INTERNATIONAL STANDARD

ISO/IEC 10861

ANSI/IEEE Std 1296

First edition
1994-04-27

Information technology— Microprocessor systems— High-performance synchronous 32-bit bus: MULTIBUS II

Technologies de l'information –
Systèmes à microprocesseurs –
Bus 32 bits synchrone à haute performance:
MULTIBUS II

Reference number
ISO/IEC 10861 : 1994(E)
ANSI/IEEE
Std 1296, 1994 Edition

Abstract: The operation, functions, and attributes of a parallel system bus (PSB), called MULTIBUS II, are defined. A high-performance backplane bus intended for use in multiple processor systems, the PSB incorporates synchronous, 32-bit multiplexed address/data, with error detection, and uses a 10 MHz bus clock. This design is intended to provide reliable state-of-the-art operation and to allow the implementation of cost-effective, high-performance VLSI for the bus interface. Memory, I/O, message, and geographic address spaces are defined. Error detection and retry are provided for messages. The message-passing design allows a VLSI implementation, so that virtually all modules on the bus will utilize the bus at its highest performance—32 to 40 Mbyte/s. An overview of PSB, signal descriptions, the PSB protocol, electrical characteristics, and mechanical specifications are covered.
Keywords: high-performance synchronous 32-bit bus, MULTIBUS II, system bus architectures

---

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

ISBN 1-55937-368-7

April 27, 1994 *SH16766*

ISO/IEC 10861 : 1994
[ANSI/IEEE Std 1296, 1994 Edition]

Information technology—Microprocessor systems—High-performance synchronous 32-bit bus: MULTIBUS II

Sponsor

Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society

Adopted as an International Standard by the International Organization for Standardization and by the International Electrotechnical Commission

Published by The Institute of Electrical and Electronics Engineers, Inc.

American National Standard

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and nongovernmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

In 1990, ANSI/IEEE Std 1296-1987 was adopted by ISO/IEC JTC 1, as draft International Standard ISO/IEC/DIS 10861. This draft was subsequently approved by ISO/IEC JTC 1 in the form of this edition, which is published as International Standard ISO/IEC 10861 : 1994.

International Organization for Standardization/International Electrotechnical Commission
Case postale 56 • CH-1211 Genève 20 • Switzerland

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Introduction

(This introduction is not a normative part of ISO/IEC 10861 : 1994 [ANSI/IEEE Std 1296, 1994 Edition], but is included for information only.)

In the last decade, the avalanche of new microcomputer technology, especially VLSI, threatened to obsolete products almost before they went into production. To buffer users from this onrush of technology, Intel helped develop standard interfaces. One of the most notable was the MULTIBUS I system bus, which was used as the basis for a standard by the IEEE in 1983 as IEEE Std 796-1983 (after going through a 5-year review and revision process).

In the early 1980s, Intel recognized that the trends toward multiprocessing and more sophisticated microcomputer-based systems called for an advanced 32-bit system bus architecture. Intel called this new bus MULTIBUS II. In continuing to pioneer the open systems technology, which included multiprocessing, four critical requirements were observed: technical credibility, processor independence, standardization, and openness to all levels of integration. Early in the development of the new bus, Intel established a "MULTIBUS II Development Consortium." The consortium gave the new bus a technical credibility that few buses, especially those defined only among board vendors, can match. The companies in the consortium also represented all microprocessor families; included in the group were 68020, 32032, 80386, and Z8000 board and system users, thus ensuring that the bus is easily adaptable to virtually any manufacturer's processor.

The primary benefits being sought in the creation of this new bus were high-performance multiprocessing, high system reliability, ease-of-use by system designers, and improved cost/performance.

Specific bus features were developed in response to these objectives. The 32 Mbyte/s message passing of the bus provides a bus that acts like a very high-speed network connection for multiple processors (or processor equivalents). There is a recognition that the bus is no longer to interconnect a CPU with its memory and I/O; instead the bus is to interconnect whole stand-alone processors with each other and with intelligent "subsystems-on-a-board."

System reliability is enhanced by the features of bus parity, synchronous operation, negative acknowledge, transfer retries, geographic addressing, and advanced backplane design. Ease-of-use by system designers is implemented primarily through the geographic addressing, which provides for dynamic system configuration. The bus encourages the use of software programmable configuration options (and discourages any use of mechanical jumpers). The standardization of the high-level message-passing protocol also gives the system designer an easy-to-use capability for interprocessor communication.

The cost/performance objective of the bus is delivered through its specification of a realizable 32 to 40 Mbyte/s bus bandwidth. Virtually all boards designed to the bus can achieve this bus utilization factor due to the high-level protocol called out in the specification, and thus the availability of standard, high-performance and cost-effective VLSI components to actually implement this level of performance. For example, this specification and the VLSI make it possible for eight concurrent 4 megabyte/second transfers to take place on the bus. This, or other combinations of transfers that add up to 32 Mbyte/s, demonstrate the real cost/performance advantages of the bus for multiprocessor applications.

In 1983 MULTIBUS II was introduced to the IEEE standards process as a part of the considerations for the P896 (Future Bus) working group activities. In the 1984/1985 time frame the MSC (Microcomputer Standards Committee, of the TCMM) formed an independent study group for MULTIBUS II. During this time the many active participants of the group proceeded to thoroughly review and make changes to the proposed draft. In early 1986 the group was assigned a formal project number P1296. During the remainder of 1986, the draft was passed by the Working Group and the MSC after thorough review, discussion, and changes. In 1987, the draft was presented for Sponsor ballot and, after passing, presented to the June 1987 meeting of the IEEE Standards Board.

The IEEE Standards Board calls attention to the fact that there are patents claimed and/or pending on many aspects of this bus by Intel Corporation. IEEE takes no position with respect to patent validity. Intel Corporation has assured the IEEE that it is willing to grant a license for these patents on reasonable and nondiscriminatory terms to anyone wishing to obtain such a license. The general terms of the license are a one-time administration fee of $100 for a nonexclusive perpetual license. Intel Corporation's undertakings in this respect are on file obtained from the legal department of Intel Corporation whose address is Intel Corporation, 5200 N.E. Elam Young Parkway, Hillsboro, OR 97124.

There were many contributors to the standards review process, but the following members deserve special mention for their active participation:

Task Force Coordinators:	Jack Blevins
	Maurice Hubert
	Hubert Kirrmann
	Jim Nebus
Secretary of Working Group:	Steve Cooper
Original Study Group Chairman:	Paul Borrill
Original Draft Editor:	Scott Tetrick

The P1296 Working Group that prepared this standard had the following membership:

Richard W. Boberg, *Chair*

Web Augustine	Gene Freehauf	Ken Smith
Jack Blevins	Maurice Hubert	Michael Thompson
Paul Borrill	Hubert Kirrmann	Scott Tetrick
Steve Cooper	Klaus Mueller	Eike Waltz
Tom Crawford	Jim Nebus	Janusz Zalewski
	Don Nickel	

The following members of the Technical Committee on Microprocessors and Microcomputers were on the balloting body:

Andrew Allison	Martin Freeman	Deene Ogden
Peter J. Ashenden	David Gustavson	Tom Pittman
Matt Biewer	Tom Harkaway	Shlomo Pri-Tal
John Black	Dave Hawley	P. Reghunathan
Jack Blevins	David James	Richard Rawson
Richard Boberg	Laurel Kaleda	Bill Shields
Paul Borrill	Richard Karpinski	Michael Smolin
Bradley Brown	Hubert Kirrman	Robert Stewart
Clyde Camp	Doug Kraft	Subramanganesan
John D. Charlton	Tom Kurihara	Michael Teener
Steve Cooper	Glen Langdon	Scott Tetrick
Randy Davis	Gerry Laws	Eike Waltz
J. Robert Davis	Tom Leonard	Carl Warren
Shirish P. Deodhar	Rollie Linser	George White
Jim Dunlay	Gary Lyons	Fritz Whittington
Wayne Fischer	James Nebus	Tom Wicklund
Jim Flournoy	Gary Nelson	Andrew Wilson
Gordon Force		Anthony Winter

When the IEEE Standards Board approved this standard on June 11, 1987, it had the following membership:

Donald C. Fleckenstein, *Chair* **Marco W. Migliaro,** *Vice Chair*
Andrew G. Salem, *Secretary*

James H. Beall
Dennis Bodson
Marshall L. Cain
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster
Kenneth D. Hendrix
Irvin N. Howell
Leslie R. Kerr
Jack Kinn
Irving Kolodny
Joseph L. Koepfinger*
Edward Lohse
John May
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
L. John Rankine
John P. Riganati
Gary S. Robinson
Frank L. Rose
Robert E. Rountree
Sava I. Sherr*
William R. Tackaberry
William B. Wilkens
Helen M. Wood

*Member Emeritus

IEEE Std 1296-1987 was approved by the American National Standards Institute on February 8, 1987 and was reaffirmed by IEEE on March 17, 1994.

Contents

Information technology—Microprocessor systems—High-performance synchronous 32-bit Bus: MULTIBUS II

1. General overview

1.1 Scope

This International Standard defines the operation, functions, and attributes of the IEEE 1296 bus standard.

a) This standard defines a high-performance 32-bit synchronous bus standard.
b) The bus standard must have a design-in lifetime of 10 years with backward compatibility.
c) The standard is intended for general purpose applications to optimize block transfers, including protocol for message passing. For real-time applications, the bus will provide a means of ensuring an upper limit to message delivery time.
d) The standard is intended to be compatible with existing IEC mechanical standards (IEC Pub 297-1,[1] 297-3, and 603-2) with recognition of the need for special front panels to address ESD, EMI, and RFI requirements.
e) Options within the standard will be clearly identified.
f) The standard is intended to support multiple processor modules in a functionally partitioned configuration and heterogeneous processor types in the same system.
g) The standard is intended to support heterogeneous processor types in the same system.
h) Message-passing format and protocol is intended for future migration to a serial system bus.

1.2 Normative references

The following standards contain provisions which, through references in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent edition of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

DIN 41612, Two Part Connectors for Printed Board, GRIB, to 54 mm, Common Mounting Features, Survey of Types.[2]

[1] Information on references can be found in 1.2.

[2] DIN publications are available from the Deutsches Institut für Normung, Burggrafenstrasse 6, D-1000 Berlin 30, Germany.

IEC 297-1 : 1986, Dimensions of mechanical structures of the 482,6 mm (19 in) series—Part 1: Panels and racks.[3]

IEC 297-3 : 1984, Dimensions of mechanical structures of the 482,6 mm (19 in) series—Part 3: Subracks and associated plug-in units.

IEC 603-2 : 1988, Connectors for frequences below 3 MHz for use with printed boards—Part 2: Two-part connectors for printed boards, for basic grid of 2,54 mm (0,1 in), with common mounting features.

IEEE Std 1101-1987, IEEE Standard for Mechanical Core Specifications for Microcomputers (ANSI).[4]

[3]IEC standards are available from the IEC Sales Department, Case Postale 131, 3 rue de Varembé, CH-1211, Genève 20, Switzerland/ Suisse.

[4]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

2. Definitions

The following definitions apply to all clauses of this International Standard.

2.1 acquisition phase: The final phase of the arbitration operation entered after determining that an agent has the highest priority and the bus is available. *See:* **arbitration operation; agent.**

2.2 address transfer: The passing of address information over the multiplexed address/data bus from the bus owner in order to select the replying agent(s). *See:* **bus owner; replying agent.**

2.3 address/data bus signal group: A set of thirty-six (36) signals, consisting of 32 address/data signals and four parity signals that are used for address and data transfers.

2.4 agent: A physical unit that has an interface to the parallel system bus, for example, a single-board computer.

2.5 agent error: An agent status that indicates an error condition in a replying agent.

2.6 agent status: The condition of the replying agent, transmitted during the reply phase of a transfer operation. *See:* **reply phase; transfer operation.**

2.7 arbitration operation: The bus operation in which agents attempt to gain exclusive access to the parallel system bus.

2.8 backplane: The physical mechanism by which signals are routed between agents.

2.9 bit (b): A binary digit.

2.10 broadcast message: A sequence of one or more data transfers from the bus owner to all replying agents, with uninterrupted bus ownership.

2.11 bus clock cycle: An amount of time equal to one bus clock period, nominally 100 nanoseconds.

2.12 bus loss: The amount of time required for a valid signal transition to occur at every point on the backplane. This value is equivalent to two bus propagation delays plus the clock skew.

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

2.14 bus owner: The agent that enters the acquisition phase of the arbitration operation and initiates one or more transfer operations. *See:* **acquisition phase; arbitration operation; transfer operation.**

2.15 bus request sequence: A set of one or more arbitration operations in which all agents that simultaneously request the bus become the bus owner, one at a time. *See:* **arbitration operation; bus owner.**

2.16 byte (B): A group of eight adjacent bits operated on as a unit.

2.17 central services module (CSM): A specific module that is required in all systems using the parallel system bus. Its services, such as starting certain bus operations and guaranteeing uniform initialization of all agents, are required by all agents on the parallel system bus. It is always located in a specific slot in the system backplane. *See:* **parallel system bus.**

2.18 client: An agent that requests services of a server. *See:* **server.**

2.19 cold-start: A sequence of events performed on the application of power that ensures a uniform initialization period for all agents, giving them the ability to begin operation from a known state.

2.20 command transfer: The passing of command information over the system control signal group, from the bus owner to the replying agent(s), during the request phase of a transfer operation. Command information includes parameters for the impending transfer operation, as well as additional address space information not transmitted with the address transfer. *See:* **request phase; system control signal group.**

2.21 data transfer: The passing of data over the multiplexed address/data bus, between the bus owner and the replying agent(s), during the reply phase of a transfer operation.

2.22 driving agent: The agent that is permitted to assert or negate a signal on the bus.

2.23 dual-mode agent: An agent that supports both memory-mode and message-mode communication on the parallel system bus. *See:* **memory-mode agent; message-mode agent.**

2.24 dual-mode system: A system that supports both memory-mode and message-mode communication. A mixture of both communication types is used on the parallel system bus.

2.25 end of transfer (EOT) status: A handshake status that indicates the last data transfer of the transfer operation. *See:* **handshake status.**

2.26 exception: An abnormal condition on the bus caused by either a bus parity error, a bus time-out, a protocol violation, or a bus owner reply phase termination.

2.27 exception operation: A bus operation in which an agent places an error indication on the parallel system bus. The error indication causes all bus agents to terminate arbitration and transfer operations.

2.28 handshake status: A status transfer that indicates the exchange of data between bus owner and replying agent(s).

2.29 I/O space: The address space used for accessing peripheral devices such as communication controllers and mass storage devices.

2.30 interconnect space: The address space used for board identification, system configuration, and board specific functions such as testing and diagnostics.

2.31 interconnect template: A definition of the contents of the interconnect space of an agent.

2.32 interface: A shared boundary between modules or agents of a computer system, through which information is conveyed.

2.33 locked: A condition of the bus that guarantees exclusive access to the parallel system bus and to resources on the replying agent(s). This inhibits transfer operations between the replying agent and any other bus interface.

2.34 memory space: The address space used for accessing physical memory devices for storage and retrieval of code and data.

2.35 memory-mode agent: An agent that communicates with others by using memory and/or I/O space on the parallel system bus.

2.36 memory-mode system: A system in which the agents communicate with one another with data structures in memory and/or I/O space.

2.37 message space: The address space used for packet based communications ranging from interrupts to negotiated data movement. *See:* **packet.**

2.38 message-mode agent: An agent that exclusively uses message space for communication with other agents.

2.39 message-mode system: A system in which communication between agents is via blocks of data transmitted in the message space.

2.40 module: A basic functional unit within an agent.

2.41 nibble: A group of four adjacent bits operated on as a unit.

2.42 packet: A block of information that is transmitted within a single transfer operation in message space. *See:* **message space; transfer operation.**

2.43 parallel system bus (PSB): The signals, media and protocol used to interconnect agents in the IEEE 1296 system.

2.44 parking: The state of the bus owner where, after the completion of the current transfer operation, ownership is retained until there is a request by another agent for the use of the bus.

2.45 power failure recovery: A sequence of events that provides orderly control of system shutdown during a temporary power failure and start-up after power is restored.

2.46 protocol: The set of signaling rules used to convey information between agents.

2.47 read data transfer: One or more data transfers from a replying agent to a bus owner, with uninterrupted bus ownership.

2.48 receiver: An agent that is the recipient of the data during a solicited message. *See:* **solicited message.**

2.49 recovery phase: The final phase of an exception operation in which the parallel system bus is allowed to sit idle for a defined amount of time. *See:* **exception operation.**

2.50 reply phase: The final phase of a transfer operation that consists of one or more consecutive data and/or status transfers on the parallel system bus.

2.51 replying agent: An agent that participates in a transfer operation with the bus owner.

2.52 request phase: The initial phase of a transfer operation in which the bus owner places command and address information on the parallel system bus.

2.53 requesting agent: An agent that has entered arbitration for bus access. *See:* **arbitration operation.**

2.54 resolution phase: The initial phase of an arbitration operation in which all agents requesting access to the bus drive an arbitration ID onto the parallel system bus. Agents mutually resolve requests and allow the agent with the highest priority to gain access to the bus. *See:* **arbitration operation.**

2.55 sender: The agent that supplies the data for a solicited message. *See:* **solicited message.**

2.56 sequential transfer: A transfer operation with multiple data transfers during the reply phase. *See:* **reply phase; transfer operation.**

2.57 server: An agent that performs a service for clients. *See:* **client.**

2.58 signal phase: The initial phase of an exception operation in which all agents are notified of an error condition. *See:* **exception operation.**

2.59 skew: The time difference between signals because of timing differences for logic and backplane delays.

2.60 solicited messages: A negotiated data transfer in message space. *See:* **data transfer; message space.**

2.61 status transfer: The passing of information over the system control signal group, between the bus owner and the replying agent, during the reply phase of a transfer operation. *See:* **agent status.**

2.62 system control signal group: A set of ten (10) signals, including two parity bits, which supply command and status information between the bus owner and the replying agent.

2.63 transfer operation: The bus operation in which a bus owner transfers data on the parallel system bus. *See:* **bus operation; bus owner.**

2.64 warm-start: A sequence of events performed to reset a running system.

