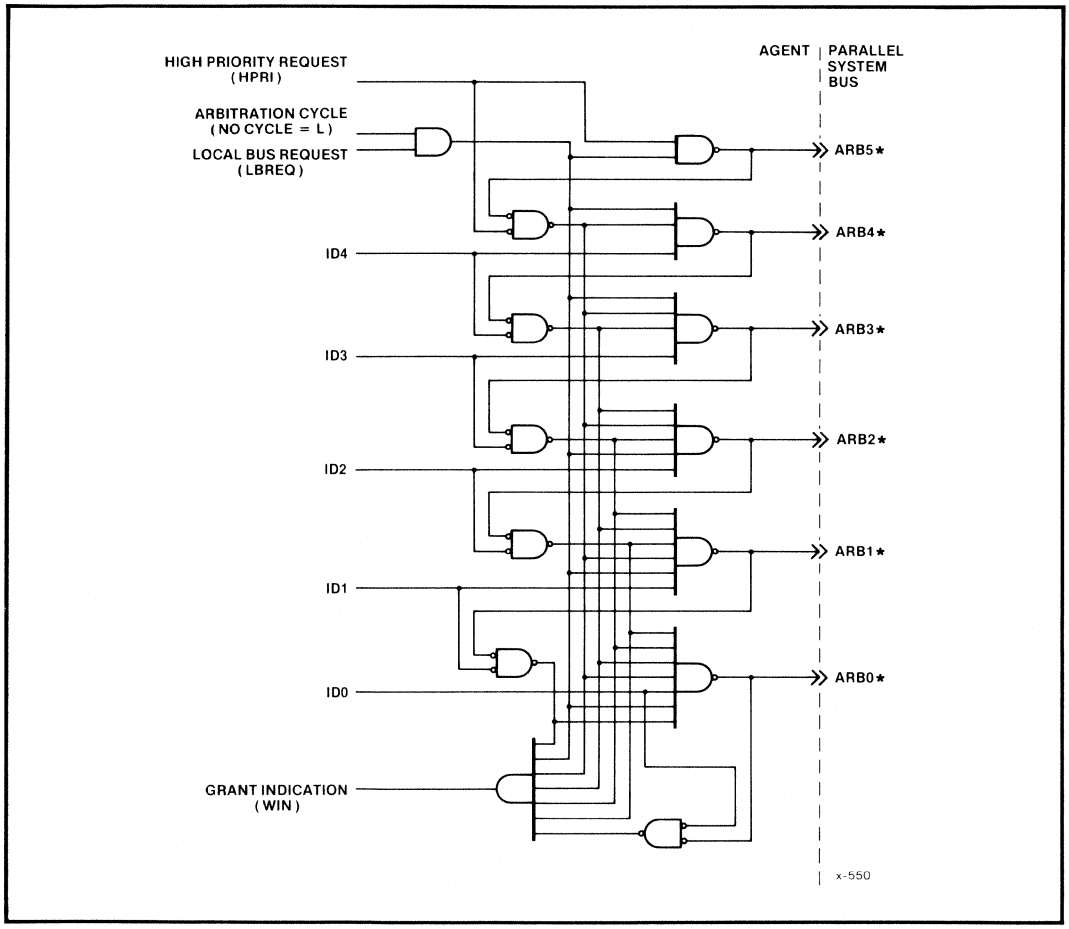



**Figure 5. iPSB™ Bus Arbitration Cycle**

and RPRDY (Replying agent ready) handshake lines. Either agent can hold off the transfer by deactivating its ready line. This handshaking supports any speed requesting or replying agent.

The transfer cycle is complete when the requesting agent signals the last data transfer via the End-Of-Cycle (EOC — SC2*). The last bus clock cycle of the transfer is when EOC, RQRDY, and RPRDY are all active simultaneously.

The replying agent has the opportunity to tell the requesting agent if it does not support the requested operation via the agent error (SC5*, SC6*, and SC7*) lines. These lines encode four types of errors: width violation, continuation error, illegal operation, and negative acknowledgment of a message. Trying to extract 32-bits of data from an 8-bit peripheral is an example of a data width violation. Continuation errors occur when attempting sequential access from an agent which does not support them or running off the ending address of a memory board. Writing to a read-only memory is an example of an illegal operation. A replying agent signals a negative acknowledgment to a message transfer cycle if its destination queue is full (the source must perform source queuing). The transfer cycle is terminated by the requesting agent when it detects that the replier is signalling an agent error. If the bus interface is intelligent, it might retry the operation with a different type that the replying agent can support. Other aspects of transfer cycle include the ability of a requesting agent to LOCK the bus via the SC1* line. SC1* is a non-multiplexed signal which inhibits alternate ports of any multi-ported resource being addressed. By locking the bus, the requesting agent guarantees itself exclusive access to a multi-ported bus resource and retains bus ownership for more than one transfer cycle.

As noted in the figure, in addition to parity protection on the address/data lines, the SC lines are also protected by parity. The requesting agent is responsible for the SC parity bits (SC8* and SC9*) during the request phase (it drives all SC lines). The reply phase requires two parity bits: one for those lines driven by the requesting agent and one for those driven by the replier. This ensures all aspects of the transfer cycle have parity protection.

### Exception Cycle

The exception cycle is an error reporting mechanism. An agent or the CSM initiates an exception cycle as a result of sensing an exception. If no exception occurs, no exception cycles occur.

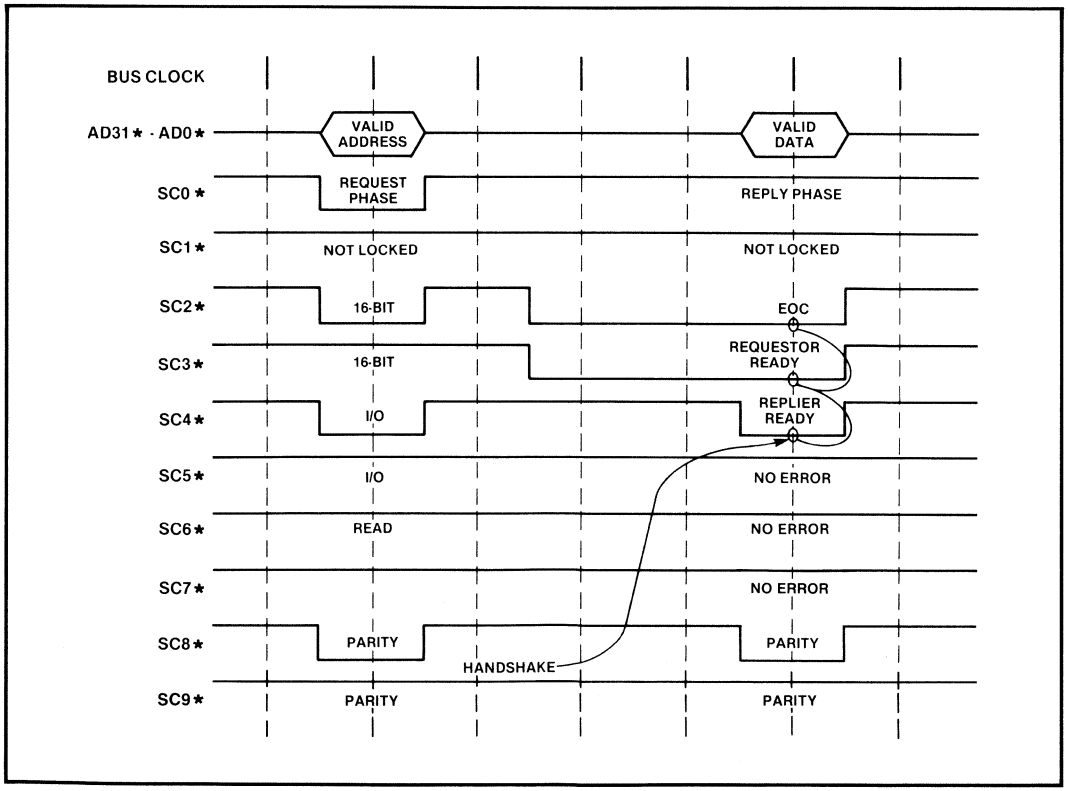



**Figure 6. Transfer Cycle Example**

The exception cycle has two purposes in the protocol: first, it provides systematic termination of activity on the iPSB bus and second, it provides a stabilization time before allowing agents to resume operation. These two purposes correspond directly to the two phases of the exception cycle: the signal and recovery phases.

The signal phase begins when an agent or a module senses an exception and activates one of the bus error lines. On receiving a bus error, all agents terminate any transfer or arbitration cycles in progress. The net effect of the signal phase is to terminate all bus activity. The signal phase continues until the error-detecting module deactivates the bus error line.

The recovery phase begins after the bus error line becomes inactive. The recovery phase is a fixed-duration delay (in terms of bus clock cycles) that allows time for the iPSB bus signals to settle before starting more transfer cycles.

There are two types of bus exceptions supported by the iPSB bus: timeout and bus error. The CSM monitors the bus to ensure that all data handshakes complete. If for some reason the handshake hangs and exceeds a maximum time limit, the CSM activates the TIMOUT* (Time Out) bus exception line to begin the exception cycle.

An agent sends a bus error exception whenever it determines that the information on the address/data (AD) or the transfer control (SC) lines is in error. Typical examples of bus error exceptions are parity errors and non-recoverable memory errors sensed during a data transfer. Once an error is detected, the agent activates the BUSERR* (Bus Error) signal line to begin the exception cycle.

## Mechanical

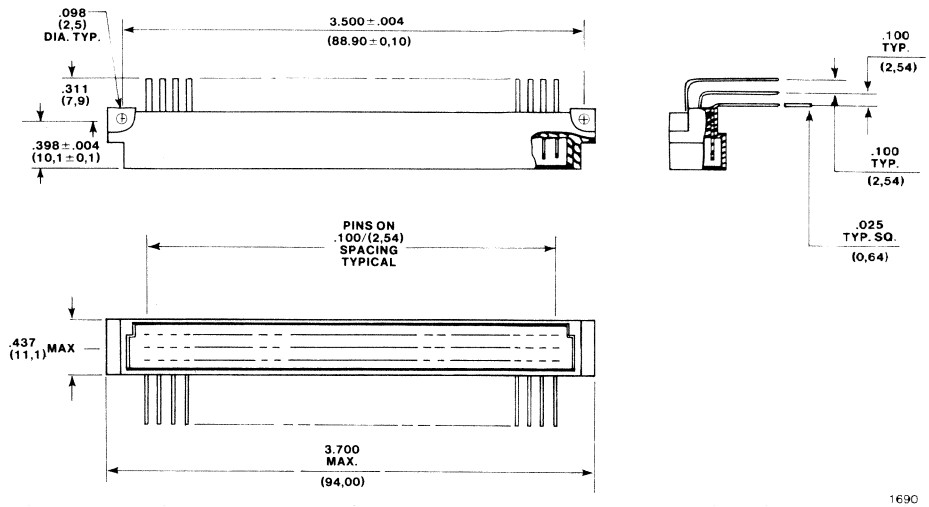
The MULTIBUS II boards, board accessories, and backplanes conform to mechanical standards defined by the International Electromechanical Commission (IEC); these standards are commonly referred to as the Eurocard mechanical standards. This mechanical system offers modular board sizes as defined in standard IEC-297-3 and reliable two-piece connectors as defined in IEC-603-2.

### Form Factor

The MULTIBUS II specification calls out two modular board form factors: 233 x 220mm and 100 x 220mm (see Figure 7). The iPSB bus and iLBX II bus portions of the MULTIBUS II system architecture are always defined on the P1 and P2 connectors respectively. However, the user can optionally define the use of the P2 connector if the iLBX II bus is not supported. (The iSSB bus is additionally defined on the P1 connector.)

## Connector

MULTIBUS II boards and backplanes use two-piece, 96-pin connectors for both the iPSB bus and the iLBX II bus. The right-angle connectors on the printed board are IEC standard 603-2-IEC-C096-M; the receptacle connectors on the backplane are IEC standard 6-03-2-IEC-C096-F (Figure 8). This connector family is noted for its reliability, availability, and low cost.



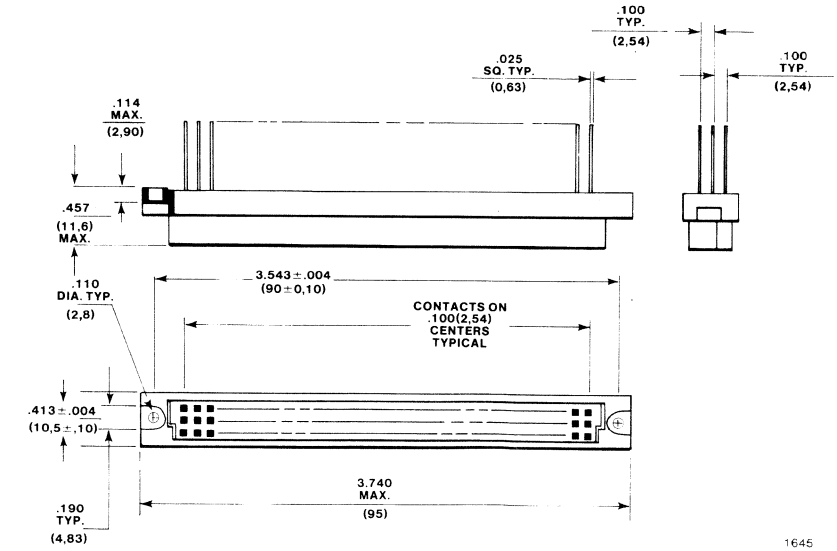




**Figure 8. MULTIBUS® II Connectors**

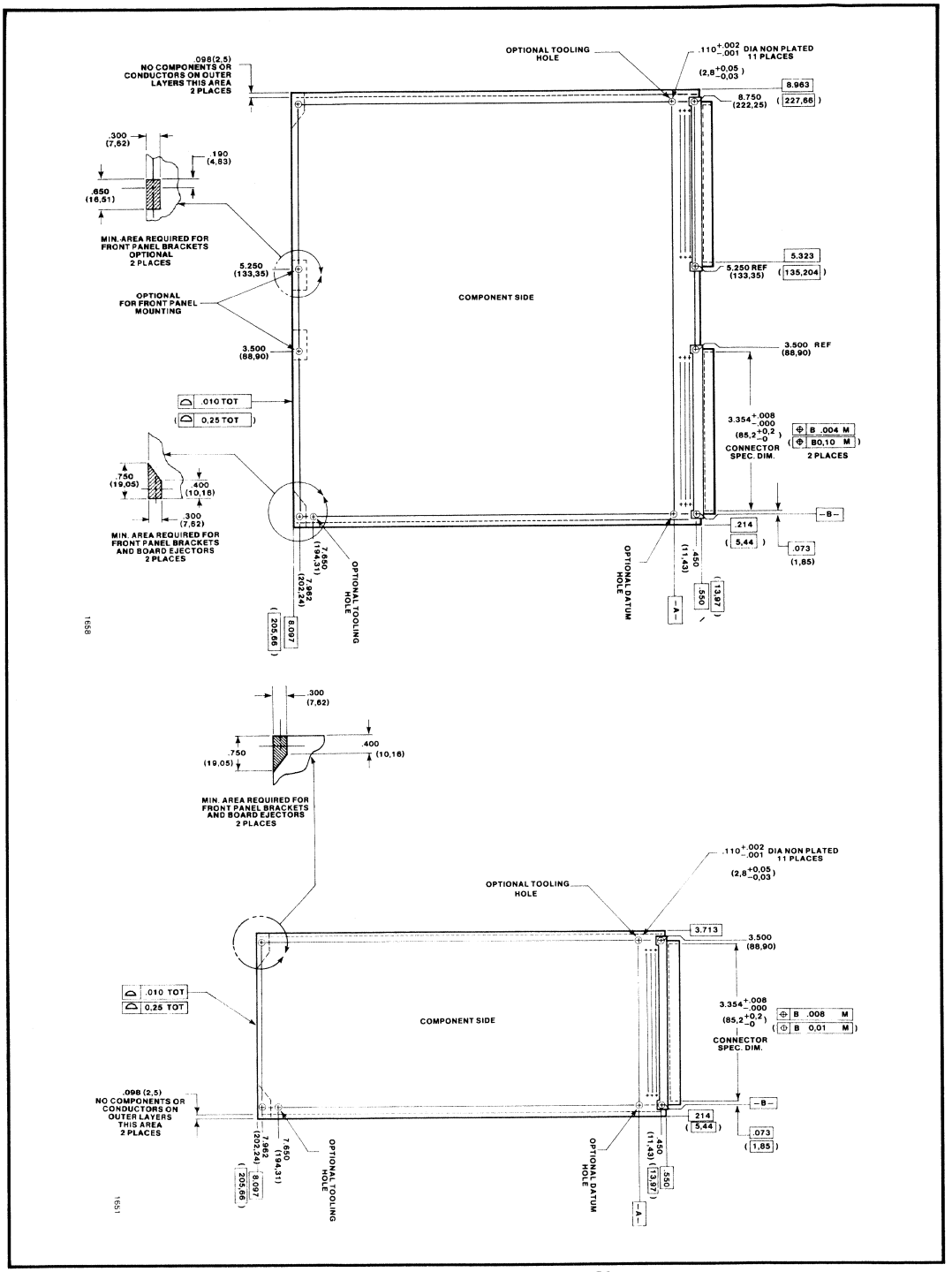



**Figure 7. MULTIBUS® II Board Sizes**

The pin assignment for the iPSB bus on P1 is shown in Table 2.

Please refer to Intel's MULTIBUS II Bus Architecture Specification Handbook for more detailed information.

**Table 2. iPSB™ Bus Pin Assignments**

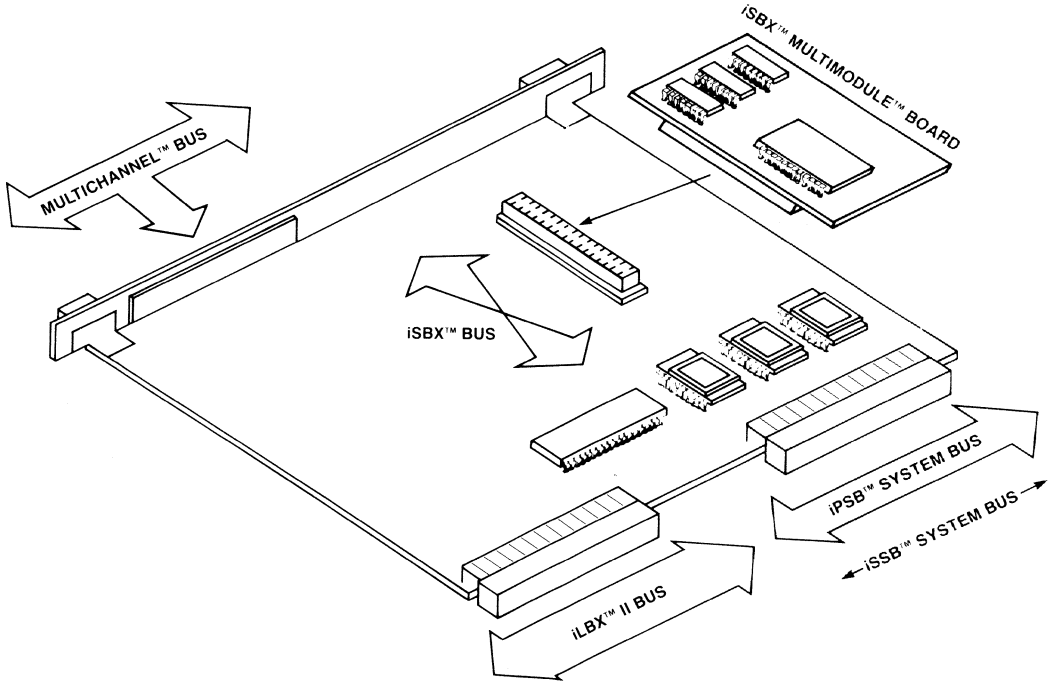
| Connector Pin Number | Row A | Row B | Row C |
|---|---|---|---|
| 1 | 0 Volts | PROT* | 0 Volts |
| 2 | +5 Volts | DCLOW* | +5 Volts |
| 3 | +12 Volts | +5 Battery | +12 Volts |
| 4 | (Note 2) | SDA (Note 3) | BCLK* |
| 5 | TIMOUT* | SDB (Note 3) | 0 Volts |
| 6 | (Note 1) LACHn | 0 Volts | CCLK* |
| 7 | AD0* | AD1* | 0 Volts |
| 8 | AD2* | 0 Volts | AD3* |
| 9 | AD4* | AD5* | AD6* |
| 10 | AD7* | +5 Volts | PAR0* |
| 11 | AD8* | AD9* | AD10* |
| 12 | AD11* | +5 Volts | AD12* |
| 13 | AD13* | AD14* | AD15* |
| 14 | PAR1* | 0 Volts | AD16* |
| 15 | AD17* | AD18* | AD19* |
| 16 | AD20* | 0 Volts | AD21* |
| 17 | AD22* | AD23* | PAR02* |
| 18 | AD24* | 0 Volts | AD25* |
| 19 | AD26* | AD27* | AD28* |
| 20 | AD29* | 0 Volts | AD30* |
| 21 | AD31* | Reserved | PAR3* |
| 22 | +5 Volts | +5 Volts | Reserved |
| 23 | BUSREQ* | RST* | BUSERR* |
| 24 | ARB5* | +5 Volts | ARB4* |
| 25 | ARB3* | RSTNC* | ARB2* |
| 26 | ARB1* | 0 Volts | ARB0* |
| 27 | SC9* | SC8* | SC7* |
| 28 | SC6* | 0 Volts | SC5* |
| 29 | SC4* | SC3* | SC2* |
| 30 | −12 Volts | +5 Battery | −12 Volts |
| 31 | +5 Volts | SC1* | +5 Volts |
| 32 | 0 Volts | SC0* | 0 Volts |

**NOTES:**
1. LACHn* for all agents but the one driving CCLK*; line contains a second CCLK* signal in systems that have more than 12 cardslots.
2. 0 Volts for all agents but the one driving BCLK*; line contains a second BCLK* signal in systems that have more than 12 cardslots.
3. Signal lines SDA and SDB are reserved for the Serial System Bus.

# MULTIBUS® II
# iLBX™ II Local Bus Extension

- **High bus bandwidth —**
  - **— 48 megabytes/sec**
- **64 megabyte (26-bit) addressing**
- **8-, 16-, 24-, and 32-bit data transfers over a 32-bit path**
- **Reliable synchronous clocking up to 12 megahertz**
- **Burst transfers up to 64 kilobytes per transfer**
- **Primary and secondary bus master exchange capabilities**
- **Supports up to 6 iLBX™ II compatible devices per bus**
- **Pipelined protocol for highest performance**
- **Optional parity protection for address and data**

The iLBX™ II Local Bus Extension is one of the family of standard bus structures resident within Intel's MULTIBUS® II Bus Architecture. The iLBX II bus is a dedicated execution bus capable of significantly increasing system performance by removing most processor execution activity from the main iPSB™ Parallel System Bus. It extends the processor board's on-board local bus to off-board resources. Acting in conjunction with the processor board, the iLBX II resources form a multiple board "virtual single board computer". The iLBX II bus preserves advantages in performance and architecture of on-board local memory, while allowing memory configurations larger than those possible on a single board.




**MULTIBUS® II Physical Diagram**

## FUNCTIONAL DESCRIPTION

### Architectural Overview

The iLBX II bus is an architectural solution for supporting large amounts of off-board memory with the same performance advantage enjoyed by on-board memory (see Figure 1). It allows the CPU board selection to be decoupled from the on-board memory requirement and still maximizes the processor's performance potential. It eliminates the processor's need to access its off-board memory resources solely over the iPSB system bus. In most systems, the processor is the only master on the iLBX II bus, so no time is required to arbitrate for the bus. This means the processor sees significantly lower memory latency than is possible if it were accessing memory over the multiple master system bus. Lower memory latency translates to higher individual processor performance.

The inclusion of the iLBX II bus in the architecture means not just higher single processor performance but higher system performance as well. The movement of execution traffic from the system bus to the iLBX II execution bus makes that much additional system bus bandwidth available to other system resources such as processors not using an execution bus or I/O devices.

For those applications which require a high bandwidth local path to I/O, such as an intelligent disk controller local to a particular processor, the iLBX II bus supports one additional bus master. This architectural enhancement allows a processor to "own" an intelligent I/O controller. All data transfers between these two modules (the processor and the controller) can occur over the low latency iLBX II bus path without disturbing activity on the system bus.

### Structural Features

#### Overview

The iLBX II bus uses a non-multiplexed, processor independent structure supporting 8-, 16-, and 32-bit processors. It supports 8-, 16-, 24-, and 32-bit data transfers over a 26-bit (64 megabyte) addressing range with a maximum bandwidth of 48 megabytes/sec.

All events performed on the bus are synchronous to a reference bus clock. This is not a fixed frequency clock as in the iPSB bus; the iLBX II bus clock runs at the basic processor bus frequency. In other words, a processor whose bus interface runs at 8 megahertz would drive the iLBX II bus at that frequency. This characteristic helps match the iLBX II bus timing to that of the processor transfer rate for best performance. The maximum iLBX II bus clock frequency is 12 megahertz. (Be careful not to confuse a processor's clock input frequency with its basic bus frequency. Many processors internally divide down their clock input by 2, 3, or 4 to obtain the basic bus frequency. It is this basic bus frequency which defines their transfer rate and which drives the iLBX II bus clock.)

#### Non-Multiplexed Structure

The iLBX II bus structure is non-multiplexed in order to simplify the interface and obtain maximum performance. The separate address, data, and control paths allow overlapped operation. This overlapping, called pipelining, means that data from a previous operation can be overlapped with the address and command information of the current operation. This characteristic substantially improves bus utilization for those processor-memory subsystems which support the feature.

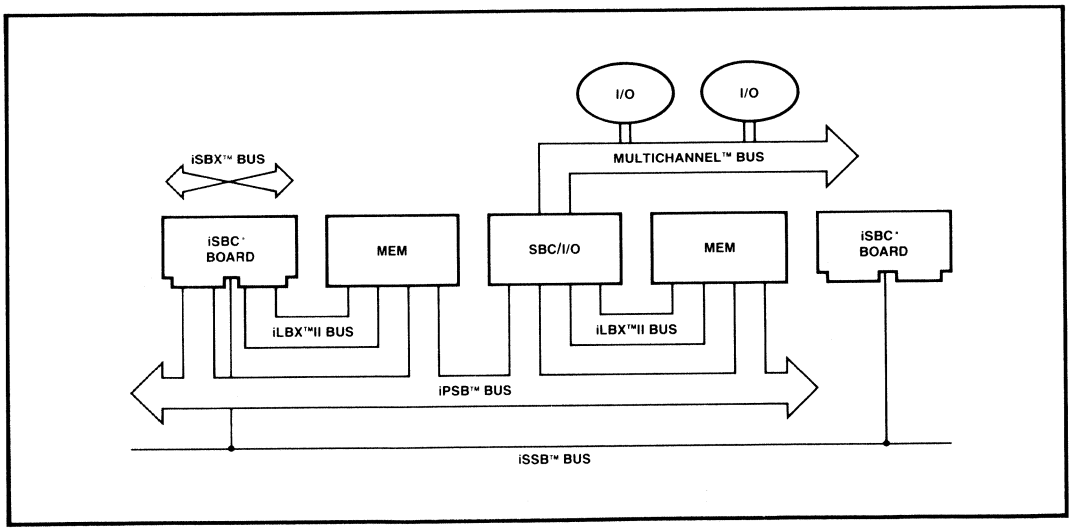



**Figure 1. MULTIBUS® II Bus Architecture**

### Interconnect Address Space

The iLBX II bus supports the slot-addressing concept of the interconnect address space found in the iPSB bus. Including this facility in the iLBX II bus allows the system to identify and configure iLBX II bus boards even though they may not contain a iPSB bus port. (Please refer to the iPSB bus data sheet for additional information on the Interconnect address space.)

### Dual Bus Masters

In order to support a wide range of system configurations, the iLBX II bus defines support for two bus masters. One master is called the Primary master; the other is known as the Secondary master. The Primary master normally "owns" the bus and does not have to spend any time arbitrating for access rights. The Secondary master must ask the Primary master for access rights. The Primary releases the bus at the first opportune time. This hierarchical structure ensures that the Primary master enjoys good memory latency while at the same time gives the Secondary the opportunity to access memory when it needs to.

The iLBX II bus also includes a dedicated interrupt line to facilitate signalling between the two masters for commands and status, and between the memory boards and the Primary master for things such as non-recoverable memory errors.

## Bus Cycle Overview

Like the iPSB bus, the iLBX II bus protocol consists of three types of bus cycles: arbitration, transfer, and exception.

### Arbitration Cycle

The arbitration cycle ensures that one and only one requesting agent is allowed access to the bus at any given time. When a requesting agent determines the need for a bus operation, it enters the arbitration cycle. For either requesting agent, this cycle lasts until it acquires the right to use the bus. In configurations with only a primary requesting agent, no time is spent for this cycle; the agent always has rights to the bus. In configurations where there are both a primary and secondary agent, the primary agent has to arbitrate for the bus only when the bus is busy under the secondary agent's control. Figure 2 illustrates the arbitration cycle.

### Transfer Cycle

The transfer cycle is the event where the request (address and command) and reply (data) information is exchanged between the bus agents. Like the iPSB bus, it consists of a request and a reply phase. During block transfers, the termination of the transfer cycle is controlled by the requesting agent. In non-block transfer cycles, the cycle's termination is implicitly recognized by both agents. Figure 3 shows a transfer cycle example.