2.65 write data transfer: One or more data transfers from the bus owner to a replying agent or agents, with uninterrupted bus ownership.

3. Guide to notation

3.1 General

A consistent convention on all signals, figures, and state-flow diagrams is used throughout this International Standard. The conventions used are described in the following subclauses.

3.2 Signal notation

Throughout this International Standard, the following convention is used for signal notation. An asterisk (*) following a signal name indicates that the signal is active when it is at a low voltage level (as described in clause 4). A signal name that is not followed by an asterisk indicates that the signal is active when it is at a high voltage level. Additionally, a high signal voltage level may be indicated by an uppercase h (H), and a low signal voltage level may be indicated by an uppercase l (L). This convention is summarized in table 3.2-1.

Table 3.2-1—Signal notation

Signal name	Electrical level	Signal notation
BREQ	Low or L	Inactive, deasserted, negated
	High or H	Active, asserted
BREQ*	Low or L	Active, asserted
	High or H	Inactive, deasserted, negated

Many signals on the parallel system bus (PSB) are more easily or conveniently discussed as a group. Names for these signals follow a decimal radix numbering convention. When discussed as an individual signal, the decimal number is simply appended to the signal name, e.g., AD15*. A group of signal lines with the same signal group name are collectively referred to by listing the group name and enclosing the decimal numbers within brackets, e.g., AD<31,23,7,0>*. A range of consecutive signals is referred to by using a double period to list the beginning and ending signals, inclusively, e.g., AD<31..24>*. Consecutive and disjoint signals are referred to by using both methods, e.g., SC<9,3..0>*. A bus signal group name with no brackets and digits afterward refers to the entire group, e.g., AD* is equivalent to AD<31..0>*.

Signal lines on the PSB are always shown in capital letters, e.g., BREQ*. In the case of open-collector drivers, where more than one agent may drive the line at any time, the signal level driven by an agent may not be the logic level currently on the PSB signal line. The signal an agent drives on the corresponding bus line is designated by the same signal name, but in lowercase letters, e.g., breq*.

3.3 Figure notation

An example of the notation used in figures is shown in figure 3.3-1. All solid vertical lines in the figure represent the sampling point for signals (the falling edge of the BCLK* signal). Dashed vertical lines

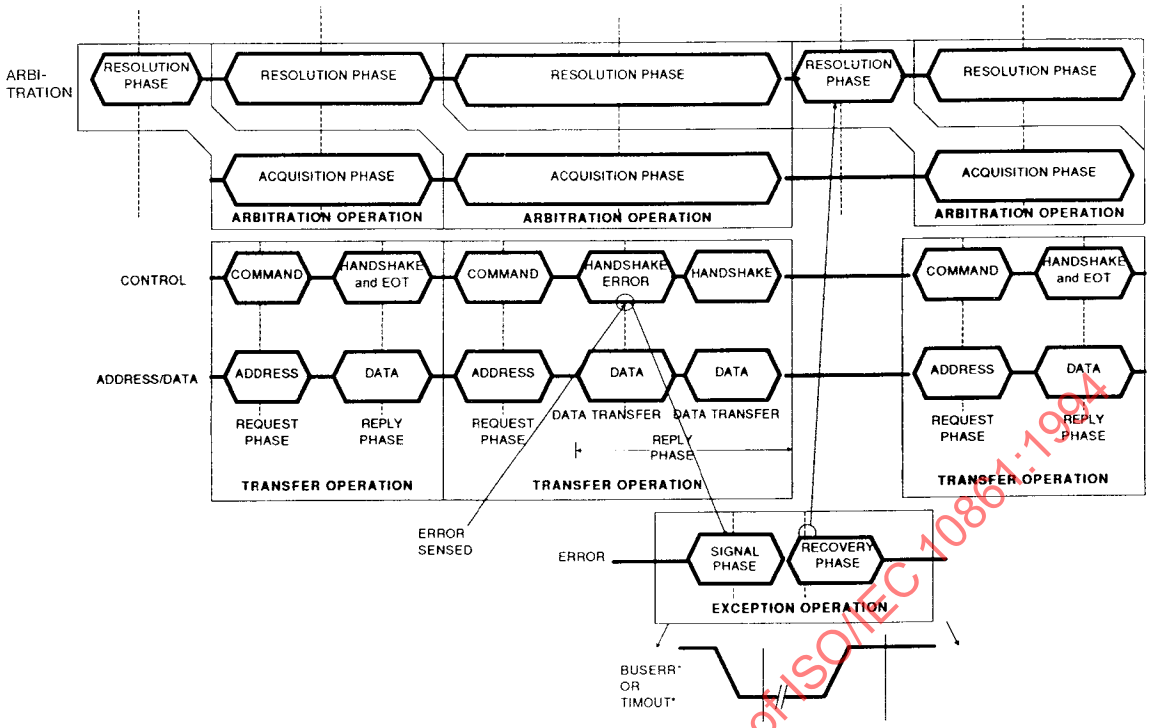


Figure 3.3-1 — Figure notation example

indicate phase transitions. Groups of signals used for phases and transfers are enclosed in polygons indicating simultaneous high and low logic levels. All bus operations (arbitration, transfer, and exception) are enclosed in rectangles. Curved lines and arrows indicate cause-and-effect relationships, not asynchronous operation. Double-slashed lines in the timing diagrams (//) indicate the occurrence of an indefinite number of bus clock cycles, unless otherwise noted.

3.4 Notation in state-flow diagrams

The state-flow diagrams use a consistent notation system in describing the transitions between states during an operation.

Within each diagram there are various components. Each component describes, either graphically or in words, some aspect of the operation of an agent. The various states that an agent may assume are represented as circles. Transitions to other states are shown as arrowed lines and are labeled with a number that corresponds to the numbered paragraph that defines their function. Bold type is used to name conditions that are derived from bus signals or name conditions that describe the internal status of an agent. Generally, such conditions are associated with transitions occurring from one state to another. A condition that causes a transition is shown next to the arrowed line for that transition. Equals signs (=) are used to indicate the state of individual lines or the components of a condition. The state equations and conditions use "AND" to indicate when the AND operator is required and "OR" to indicate when the OR operator is used.

3.5 Notation for multiple bit data representation

Data values for spanning multiple bits are represented in a consistent manner. The value for the multiple bit quantity is shown as a decimal number, except where noted. The value is shown as high-true, making it independent of bus level inversions. For byte values, the highest numbered bit is the most significant bit of the byte. This International Standard does define the format of multiple data byte specification, except when noted.

If the data is in ASCII format, the data consists of eight consecutive bits, with the value of the eighth (most significant bit) always 0.

If the data is BCD format, each nibble within the byte represents a decimal digit. The value of bits <3..0> is used for the units digit of the number and the value of bits <7..4> is used for the tens digit of the number.

4. PSB overview

4.1 General

The PSB is a general purpose interface for multiple agents. Figure 4.1-1 shows a functional diagram of the PSB. As shown, the PSB consists of five signals groups that requesting and replying agents use to communicate with one another.

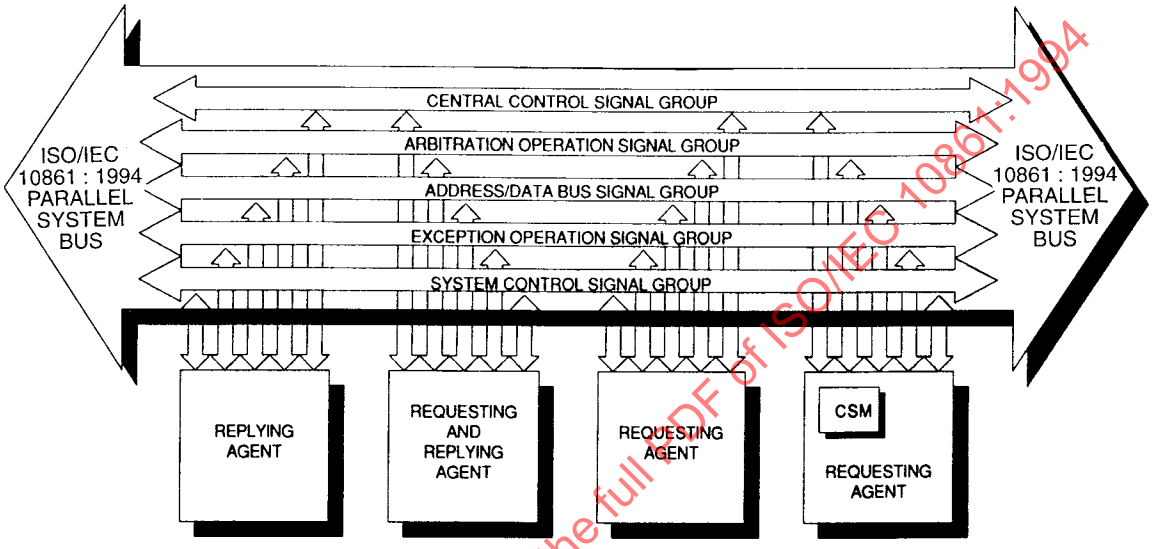


Figure 4.1-1—Block diagram of the PSB interface

The PSB has several specific attributes, as follows:

a) The address/data path is 32-bit plus four parity bits, providing a 4-Gbyte address range and capable of 8-, 16-, 24-, or 32-bit transfers. Ten system control signals, including nibble parity, provide command, status, and handshake information for the address/data path.
b) The message-passing facility provides interagent communication.
c) The interconnect facility allows configuration of agents and modules.
d) The bus operations are synchronous; the bus uses a handshaking protocol that is synchronous with the basic clock rate (10 MHz) for the system resulting in a 40 Mbyte/s maximum transfer rate.
e) The bus operates in defined bus operations. The three types of bus operation are arbitration operation, transfer operation, and exception operation.
f) The bus uses a central services module to provide some centralized system functions for up to 21 agents.

Each of these attributes is explained further in the following subclauses.

4.2 Address/data path and system control signals

The address/data path on the PSB consists of 36 signal lines, including four parity lines, that are time-multiplexed. During the request phase of a transfer operation, the signal lines provide address information; during the reply phase of a transfer operation, the signal lines contain data. Byte parity is provided for all address and data transfers on the address/data path.

Ten system control signals, including two parity signals, provide command, status and handshake information for the address/data path. These signals are also time-multiplexed. During the request phase of a transfer operation, the system control signals provide command information; during the reply phase of a transfer operation, the signals are used for status and handshake information. Nibble parity is provided for the system control lines at all times.

4.3 Message-passing facility

The PSB defines a dedicated address space for use by agents in passing messages. The message-passing facility provides a standardized method for performing direct transfers of messages (command and data) from one agent to another. Message passing on the PSB provides for interagent communications (interrupts and data movement) through the use of the dedicated message space.

4.4 Interconnect facility

The interconnect facility within the PSB defines geographic addressing for ease of configuration, initialization, and diagnosis of an agent on the PSB. This allows software to identify, diagnose, configure, and initialize an agent within a system.

4.5 Synchronous operation of the PSB

The PSB is referred to as being "synchronous" in that all operations on the PSB occur relative to the active edge of a bus clock that is distributed to all agents on the bus. The synchronous nature of the bus does not place rigid time constraints on the length of a transfer operation. It requires that agents sample and drive signals at a specific time with respect to the bus clock.

The synchronous nature of the PSB allows each operation on the bus interface to be defined as a sequence of bus states. Agents on the bus proceed from one bus state to the next by monitoring and asserting specific status and control signals in a state control sequence defined by the operation. The specific meaning or function of some status or control signals can vary and is dependent on the bus operation and state.

Bus operations and associated state sequences are diagrammed later in this International Standard.

4.6 Bus operations on the PSB

Agents perform three types of bus operation on the PSB: an arbitration operation, a transfer operation, or an exception operation. Each operation has a definite event that signals the start and end of the operation. The three types of bus operation can occur simultaneously, maximizing bus utilization.

A typical read or write sequence on the PSB consists of a series of two operations. First, the requesting agent participates in the arbitration operation for access to the bus. When the requesting agent becomes the bus owner, it performs one or more transfer operations to read or write data. If any agent senses an exception (error) on the bus, that agent places an exception indication on the bus. The exception indication initiates an exception operation, which terminates the current arbitration and transfer operations.

Figure 4.6-1 shows how the two types of operation form a typical read or write sequence on the bus. The arbitration operation is a time period in which agents decide which will be the next to control the bus; the transfer operation is a subsequent time period in which that agent performs the address, command, data, and status transfers. Note that an exception operation is an error reporting time period that occurs only when an error is sensed.

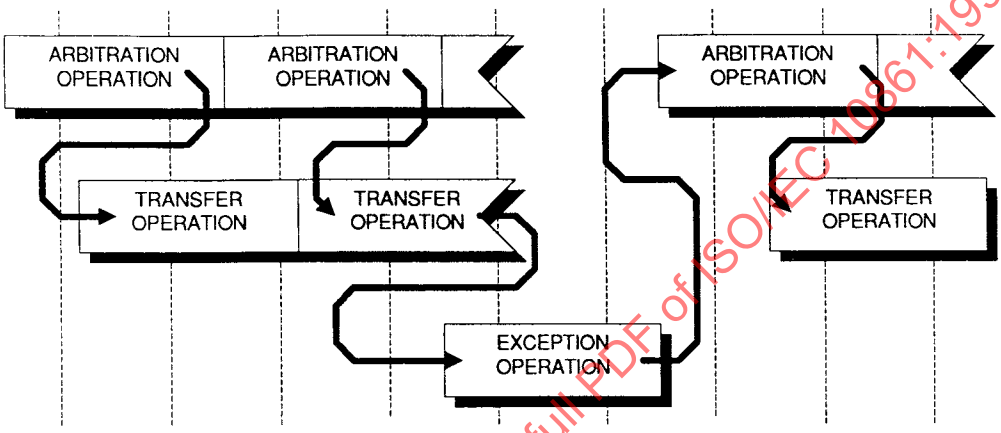


Figure 4.6-1—Bus operation relationships

Each of the three types of bus operation is described further in the following paragraphs.

4.6.1 Arbitration operation

The arbitration operation is the bus operation in which agents attempt to gain exclusive access to the PSB. For each agent that requests access, the arbitration operation includes two phases: the resolution phase and the acquisition phase.

The resolution phase is the time period during which agents that desire the bus collectively arbitrate for access rights to the bus. The agents decide among themselves which one of them is going to control the bus on completion of the current transfer operation.

The arbitration method used during the resolution phase for the PSB is referred to as a parallel contention or self-selecting method. In this method, each agent on the bus is assigned an encoded bit pattern that gives it a priority level when it makes a request for bus use.

The agent that has the highest priority request during the resolution phase obtains access rights to the bus and begins the acquisition phase while the remaining agents begin the resolution phase of the next arbitration operation. The agent in the acquisition phase is referred to as the bus owner.

Once in the acquisition phase, the bus owner begins its transfer operation while the other requesting agents conduct another resolution phase (resolving for next access rights). Figure 4.6-2 shows the end-of-a-transfer operation (EOT) initiating an acquisition phase of the arbitration operation to allow the next bus owner to acquire control of the bus.

The arbitration scheme is such that no agent is prevented from gaining access to the bus. Agents that enter arbitration enter collectively as a group, and each agent within the group receives bus access before any new agent with normal priority is allowed to participate in the bus arbitration process. A bus request sequence is a period consisting of at least one arbitration operation in which all requesting agents (at least one) within a group arbitrate for and receive bus access. When all agents within the group have received bus access, a new bus request sequence is entered and a new group of requesting agents begin arbitration for bus access. An agent may enter the bus request sequence if it has a high priority request as defined in 6.2.2.2.

Figure 4.6-2 shows bus operations that include three consecutive, errorless transfer operations on the bus. This type of operation sequence is typical of a system with more than one requesting agent. The first operation consists of a single data transfer, the second consists of three data transfers, and the third consists of one data transfer. The diagram shows the relationship between the arbitration operation and the transfer operation and shows the two phases of each arbitration operation: the resolution phase and the acquisition phase.

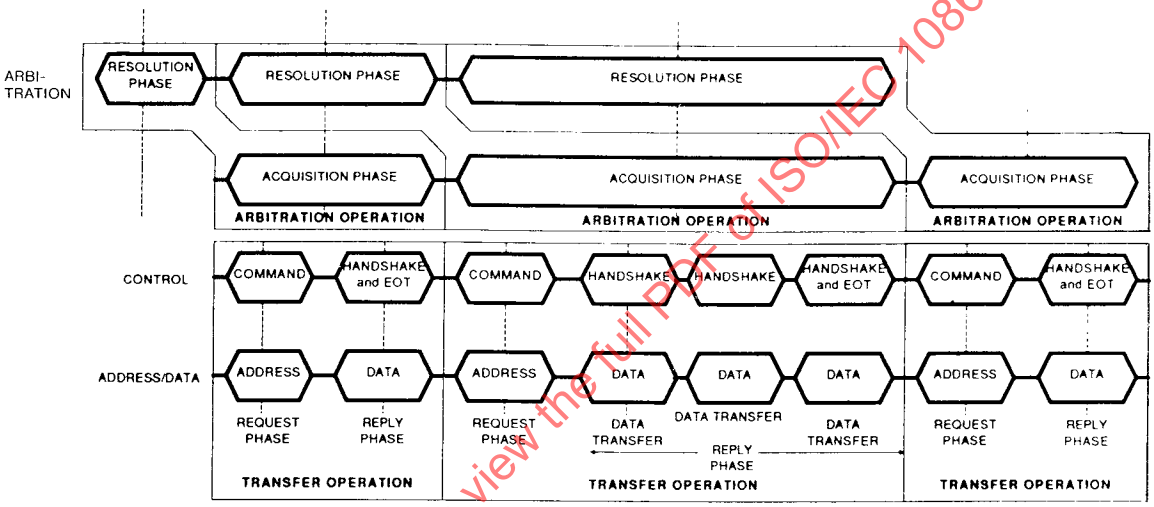


Figure 4.6-2 — Block diagram of bus operations

Note that the resolution phase of the next arbitration operation overlaps the acquisition phase of the current arbitration operation.

4.6.2 Transfer operation

After arbitrating for and winning control of the bus, the bus owner performs transfer operations to move data to or from another agent.