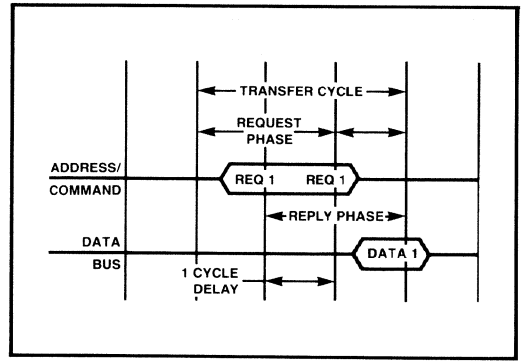



**Figure 3. iLBX™ II Transfer Cycle**

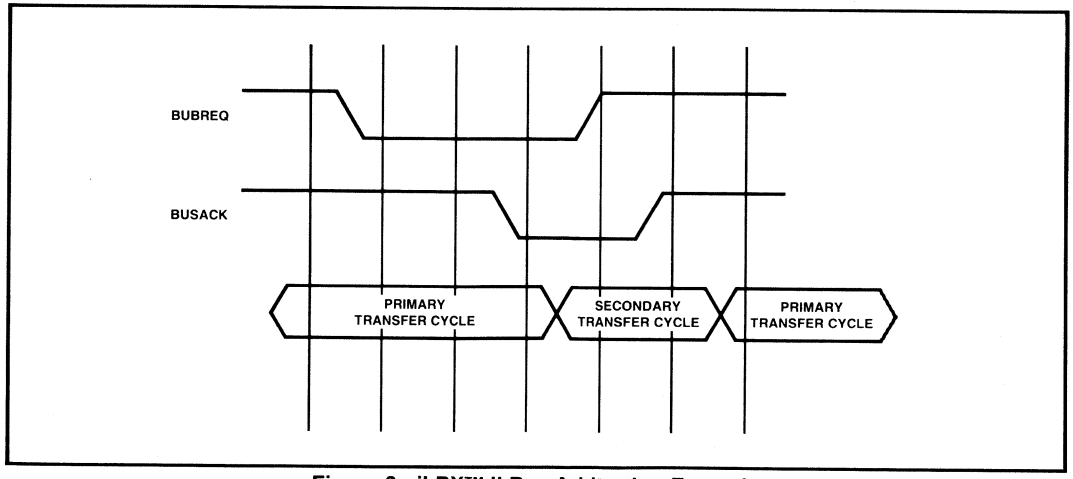



**Figure 2. iLBX™ II Bus Arbitration Example**

### Exception Cycle

Exception cycles allow the bus agents to signal any detected error or exceptional condition which might arise during a transfer cycle. Typical exceptions are uncorrectable ECC errors, parity errors, or physical boundary overflows.

## Signal Groups

### Overview

There are five categories of signals used in the iLBX II bus: address/command, data transfer, access control/status, bus control/status, and miscellaneous. An asterisk following the signal name or group indicates that the signal or group use their low electrical state as the active state.

### Address/Command

The requesting agent uses this group of signals to transfer address and command information to the potential replying agents during the request phase of a transfer cycle. This signal group consists the non-multiplexed address lines, **XA25 through XA00** (Extension bus address), the command specification lines, **XC3 through XC0** (Extension bus command), and an associated parity line, **XAPAR** (Extension bus address/command parity).

The XA25 through XA00 lines define the starting physical byte address. The command specification lines select the address space (memory or interconnect), data width (1, 2, 3, or 4 bytes), and whether the operation is a read or write cycle. The command encodings for XC3 through XC0 are shown in Figure 4.

| XC3* | XC2* | XC1* | XC0* |
|---|---|---|---|
| **Address Space** | **Access Type** | **Width Specification** | |
| Memory | Read | 1 byte<br>2 bytes | |
| Interconnect | Write | 3 bytes<br>4 bytes | |

**Figure 4. iLBX II Command Encoding**

Parity for the address/command group is not required. The bus does allow for a single parity bit covering the address and command lines as a compliance level. The iLBX II bus environment is much different than that of the iPSB system bus. It extends only a short distance (6 card slots maximum) and employs lower switching currents. This more restrictive environment reduces the need for data integrity protection in all but the larger systems.

### Data Transfer Group

This signal category consists of the 32 bi-directional data lines and their optional parity line. **XD31 through XD0** (Extension bus data) transfer the read or write data between the requesting and replying agents. Each byte in the iLBX II bus memory is mapped to one of the four byte locations of the XD lines. This technique is commonly referred to as "byte lanes" and is illustrated in Figure 5.

Like with the address/command group, the **XDPAR** (Extension bus data parity) line is optional.

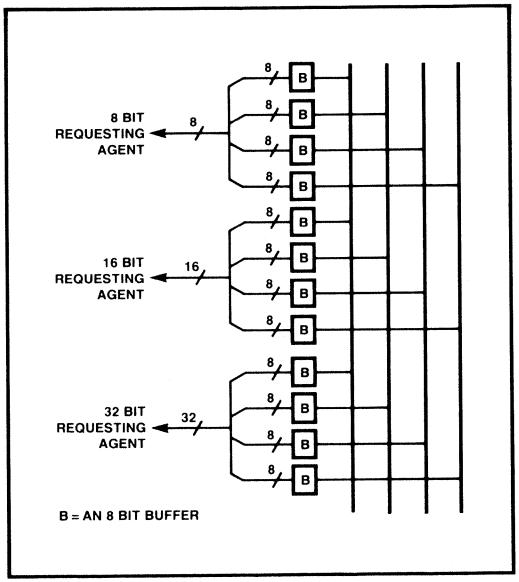



**Figure 5. iLBX™ II Data Bus Alignment Interface Requirements**

### Access Control/Status Group

This signal category consists of 5 lines which determine the start of an access request, its execution, and finally, its termination.

The **XACCREQ*** (Extension bus access request) signal indicates that the address/command information is valid during the current and next bus clock cycles. It signals the presence of the request phase of the transfer cycle. Replying agents which require more time to decode the command information can extend XACCREQ* using the XWAIT* handshake line.

The **XWAIT*** (Extension bus wait) signal has a two-fold meaning in the access protocol: it can extend the duration of the request phase and it serves as a "not ready" replier indication during the reply phase. If asserted in the first clock cycle of the request phase, it extends the phase, otherwise, it will signal "not ready" during the reply phase.

In many system configurations the iLBX II bus memory boards are dual-ported to both the iLBX II and iPSB buses. This requires a mutual exclusion facility when implementing semaphores and other data structures in this shared memory. The **XLOCK*** (Extension bus lock) signal allows the iLBX II bus requesting agents to lock out the other port while performing indivisible accesses to shared structures.

To perform block transfers on the iLBX II bus, the requesting agent asserts the **XBTCTL*** (Extension bus block transfer control) signal. This line informs the replying agents that two or more data transfer periods will accompany a single request phase. XBTCTL* is de-asserted by the requesting agent to signal the end of the block transfer.

### Bus Control/Status Group

The signals in this group control the passing of bus ownership between the primary and secondary requesting agents. When the bus is in use, they also indicate which agent is in control.

The **XBUSREQ*** (Extension bus request) signal is driven by the secondary requesting agent to acquire the bus from the primary agent. Only the primary requesting agent receives this signal. When the primary detects that the secondary is requesting the bus, it replies with the **XBUSACK*** (Extension bus acknowledge) signal to inform the secondary that the bus is now his. This bus exchange occurs at the discretion of the primary.

The secondary owns the bus after asserting XBUSREQ* and receiving XBUSACK* active. The primary can request that the bus be returned at any time by removing XBUSACK*. The secondary must return the bus at the earliest time; typically when it completes its current transfer cycle.

### Miscellaneous Control Group

The **XRESET*** (Extension bus reset) is driven by the primary requesting agent to locally initialize its iLBX II bus environment. It is typically asserted after the agent receives a reset indication on the iPSB system bus.

The **XINT*** (Extension bus interrupt) allows the secondary requesting agent and any of the replying agents to signal the primary requesting agent for inter-module communication. Since the secondary agent is usually performing tasks on behalf of the primary agent, this interrupt line removes the need for the primary to continuously poll the secondary for completion of its tasks.

The **XID2* through XID0*** (Extension bus identity) lines are hardwired lines on the backplane to allow any iLBX II bus board to determine its position on the bus. They encode the interconnect space least significant three bits of the slot ID field. (See the iPSB bus data sheet for an explanation of the interconnect address space.)

The final line is the **XBCLK*** (Extension bus clock) line. It provides the reference timing signal for the synchronous bus operations. It is driven by the primary requesting agent at its processor bus frequency.

The iLBX II bus also defines additional +5 volt and ground pins.

## Bus Protocol

In the MULTIBUS II specification, both timing diagrams and state-flow diagrams describe the iLBX II bus protocol. The state-flow diagrams present the lowest level and most rigorous definition while the timing diagrams help conceptual understanding. For the purposes of this data sheet, only the timing diagram description is used. The following sections use Figure 6 as an example of the protocol.

### Arbitration Cycle

With only two potential requesting agents contending for access rights to the bus, the arbitration cycle is very simple. The figure illustrates the secondary requesting agent requesting the bus from the primary and then running a simple transfer cycle. The secondary requesting agent makes its request by asserting XBUSREQ*. The primary gives up the bus by returning XBUSACK* active. In this example, the secondary uses the bus for only a single transfer cycle so it de-asserts XBUSREQ* when complete. The primary agent responds by withdrawing XBUSACK* to indicate it now owns the bus.

### Transfer Cycle

Like in the iPSB bus, the transfer cycle proceeds as a request phase and a reply phase. The requesting agent (either the primary or the secondary depending upon who currently owns the bus) informs the potential replying agents of the request phase by driving valid information on the address/command signal group and asserting XACCREQ*. The request phase normally lasts two clock cycles although the replying agents have the opportunity to extend the phase as long as necessary by asserting XWAIT* during the first clock period of the phase. The phase is extended as long as XWAIT* is active. In the example, the request phase is extended one additional clock.

The reply phase begins when XWAIT* is de-asserted. At this point, the meaning of XWAIT* changes to become a "not ready" indication from the selected replying agent. In the example, the replying agent requires one additional clock period to supply the data so XWAIT* is asserted for one clock. The reply phase terminates on the same clock that data is valid.

### Exception Cycle

If transfer integrity checking is implemented on the iLBX II bus, errors are signalled on the clock following the last valid information period. In the example, errors

detected on the address/command lines during the request phase are signalled on the clock following the removal of valid request information. The same applies to errors detected on the data lines during the reply phase.

## Mechanical

The iLBX II bus is defined on the P2 connector of two-connector MULTIBUS II boards. Since the iLBX II bus environment is local to a particular processor board, the iLBX II bus backplane does not extend the entire length of the iPSB bus backplane. This allows for multiple iLBX II bus environments in a given system.

The pin assignment for the iLBX II bus on P2 is shown in iLBX II specification section in the MULTIBUS II Bus Architecture Specification Handbook.

Please refer to Intel's MULTIBUS II Bus Architecture Specification Handbook for more detailed information.

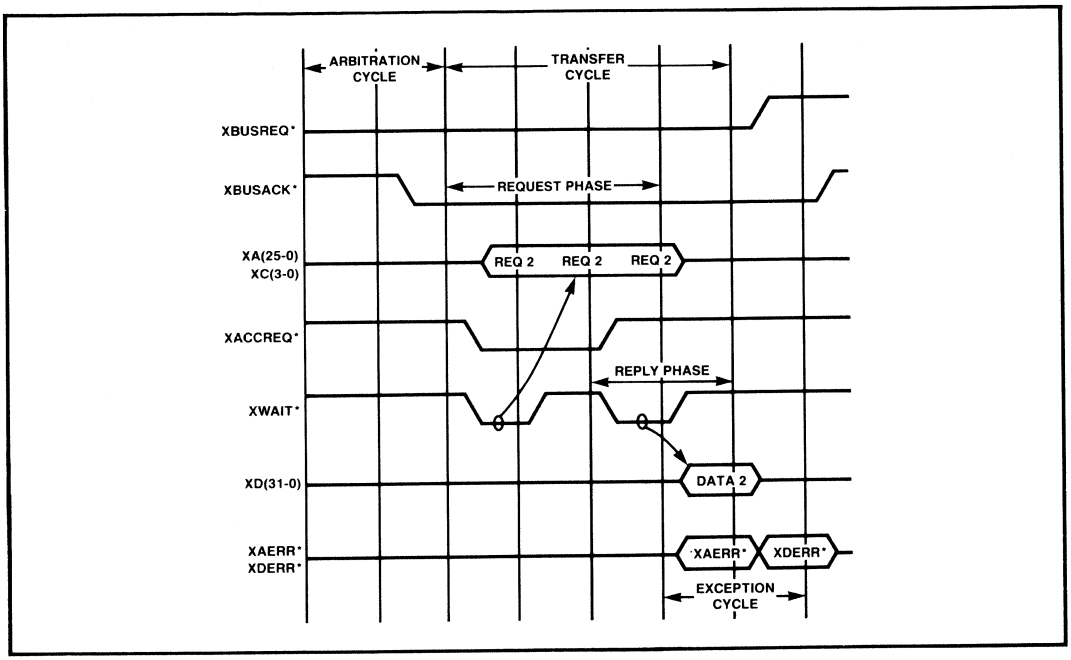
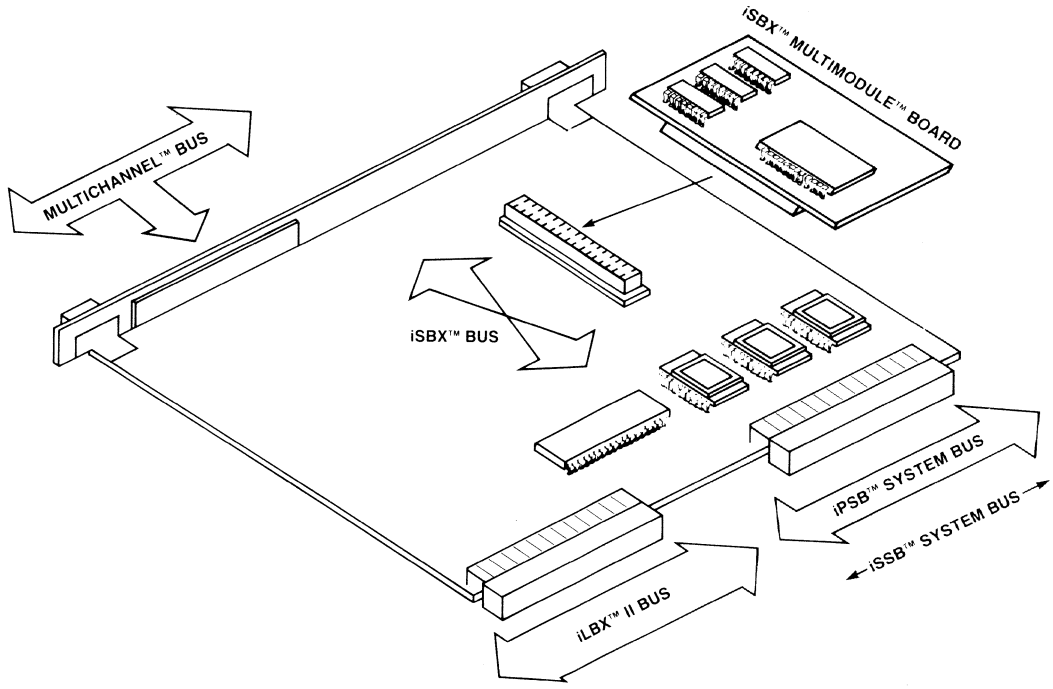



**Figure 6. iLBX™ Transfer Cycle Example**

# MULTIBUS® II
# iSSB™ Serial System Bus

- **Logical equivalent to the iPSB™ bus message space**
- **2 megabits/sec serial data rate**
- **Multi-master capability up to 32 nodes**
- **Physical distribution up to 10 meters**
- **Deterministic access protocol**
- **Based upon CSMA/CD (Carrier Sense Multiple Access with Collision Detection)**

The iSSB™ Serial System Bus is a simple, low cost alternative to the iPSB™ Parallel System Bus message address space. The message passing interface is identical for both buses; this allows easy migration from one bus to the other with no software changes. The iSSB bus serves as a low-cost replacement for the iPSB bus in applications where cost reduction is required and serves as a complement to the iPSB bus where an alternative bus path is needed for interface control, diagnostics, or redundancy changes.




**MULTIBUS® II Physical Diagram**

## FUNCTIONAL DESCRIPTION

### Architectural Overview

The trend toward a more functional VLSI has driven the cost-functionality vector to allow system designers to pack more and more functionality on a given size board while maintaining approximately constant cost. The iSSB Serial System Bus lets VLSI drive the cost-functionality vector in the other direction; dramatically reduce the cost while maintaining roughly constant functionality. It accomplishes this by reducing the **interconnect cost** and allowing **physical distribution** of modules.

#### Reduced Interconnect Cost

Most systems today use a parallel interface to interconnect boards within the system. Frequently the cost to provide this interconnect is a significant percentage of the total system cost. Connectors, backplanes and interface logic are all part of this interconnect cost.

The iSSB Serial System Bus dramatically reduces the interconnect cost by replacing the parallel interface's multiple-line connector and backplane with a simple twisted-pair interface using telephone-type connectors. It also reduces the interface logic to a single VLSI component as opposed to the multiple components required in a parallel interface.

#### Physical Distribution

Being tied to a backplane or bulky ribbon cable limits the system designer's mechanical flexibility in constructing a system from multiple modules. The iSSB bus frees him of these restrictions by letting him physically distribute the system modules up to 10 meters apart.

### Structural Features

#### Physical Characteristics

The iSSB bus consists of a maximum of 32 nodes which can be distributed over a maximum of 10 meters of cable. The nodes may be distributed along an external cable segment or clustered into backplanes as shown in Figure 2. Each backplane may contain up to 20 nodes, the maximum number of cardslots in a iPSB bus backplane.

Clustered systems use repeaters as a connection between backplanes and the iSSB bus cable. The repeaters isolate the cable from the excessive capacitive load on the backplane.

#### Access Protocol

The iSSB bus employs an access protocol called Carrier-Sense-Multiple-Access with collision detection (CSMA/CD). The CSMA/CD protocol allows agents to transmit data whenever they are ready.

In CSMA/CD operation, an agent with data to transmit looks at the iSSB bus for traffic before beginning a transmission. If the bus is not idle, the agents waits until the line becomes idle and until an interframe space has passed. After both events, the agent begins transmission of the message.

It is possible for more than one agent to initiate a transmission at the same time; in that case, a collision occurs on the bus. The protocol handles collisions on the iSSB bus via a deterministic collision resolution algorithm that uses time slotting.

The deterministic collision algorithm guarantees a time slot during which each agent can transmit without interference from other agents. The resolution guaran-

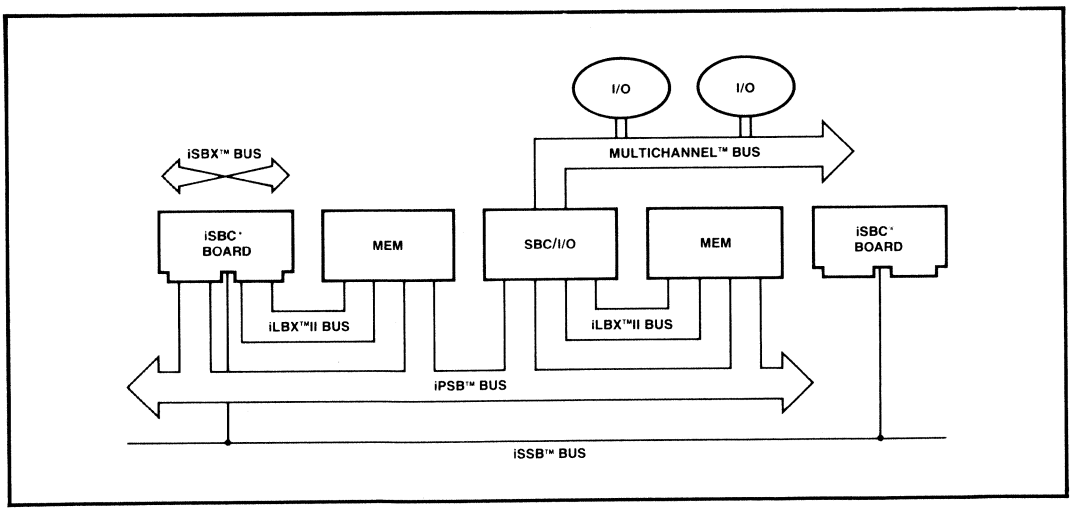



**Figure 1. MULTIBUS® II Bus Architecture**

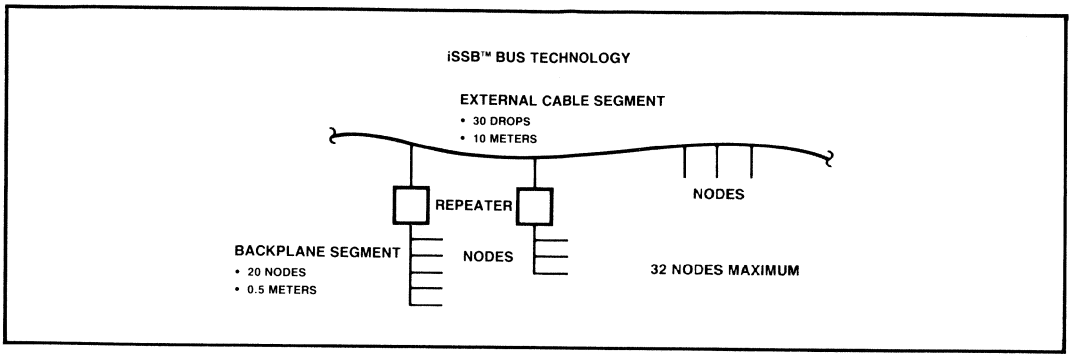



**Figure 2. Typical iSSB™ Bus System Configurations**

tees fair access to all agents. This type of collision resolution provides a real-time response that allows agents to resolve collisions in a finite time period.

### Error Control

The iSSB bus uses a 16-bit CRC (Cyclic Redundancy Check) in order to provide error detection. Used in conjunction with an intelligent interface, this allows the iSSB to look as reliable as the iPSB bus even though it is up to 10 meters long.

### Physical Interface

The physical iSSB bus interface consists of two signal lines (the SDA and SDB lines) that are included as part of the iPSB bus backplane design and may be extended via a 2-wire cable that connects to a repeater, typically located on the CSM. Agents encode data on the complementary, open-collector signal lines as shown in Table 1.

**Table 1. iSSB™ Bus Signal Line Encoding**

| SDA Line | SDB Line | Line Condition |
|---|---|---|
| 0 | 0 | collision |
| 0 | 1 | logic 0 |
| 1 | 0 | logic 1 |
| 1 | 1 | idle |

The portion of the signal lines within the backplane is designed to operate in a high-noise environment such as a heavily loaded backplane. Cable extensions to the iSSB bus must adhere to normal transmission line requirements.

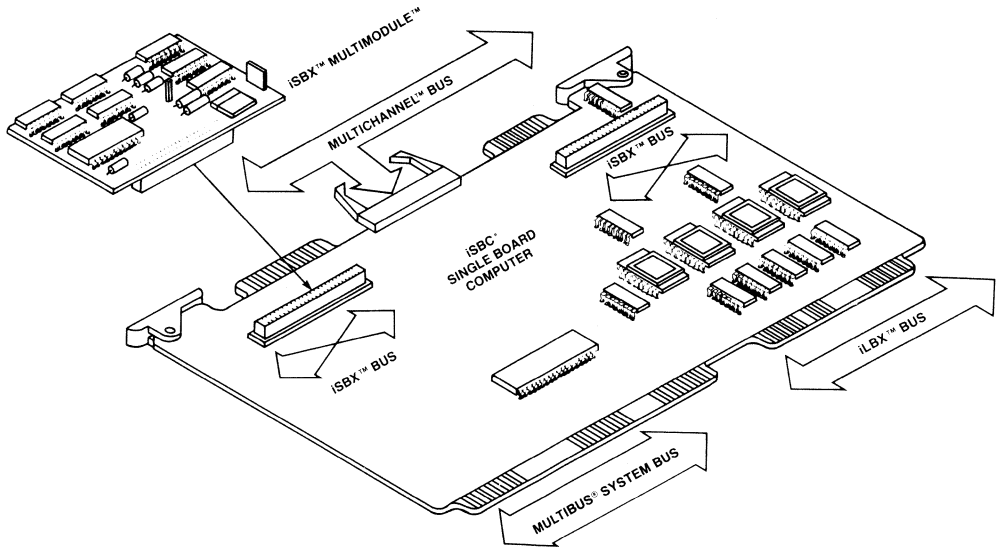
To further improve reliability, the bus interface includes receivers that sample the data and filter out noise which may be coupled from the surrounding environment.

Please refer to Intel's MULTIBUS® II Bus Architecture Specification Handbook for more detailed information.

# iSBX™ I/O EXPANSION BUS

- **IEEE P959 industry standard I/O expansion bus**
- **Provides on-board expansion of system resources**
- **Small iSBX™ MULTIMODULE™ boards plug directly into iSBC® boards**
- **Supports compatible 8- and 16-bit data transfer operations**
- **Part of Intel's Total System Architecture: MULTIBUS,® iLBX,™ MULTICHANNEL,™ and iSBX™**
- **Low-cost "vehicle" to incorporate the latest VLSI technology into iSBC®-based systems**
- **Provides increased functional capability and high performance**
- **Supported by a complete line of iSBC® base boards and iSBX™ MULTIMODULE™ boards, providing analog and digital I/O, high-speed math, serial and parallel I/O, video graphics, and peripheral controllers**

The iSBX™ I/O Expansion Bus is one of a family of standard bus structures resident within Intel's total system architecture. The iSBX bus is a modular, I/O expansion bus capable of increasing a single board computer's functional capability and overall performance by providing a structure to attach small iSBX MULTIMODULE™ boards to iSBC® base boards. It provides for rapid incorporation of new VLSI into iSBC MULTIBUS® systems, reducing the threat of system obsolescence. The iSBX bus offers users new economics in design by allowing both system size and system cost to be kept at minimum. As a result, the system design achieves maximum on-board performance while allowing the MULTIBUS interface to be used for other system activities. The iSBX bus enables users to add-on capability to a system as the application demands it by providing off-the-shelf standard MULTIMODULE boards in the areas of graphics controllers, advanced mathematics functions, parallel and serial I/O, disk and tape peripheral controllers, and magnetic bubble memory. A full line of MULTIBUS boards and iSBX MULTIMODULE boards are available from Intel and other third party sources in the industry.



## FUNCTIONAL DESCRIPTION

### Bus Elements

The iSBX™ MULTIMODULE™ system is made up of two basic elements: base boards and iSBX MULTIMODULE boards. In an iSBX system, the role of the base board is simple. It decodes I/O addresses and generates the chip selects for the iSBX MULTIMODULE boards.

The iSBX bus supports two classes of base boards, those with direct memory access (DMA) support and those without. Base boards with DMA support have DMA controllers that work in conjunction with an iSBX MULTIMODULE board (with DMA capability) to perform direct I/O to memory or memory to I/O operations. Base boards without DMA support use a subset of the iSBX bus and simply do not use the DMA feature of the iSBX MULTIMODULE board.

The iSBX MULTIMODULE boards are small, specialized, I/O mapped boards which plug into base boards. The iSBX boards connect to the iSBX bus connector and convert iSBX bus signals to a defined I/O interface.

### Bus Interface/Signal Line Descriptions

The iSBX bus interface can be grouped into six functional classes: control lines, address and chip select lines, data lines, interrupt lines, option lines, and power lines. The iSBX bus provides nine control lines that define the communications protocol between base board and iSBX MULTIMODULE boards. These control lines are used to manage the general operation of the bus by specifying the type of transfer, the coordination of the transfer, and the overall state of the transfer between devices. The five address and chip select signal lines are used in conjunction with the commmand lines to establish the I/O port address being accessed, effectively selecting the proper iSBX MULTIMODULE. The data lines on the iSBX bus can number 8 or 16, and are used to transmit or receive information to or from the iSBX MULTIMODULE ports. Two interrupt lines are provided to make interrupt requests possible from the iSBX board to the base board. Two option lines are reserved on the bus for unique user requirements, while several power lines provide +5 and ±12 volts to the iSBX boards.

### Bus Pin Assignments

The iSBX bus uses widely available, reliable connectors that are available in 18/36 pin for 8-bit devices and 22/44 pin for 16-bit devices. The male iSBX connector is attached to the iSBX MULTIMODULE board and the female iSBX connector is attached to the base board. Figure 2 shows the dimensions and pin numbering of the 18/36 pin iSBX connector, while Figure 3 does the same for the 22/44 pin iSBX connector. A unique scheme allows the 16-bit female connector to support 8 or 16-bit male MULTIMODULE boards. Table 1 lists the signal/pin assignments for the bus.

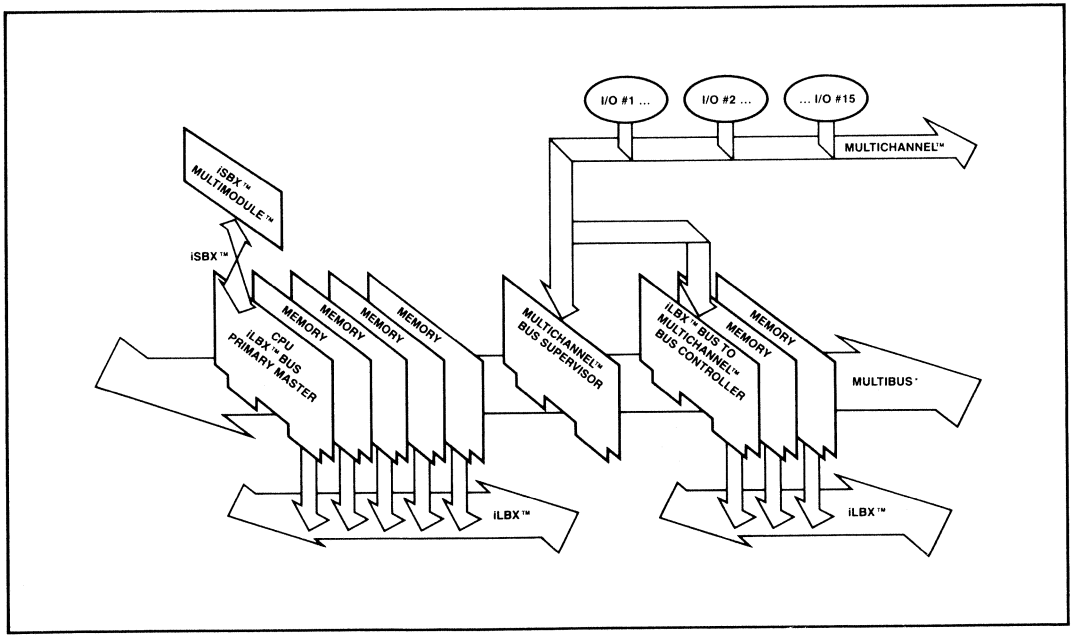



**Figure 1. MULTIBUS® System Architecture**

Table 1. iSBX™ Signal/Pin Assignments

| Pin[1] | Menmonic | Description | Pin[1] | Mnemonic | Description |
|---|---|---|---|---|---|
| 43 | MD8 | MDATA Bit 8 | 44 | MD9 | MDATA Bit 9 |
| 41 | MDA | MDATA Bit A | 42 | MDB | MDATA Bit F |
| 39 | MDC | MDATA Bit C | 40 | MDD | MDATA Bit D |
| 37 | MDE | MDATA Bit E | 38 | MDF | MDATA Bit F |
| 35 | GND | Signal Gnd | 36 | +5V | +5 Volts |
| 33 | MD0 | MDATA Bit 0 | 34 | MDRQT | M DMA Request |
| 31 | MD1 | MDATA Bit 1 | 32 | MDACK/ | M DMA Acknowledge |
| 29 | MD2 | MDATA Bit 2 | 30 | OPT0 | Option 0 |
| 27 | MD3 | MDATA Bit 3 | 28 | OPT1 | Option 1 |
| 25 | MD4 | MDATA Bit 4 | 26 | TDMA | Terminate DMA |
| 23 | MD5 | MDATA Bit 5 | 24 | | Reserved |
| 21 | MD6 | MDATA Bit 6 | 22 | MCS0/ | M Chip Select 0 |
| 19 | MD7 | MDATA Bit 7 | 20 | MCS1/ | M Chip Select 1 |
| 17 | GND | Signal Gnd | 18 | +5V | +5 Volts |
| 15 | IORD/ | I/O Read Cmd | 16 | MWAIT/ | M Wait |
| 13 | IOWRT/ | I/O Write Cmd | 14 | MINTR0 | M Interrupt 0 |
| 11 | MA0 | M Address 0 | 12 | MINTR1 | M Interrupt 1 |
| 9 | MA1 | M Address 1 | 10 | | Reserved |
| 7 | MA2 | M Address 2 | 8 | MPST/ | iSBX Multimodule Board Present |
| 5 | RESET | Reset | 6 | MCLK | M Clock |
| 3 | GND | Signal Gnd | 4 | +5V | +5 Volts |
| 1 | +12V | +12 Volts | 2 | -12V | -12 Volts |

Notes:
1. Pins 37-44 are used only on 8/16-bit systems
2. All undefined pins are reserved for future use.