Figure 4.6-2 shows three consecutive transfer operations on the bus. The diagram shows the two phases of the transfer operation, the request phase and the reply phase, and shows the transfers within each phase, including the command, the address, the data, the handshake, and the EOT.

The request phase is controlled by the bus owner. During the request phase, the bus owner places address and control information onto the bus. The address and control information defines the replying agent(s), the type of operation, and the type of address space involved in the transfer operation.

After the bus owner transmits the address and control information, the reply phase of the transfer operation begins, in which the replying agent(s) satisfies the request.

During the reply phase, the bus owner and replying agent engage in a handshake that synchronizes the data transfer. In the case of a read transfer, the handshake status signifies that the replying agent has placed the data on the bus and the bus owner has received it. In the case of a write transfer, the handshake status signifies that the bus owner has placed the data on the bus and that the replying agent has received it. The bus owner and replying agent may perform one or more data transfers in a reply phase. The final data transfer is accompanied by an EOT indication. With the EOT, the bus owner releases ownership of the bus if other agents request access to the bus. Otherwise, the bus owner retains ownership of the bus.

4.6.3 Exception operation

The exception operation is an error-reporting and recovery tool. If an agent detects an error (exception), that agent provides an exception indication on the bus and all agents begin an exception operation. The exception operation terminates any arbitration and transfer operations.

Figure 4.6-3 shows the same operation sequence presented in figure 4.6-2, except that in figure 4.6-3 an agent detects and signals an exception, thus initiating an exception operation. (Any agent can detect and signal an exception.) The diagram shows the two phases of the exception operation, the signal phase and the recovery phase, and shows how the exception operation terminates transfer and arbitration operations.

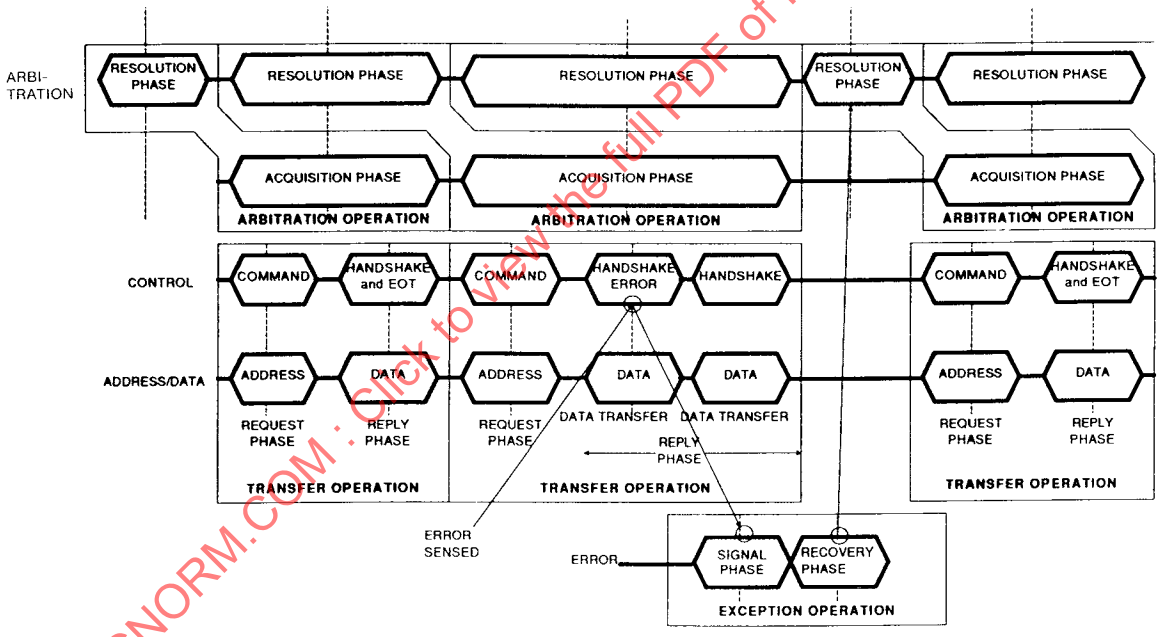


Figure 4.6-3—Block diagram of bus operations with errors

During the signal phase, the agent that detected the exception condition indicates the problem to all agents on the bus by asserting an exception line. As a result of the signal phase, all agents terminate any arbitration and transfer operation that may be in progress.

After the exception line is negated by the agent that signalled the error, the recovery phase begins. One bus clock cycle following the negation of the exception line, arbitration operations may begin. Four bus clock cycles following the negation of the exception line, transfer operations may begin.

As figure 4.6-3 shows, the exception operation has a specific time relationship with respect to sensing an exception in the transfer operation. That is, the current exception operation reflects an exception condition sensed during the previous transfer operation.

4.7 Central services module

The PSB uses a single central services module (CSM) to provide central control of the bus functions. The CSM is responsible for clock and time out generation, monitoring the state of the power supplies and initialization of the bus interface logic. One and only one CSM is active in the PSB backplane.

5. Signal descriptions

5.1 General

This clause lists and defines the five signal groups that make up the PSB. Each signal is discussed in detail.

5.2 Signal groups

The PSB contains five groups of signals over which requesting and replying agents enact the bus protocol. All signals are sampled and driven relative to the falling edge of the bus clock unless otherwise noted. The signal groups, signal names, and their functions are listed in table 5.2-1. Each signal is discussed in the following text. Refer to 7.5.1 for information regarding physical signal position on the connector.

Table 5.2-1—Signal groups on the PSB

Signal group name	Signals in group	Number of signals	Description
Arbitration operation signal group	BREQ*, ARB*	7	Provides the signals to request the bus and resolve arbitration for the next bus owner.
Address/data bus signal group	AD*, PAR*	36	Provides the address, data, and parity signals for read and write sequences.
System control signal group	SC*	10	Provides the control signals needed to transfer address and data on the address/data signal group.
Exception operation signal group	TIMOUT*, BUSERR*	2	Provides the error indication signals to stop transfer operations.
Central control signal group	RST*, RSTNC*, DCLOW*, PROT*, BCLK*, CCLK*, LACHn*	7	Provides system-wide services such as reset and initialization control.

Refer to clause 7 for electrical specifications of the signals.

5.2.1 Arbitration operation signal group

The arbitration signals on the PSB provide for the requesting, arbitrating, and granting of bus ownership to bus agents. All agents that require access to bus resources must arbitrate for use of the bus.

The seven signals within the arbitration operation signal group that implement the arbitration operation protocol on the PSB are bus request (BREQ*) and the arbitration ID signals ARB<5..0>*. These signals are described in the following subclauses.

5.2.1.1 BREQ* (bus request)

BREQ*, when asserted, indicates that a bus request sequence is in progress. Any bus agent that requires access to the bus asserts its breq*. All breq* signals are ORed to become the BREQ* signal. Each requesting agent negates breq* as it receives bus access. As the last requesting agent in the current bus request operation is granted the bus, it negates breq*, which causes the negation of BREQ* on the bus. The negation of BREQ* indicates to all bus agents that the bus request sequence has ended. BREQ* is used only in arbitration operations; once an agent has ownership of the bus, it negates breq* and performs the transfer operation. Other agents wishing access to the bus must sample BREQ* to determine if arbitration is already underway. If arbitration has not begun, an agent may assert its breq* and enter arbitration.

5.2.1.2 ARB* (arbitration)

The arbitration signals are ARB<5..0>*. The arbitration signals provide three functions:

a) Carry cardslot ID assignment during reset
b) Carry arbitration ID assignment during reset
c) Arbitration

During reset, the CSM uses the ARB* lines to assign the agent in each cardslot a cardslot ID and an arbitration ID. The CSM drives the cardslot ID when ARB5* is asserted and the arbitration ID when ARB5* is negated. The cardslot and arbitration IDs are placed on ARB<4..0>*. The cardslot ID gives the agent a geographical address that is used in addressing interconnect space. The arbitration ID defines the arbitration priority level for the agent. During reset, all agents are required to latch their cardslot ID and requesting agents are required to latch their arbitration ID from the ARB<4..0>* lines.

The ARB* lines are used to resolve priorities for agents requesting access to the bus. During the resolution phase, a requesting agent places its arbitration ID on its arb* lines. Then, the agent with an arbitration ID that matches the ORed arbitration code present on ARB* gains access to the bus. As an agent gains bus access, it stops asserting its arb* and breq* lines and leaves arbitration to perform transfer operations as the bus owner. The remaining agents in the bus request sequence successively gain access to the bus in the same fashion as described.

An agent uses the ARB5* line to request high-priority access to the bus during arbitration. If two or more agents assert ARB5* at the same time, their bus access is determined by the priority of their arbitration IDs on ARB<4..0>*.

5.2.2 Address/data bus signal group

Only the bus owner and the selected replying agent(s) use the multiplexed address/data bus of the PSB. The address/data bus signal group includes two sets of signals: address/data signals and parity signals. The address/data signals and the parity signals of the address/data bus signal group are described in the next subclauses.

5.2.2.1 AD* (address/data bus)

The address/data signals on the PSB are AD<31..0>*. These 32 signals are multiplexed and serve a dual purpose depending on the phase of the transfer operation.

During the request phase of the transfer operation, AD* signals contain the address for the ensuing data transfer (AD0* is the least significant and AD31* is the most significant address bit). This address corresponds to a location in memory or I/O space, a processing agent in message space, or a cardslot/register location in interconnect space. The bus owner drives these lines during the request phase.

During the reply phase of the transfer operation, AD* signals provide for a variable-width data transfer, carrying 8, 16, 24, or 32 bits of data. The replying agent drives the AD* lines during the reply phase of a read transfer. The bus owner drives the AD* lines during the reply phase of a write transfer.

5.2.2.2 PAR* (parity)

The parity signals on the address/data bus signal group are PAR<3..0>*. The parity signals help ensure the integrity of address and data information transferred on the AD* lines. Each parity signal carries an even-parity bit for one byte of a 4-byte (32-bit) address/data bus. An agent that drives less than 4 bytes need not place parity for the unused bytes onto the bus. The parity signals are assigned to bytes as follows:

PAR0* — parity for AD<7..0>*
PAR1* — parity for AD<15..8>*
PAR2* — parity for AD<23..16>*
PAR3* — parity for AD<31..24>*

Even parity means that when the number of asserted bits within a byte on the bus (not including the parity bit) is even, the corresponding parity signal is negated. If the number is odd, the parity signal is asserted.

5.2.3 System control signal group

The system control signal group on the PSB consists of ten signals, SC<9..0>*, that provide control between agents during transfer operations. Agents use the SC* signals to define commands or report status, depending on the phase of the transfer operation.

During the request phase of a transfer operation, the bus owner drives the SC* signals providing command information to the replying agent(s).

During the reply phase of a transfer operation (not broadcast messages), the bus owner drives the SC<9,3..0>* signals and the replying agent drives the SC<8..4>* signals. The SC* signals provide for handshake and the transfer of status information between the bus owner and the replying agent(s). During the reply phase of broadcast messages, the bus owner continues to drive all SC* signals.

The following subclauses describe the functions of each SC* signal during the request and reply phases. Table 5.2-2 summarizes the functions of the SC* signals. Tables 5.2-3 and 5.2-4 list the request phase and reply phase functions for each of the SC* signals.

5.2.3.1 SC0* (request phase or reply phase)

SC0* is always driven by the bus owner and serves the same function during the request and reply phases of a transfer operation. When asserted, SC0* indicates to all agents that the bus owner is in the request phase of a transfer operation and that the address/data bus and the SC* lines contain valid request phase information. When negated, SC0* indicates to all agents that the bus owner is in the reply phase of the transfer operation.

5.2.3.2 SC1* (lock)

SC1* is always driven by the bus owner and serves the same function during the request and reply phases of a transfer operation. When the bus owner asserts SC1*, pending bus requests are unheeded and the

Table 5.2-2—Summary of functions of SC* signals

Signal	Function during request phase	Function during reply phase
SC0*	Request phase	Reply phase
SC1*	LOCK	LOCK
SC2*	Data width	End-of-transfer
SC3*	Data width	Bus owner ready
SC4*	Address space	Replying agent ready
SC5*	Address space	Agent status
SC6*	Read/write data transfer	Agent status
SC7*	Reserved	Agent status
SC8*	Even parity on SC<7..4>*	Even parity on SC<7..4>*
SC9*	Even parity on SC<3..0>*	Even parity on SC<3..0>*

Table 5.2-3—Functions of SC* during request phase

Signal	Driving agent	Functions during request phase	Meaning
SC0*	Bus owner	Identify transition between request phase and reply phase of the transfer operation.	L=request phase
SC1*	Bus owner	Locked access	L=lock active H=lock inactive
SC<3..2>*	Bus owner	Identify the width of the data during the transfer operation.	SC3* SC2* H H = 8-bit transfers H L = 16-bit transfers L H = 24-bit transfers L L = 32-bit transfers
SC<5..4>*	Bus owner	Identify the address space for the transfer operation.	SC5* SC4* H H = memory space H L = I/O space L H = message space L L = interconnect space
SC6*	Bus owner	Identify the type of data transfer in the reply phase.	L = write data transfer H = read data transfer
SC7*	Bus owner	Not used; must be high.	
SC8*	Bus owner	Provide even parity for SC<7..4>*:	L = odd number of lows on SC<7..4>* H = even number of lows on SC<7..4>*
SC9*	Bus owner	Provide even parity for SC<3..0>*:	L = odd number of lows on SC<3..0>* H = even number of lows on SC<3..0>*

Table 5.2-4—Functions of SC* during reply phase

Signal	Driving agent	Functions during request phase	Meaning
SC0*	Bus owner	Identify transition between request phase and reply phase of the transfer operation.	H = not request phase
SC1*	Bus owner	Locked access	L = lock active H = lock inactive
SC2*	Bus owner	Place an end-of-transfer indication onto the bus.	L = end-of-transfer for transfer operation H = not end-of-transfer
SC3*	Bus owner	Provide a bus owner ready indication on the bus (part of the reply phase handshake).	L = agent ready for data transfer operation H = agent not ready
SC4*	Replying agent	Provide a replying-agent-ready indication on the bus (part of the reply phase handshake).	L = agent ready for data transfer operation H = agent not ready
SC<7..5>*	Replying agent	Place an agent status indication onto the bus.	SC7* SC6* SC5* Description L L L Reserved. L L H Reserved. L H L Agent data error. L H H NACK; Replying agent cannot respond because of a temporarily unavailable resource. H L L Transfer not understood error; multiple errors sensed. H L H Continuation error; replying agent is unable to continue. H H L Width error; width of transfer is not compatible with replying agent. H H H No error; transfer completed without errors.
SC8*	Replying agent	Provide even parity for SC<7..4>*:	L = odd number of lows on SC<7..4>* H = even number of lows on SC<7..4>*
SC9*	Bus owner	Provide even parity for SC<3..0>*:	L = odd number of lows on SC<3..0>* H = even number of lows on SC<3..0>*

bus owner retains ownership of the bus. Agents in the resolution phase remain there until the current bus owner relinquishes the bus. Additionally, when SC1* is asserted, all multiported resources on the PSB that are accessed during the time that the bus owner asserted SC1* are locked from access by other devices. The locked resources are dedicated to the current bus owner until SC1* is negated.

5.2.3.3 SC2* (data width or end-of-transfer operation)

SC2* changes functions depending on the phase of the transfer operation.

During the request phase, the bus owner uses SC2* (with SC3*) to tell the replying agent(s) the data-width format for the ensuing data transfer. The bus owner codes SC<3..2>* to select one of four possible data widths: 8-, 16-, 24-, or 32-bits. Table 5.2-3 shows the four codes and the associated data widths.

During the reply phase, the bus owner asserts SC2* to inform all agents that the current data transfer is the last in the transfer operation, the end-of-transfer (EOT).

5.2.3.4 SC3* (data width or bus owner ready)

SC3* changes functions depending on the phase of the transfer operation.

During the request phase, the owner uses SC3* (with SC2*) to tell the replying agent(s) the data-width format for the ensuing data transfer. The bus owner drives SC<3..2>* to select one of four possible data widths: 8-, 16-, 24-, or 32-bits. Table 5.2-3 shows the encoding of each data width.

During the reply phase, the bus owner asserts SC3* to notify the replying agent when it is ready to send or receive data. SC3* and SC4* provide a two-directional handshake during the reply phase of the transfer operation.

5.2.3.5 SC4* (address space or replier ready)

SC4* changes functions depending on the phase of the transfer operation.

During the request phase, the bus owner drives SC4* (with SC5*) to select which address space will be used for the impending data transfer. For a read transfer, the address space is a source of data; for a write sequence, the address space is a destination. The bus owner has four address space options from which to choose: memory, I/O, message, or interconnect. Table 5.2-3 shows the four codes available using SC4* and SC5* and the associated address space selection.

During the reply phase, the replying agent asserts SC4* to indicate to the bus owner that it is ready to send or receive data. SC4* and SC3* provide a two-directional handshake during the reply phase of the transfer operation.

5.2.3.6 SC5* (address space or agent status)

SC5* changes functions depending on the phase of the transfer operation.

During the request phase, the bus owner drives SC5* and SC4* to select which address space will be used for the impending data transfer. For a read transfer, the address space is a source of data; for a write sequence, the address space is a destination. Table 5.2-3 shows the four codes available and the associated address space selection.

During the reply phase, SC5* is driven (with SC<7..6>*) by the replying agent to provide agent status information. Table 5.2-4 shows the assigned agent status codes.

5.2.3.7 SC6* (read-write or agent status)

SC6* changes functions depending on the phase of the transfer operation.

During the request phase, the bus owner uses SC6* to indicate the direction of data flow for the transfer. When the bus owner negates SC6*, the reply phase of the transfer operation is a read data transfer. When the bus owner asserts SC6*, the transfer operation is a write data transfer.

During the reply phase, SC6* is driven with SC<7,5>* by the replying agent to provide agent status information. Table 5.2-4 shows the assigned agent status codes.

5.2.3.8 SC7* (reserved or agent status)

SC7* changes functions depending on the phase of the transfer operation.

During the request phase, SC7* is not used and must be negated by the bus owner.

During the reply phase, SC7* is driven with SC<6..5>* by the replying agent to provide agent status information. Table 5.2-4 shows the assigned agent status codes.