## Bus Operation Protocol

### COMMAND OPERATION

The iSBX bus supports two types of transfer operations between iSBX elements: I/O Read and I/O Write. An iSBX board can respond to these I/O transfers using either full speed mode or extended mode.

For a full speed I/O Read (Figure 4) the base board generates a valid I/O address and a valid chip select for the iSBX MULTIMODULE board. After set-up, the base board activates the I/O Read line causing the iSBX board to generate valid data from the addressed I/O port. The base board then reads the data and removes the read command, address, and chip select. The full speed I/O Write (Figure 5) operation is similar to the I/O Read except that the base board generates valid data on the lines and keeps the write command line active for the specified a hold time.

The extended Read operation (Figure 6) is used by iSBX MULTIMODULE boards that aren't configured to meet full speed specifications. It's operation is similar to full speed mode, but must use a wait signal to ensure proper

data transfer. The base board begins the operation by generating a valid I/O address and chip select. After set-up, the base board activates the Read line causing the iSBX board to generate a Wait signal. This causes the CPU on the base board to go into a wait state. When the iSBX board has placed valid Read data on the data lines, the MULTIMODULE board will remove the Wait signal and release the base board CPU to read the data and deactivate the command, address, and chip select. The extended Write operation (Figure 7) is similar to the extended Read except that the Wait signal is generated after the base board places valid Write data on the data lines. The iSBX board removes the Wait signal when the write pulse width requirements are satisfied, and the base board can then remove the write command after the hold time is met.

### DMA OPERATION

An iSBX MULTIMODULE system can support DMA when the base board has a DMA controller and the iSBX MULTIMODULE board can support DMA mode. Burst mode DMA is fully supported, but for clarity and simplicity, only a single DMA transfer for an 8-bit base board is discussed.

A DMA cycle (Figure 8) is initiated by the iSBX board when it activates the DMA request line going to the DMA controller on the base board. When the DMA controller gains control of the base board bus, it acknowledges back to the iSBX board and activates an I/O or Memory Read. The DMA controller then activates an I/O or Memory Write respectively. The iSBX board removes the DMA request during the cycle to allow completion of the DMA cycle. Once the write operation is complete, the DMA controller is free to deactivate the write and read command lines after a data hold time.

### INTERRUPT OPERATION

The iSBX MULTIMODULE board on the iSBX bus can support interrupt operations over its interrupt lines. The iSBX board initiates an interrupt by activating one of its two interrupt lines which connect to the base board. The CPU processes the interrupt and executes the interrupt service routine. The interrupt service routine signals the iSBX MULTIMODULE board to remove the interrupt, and then returns control to the main line program when the service routine is completed.

Please refer to the Intel iSBX Bus Specification for more detailed information on its operation and implementation.

## SPECIFICATIONS

### Word Size

**Data** — 8, 16-bit

### Port Assignments

### Power Supply Specifications

**Table 3.**

| Minimum (volts) | Nominal (volts) | Maximum (volts) | Maximum (current)* |
|---|---|---|---|
| +4.75 | +5.0 | +5.25 | 3.0A |
| +11.4 | +12 | +12.6 | 1.0A |
| −12.6 | −12 | −11.4 | 1.0A |
| — | GND | — | 3.0A |

* Per iSBX Multimodule board mounted on base board.

**Table 2. iSBX™ MULTIMODULE™ Base Board Port Assignments**

| iSBX™ Connector Number | Chip Select | 8-Bit Base Board Address | 16-Bit Base Board Address (8-bit mode) | 16-Bit Base Board Address (16-bit mode) |
|---|---|---|---|---|
| iSBX™ 1 | MCS0/<br>MCS1/ | F0-F7<br>F8-FF | 0A0-0AF<br>0B0-0BF | 0A0, 2, 4, 6, 8, A, C, E<br>0A1, 3, 5, 7, 9, B, D, F |
| iSBX™ 2 | MCS0/<br>MCS1/ | C0-C7<br>C8-CF | 080-08F<br>090-09F | 080, 2, 4, 6, 8, A, C, E<br>081, 3, 5, 7, 9, B, D, F |
| iSBX™ 3 | MCS0/<br>MCS1/ | B0-B7<br>B8-BF | 060-06F<br>060-06F | 060, 2, 4, 6, 8, A, C, E<br>061, 3, 5, 7, 9, B, D, F |

## DC Specifications

**Table 4. iSBX™ MULTIMODULE™ Board I/O DC Specifications**

**Output**[1]

| Bus Signal Name | Type[2] Drive | $I_{OL}$ Max Min (mA) | @ Volts ($V_{OL}$ Max) | $I_{OH}$ Max Min ($\mu$A) | @ Volts ($V_{OH}$ Min) | $C_O$ (Min) (pf) |
|---|---|---|---|---|---|---|
| MD0-MDF | TRI | 1.6 | 0.5 | −200 | 2.4 | 130 |
| MINTR0-1 | TTL | 2.0 | 0.5 | −100 | 2.4 | 40 |
| MDRQT | TTL | 1.6 | 0.5 | − 50 | 2.4 | 40 |
| MWAIT/ | TTL | 1.6 | 0.5 | − 50 | 2.4 | 40 |
| OPT1-2 | TTL | 1.6 | 0.5 | − 50 | 2.4 | 40 |
| MPST/ | TTL | Note 3 | | | | |

**Input**[1]

| Bus Signal Name | Type[2] Receiver | $I_{IL}$ Max (mA) | @ $V_{IN}$ MAX (volts) Test Cond. | $I_{IH}$ Max ($\mu$A) | @ $V_{IN}$ MAX (volts) Test Cond. | $C_I$ Max (pf) |
|---|---|---|---|---|---|---|
| MD0-MDF | TRI | −0.5 | 0.4 | 70 | 2.4 | 40 |
| MA0-MA2 | TTL | −0.5 | 0.4 | 70 | 2.4 | 40 |
| MCS0/-MCS1/ | TTL | −4.0 | 0.4 | 100 | 2.4 | 40 |
| MRESET | TTL | −2.1 | 0.4 | 100 | 2.4 | 40 |
| MDACK/ | TTL | − 1.0 | 0.4 | 100 | 2.4 | 40 |
| IORD/ IOWRT/ | TTL | −1.0 | 0.4 | 100 | 2.4 | 40 |
| MCLK | TTL | − 2.0 | 0.4 | 100 | 2.4 | 40 |
| OPT1-OPT2 | TTL | −2.0 | 0.4 | 100 | 2.4 | 40 |

NOTES:
1. Per iSBX Multimodule I/O board.
2. TTL = standard totem pole output. TRI = Three-state.
3. iSBX Multimodule board must connect this signal to ground.