5.2.3.9 SC8* (parity on SC<7..4>*)

The function of SC8* remains the same in both the request phase and the reply phase of the transfer operation. Agents use SC8* to provide even parity for the SC<7..4>* signals. If SC<7..4>* contain an odd number of asserted signals on the bus, then the agent driving SC<7..4>* must also assert SC8* on the bus.

5.2.3.10 SC9* (parity on SC<3..0>*)

The function of SC9* remains the same in both the request phase and the reply phase of the transfer operation. Agents use SC9* to provide even parity for the SC<3..0>* signals. If SC<3..0>* contain an odd number of asserted signals on the bus, then the agent driving SC<3..0>* must also assert SC9* on the bus.

5.2.4 Exception operation signal group

The PSB provides the exception operation signal group for passing indications of exception errors to all agents. This group contains two signals: bus error (BUSERR*) and bus time out (TIMOUT*). Each of these signals is on a dedicated line. They are defined in the following subclauses.

5.2.4.1 BUSERR* exception

An agent asserts bus error to indicate the detection of a data integrity problem during a transfer. Problems reported through the BUSERR* signal are: detection of a parity error on the valid bytes of the AD* lines or the SC<7..0>* lines, a bus owner reply phase termination, or a protocol violation occurring during the reply phase. BUSERR* is received by all agents and can be asserted by any agent capable of detecting transfer integrity problems.

5.2.4.2 TIMOUT* exception

An agent monitors TIMOUT* to determine when a bus time-out condition occurs. Bus time out on the PSB is a result of the CSM determining that an agent is taking too much time to assert a handshake signal on the bus. TIMOUT* may also be generated by the CSM after determining that the bus owner has maintained ownership for an excessive length of time, which is user defined. TIMOUT* is received by all active agents on the bus and is generated from time-out circuitry located on the CSM.

5.2.5 Central control signal group

The central control signal group provides bus status and control information for devices operating on the PSB. This signal group contains seven signals. They are as follows:

- Reset (RST*)
- Reset not complete (RSTNC*)
- DC power low (DCLOW*)
- Protect (PROT*)
- Bus clock (BCLK*)
- Central clock (CCLK*)
- ID latch (LACHn*)

Each of the signals is described in more detail in the following subclauses. Electrical specifications of the signals are contained in clause 7.

5.2.5.1 Reset (RST*)

The CSM asserts RST* as a system-level initialization signal to all agents. Agents may monitor RST* along with DC power low (DCLOW*) and protect (PROT*) to distinguish between the types of reset: warm-start, cold-start or power failure recovery (see 6.5.2).

5.2.5.2 Reset not complete (RSTNC*)

Agents may assert RSTNC* during reset to extend the initialization time period provided by the CSM. RSTNC* is an OR-tied signal that is low when one or more agents on the PSB have not completed their reset requirements. Agents cannot perform bus operations while RSTNC* is asserted. The rstnc* signal must be driven by any agent that needs more initialization time than the RST* signal provides. The rstnc* signal should be negated by the agent as soon as possible, as no bus operations can be performed until the RSTNC* signal is negated. RSTNC* is received by all bus agents.

5.2.5.3 DC power low (DCLOW*)

The CSM asserts DCLOW* to all agents as a warning of an imminent power failure. The CSM monitors the power level from the power supply and holds DCLOW* high during normal system operation. When the power supply identifies a power failure for the CSM, the CSM asserts DCLOW* to indicate to all agents that a power failure is imminent. Systems use DCLOW* as the control signal for enabling back-up power and storage. DCLOW* is asynchronous to the bus clock.

5.2.5.4 Protect (PROT*)

The CSM asserts PROT* as a power-failure protection signal to other system agents. PROT* is a delayed result of the CSM sensing a power failure and indicates that system-supplied voltages are not within tolerance. Agents in the system that contain battery-backed resources receive PROT*. The CSM negates PROT* after ac line voltage is stable and system voltages are available. PROT* is asynchronous to the bus clock.

5.2.5.5 Bus clock (BCLK*)

Only the CSM generates BCLK* and all agents in a system receive BCLK*. An agent uses BCLK* to drive the PSB state machines. BCLK* has a fixed frequency of 10 MHz on the bus for normal operation, though the frequency can be reduced to dc for testing of the PSB. Agents should be designed to operate at BCLK* frequencies from 10 MHz to dc. The falling edge of BCLK* provides all system timing references.

5.2.5.6 Central clock (CCLK*)

CCLK* is generated by the CSM and may be used for additional timing among agents on the PSB. CCLK* operates with a specific relationship to BCLK* and at twice the frequency.

5.2.5.7 ID latch (LACHn*)

An agent uses its ID latch signal only during a reset. The LACHn* signal is driven by the CSM and serves two functions: it tells the agent when to latch the arbitration ID and when to latch the cardslot ID from the ARB<4..0>* lines, depending on the state of ARB5*.

The ID latch signal is called LACHn* (where "n" is the cardslot to which the ID is assigned). Each of the cardslots, except 0, contains a LACHn* signal that is activated when the CSM drives one of the address/data lines (AD<20..1>*). For example, the agent in cardslot 2 accepts its ID when the CSM asserts AD2* which is connected on the backplane to LACHn* at cardslot 2. This is shown in figure 5.2-1.

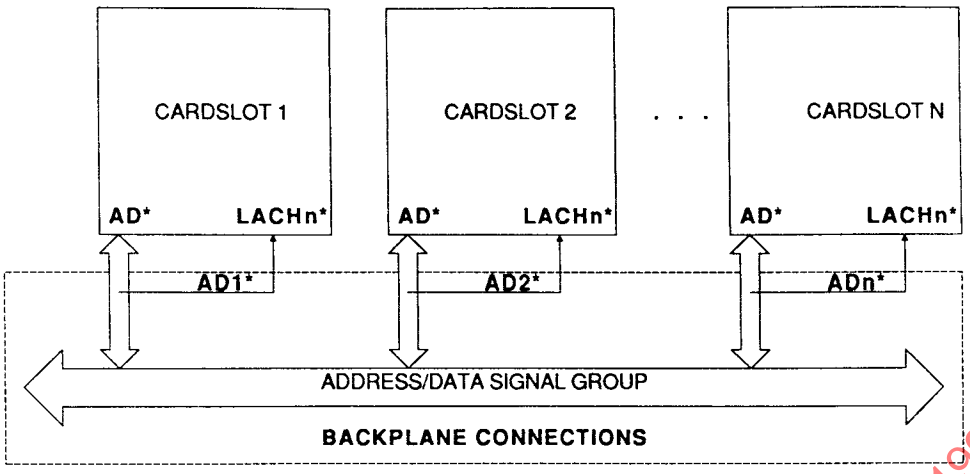


Figure 5.2-1 — Backplane connection of LACHn*

5.2.6 Power

The PSB provides power for bus agents. The system power includes dc power at +5 V, +12 V, –12 V, ground, and facilities for +5 V battery backup.

6. PSB protocol

6.1 General

This clause describes the communication protocol for the PSB. The communication protocol among agents on the PSB is consistent for all bus operations. All agents implement the protocol by monitoring and controlling the signals within the five signal groups described in clause 5. The description of the protocol is organized according to the three general bus operations that occur: the arbitration operation, the transfer operation, and the exception operation. Following the discussion of bus operations, the systemwide functions of the PSB are presented.

The protocol for all bus operations is presented in two stages, first in timing diagrams and then in state-flow diagrams. The timing diagrams provide a time-ordered representation of bus operations by showing the transition relationships among the signals that define the bus protocol. The timing diagrams illustrate the synchronous nature of the PSB by showing the bus clock cycles as vertical lines. The bus clock provides a common signal for all bus agents and is used to synchronize bus operations.

State-flow diagrams present the lowest level of detail in defining the protocol responsibilities of each agent in an operation. The state-flow diagrams list signal conditions required for each state-transition in an operation.

Figure 6.1-1 shows the three types of bus operation and the signal groups that create the protocol for each operation. The three types of bus operation are as follows:

a) Arbitration operation
b) Transfer operation
c) Exception operation

These three bus operations are discussed in detail in the following subclauses.

6.2 Arbitration operation

An agent that wishes to transfer data on the PSB must participate in a bus request sequence and participate in at least one arbitration operation. During an arbitration operation, one or more agents arbitrate for control of the bus. As a result of a single arbitration operation, one agent gains control of the bus and becomes the bus owner. Agents must gain control of the PSB before they can transfer data.

The arbitration operation performs two functions: first, it allows all requesting agents equal opportunity to access 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 a case where more than one agent requests access to the bus at the same instant (or during the same bus request sequence), those agents that are refused access to the bus engage in additional arbitration operations until all requesting agents are granted bus access, in a sequential fashion, based on their priority.

6.2.1 Agent arbitration ID and cardslot ID assignment

During reset, the CSM in each system assigns cardslot IDs (1 through 20) to each agent and assigns arbitration IDs to each agent on the bus. The sequence involves the use of the arbitration ID lines

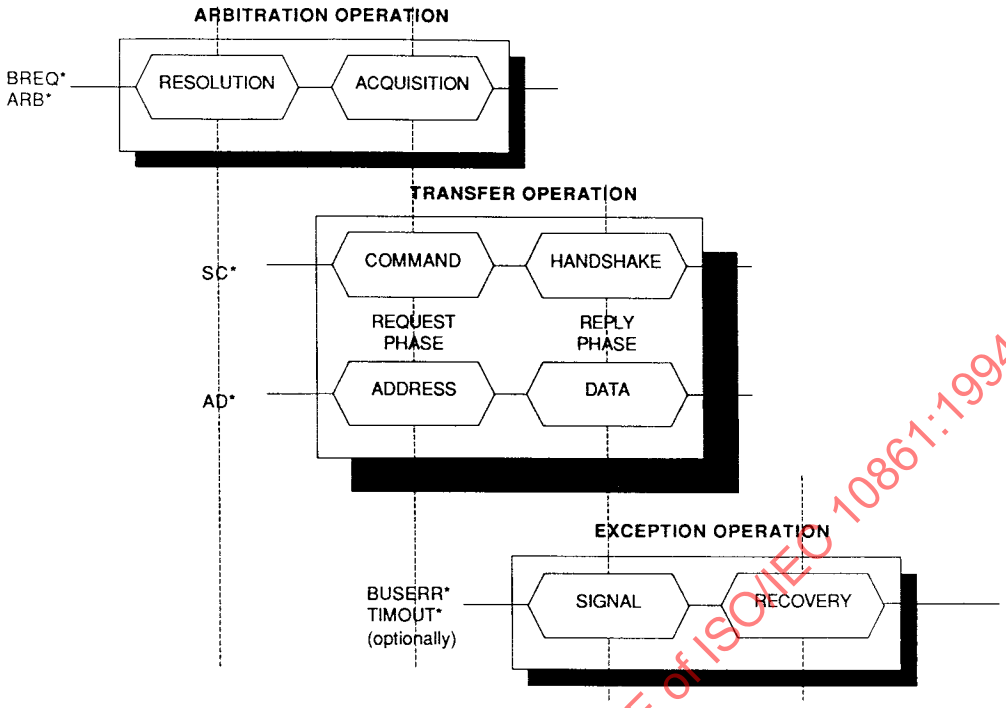


Figure 6.1-1 — Bus operation details

(ARB*), the address/data bus lines (AD<20..1>*), the ID latch signal (LACHn*), and the reset signal (RST*). When ARB5* is asserted, the CSM drives cardslot IDs on ARB<4..0>*; when ARB5* is negated, the CSM drives arbitration IDs on ARB<4..0>*.

With each arbitration/cardslot ID, the CSM drives a corresponding address/data line (connected to the LACHn* signal for each cardslot) to identify the destination for the ID. The CSM asserts one and only one of the AD* lines at any time during the assignment sequence. Each agent uses its LACHn* line (connected to one of the address/data lines) to identify the ID from the ARB<4..0>* lines. Whenever an agent detects RST*, LACHn* and ARB5* asserted, the agent must latch the values on ARB<4..0>* for use as its cardslot ID. Whenever an agent detects RST* and LACHn* asserted and ARB5* negated, the agent must latch the values on ARB<4..0>* for use as its arbitration ID.

Figure 6.2-1 details the relationship among RST*, ARB*, and LACHn*. The LACHn* signal remains asserted for at least one bus clock cycle. The arbitration or slot ID is guaranteed to be stable for at least one clock cycle before and after the LACHn* signal is active. Refer to 6.5 for more information on the assignment of cardslot and arbitration IDs.

6.2.2 Bus arbitration priority

All agents in a system share access to six signal lines on the bus, ARB<5..0>*. These signals provide the mechanism that allows agents to decide among themselves which agent has first access rights to the

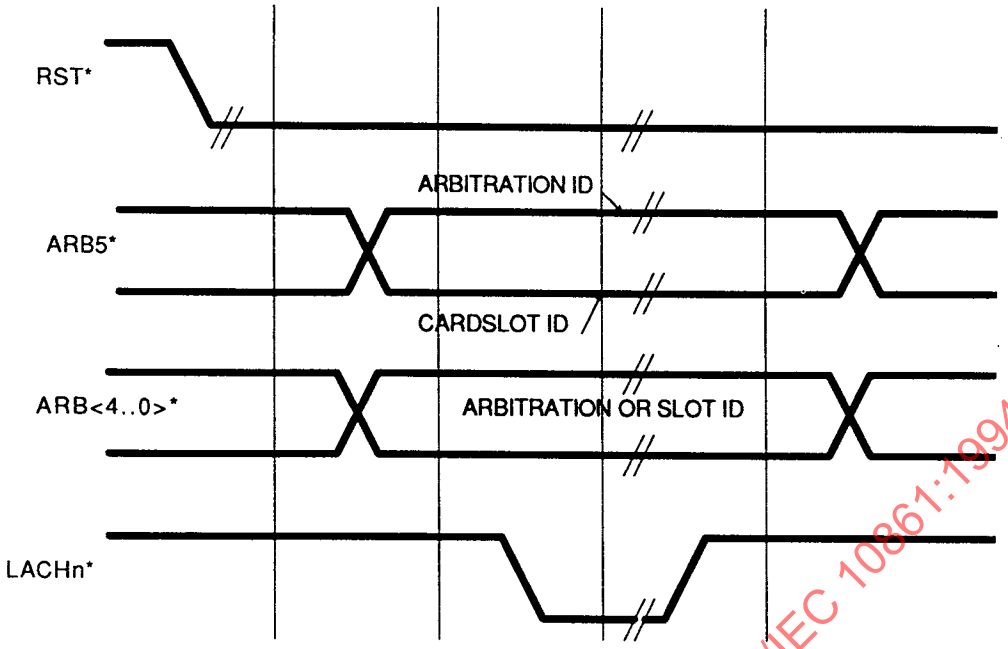


Figure 6.2-1 — Reset sequence

bus. At the beginning of a resolution phase, each agent desiring access to the bus places its arbitration ID and priority onto its arb* lines. The arbitration operation signal group supports two different priorities at which an agent can request use of the bus: normal priority (order of access determined by the arbitration ID on ARB<4..0>*) and high priority (access based on condition of ARB5* and arbitration ID on ARB<4..0>*). It is possible that the implementations could utilize different values of ARB<4..0>* for high and normal priority arbitration. The protocol for each priority method is discussed in the next subclauses.

6.2.2.1 Normal priority

An agent places a normal priority request by asserting its breq*, holding arb5* inactive, and placing its arbitration ID onto arb<4..0>*. Then each agent that requests access to the bus determines whether it is making the highest priority request (higher arbitration IDs have higher priority—see table 6.5-1) by monitoring the condition of all arbitration lines on the bus.

At the beginning of the resolution phase, each agent requesting the bus must place its arbitration ID on its arb<4..0>* lines. If, for the given agent, the arbitration ID that it asserts does not match the ID present on the ARB* lines, the agent stops driving the arbitration ID lines below the bit that doesn't match (toward the LSB). The agent continues to compare its arb* lines to the ARB* lines throughout the resolution phase. The settling time for the ARB* lines is a maximum of three bus clock cycles, but the driving of the arb* lines by requesting agents continues until the bus ownership can be exchanged. At the end of the resolution phase, the agent whose arbitration ID matches the ARB* lines becomes the bus owner and removes its breq* and arb* signals. Agents whose arbitration ID lines do not match the ARB* lines continue into the resolution phase of the next arbitration operation. One arbitration operation is executed for each agent that receives access to the bus. To service all agents that have requested bus access, a series of arbitration operations are executed. This series of arbitration operations, during which all requesting agents receive bus ownership access sequentially, based on their priority, is called a bus request sequence.

Agents making a normal priority request may request bus access and enter a bus request operation only when no bus request sequence is in progress, as indicated by the negation of BREQ*. All agents engaged in the current bus request sequence must be given bus access, and the current bus request sequence must end before a new bus request sequence can begin. This ensures that no normal priority agent in a given bus request sequence is locked out of bus access by another normal priority agent.

6.2.2.2 High priority

If an agent requires urgent bus operations, that agent uses the high priority feature. An agent's high priority request does not wait for the negation of BREQ* before driving its arb* lines. However, if BREQ* is asserted, an agent with a high priority request must wait until the beginning of the next resolution phase to drive its arb* lines. In this case, the beginning of the next resolution phase immediately follows an EOT status transfer without a lock indication. If BREQ* is not asserted at the time of a high priority request, the requesting agent enters a resolution phase in the same manner as agents with normal priority requests.

Once having entered the resolution phase, an agent places a high priority request by asserting arb5* while placing its arbitration ID onto its arb<4..0>* lines. By asserting arb5*, an agent guarantees itself access to the bus before any simultaneous or pending requests from other agents asserting a normal priority request. When more than one agent simultaneously places a high priority request, (asserts arb5*) the agent with the higher priority specified by its arbitration ID gains first access.

Table 6.2-1 summarizes the interaction between agents of different priority levels on the bus.

Table 6.2-1—Arbitration priority protocol

Priority level of Agent A's request	Priority level of Agent B's request	Next bus owner	Description
Normal	Normal	Either	Arbitration based on ARB<4..0>* to select next bus owner.
Normal	High	Agent B	Allow the agent making the high priority request to be the next bus owner.
High	Normal	Agent A	Allow the agent making the high priority request to be the next bus owner.
High	High	Either	Arbitrate among high priority agents based on ARB<4..0>*

6.2.3 Bus ownership

If, at the end of the resolution phase, an agent recognizes that its arbitration ID matches the ID on the bus, the agent begins an acquisition phase.