All Inputs: Max $V_{IL}$ = 0.8V
Min $V_{IH}$ = 2.0V

## Connectors

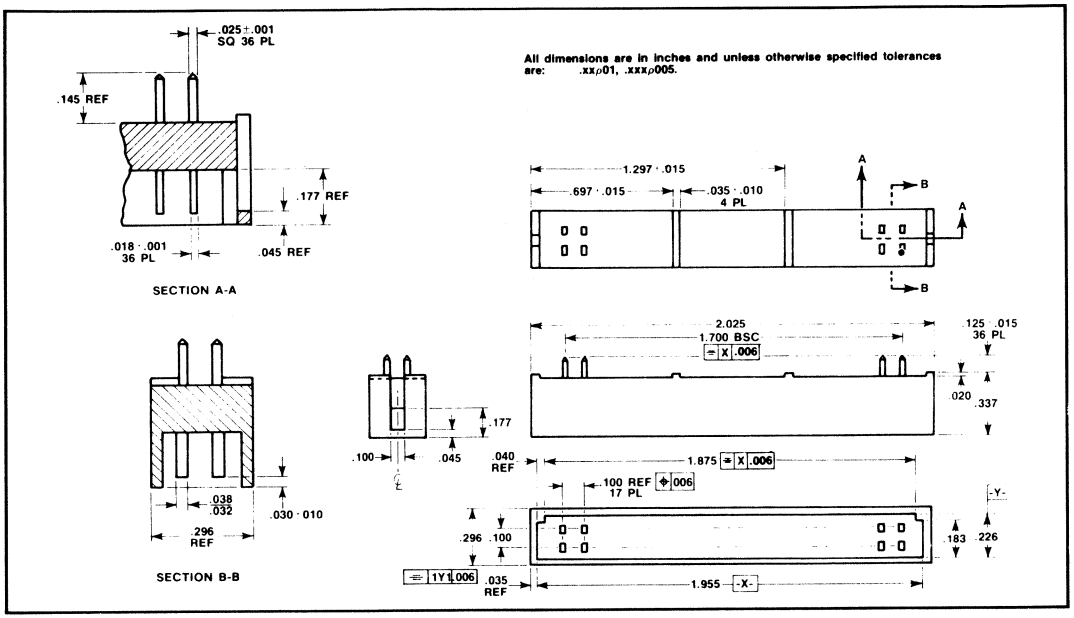



Figure 2. 18/36 Pin iSBX™ Connector

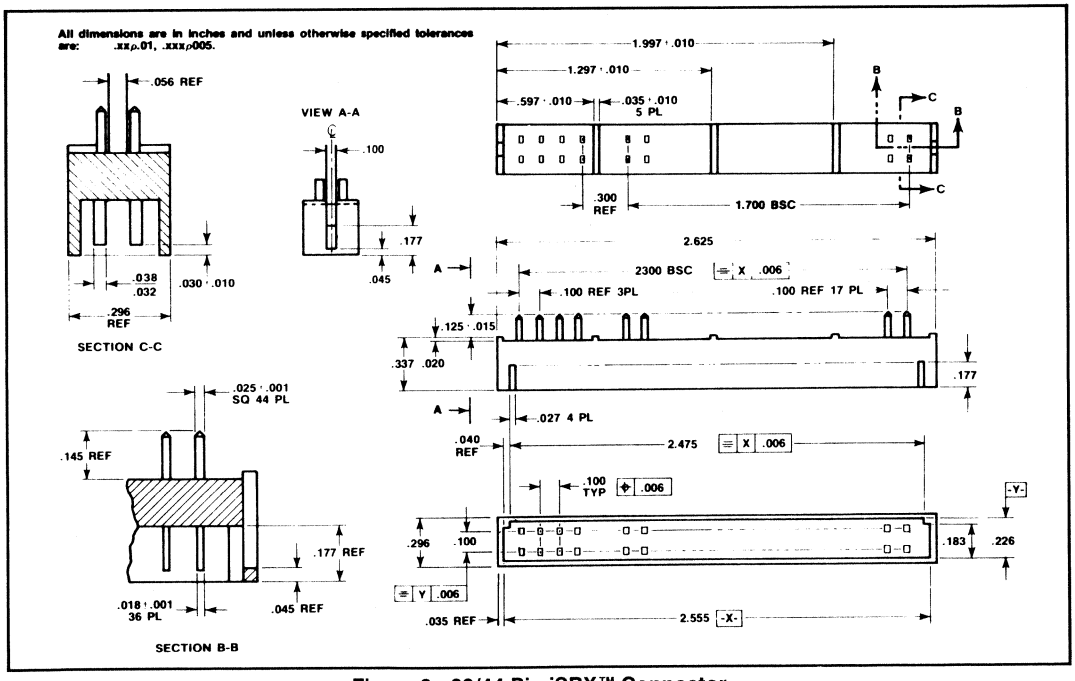



Figure 3. 22/44 Pin iSBX™ Connector

## Bus Timing Diagrams

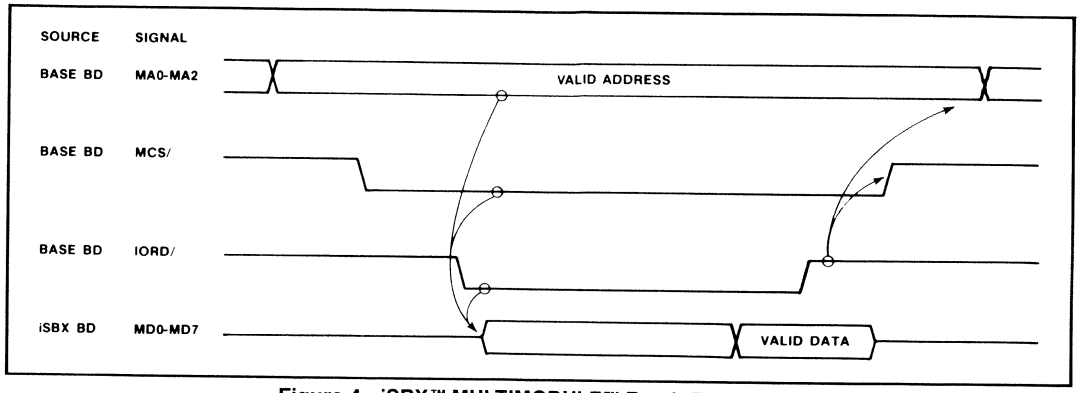



Figure 4. iSBX™ MULTIMODULE™ Read, Full Speed

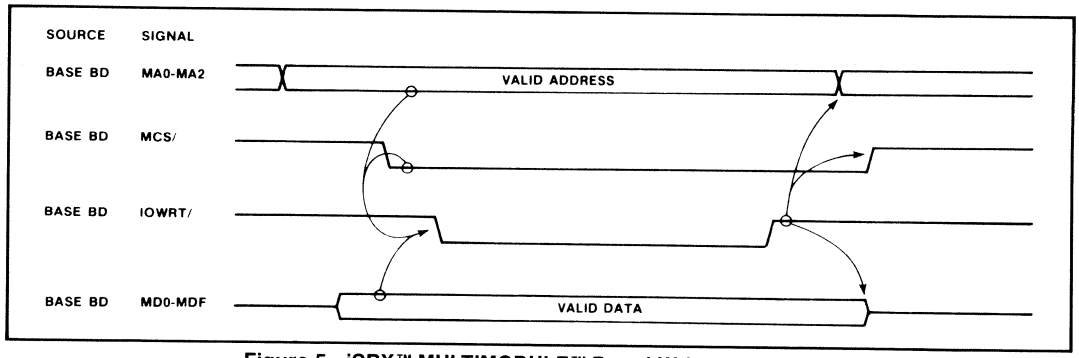



Figure 5. iSBX™ MULTIMODULE™ Board Write, Full Speed

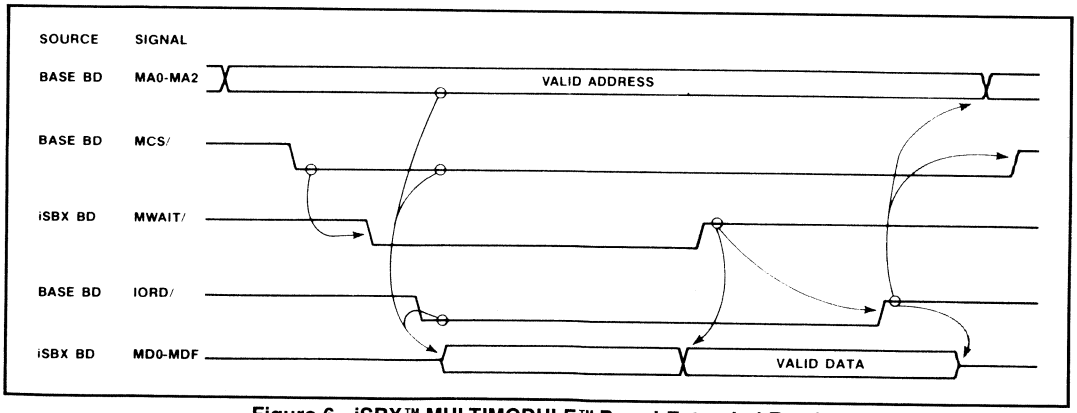



Figure 6. iSBX™ MULTIMODULE™ Board Extended Read

## Bus Timing Diagram (Con't)

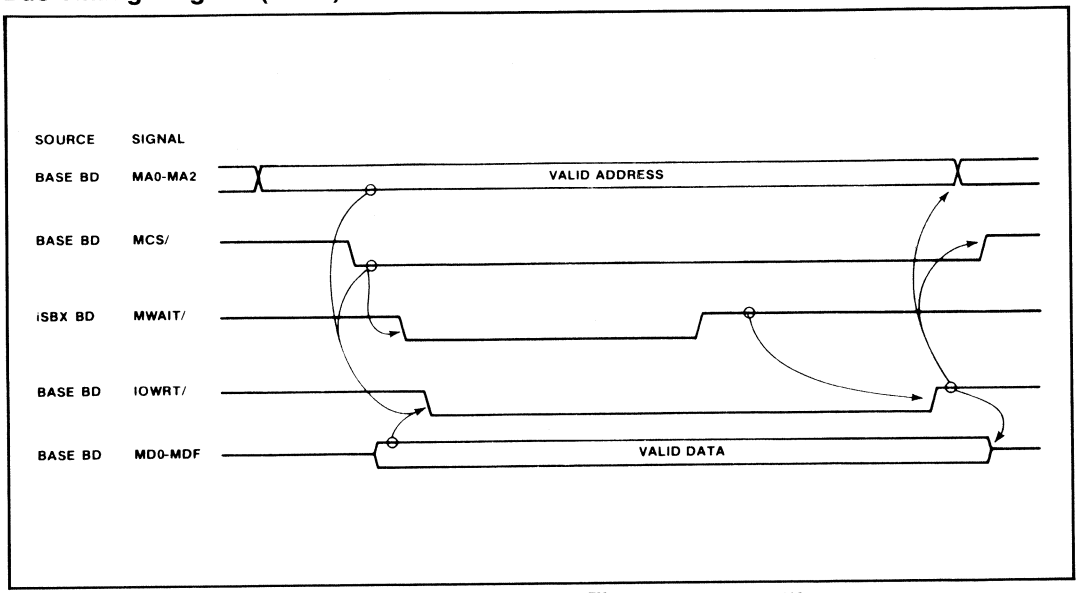



**Figure 7. iSBX™ MULTIMODULE™ Board Extended Write**

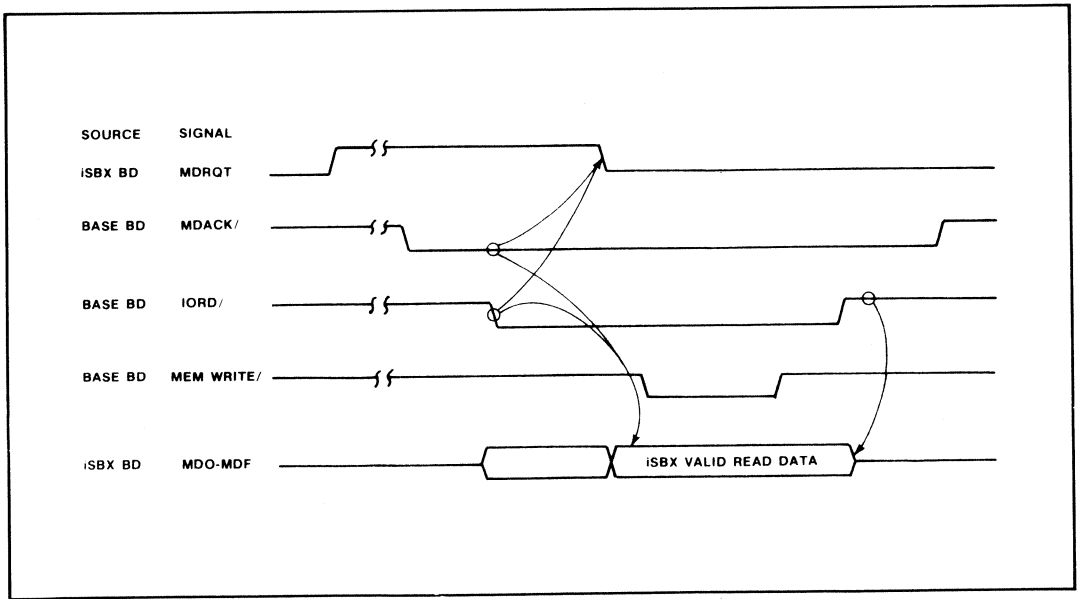



**Figure 8. iSBX™ MULTIMODULE™ Board DMA Cycle (iSBX™ MULTIMODULE™ to Base Board Memory)**

## Board Outlines

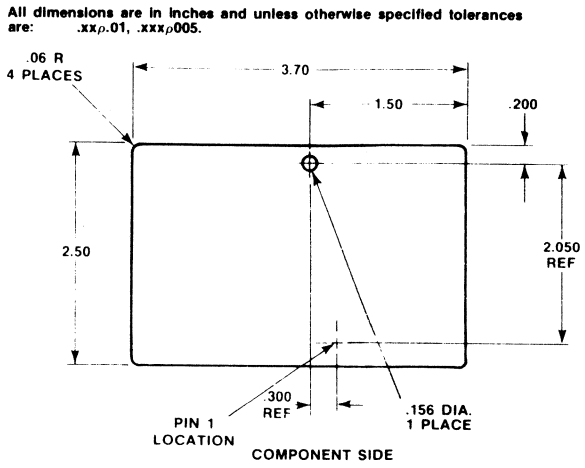



Figure 9. iSBX™ Board Outline

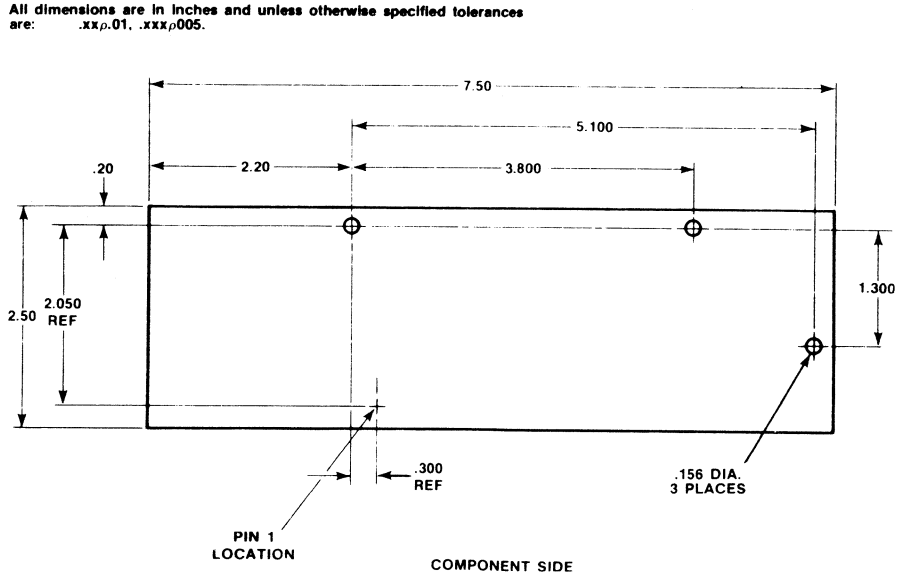



Figure 10. Double Wide iSBX™ Board Outline

## Environmental Characteristics

**Operating Temperature** — 0 to 55°C

**Humidity** — 90% maximum relative; non-condensing

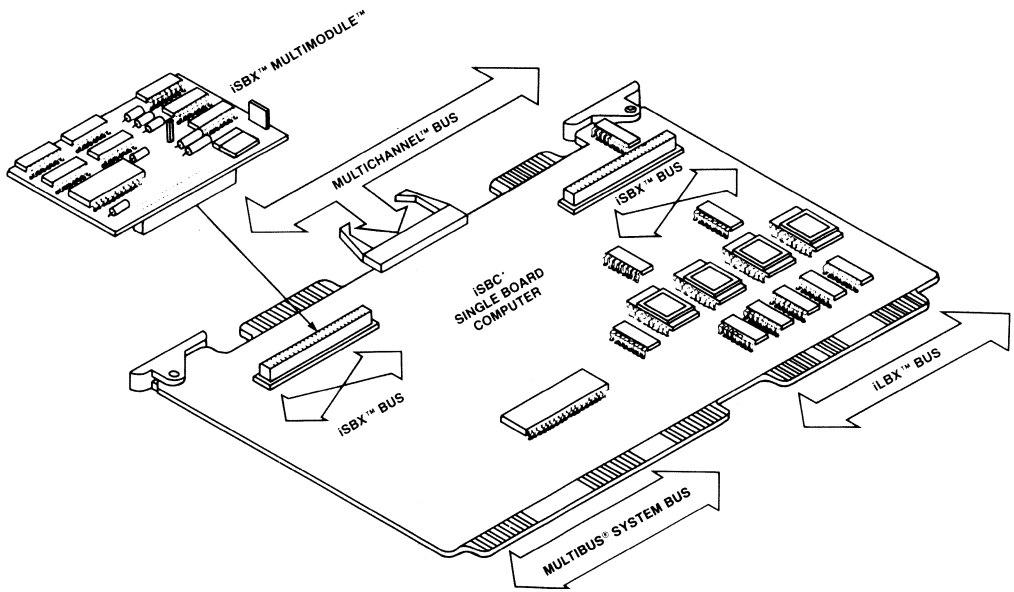
## Reference Manuals

**210883** — MULTIBUS Handbook

# MULTICHANNEL™ I/O BUS

- **High speed 8- or 16-bit block transfers between memory and/or I/O**
- **Transfer rates up to 8 megabytes/sec.**
- **Full speed operation at distances of up to 15 meters.**
- **Supports Supervisor, Controller, or basic Talker/Listener capabilities**
- **Off-loads burst mode I/O activities from host CPU and MULTIBUS® system bus**
- **Up to 16 devices may be interfaced to the bus.**
- **16 megabytes of memory and 16 megabytes of I/O are addressable on each device**

The MULTICHANNEL™ I/O Bus is one of a family of standard bus structures resident within Intel's total system architecture. The MULTICHANNEL bus is a general purpose, high-speed I/O bus capable of significantly increasing system performance by providing a separate data path for DMA I/O activities. By isolating I/O transfers from the system bus, the MULTICHANNEL bus off-loads I/O activity from the host CPU, reduces the probability of bus saturation on the system bus, and reduces contention between I/O and data processing activities on the system bus. The MULTICHANNEL bus can support up to 16 devices at distances up to 15 meters with a maximum burst throughput of 8 megabytes per second. These 16 devices are classified in a manner similar to the IEEE 488 bus concept: Supervisors, Controllers, or Talker and Listeners. As a non-proprietary, standardized I/O bus, the MULTICHANNEL bus is a cost-effective DMA interface ideal for applications such as computer graphics, specialized peripheral control, automatic test equipment, video camera image processing, data acquisition, and high-speed MULTIBUS® system-to-system communication.



## FUNCTIONAL DESCRIPTION

### Architectural Overview

The MULTICHANNEL bus is the standard high speed I/O interface to MULTIBUS-based systems. Its general purpose design and high performance (8 MB/sec) augment the overall system design by improving I/O interface flexibility and system throughput. The flexibility is realized by using an easy-to-use public standard interface that can support up to sixteen 8-bit or 16-bit devices at up to 15 meters. This structure allows the MULTICHANNEL bus to provide easy I/O system expansion, effective box-to-box communication, and a growth path capable of supporting new generations of high-performance I/O devices. The MULTICHANNEL bus increases system throughput by providing a high-performance data path for efficient movement of large amounts of data.

### Structural Features

#### MULTICHANNEL™ BUS CONFIGURATION

The MULTICHANNEL bus is a multiplexed, asynchronous block transfer, 16-bit I/O bus designed to handle 8-bit and 16-bit transfers between peripherals and single board computers. Its structure (pictured in Figure 2) consists of 16 address/data lines, 6 control lines, 2 interrupt lines, plus parity and reset. These signal lines are implemented as either a 60 conductor flat ribbon cable or a twisted-pair cable spanning a distance of up to 15 meters. A 30/60-pin 3M® connector is recommended for device connection to the MULTICHANNEL bus. The male connectors are installed on each MULTICHANNEL device and the female connectors are mounted on the cable. To insure system integrity, the MULTICHANNEL cable is terminated at both ends.

#### BUS ELEMENTS

Three device types — the Basic device, the bus Controller device, and the bus Supervisor device — each provide a different level of capability. The Basic Talker/Listener device has lowest capability, responding only to data transfer requests issued by a Supervisor or Controller. The bus Controller device has higher capability than a Basic Talker/Listener on the bus. It can respond to data transfer requests, control data transfers, and can program other MULTICHANNEL devices under direction from a bus Supervisor. Operating at the highest capability is the bus Supervisor device. It provides major control and management of the MULTICHANNEL bus. The bus Supervisor resolves and grants MULTICHANNEL bus priority, monitors bus status, handles interrupts, and controls the reset line, in addition to performing all bus Controller functions.

3M is a registered trademark of 3M Corporation.

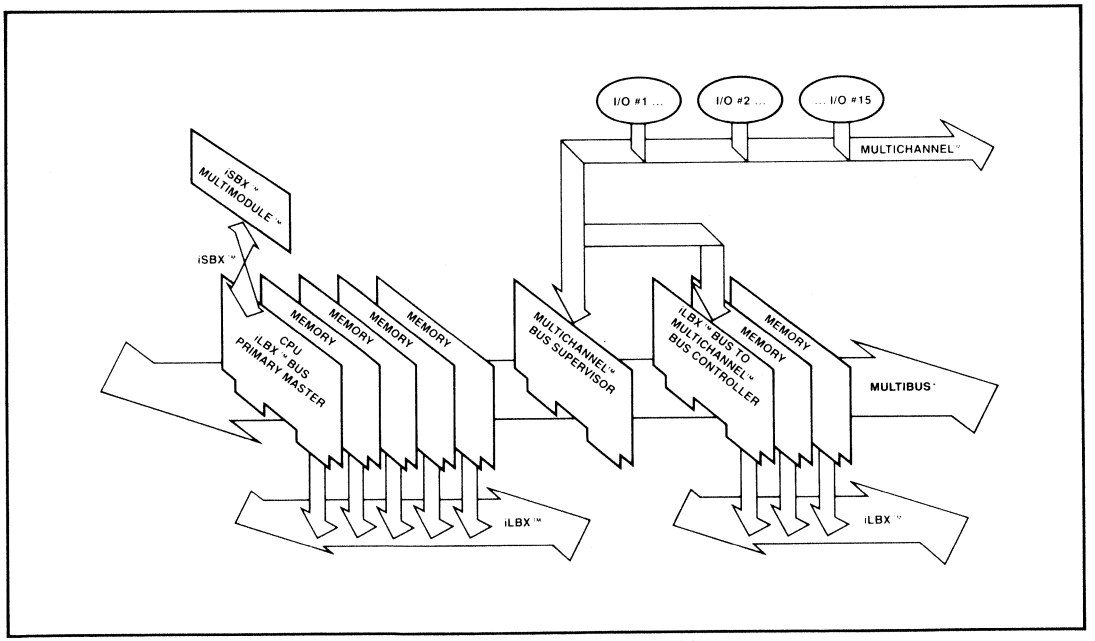



**Figure 1. MULTIBUS® System Architecture**

MULTICHANNEL bus devices are functionally flexible, creating overlaps between types of bus functions and types of bus devices performing those functions. These devices perform functions in various states of operation: master, slave, talker, listener. When a device is controlling the command/action lines, it is in the master state, and both the bus Supervisor and the bus Controller can operate in this state, although not simultaneously. The slave state indicates a device that can monitor the command/action lines. Only Controllers and Basic Talker/Listeners operate as slaves. All three device types can operate in the talker state or the listener state, but not all at the same time. A Talker is any device selected by the bus master which is writing data to the bus. A Listener is any selected device which is reading data from the bus.

## BUS INTERFACE/SIGNAL LINE DESCRIPTIONS

The MULTICHANNEL bus signal lines are grouped into five classes based on the functions they perform: address/data, control, interrupt, parity, and reset. The 16 address/data lines are multiplexed by a control line to act either as 16 unidirectional address lines or 16 bidirectional data lines. When used as address lines, they transmit the device address to all devices attached to the MULTICHANNEL bus. When used as bidirectional data lines, they transmit and receive data to or from MULTICHANNEL devices. The six control lines determine the overall operation of the bus from specifying the type of data transfer to providing the handshake for data transfers between MULTICHANNEL devices. Two interrupt lines are supplied to initiate and terminate data transfers, and to indicate device failures, memory failures, or parity errors. A parity line and a reset line provide support for a parity option and system reset capability whenever required.

## BUS PIN ASSIGNMENTS

For proper MULTICHANNEL implementation, a 60 conductor (twisted pair or flat) cable using a 30/60 pin 3M connector, is used for device connection to the bus. Figure 3 is an outline drawing of the iSBC® MULTICHANNEL connector which also shows the pin numbering. The MULTICHANNEL bus connector signal pin assignments are listed in Table 1. Cable termination is implemented at both cable ends to insure proper system integrity over a 15-meter cable. Figure 4 is a schematic of the cable termination circuits. A cable termination module could be created that would then be connected to the cable end via a 30/60 pin connector.

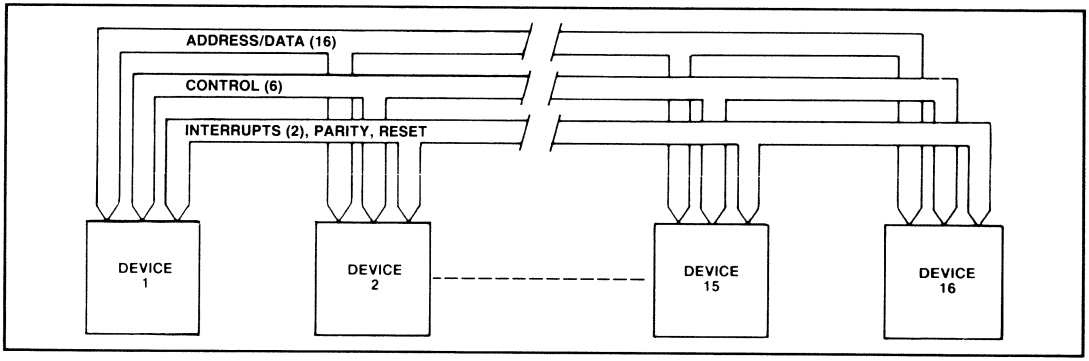



**Figure 2. Block Diagram of MULTICHANNEL™ Bus Structure**

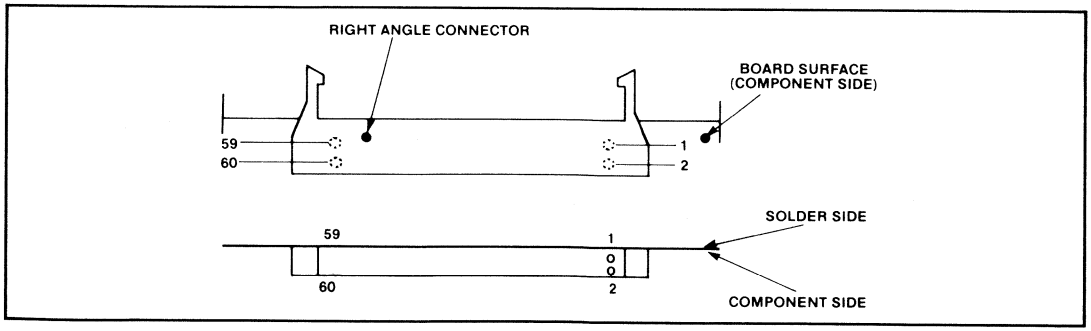



**Figure 3. Connector Example**

**Table 1. MULTICHANNEL™ Bus Pin Assignments**

| Lower Row | | | Upper Row | | |
|---|---|---|---|---|---|
| **Pin** | **Mnemonic** | **Signal Name** | **Pin** | **Mnemonic** | **Signal Name** |
| 1 | GND | GROUND | 2 | AD0/ | ADDRESS DATA LINE 0 |
| 3 | GND | GROUND | 4 | AD1/ | ADDRESS DATA LINE 1 |
| 5 | GND | GROUND | 6 | AD2/ | ADDRESS DATA LINE 2 |
| 7 | GND | GROUND | 8 | AD3/ | ADDRESS DATA LINE 3 |
| 9 | GND | GROUND | 10 | AD4/ | ADDRESS DATA LINE 4 |
| 11 | GND | GROUND | 12 | AD5/ | ADDRESS DATA LINE 5 |
| 13 | GND | GROUND | 14 | AD6/ | ADDRESS DATA LINE 6 |
| 15 | GND | GROUND | 16 | AD7/ | ADDRESS DATA LINE 7 |
| 17 | GND | GROUND | 18 | AD8/ | ADDRESS DATA LINE 8 |
| 19 | GND | GROUND | 20 | AD9/ | ADDRESS DATA LINE 9 |
| 21 | GND | GROUND | 22 | ADA/ | ADDRESS DATA LINE 10 |
| 23 | GND | GROUND | 24 | ADB/ | ADDRESS DATA LINE 11 |
| 25 | GND | GROUND | 26 | ADC/ | ADDRESS DATA LINE 12 |
| 27 | GND | GROUND | 28 | ADD/ | ADDRESS DATA LINE 13 |
| 29 | GND | GROUND | 30 | ADE/ | ADDRESS DATA LINE 14 |
| 31 | GND | GROUND | 32 | ADF/ | ADDRESS DATA LINE 15 |
| 33 | GND | GROUND | 34 | RESET/ | RESET |
| 35 | GND | GROUND | 36 | AACC | ADDRESS MODE ACCEPT |
| 37 | GND | GROUND | 38 | SRQ/ | SERVICE REQUEST |
| 39 | GND | GROUND | 40 | STO/ | SUPERVISOR TAKE OVER |
| 41 | GND | GROUND | 42 | DACC/ | DATA MODE ACCEPT |
| 43 | GND | GROUND | 44 | SA/ | SUPERVISOR ACTIVE |
| 45 | PB*/ | PARITY BIT (INV.) | 46 | PB/ | PARITY BIT |
| 47 | R/W/ | READ NOT WRITE (INV.) | 48 | R/W | READ NOT WRITE |
| 49 | A/D/ | ADDRESS NOT DATA (INV.) | 50 | A/D | ADDRESS NOT DATA |
| 51 | DRDY*/ | DATA READY (INV.) | 52 | DRDY/ | DATA READY |
| 53 | RES | RESERVED | 54 | RES | RESERVED |
| 55 | RES | RESERVED | 56 | RES | RESERVED |
| 57 | RES | RESERVED | 58 | RES | RESERVED |
| 59 | RES | RESERVED | 60 | RES | RESERVED |

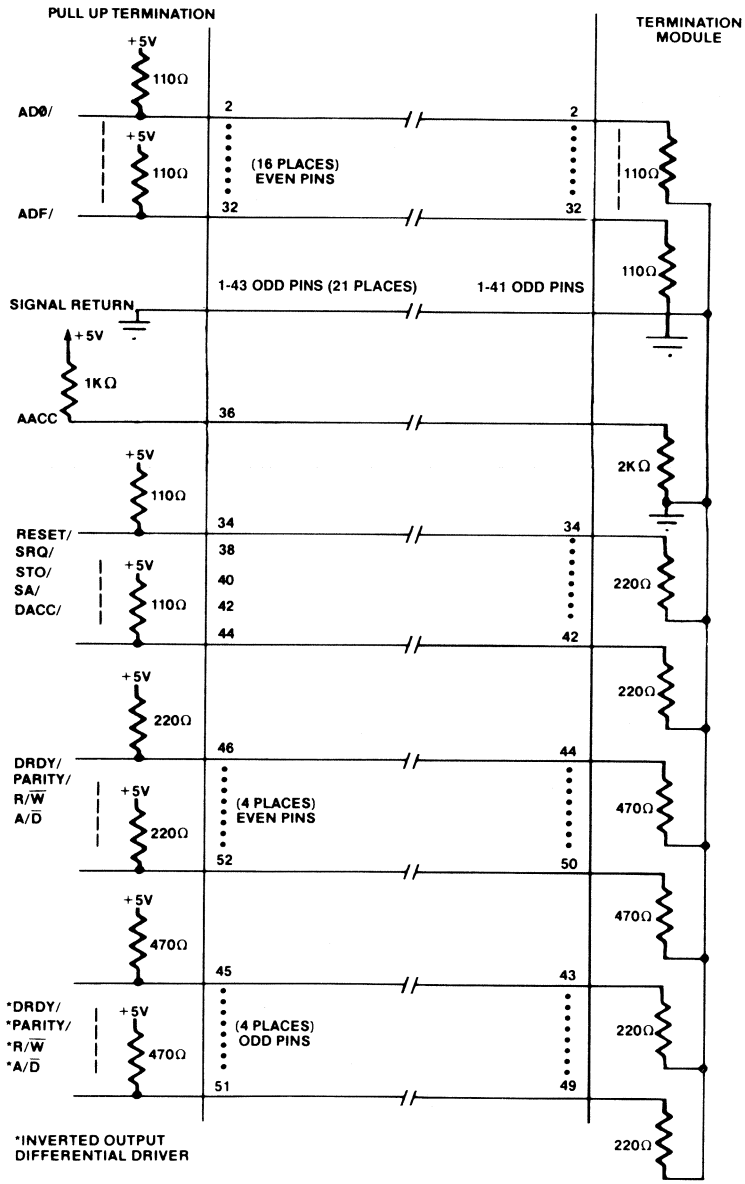



**Figure 4. Bus Termination Schematic**

## Bus Operation Protocol

### DATA TRANSFER OPERATION

There are three modes of communication in the operation protocol: address mode, data mode, and control transfer mode. Using these transfer modes, each MULTICHANNEL device provides handshaking capability for totally asynchronous block data transfers. Address mode is the time when the address/data control line is high. Information placed on the address/data lines of the MULTICHANNEL bus as two successive 16-bit words is interpreted to select or deselect a device on the bus and address the specific resource on the device. Typically, these address mode transfers are only 2 word sequences. Figure 5 is a timing diagram of the handshake routine in address mode. The data mode is the time when the address/data control line is low. Valid data is placed on the address/data lines of the bus and can occur only after an address mode has been performed. Transfers during data mode are usually large quantities of either 8- or 16-bit data, and are passed to or from the addressed device until it is deselected. Figure 6 is a timing diagram of a data transfer sequence.

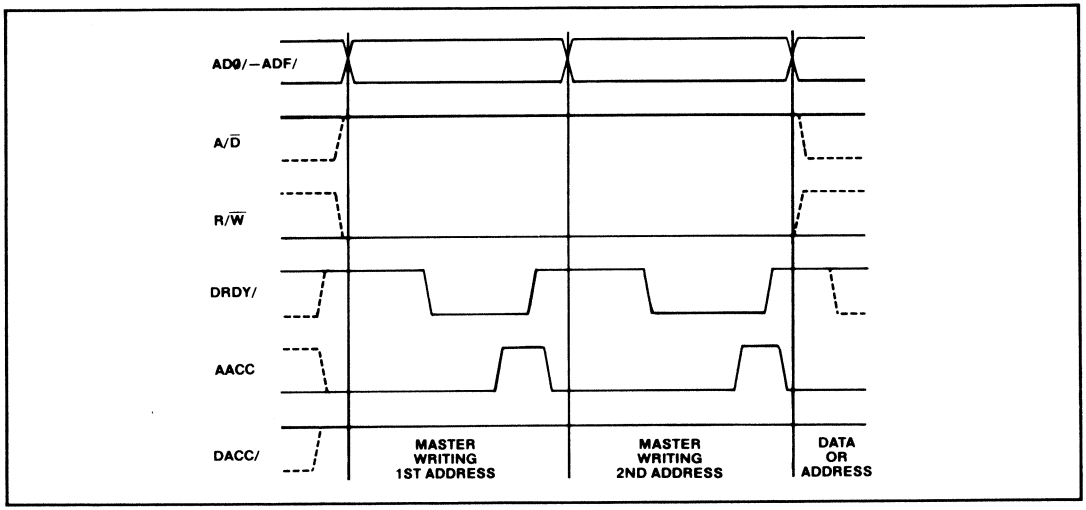



**Figure 5. MULTICHANNEL™ Bus Address Cycle**

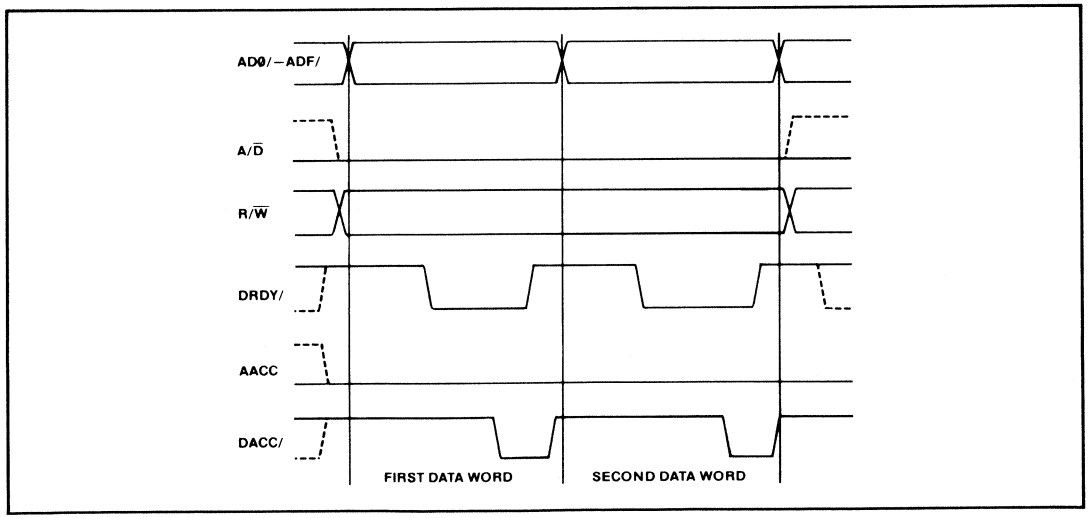



**Figure 6. MULTICHANNEL ™ Bus Data Transfer Sequence**

Control transfer mode is the time when the bus Supervisor selects the bus Controller and programs its registers with required information. Once programmed, a bus Controller may select a device and originate a data transfer operation.