During the acquisition phase, the bus owner performs transfer operations and all agents with pending requests begin arbitrating again to determine the next bus owner. During the last handshake of the transfer operation, the bus owner asserts SC2*. If another agent has won access rights to the bus, then the current bus owner releases control of the bus and the new bus owner immediately enters the bus acquisition phase.

A bus owner can perform several consecutive transfer operations by asserting lock via SC1*. By asserting SC1*, the bus owner guarantees itself exclusive use of the bus until it negates SC1*; other agents are prohibited from gaining ownership of the bus while SC1* is asserted.

Figure 6.2-2 shows the timing sequence for an operation during which the bus owner has locked the bus. While SC1* is asserted, the bus owner may perform several consecutive transfer operations without interruption from any other device.

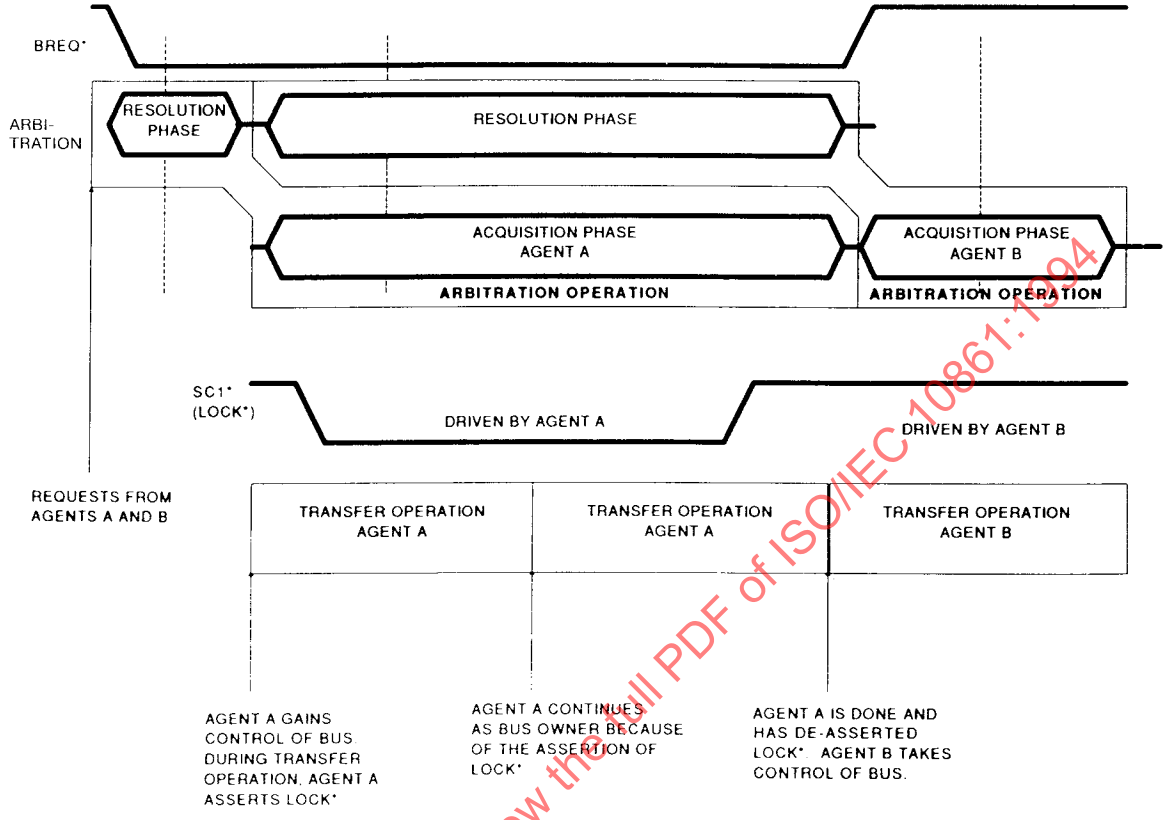


Figure 6.2-2—Timing sequence with bus locked

6.2.4 Parking and bus release

A bus owner releases control of the bus if it has completed its transfer operation, SC1* (lock) is not asserted, and another agent has entered the resolution phase of an arbitration operation. If the current bus owner is the last agent in the bus request sequence to gain access to the bus, then as it negates breq*, the BREQ* signal goes inactive on the bus. The negation of BREQ* allows all agents to participate in the next bus request sequence. As the bus owner completes its operation, the EOT status transfer on the bus allows the next bus owner to assume ownership of the bus.

If no other agents request access to the bus, the last agent to be the bus owner retains ownership of the bus. As such, this agent can perform another transfer operation without arbitrating for access to the bus. This condition is known as parking.

If a bus owner performs multiple data transfers while other agents are in the resolution phase of an arbitration operation, the resolution phase is extended until the EOT status transfer occurs. The occurrence of the EOT status transfer permits the new bus owner to enter the acquisition phase and perform transfer operations.

Figure 6.2-3 shows the three conditions that must be satisfied to exchange ownership of the bus: a pending request for the bus, no transfer operation in progress, and SC1* (lock) negated.

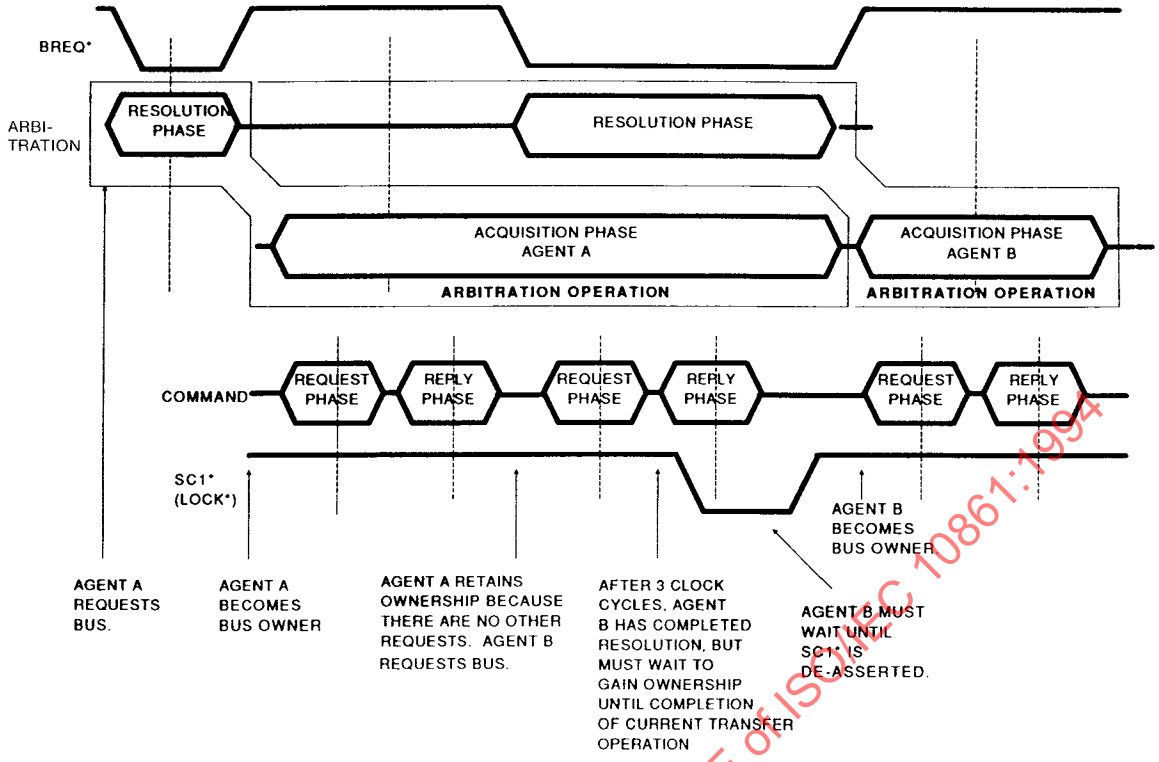


Figure 6.2-3—Timing sequence for bus release

6.2.5 Arbitration priority example

Figure 6.2-4 shows two consecutive bus request sequences during which four agents (Agent A, Agent B, Agent C, and Agent X) make normal priority requests for bus access assuming an idle bus. These two bus request sequences occur as described in the following steps:

a) Agents A, B, and C make normal priority requests for access to the bus by asserting their breq* lines and placing their arbitration IDs onto their arb<4..0>* lines with arb5* negated.

b) The breq* signals from the three agents are ORed together and placed on the bus as BREQ*, thus closing the current bus request sequence to any additional normal priority requests.

c) The first arbitration operation begins. During the resolution phase, Agent A is determined to have the next bus ownership by virtue of its arbitration ID and is given access rights to the bus.

d) Agent A now negates its breq* and arb*, enters the acquisition phase, and begins the transfer operation over the PSB. As the first arbitration operation enters its acquisition phase, the second arbitration operation begins its resolution phase with Agents B and C. Note that the acquisition phase of the first arbitration operation occurs coincident with the resolution phase of the second arbitration operation.

e) During the resolution phase of the second arbitration period, agent X desires to make a normal priority bus request. However, a bus request sequence is currently in progress, and this is by the assertion of BREQ*. Agent X is not permitted to make the normal priority bus request or engage in arbitration until the current bus request sequence is completed and all current requesting agents have been satisfied.

f) At the end of the resolution phase of the second arbitration operation, Agent B is determined to have next bus ownership, and is given access rights to the bus. Agent B negates its breq* and arb* after Agent A has relinquished bus ownership. Agent B then enters the acquisition phase and begins the transfer operation over the PSB.

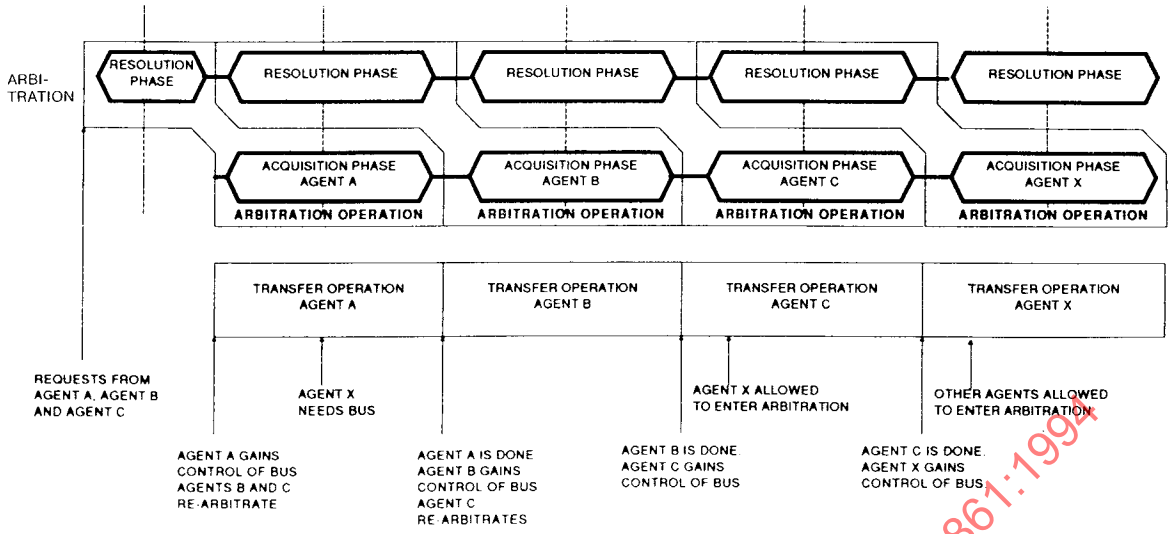


Figure 6.2-4—Typical arbitration operation sequence

g) As Agent B enters the acquisition phase described in the last step, Agent C enters the resolution phase of the third arbitration operation.

h) At the end of the resolution phase, Agent C wins access rights to the bus. As Agent B finishes its transfer operation, it relinquishes control of the bus and Agent C enters the acquisition phase of the third arbitration operation and negates its breq* and arb*.

i) The negation of breq* by Agent C results in the negation of BREQ* on the bus. This indicates to all bus agents that the current bus request sequence has ended. At this point, any agent requiring access to the PSB (excluding Agent C, which currently has access) can make a bus request in the normal fashion described previously. In this example, Agent X makes a normal priority bus request by placing its arbitration ID on arb<4..0>* and asserts breq*.

j) Agent X enters the resolution phase of the first arbitration operation of the new bus request sequence. Agent C is performing its transfer operation over the PSB.

k) Agent X wins the resolution phase of the arbitration operation and is given access rights to the bus. As Agent C completes its transfer operation, Agent X enters the acquisition phase and negates its breq* and arb*. Agent X then begins the transfer operation.

In the previous example, if Agent X had entered a high priority bus request, it would have entered the very next resolution phase of the first bus request sequence. Agent X would have been granted bus access in place of Agent B, and Agent C would have been the last agent to receive bus access. In this case, there would have been only one bus request sequence, containing four arbitration operations, providing bus access to four bus agents, although only three bus agents requested the bus at the beginning of the bus request period.

6.2.6 Resolution algorithm example

As described earlier, a bus agent uses its arbitration for bus access. The function of the arbitration ID is described further in the following paragraphs.

When an agent attempts to gain control of the bus, the agent places its arbitration ID onto the bus and compares its own ID, bit-by-bit, with the ID present on the bus. At the bit location where the agent ID does not match the ID on the bus, the agent uses a back-off algorithm. If an agent asserts an arbitration line, that signal level is dominant in forming the bus ID; the bus ID consists of the "wired OR" function performed on all agent IDs. Figure 6.2-5 shows an example of the arbitration circuit on a requesting agent. Note that the value of the ARB* lines affect the value driven by the agent on its arb* lines.

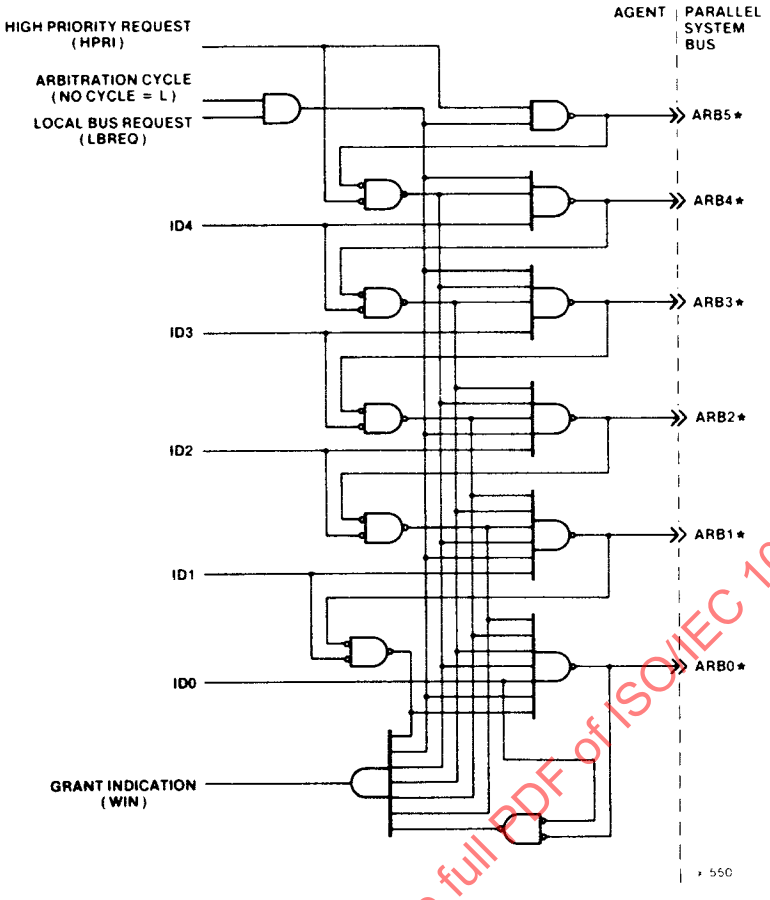


Figure 6.2-5—Arbitration ID interface example (at each requesting agent)

6.2.6.1 Example 1

To understand how the back-off algorithm works, consider the following example. This example describes the resolution phase of an arbitration operation. Table 6.2-2 shows the values of the ARB* signals during this example. It is important to note that the periods in the table represent the settling time for the ARB* lines, not bus clock cycles. Three agents simultaneously make normal priority requests for bus access. At the start of the resolution phase, Agent 1 places a value of HLLLHL on arb<5..0>*, Agent 2 places HLLLHH, and Agent 3 places HLLHLL. The IDs for all three agents are listed in the Start column of table 6.2-2. The arbitration process is described in the following steps:

a) Since these are all normal priority requests, ARB5* is negated. The arbitration ID present on the bus is the wired OR of the IDs being placed on the bus by the three agents.
b) The agents begin comparing their IDs with the bus ID. During Period 1, agents move from the high-order to low-order bits comparing, each in turn. When an agent detects a match, it moves to the next lower bit (LSB) and compares.
c) If the agent detects the assertion of a bit which it is not asserting, it immediately stops driving all bits below it (that is, from the mismatched bit toward the LSB, all bits are negated, denoted in the table by 'h'). This may cause an agent to drive its arb* lines with a value differing from its arbitration ID, which is referred to as a pseudo-ID. Each arbitrating agent creates its pseudo-ID is this fashion. The pseudo-IDs are driven onto the bus and compared throughout the resolution phase.

Table 6.2-2—Arbitration ID comparison example

Settling periods are not related to bus clock cycles.