The operational sequences of these transfer modes are similar in handling read and write operations to and from the 16 megabytes of memory and the 16 megabytes of registers addressable on each MULTICHANNEL device.

A typical transfer sequence begins when the master sends a two-word address sequence to select a MULTICHANNEL device and specify address, direction and resource (memory vs. I/O) of the data transfer. Following device selection, the Talker proceeds to send the data as a continuous 8 or 16-bit data word stream until the block data move is complete. The master terminates the transfer by issuing another two-word address sequence for device deselection.

The transfer sequence described is identical for both memory and register type transfers. The master controls similar read and write operations between devices, and the address select and deselect sequences use the same address format. Figure 7 contains the MULTICHANNEL bus address format.

## DEVICE REGISTER DEFINITION

Of the 16 megabytes of register space per device, the first 16 registers are pre-defined to provide a standard register area common to all devices. The remaining registers are user definable. Table 2 lists the 16 defined registers along with their function. The use of this register concept allows for standard interface between all MULTICHANNEL devices. Please refer to the MULTICHANNEL Bus Specification for more detailed information.

**Table 2. MULTICHANNEL™ Device Register Definitions**

| Register Number | Definition | Mode |
|---|---|---|
| 0 | STO/ Flag/Status | Read Only |
| 1 | SRQ/ Flag/Status | Read Only |
| 2 | SRQ/ Mask | Write Only |
| 3 | Device Command | Write Only |
| 4 | Device Parameter | Write Only |
| 5 | Data Address 1 | Read or Write |
| 6 | Data Address 2 | Read or Write |
| 7 | Block Length 1 | Read or Write |
| 8 | Block Length 2 | Read or Write |
| 9 | Error Address 1 | Read Only |
| 10 | Error Address 2 | Read Only |
| 11 | Address Extension | Write Only |
| 12-15 | Reserved | |
| 16-16 Mbyte | User Defined | Read or Write |

## BUS INTERRUPT HANDLING

The MULTICHANNEL bus Supervisor, being responsible for bus access and control, monitors the two bus interrupt lines. The Supervisor Take-Over interrupt (STO) is used to inform the bus Supervisor that a device wants to return control of the bus to the Supervisor or that an error has occurred. The Service Request Interrupt (SRQ) is used by devices which do not have control of the bus, but require service from the bus Supervisor. To locate a device transmitting a bus interrupt, the bus Supervisor

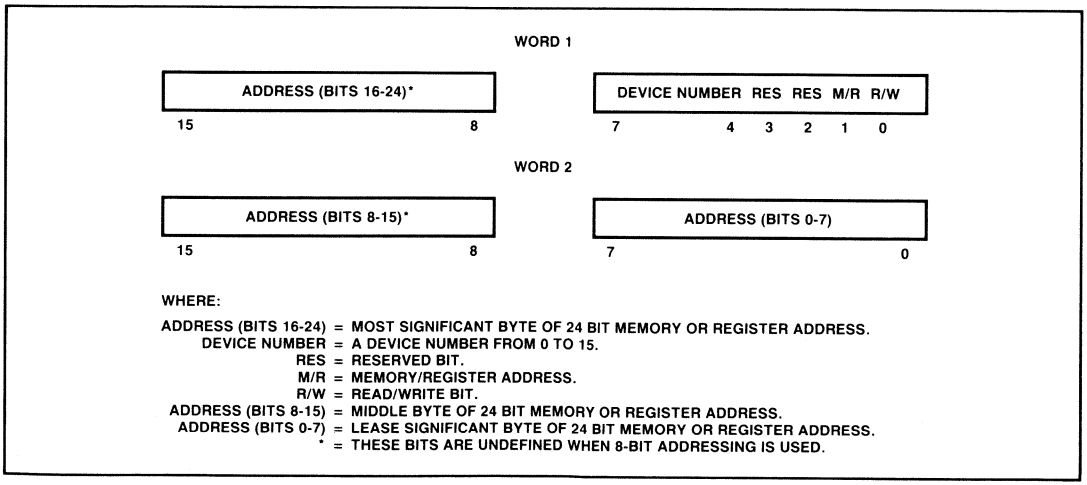



**Figure 7. MULTICHANNEL™ Bus Address Format**

polls each device attached to the bus by reading the appropriate register of each device and testing for a non-zero value. In current implementations, the Supervisor polls each device only once. If the interrupt is not removed an error occurs.

### PARITY AND RESET

Parity operation on the MULTICHANNEL bus is provided, but is not required. The bus Supervisor selects between parity mode and non-parity mode depending upon system requirements. If parity mode is selected all Talkers must generate odd parity. All active Listeners monitor the parity line and generate an STO interrupt signal if there is a parity error.

A reset function is also supported by the MULTICHANNEL bus, and is controlled by the bus Supervisor to bring the bus to a known state. It is used to reset all devices after power-up, and when required to gain control of the bus.

## SPECIFICATIONS

### Word Size

**Data** — 8, 16-bit

### Memory Addressing

**24-bits** — 16 megabyte - direct access - automatic incrementing

### Register Addressing

**24-bits** — 16 megabyte - direct access

### Electrical Characteristics

#### DC SPECIFICATIONS

### Maximum Bus Length

15 meters (50 feet)

### Bus Devices Supported

**16 total devices** — (Supervisor, Controller, and Talker/Listener)

### Bus Bandwidth

**8 megabytes/sec.** — 16-bit

**4 megabytes/sec.** — 8-bit

**Table 3. DC Specifications**

| Signal Name | Driver Type | Termination (see Note) | Min. Driver Requirements | | | Max. Receiver Requirements | | |
|---|---|---|---|---|---|---|---|---|
| | | | High | Low | Load Cap | High | Low | Load Cap |
| AD15-0/ | TRI-STATE | 110 Ohms | - 5 ma | 48 ma | 300 pf | 0.2 ma | 0.8 ma | 15 pf |
| SA/ | OPEN COLL | 110/220 Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| RESET/ | OPEN COLL | 110/220 Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| AACC | OPEN COLL | 1K/2K Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| DACC/ | OPEN COLL | 110/220 Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| SRQ/ | OPEN COLL | 110/220 Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| STO/ | OPEN COLL | 110/220 Ohms | N.A. | 48 ma | 300 pf | 0.4 ma | 0.6 ma | 15 pf |
| R/W | DIF, NON-INV | 220/470 Ohms | - 20 ma | 40 ma | 300 pf | 0.5 ma | 0.5 ma | 15 pf |
| R/W/ | DIF, INV | 470/220 Ohms | - 20 ma | 40 ma | 300 pf | 0.5 ma | 0.5 ma | 15 pf |
| A/D | DIF, NON-INV | 220/470 Ohms | - 20 ma | 40 ma | 300 pf | 0.5 ma | 0.5 ma | 15 pf |
| A/D/ | DIF, INV | 470/220 Ohms | - 20 ma | 40 ma | 300 pf | 0.5 ma | 0.5 ma | 15 pf |
| PB/ | DIF, NON-INV | 220/470 Ohms | - 20 ma | 40 ma | N.A. | 0.5 ma | 0.5 ma | N.A. |
| PB*/ | DIF, INV | 470/220 Ohms | - 20 ma | 40 ma | N.A. | 0.5 ma | 0.5 ma | N.A. |
| DRDY/ | DIF, NON-INV | 220/470 Ohms | - 20 ma | 40 ma | N.A. | 0.5 ma | 0.5 ma | N.A. |
| DRDY*/ | DIF, INV | 470/220 Ohms | - 20 ma | 40 ma | N.A. | 0.5 ma | 0.5 ma | N.A. |

**NOTE:** Termination provided only at the physically ends of the interconnect cable. Where the positive termination (pull-up) resistance is different from the negative termination (pull-down) resistance, the positive termination resistance is listed first.

## BUS TIMING

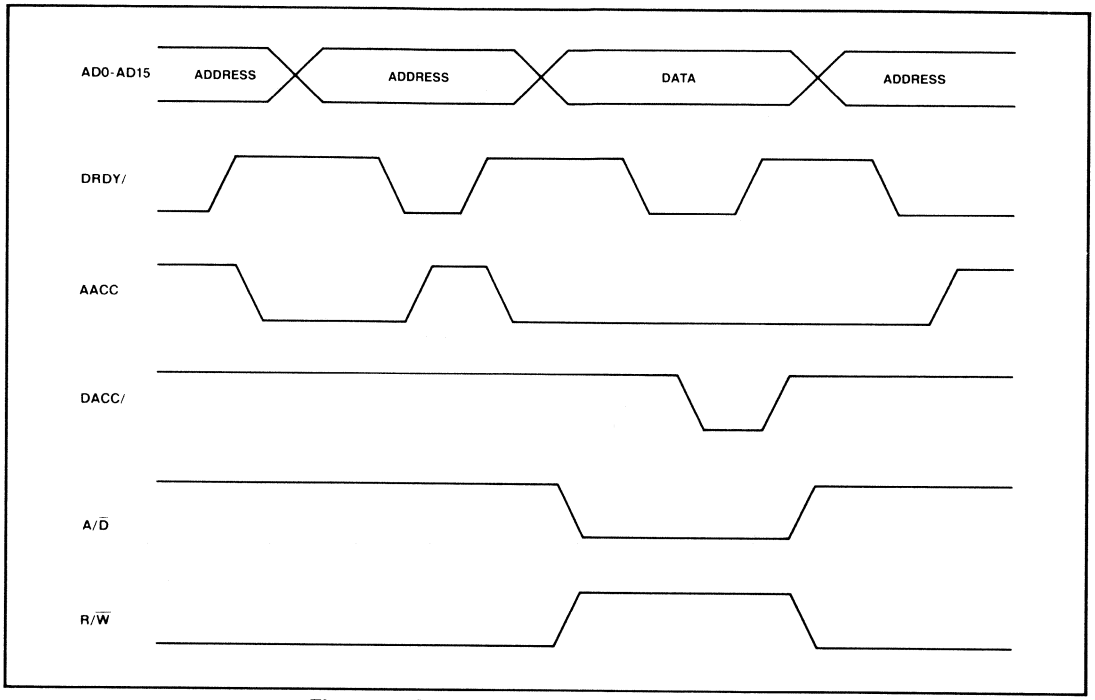



**Figure 8. Address-Read-Address-Write Cycles**

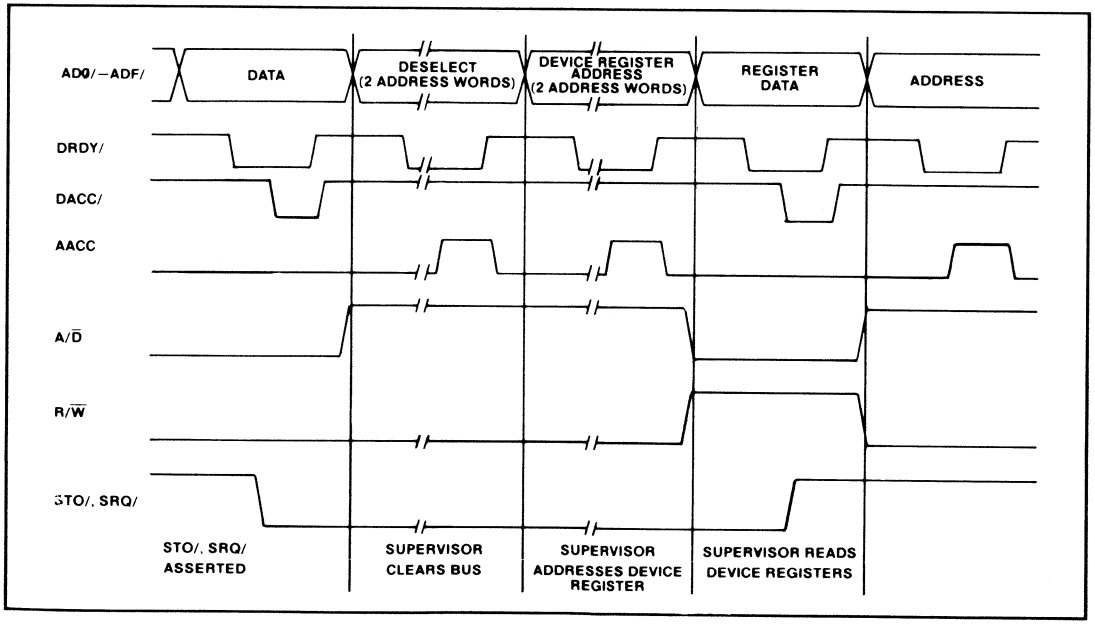



**Figure 9. Supervisor Interrupt Timing**

## Cables and Connectors

**Table 3. Cable and Receptacle Vendors**

| MULTICHANNEL™ Bus Compatible Cable | | | |
|---|---|---|---|
| **Vendor** | **Ribbon Type** | **Vendor No.** | **Conductor** |
| Belden | Plain Flat | 9L28060 | 60 |
| Belden | Twisted-Pair | 9V28060 | 60 |
| Belden | Insulated Flat | 9L28260 | 60 |
| Spectrastrip | Plain Flat | 455-240-60 | 60 |
| Spectrastrip | Twisted-Pair | 455-248-60 | 60 |
| Spectrastrip | Insulated Flat | 151-2830-060 | 60 |
| **MULTICHANNEL™ Bus Compatible Receptacles** | | | |
| **Vendor** | **Type** | **Vendor No.** | **Pins** |
| Berg | Male | 65823-103 | 60 |
| Berg | Female | 65949-960 | 60 |
| 3M | Male | 3372-1302 | 60 |
| 3M | Female | 3334-6000 | 60 |

### PHYSICAL PROPERTIES

**Conductors** — 28 AWG, 7/36 strand, tinned copper

**Conductor Insulation** — 0.010 inch wall, nominal

**Conductor Spacing** — Twisted pair – 0.10 inch, nominal; Flat – 0.050 inch, ± 10%

**Cable Thickness** — Flat – 0.042 inch, nominal

**Temperature Rating** — 80°C

### ELECTRICAL PROPERTIES

**Impedance (nominal)** — 105 ohms ± 10%

**Propagation Velocity (nominal)** — 1.7 ns/ft

**Capacitance (nominal)** — 22 pf/ft

### INSULATION REQUIREMENTS

**Voltage Rating (minimum)** — 100 Vdc

**Insulation Resistance (minimum)** — $1 \times 10^{10}$ ohms

## Environmental Characteristics

**Temperature** — 0 - 55°C

**Humidity** — 90% max. relative (no condensation)

## Reference Manuals

**210883** — MULTIBUS Architecture Handbook.

Intel Corporation
OEM Systems Operation
5200 NE Elam Young Parkway
Hillsboro, Oregon 97123
(503) 640-7800

Printed in U.S.A./OMO/15K/BA/SL
**Order Number: 230893**