Settling periods	Participants	Initial value	Resultant value	
Period 1	Initial ARB*	HHHHHH		
	Agent 1 ID* = HLLLHL Agent 2 ID* = HLLLHH Agent 3 ID* = HLLHLL		HLLLHL HLLLHH HLLHLL	
	Resulting ARB*		HLLLLL	
Period 2	Previous ARB*	HLLLLL		
	Agent 1 arb* Agent 2 arb* Agent 3 arb*		HLLLHh HLLLHh HLLHhh	Pseudo-ID lose Lose Pseudo-ID lose
	Resulting ARB*		HLLLHH	
Period 3	Previous ARB*	HLLLHH		
	Agent 1 arb* Agent 2 arb* Agent 3 arb*		HLLLHL HLLLHH HLLHhh	Win Win Pseudo-ID lose
	Resulting ARB*		HLLLHL	
Period 4—Steady state, does not cause ARB* transitions	Previous ARB*	HLLLHL		
	Agent 1 arb* Agent 2 arb* Agent 3 arb*		HLLLHL HLLLHH HLLHhh	Win Lose Pseudo-ID lose
	Resulting ARB*		HLLLHL	

NOTE—h = value in the corresponding bit position is deasserted by agent because it must drive a pseudo-ID.

d) After the first settling time for the ARB* lines agents on the bus have determined pseudo-IDs to be driven onto the PSB.
e) After the pseudo-IDs calculated in Period 1 have settled on the bus, the ARB* lines have a new value. This causes all agents to determine new pseudo-IDs and drive them onto the bus.
f) After the pseudo-IDs calculated in Period 2 have settled on the bus, the ARB* lines again have a new value. This process continues until the ARB* lines are stable (at most three bus clock cycles). Agent 1 has now determined that at the end of the resolution phase, it becomes the bus owner, and Agents 2 and 3 have determined that they are not to be the next bus owner.
g) After Agent 1 takes possession of the bus, Agents 2 and 3 arbitrate for the next ownership. Note that the higher the value of the arbitration ID (lower value after inversion on the ARB* lines) the higher the priority.

6.2.6.2 Example 2

This example illustrates the timing sequence for normal priority bus acquisition (see figure 6.2-6). In this example there are three agents: A, B, and C. In terms of their arbitration IDs (those assigned during reset), Agent A obtains bus ownership first, followed by Agent B, then Agent C (last). The sequence is detailed in the following steps:

a) During the same bus clock cycle both Agents A and B desire normal priority use of the bus. Since no bus request sequence is in progress (BREQ* = H), both agents enter arbitration by asserting

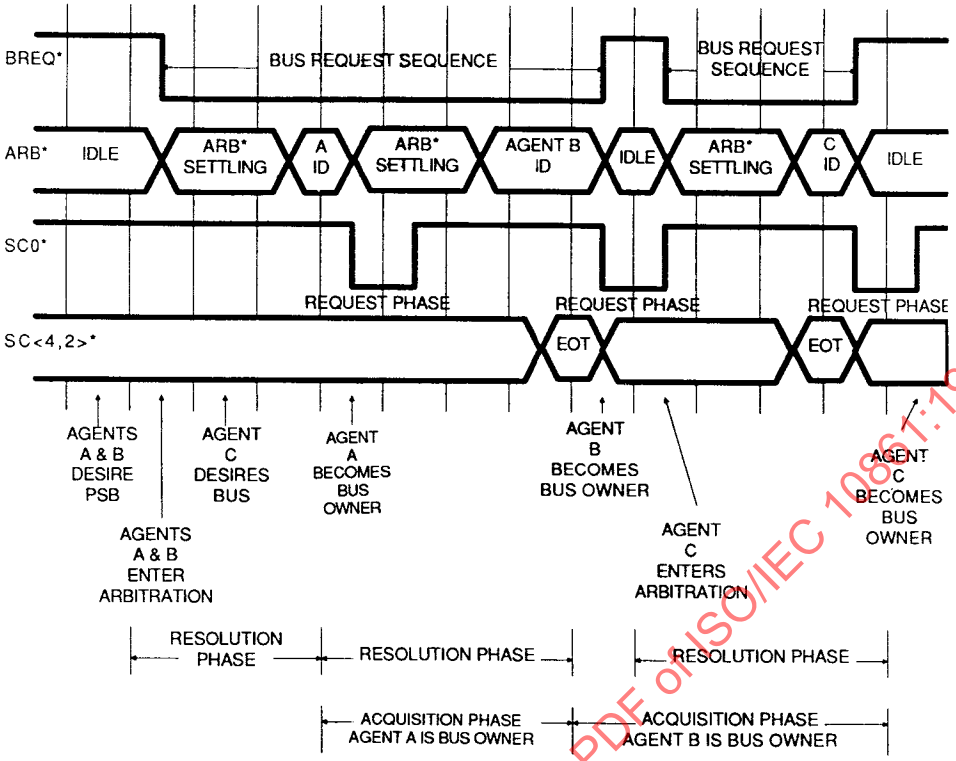


Figure assumes SC1* is negated

Figure 6.2-6—Timing sequences for bus acquisition

breq* and placing their arbitration IDs onto the arb<4..0>*. In both cases, arb5* is negated, indicating that both are normal priority requests.

b) During this bus clock cycle and the next two, the ARB* lines settle according to the back-off algorithm described previously, resulting in a valid arbitration ID by the end of the third bus clock cycle.

c) Meanwhile, during the current resolution phase just described, Agent C desires normal priority use of the bus. At the end of the third bus clock cycle in the current resolution phase, Agent A's arbitration ID is stable on ARB* indicating that it is the next bus owner.

d) On the next bus clock cycle (fourth bus clock cycle), Agent A gets ownership of the bus. Agent A removes its arb* and breq* signals and begins transfer operations. Agent B remains in the resolution phase, continues to assert breq*, and drives its arbitration ID. Agent C is prevented from entering into arbitration with Agent B because Agent C has a normal priority request and a bus request sequence is currently active.

e) By the end of the third bus clock cycle of the second resolution phase, Agent B's arbitration ID is stable on ARB* (indicating it is the next bus owner). However the second resolution phase is extended because the current transfer operation is not complete. When the current transfer operation completes, Agent A relinquishes bus ownership. Agent B becomes the bus owner and removes arb* and breq* and begins transfer operations.

f) Since all requesting agents participating in the bus request operation have obtained bus ownership, BREQ* is negated. Agent C, sensing that no bus request sequence is currently in progress, enters arbitration. Agent C progresses through a resolution phase and an acquisition phase as described above.

6.2.6.3 Timing sequence for high priority bus acquisition

Figure 6.2-7 shows the timing sequence for high priority bus acquisition. In this example there are three agents: A, B, and C. In terms of their arbitration IDs (those assigned during reset) Agent A has the highest priority, Agent B next highest, and Agent C the lowest. The sequence of events for this bus acquisition (assuming an idle bus) is given in the following steps:

a) In the same bus clock cycle, both Agents A and B desire normal priority use of the bus. Since no bus request sequence is in progress (BREQ* is negated), both agents place normal priority bus requests by asserting breq*, placing their IDs on arb<4..0>* with arb5* negated.
b) During the first bus clock cycle and the next two, the ARB* lines settle according to the back-off algorithm described previously, resulting in steady-state ARB* lines by the end of the third bus clock cycle.
c) Meanwhile, during the resolution phase described in the last step, Agent C desires high priority use of the bus. At the end of the third bus clock cycle, Agent A's arbitration ID is stable on the ARB* lines indicating that it is the next bus owner. During the next bus clock cycle, Agent A gets ownership of the bus. Agent A removes its arbitration ID from arb*, negates breq*, and begins transfer operations.
d) Agent B remains in the resolution phase and continues to assert breq* and drive its arbitration ID. Agent C enters into arbitration with Agent B because Agent C has a high priority request and a new arbitration operation (and a new resolution phase) has begun. This is true even though the current bus request sequence has not ended. When Agent C enters the arbitration, it asserts its breq*, drives its arbitration on the arb<4..0>* lines with arb5* asserted, indicating a high priority request.

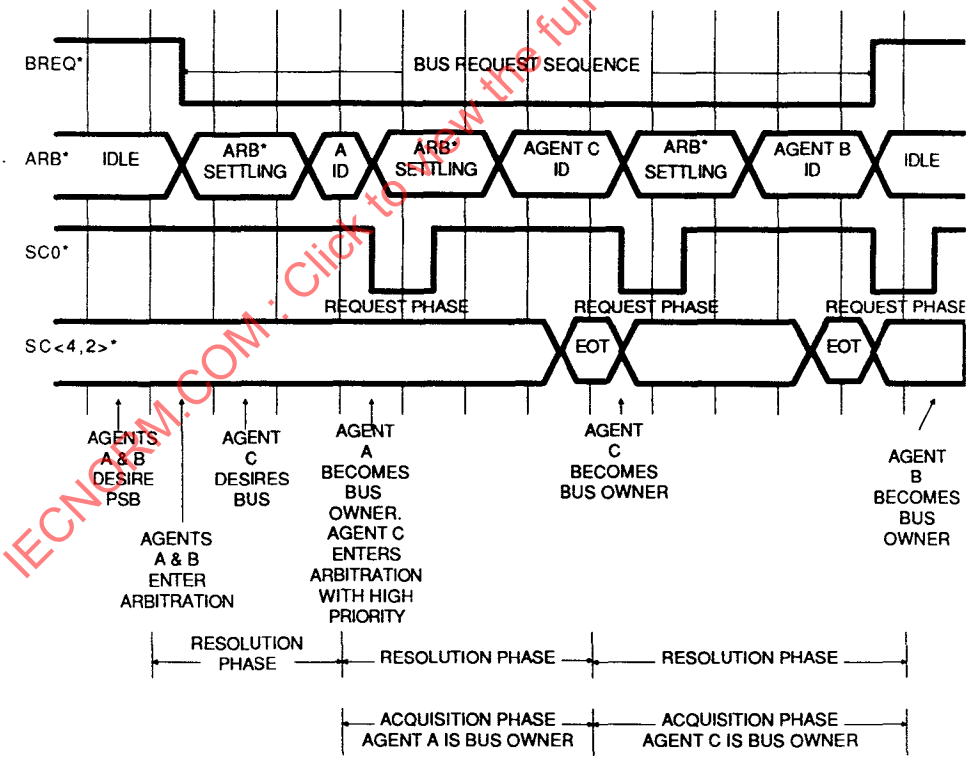


Figure assumes SC1* is negated

Figure 6.2-7—Timing sequence for high priority bus acquisition

e) By the end of the third bus clock cycle of the second arbitration operation, Agent C's arbitration ID is stable on the ARB* lines, indicating it is the next bus owner. This is true because Agent C placed a high priority request while Agent B placed a normal priority request. The resolution phase of the second arbitration operation is extended because the current transfer operation is not complete. When the current transfer operation completes, Agent A relinquishes bus ownership and Agent C, entering the acquisition phase, becomes the new bus owner. As Agent C leaves the resolution phase, it negates its arbitration ID from the arb* (including arb5*) and breq*, and begins transfer operations.

f) Agent B remains in the resolution phase with breq* asserted and its arbitration ID driven on the arb* lines. It then progresses through an arbitration operation (a resolution phase and an acquisition phase) as described previously.

6.3 Transfer operation

An agent must complete a successful arbitration operation or be parked before it can initiate a transfer operation. On completing an arbitration operation, one agent gains ownership of the bus and starts a transfer operation immediately. Only the agent that owns the bus may conduct transfer operations on the PSB.

An agent initiates a transfer operation by manipulating signals within two signal groups on the PSB: the address/data bus signal group and the system control signal group.

The address/data bus signal group (AD* and PAR* signal lines) is a set of 36 signal lines that contains address information during the request phase and data information during the reply phase of a transfer operation. The function of the AD* lines change depending on the phase of the transfer operation.

The system control signal group (SC*) is a set of 10 signals that defines the specific operation to be performed within the transfer operation. As with the address/data bus signal group, the function performed by each of the SC* signals is dependent on the phase of the transfer operation.

In the request phase, the bus owner uses the system control signal group to specify for the replying agent(s) the parameters of the impending transfer operation. These parameters are the specific command or operation to be performed, the address space of the command, and the width of the data transfer. The bus owner also uses the address/data bus signal group to transfer the address for the operation during the request phase.

During the reply phase, the functions performed by the system control signal group are redefined. The protocol divides the system control signal group into two smaller groups (except for broadcast messages), one driven by the bus owner and one driven by the replying agent. These two signal groups permit both agents to successfully complete the transfer operation. The handshake signals synchronize the data transfer operation. The bus owner identifies the last data transfer by asserting SC2* (EOT), which concludes the transfer operation. The replying agent returns agent status to the bus owner with its part of the handshake. During the reply phase of broadcast messages, the bus owner drives all SC* lines.

6.3.1 Types of transfer operation

There are three types of data transfers: the single data transfer, the sequential data transfer, and the broadcast message. Each type implements a precise handshake sequence. Each is described further in the following subclauses.

6.3.1.1 Single data transfer

The single data transfer supports one data transfer.

The transfer may be 8-, 16-, 24- or 32-bits wide and is completed with one handshake. During the request phase, the bus owner places address information on the AD* lines. Parity for the address word is placed on PAR<3..0>*. The bus owner uses the system control signal group to specify the data width, the address space, the phase (request), the transfer direction (read or write) and provide parity for the binary value present on the SC* lines.

During the reply phase, the bus owner negates SC0* to indicate that the request phase has completed and the reply phase is underway. The bus owner also asserts SC3* (bus owner ready) to indicate that it is ready to receive or send data (depending on whether the transfer operation is a read or write, respectively) and SC2* (EOT) to indicate that the current transfer is the last transfer. Additionally, the bus owner places parity for SC<3..0>* on SC9*.

During the reply phase, the replying agent asserts SC4* (replying agent ready) to indicate that it is ready to send or receive data. The replying agent also places agent status on SC<7..5>*, as well as parity for bits SC<7..4>* on SC8*.

When SC<4..2>* are asserted, the data transfer occurs and the transfer operation completes.

A single data transfer operation may be either a read (data transfer is from replying agent to bus owner) or a write (data transfer is from bus owner to replying agent) operation. The supplier of data, either the replying agent for a read operation or the bus owner for a write operation, must signal its part of the handshake status (SC3*) without waiting for the other agent.

During the reply phase of a read data transfer, the replying agent provides data on the AD* lines. The replying agent asserts SC4* (replying agent ready) when it has placed valid data on the AD* lines. The bus owner asserts SC2* (EOT) and SC3* (bus owner ready) when it is ready to accept data from the bus. When both SC3* and SC4* are asserted, the handshake occurs and the data transfer occurs synchronous with the falling edge of the bus clock.

Figure 6.3-1 shows a block diagram of the read data transfer and figure 6.3-2 shows the signal sequences for the read data transfer handshake.

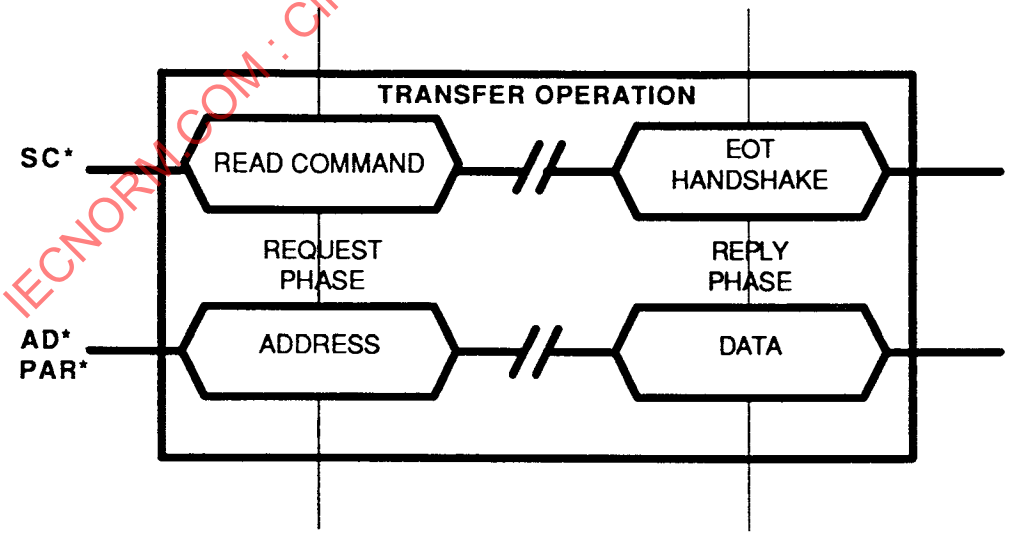


Figure 6.3-1 — Block diagram of single read data transfer

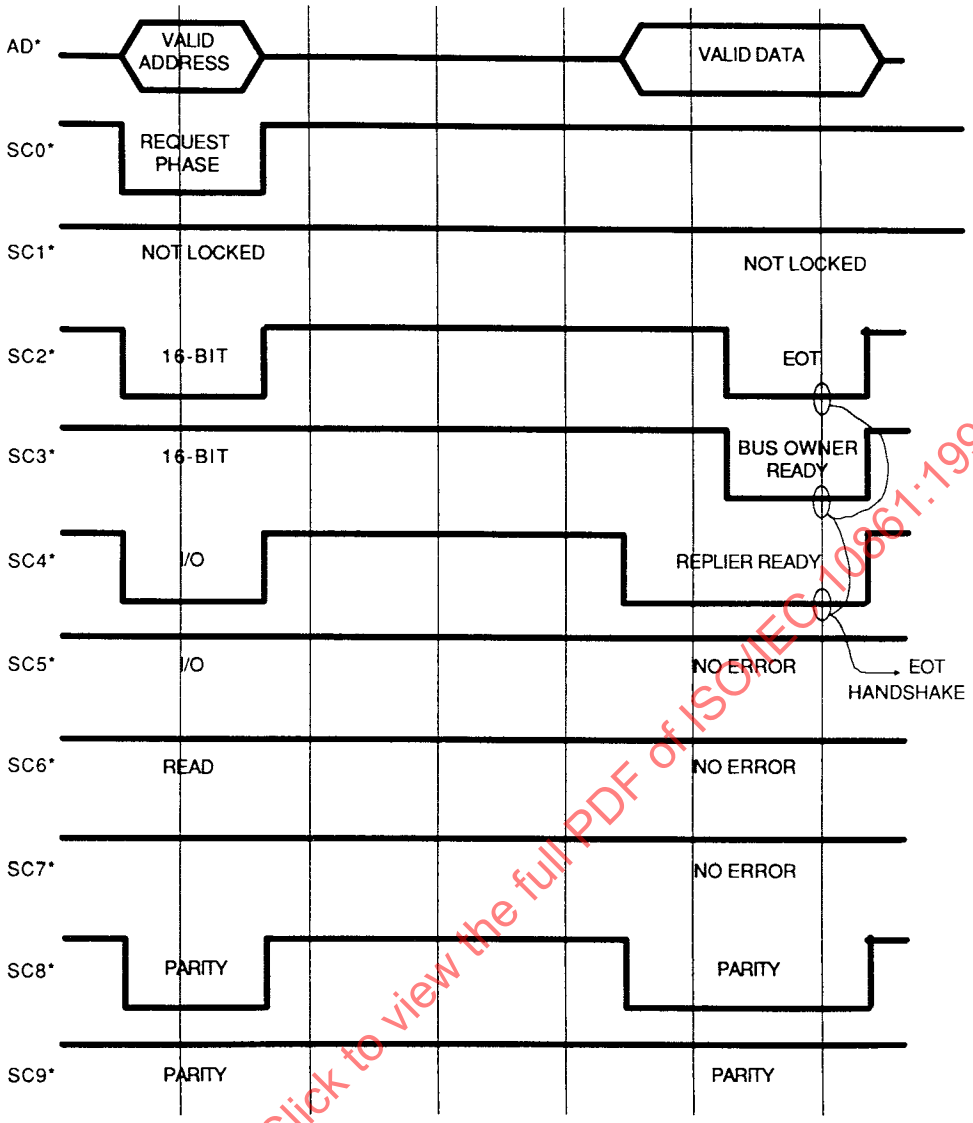


Figure 6.3-2—Timing for a single read data transfer

During the reply phase of a write data transfer, the bus owner provides data on the AD* lines. The bus owner asserts SC2* (EOT) and SC3* (bus owner ready) when it has placed valid data on the AD* lines. The replying agent asserts SC4* (replying agent ready) when it is ready to accept data from the bus. When both SC3* and SC4* are asserted, the handshake occurs and the data transfer occurs synchronous with the falling edge of the bus clock.

Figure 6.3-3 shows a block diagram of the write data transfer and figure 6.3-4 shows the signal sequences for the write data transfer handshake.

6.3.1.2 Sequential data transfer

The sequential data transfer differs from the single data transfer in that the operation consists of multiple data transfers during the reply phase. Figure 6.3-5 shows a block diagram of a typical sequential data transfer. Figure 6.3-6 shows the timing sequence for a sequential read data transfer and figure 6.3-7 shows the timing sequence a sequential write data transfer.

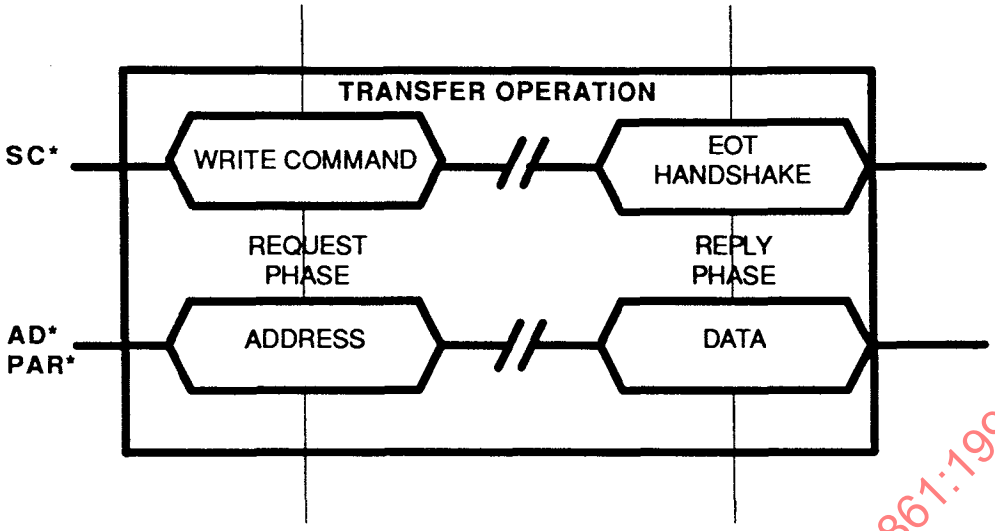


Figure 6.3-3—Block diagram of single write data transfer

A replying agent recognizes the difference between a single data transfer and a sequential data transfer by inspecting SC2* at the time of the handshake for the data transfer. The status transfer that occurs for the intermediate data transfers is different from the handshake that occurs at the final data transfer, in that SC2* (EOT) is not asserted, as figure 6.3-7 shows. As before, the sequential data transfer is terminated when the bus owner asserts SC2* (EOT). In a sequential data transfer, the replying agent must retain the request phase information for all data transfers.

6.3.1.3 Broadcast message

The broadcast message differs from the single and sequential data transfers in two areas. First, the request phase addresses all replying agents supporting message space rather than only one agent, and its use is limited only to message space. Second, replying agents do not provide agent status during the reply phase of the transfer operation. Figure 6.3-8 shows a diagram of a broadcast message and figure 6.3-9 shows the timing sequence for the handshake.

The broadcast message is only for message space operations and requires a destination address of AD<7..0>* asserted during the request phase.

In a broadcast, the replying agents do not respond with the normal handshake signal (on SC4*). Instead, the bus owner drives valid data onto the bus and asserts SC3* for at least eight bus clock cycles. To complete the data transfer in the broadcast message, the bus owner asserts SC4* for a single bus clock cycle to signal the handshake. During the final data transfer in the broadcast message, the bus owner asserts SC2* (EOT) in addition to SC3* and SC4*. For broadcast messages, the bus owner drives SC<8..4>*, signals that are normally driven by the replying agent. Since SC<7..5>* are used by replying agents to signal agent status, these lines are not used (negated by the bus owner) for broadcast messages to indicate successful completion of the operation. The handshake for this operation is detailed in figure 6.3-9.

6.3.2 Transfer operation lock

By asserting SC1* (lock), the bus owner ignores any pending requests and is permitted to lock the bus from use by other agents, subject to possible system time constraints. If the bus owner locks the bus, then all agents in the resolution phase remain in that phase until the bus owner unlocks the bus. When SC1*

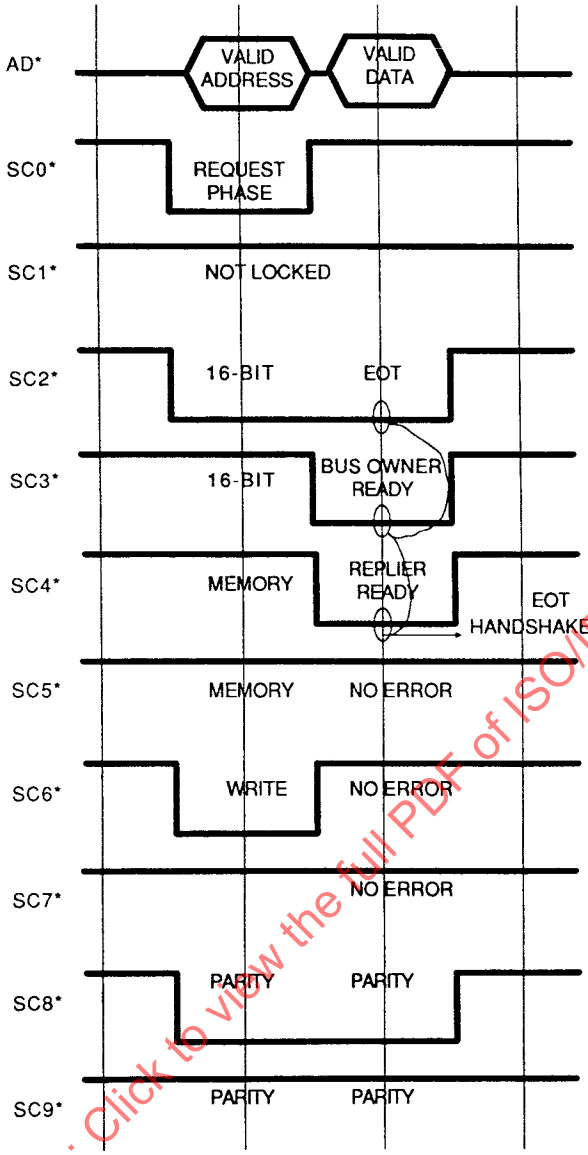


Figure 6.3-4 — Timing for single write data transfer

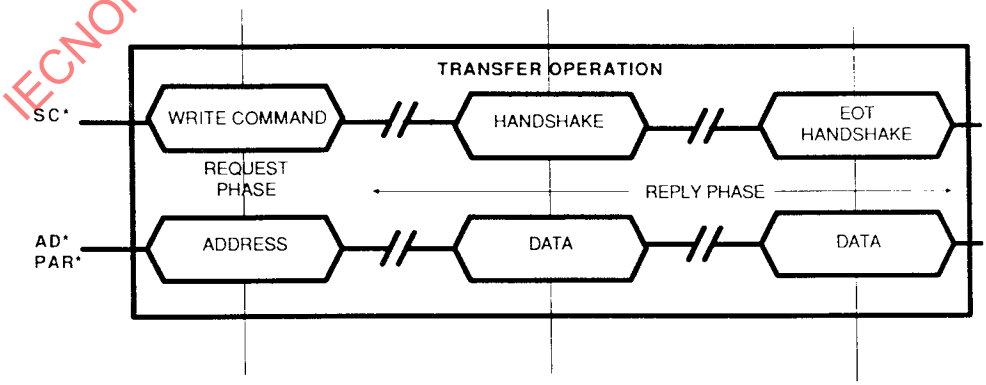


Figure 6.3-5 — Block diagram of sequential data transfer

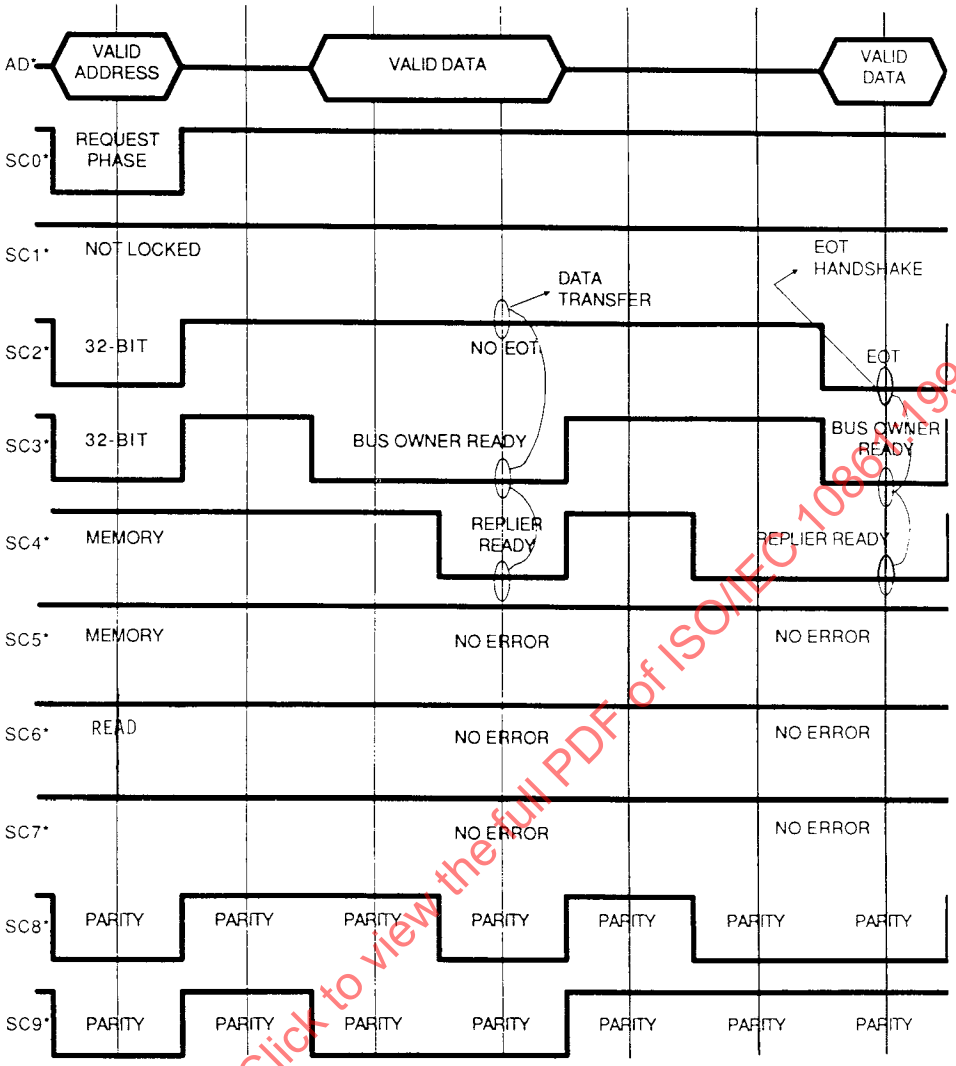


Figure 6.3-6—Timing for sequential read data transfer

(lock) is asserted by the current bus owner, no other agent can become the bus owner and any multi-ported resources that have been accessed remain unavailable to all other agents through all ports until the bus owner negates SC1*.

SC1* (lock) can be asserted upon obtaining bus ownership prior to the request phase to maintain bus ownership. Without the assertion of SC1*, the bus owner has three bus clock cycles to begin the transfer operation. The assertion of SC1* (lock) prior to the request phase has the effect of maintaining bus ownership, even though no transfer operation is in progress.

To guarantee mutual exclusion of multiported resources, it is required that the bus owner assert SC1* immediately following the request phase, at the latest, for the first data transfer to be locked and keep SC1* asserted until SC4* (replying agent ready) is asserted on the final data transfer of the locked transfer operation. Figure 6.3-10 shows the timing sequence for a lock operation.

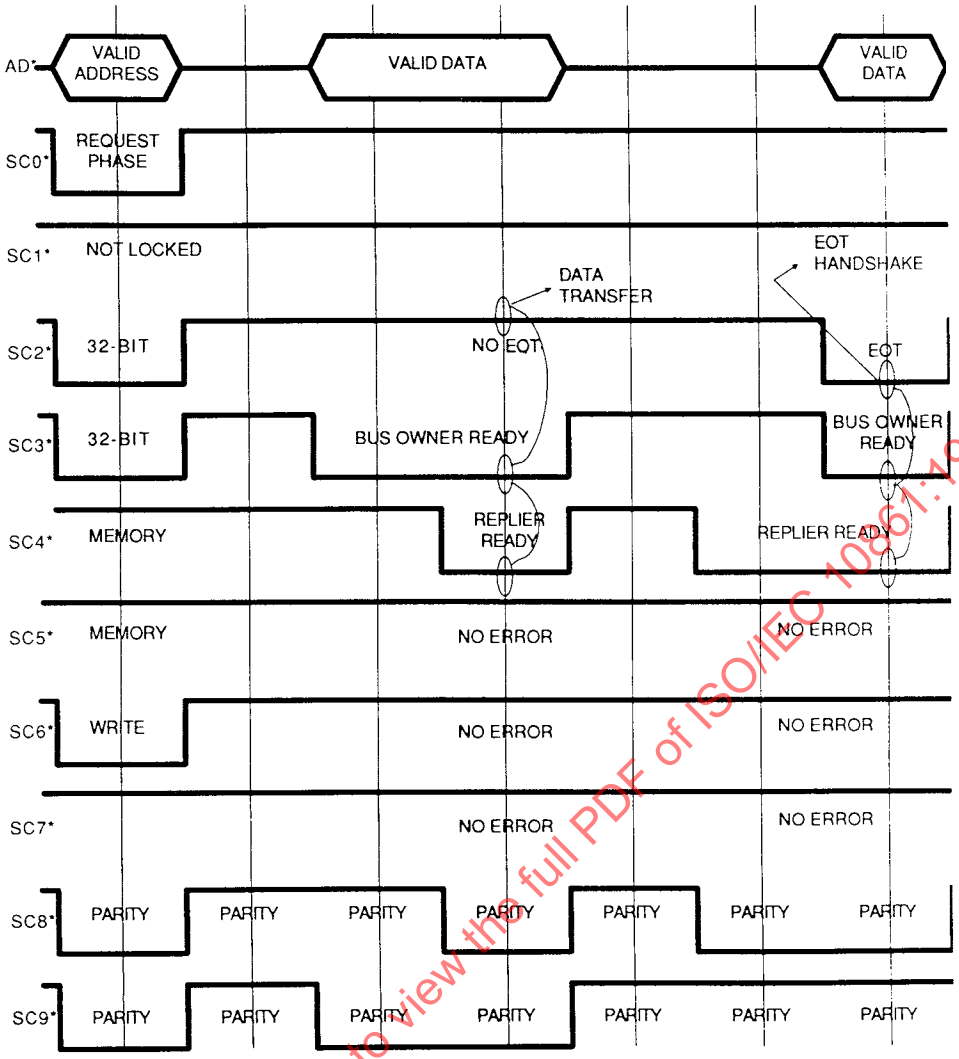


Figure 6.3-7 — Timing for sequential write data transfer

6.3.3 Agent status

During the reply phase of transfer operations, the bus owner continually monitors the SC<7..5>* lines for status information from the replying agent. Such information is called *agent status*. Agent status is sent to the bus owner during the execution of the transfer operations and indicates that the transfer operation is completing successfully or with one of several possible errors. Agent status conditions do not cause the execution of exception operations, nor do they terminate arbitration operations. However, an agent status that indicates an error does cause the current transfer operation to terminate.

There are six possible agent status codes that can be returned by the replying agent. Agent status is placed on SC<7..5>* simultaneously with the assertion of SC4* (replying agent ready) by the replying agent. These agent status codes are returned by the replying agent for accesses to any of the four address spaces of the PSB.

With the detection and signalling of the error status by the replying agent, all bus operations on the part of the replying agent cease until the bus owner asserts SC2* and SC3* to complete the operation (unless the bus owner has already completed the operation with the assertion of SC2*).

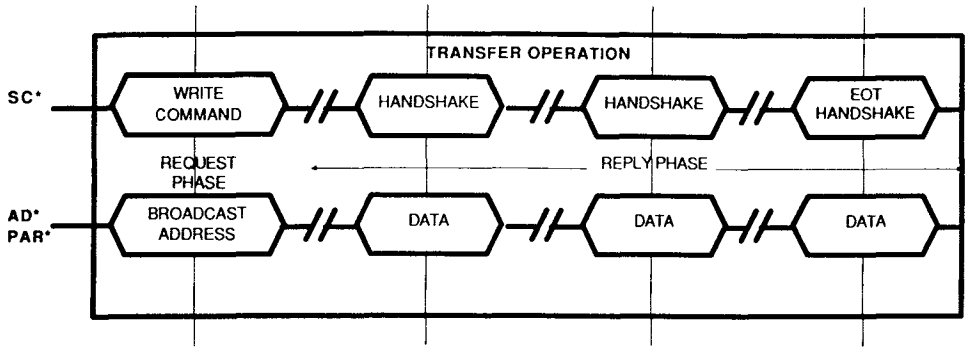


Figure 6.3-8—Block diagram of broadcast message

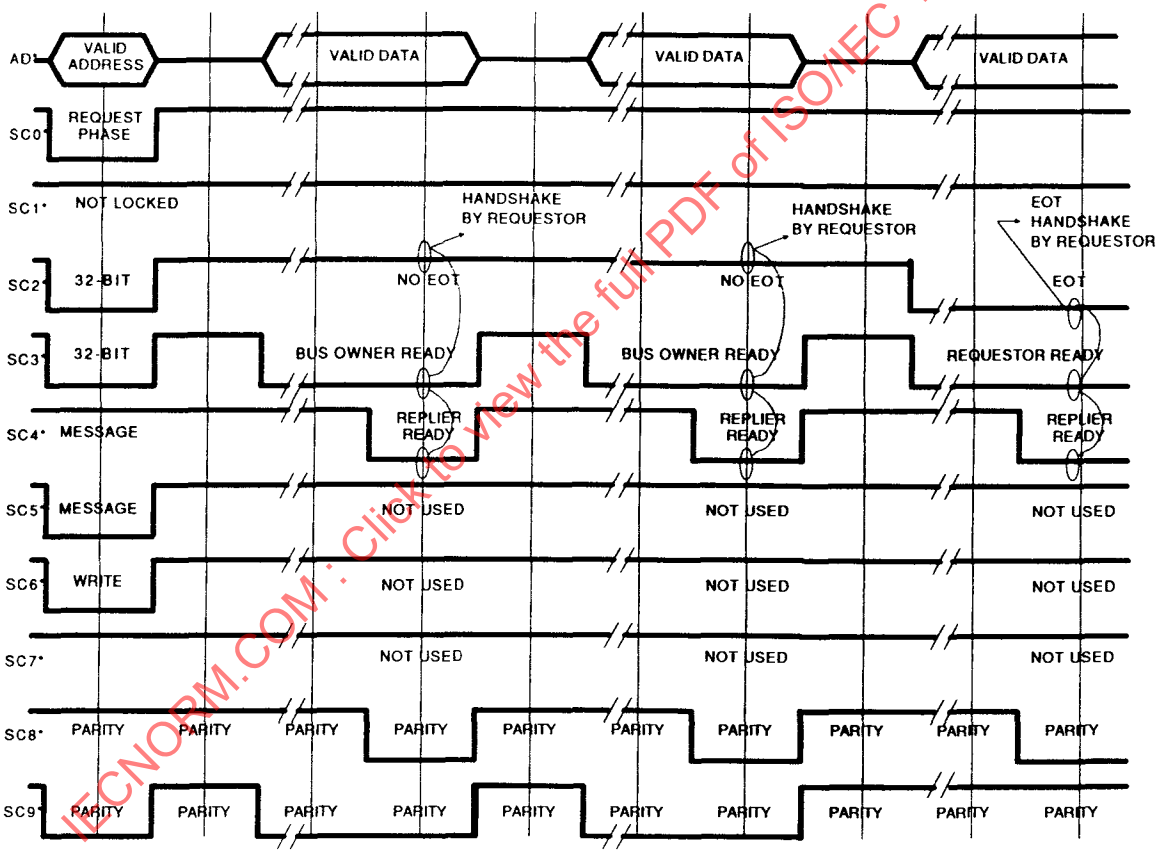


Figure 6.3-9—Timing for broadcast message

Table 6.3-1 shows the six possible agent status conditions and the associated codes that are placed on SC<7..5>* to indicate the conditions. Following table 6.3-1 is a discussion of each of the agent status conditions, including when they are detected, when they are signalled to the bus owner, and as an aid, possible recovery actions that can be performed by the bus owner.

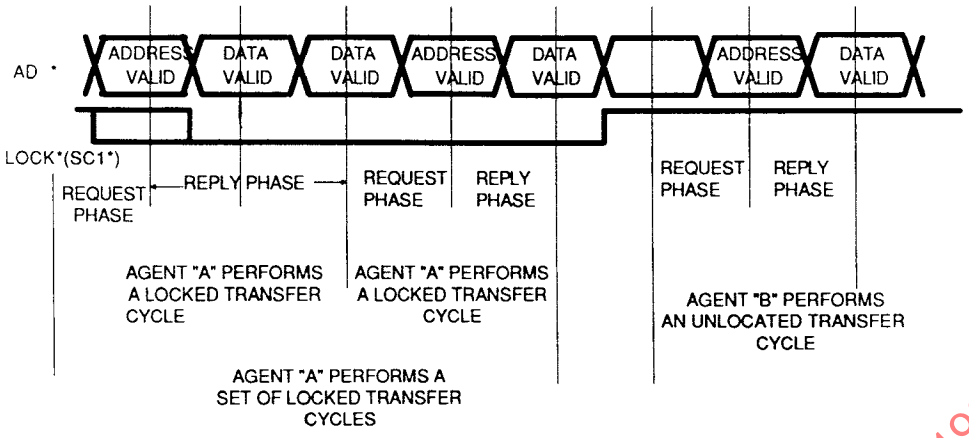


Figure 6.3-10—Timing sequence for LOCK operation

Table 6.3-1—Agent status codes

Agent status	SC7*	SC6*	SC5*
Reserved	L	L	L
Reserved	L	L	H
Data error	L	H	L
Negative acknowledge error	L	H	H
Transfer-not-understood error	H	L	L
Continuation error	H	L	H
Transfer-width error	H	H	L
No error	H	H	H

6.3.3.1 No error

The replying agent returns this agent status code when ready to exchange data and no error has occurred. The replying agent indicates this status by deasserting SC<7..5>*.

6.3.3.2 Negative acknowledge error

The negative acknowledge (NACK) error occurs when an otherwise legal operation is attempted to a busy replying agent. It is detected during the request phase of a transfer or after the request phase during replying agent decode and signalled on the first handshake after detection. Recovery may typically consist of trying the operation again at a later time. The replying agent indicates this status by deasserting SC5* and SC6* and asserting SC7*.

6.3.3.3 Transfer-width error

This error occurs when the requested transfer width (indicated by SC<3,2>* during the request phase) is not supported by the replying agent. Refer to 6.3.6 for the conditions necessary for the replying agent

to signal a transfer-width error. It is signalled in the first handshake after the request phase. As a possible recovery action the operation may be tried as multiple transfers with reduced data width. The replying agent signals this error by asserting SC5* and deasserting SC6* and SC7*.

Figure 6.3-11 shows the timing sequence for a transfer operation that contains a transfer-width error.

6.3.3.4 Continuation error

This error is caused by an attempt to perform an otherwise legal sequential transfer to an agent that does not support sequential transfers or that exceeds the memory or buffer space of the replying agent. Detection occurs during the reply phase for the handshake with SC2* not asserted. The continuation error status is reported in the next handshake. A possible recovery action would be to perform another

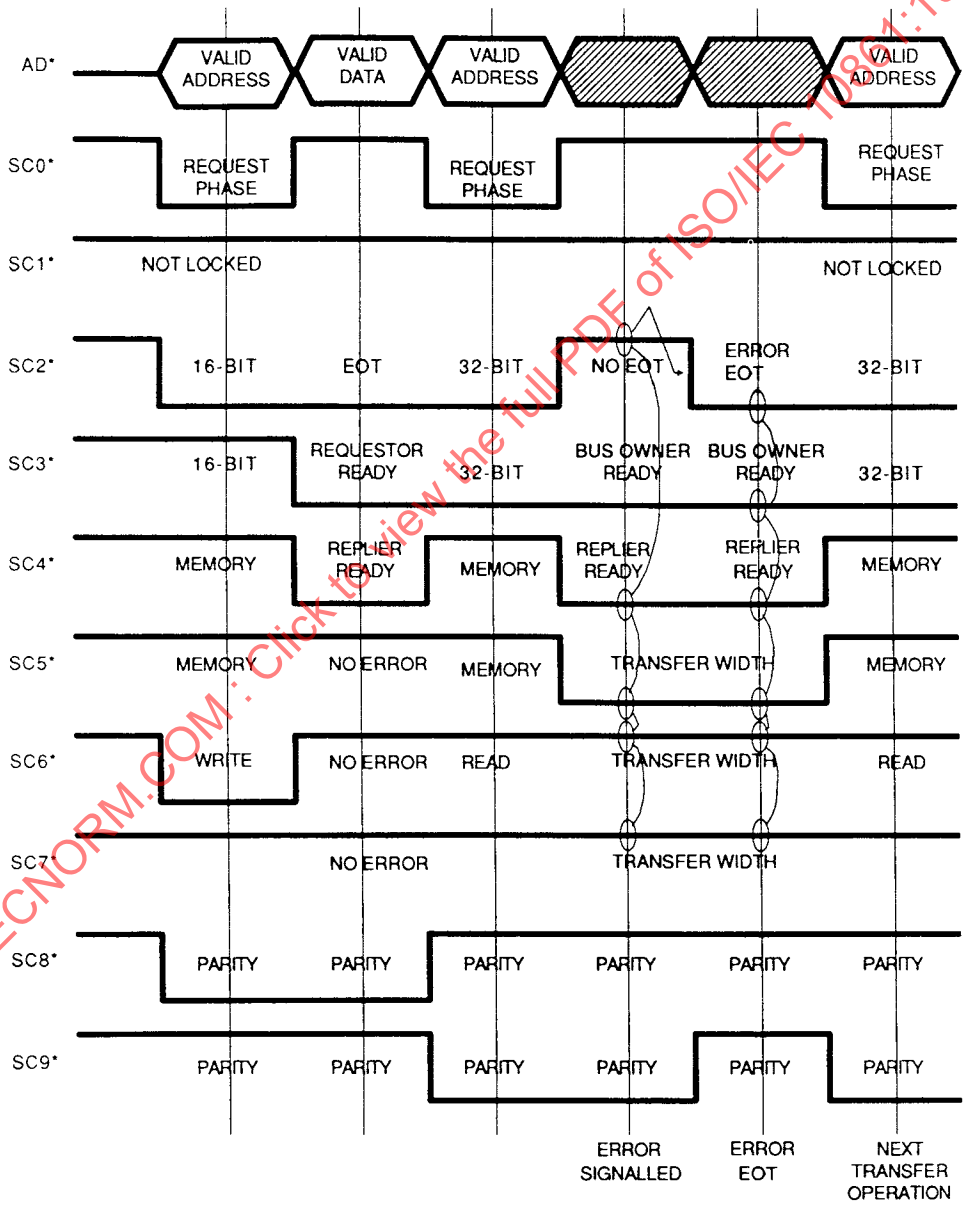


Figure 6.3-11—Timing sequence for transfer-width error

transfer operation with the address of the location accessed when the continuation error was signalled. The replying agent signals this error by deasserting SC<7,5>* and asserting SC6*.

Figure 6.3-12 shows the timing for a transfer operation that contains a continuation error.

6.3.3.5 Data error

This error occurs when there is a data integrity problem within the replying agent; for example, if parity on a memory board is incorrect. This error is detected during data access and signalled during the reply phase, for the handshake associated with the questionable data. Recovery from this error type depends on the particular implementation of the system. The replying agent indicates this error by asserting SC<7,5>* and deasserting SC6*.

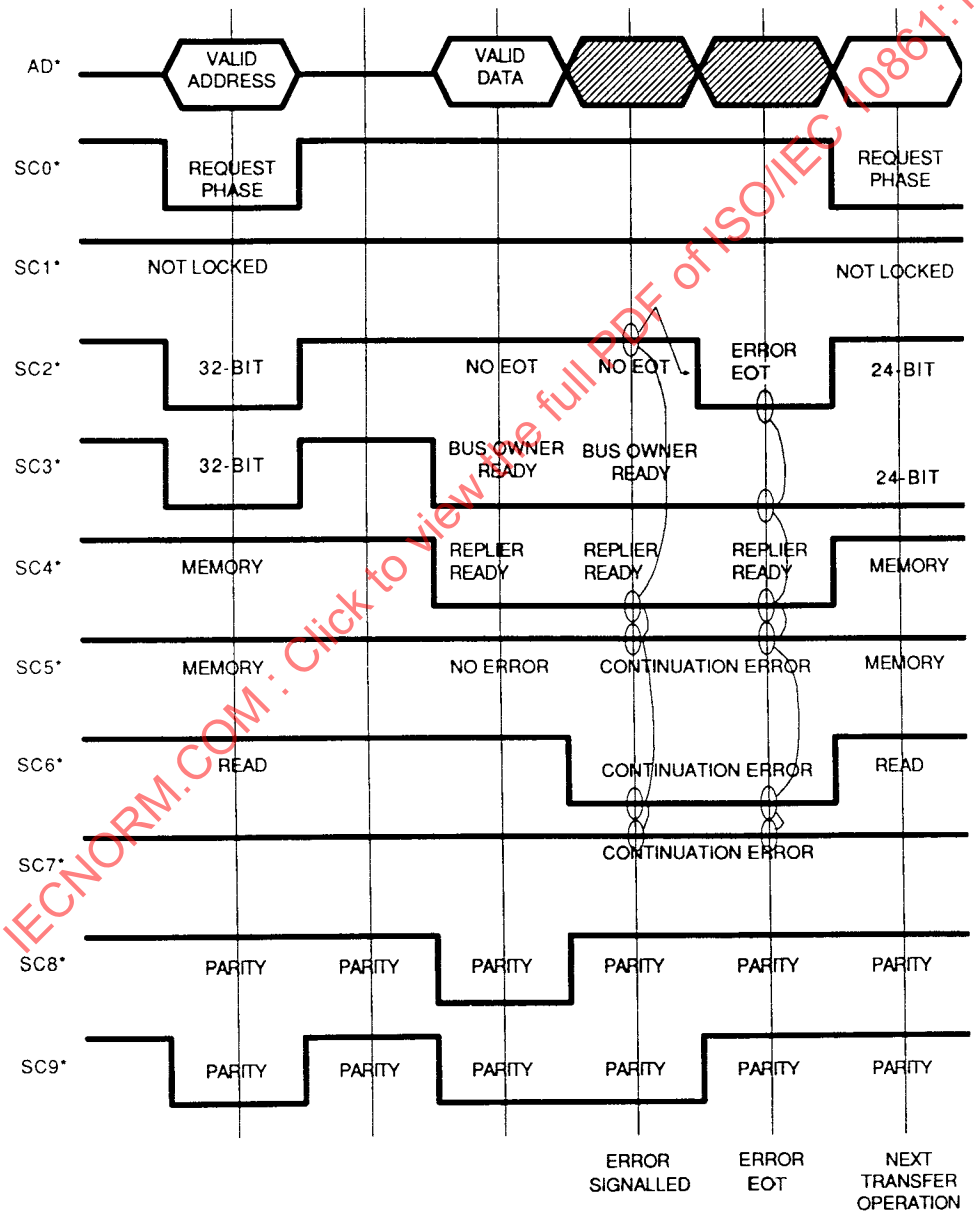


Figure 6.3-12—Timing sequence for continuation error

6.3.3.6 Transfer-not-understood error

This error is caused by any illegal request (e.g., 8- or 24-bit wide messages), any illegal data phase (e.g., 24-bit block transfer), or any legal bus operation that is not supported by the replying agent and is not recoverable (e.g., a write to a read-only memory). This condition may be detected at any time during the transfer operation and is signalled for the first handshake that is affected by the condition. Recovery from this error is usually not possible. The replying agent indicates this error by asserting SC<6,5>* and deasserting SC7*.

6.3.3.7 Priority of errors

Generally, the first detected error is reported and subsequent errors occurring within the same operation are ignored. If errors occur at the same time, agent errors have a lower priority than exception errors. For more information concerning exception errors, see Priority of Errors under 6.4, Exception operation. The detection and signalling of agent status errors are prioritized as follows:

a) Request phase agent errors take priority over reply phase errors.
b) Multiple request phase errors are prioritized in the following order:
   1) Transfer not understood (highest priority)
   2) Data width
   3) NACK (lowest priority)
c) Multiple reply phase errors are prioritized in the following order:
   1) Transfer not understood (highest priority)
   2) NACK
   3) Continuation
   3) Data error (lowest priority)

6.3.4 Address space definitions in transfer operations

During the request phase of a transfer operation, the bus owner uses the SC<5,4>* signals to select one of four independent address spaces on the PSB:

SC5*	SC4*	Address space selected
H	H	Memory space
H	L	I/O space
L	H	Message space
L	L	Interconnect space

Each of the four address spaces has different characteristics, capabilities, and uses. When an agent performs a transfer operation involving one of the address spaces, that agent is responsible for adhering to the protocol governing access to and use of that address space.

The attributes for each address space are defined in the following paragraphs and summarized in table 6.3-2. The description of each address space includes a figure showing the format for the address that is required within that address space.

6.3.4.1 Memory space attributes

The memory space defines the memory available on the PSB. The IEEE 1296 standard defines the use of the memory space with specific protocol, as follows:

Table 6.3-2—Address space summary

Address space	Address space size	Sequence type	Transfer width	Sequential transfers	Number of replying agents
Memory	2^{32} bytes	Read/write	8, 16, 24, or 32 bits	Supported with increment	one
I/O	2^{16} 8-bit ports	Read/write	8, 16, 24, or 32 bits	Supported without increment	one
Message	$2^{8}-1$ agents with 1 broadcast address	Write only	32 bits	Supported without increment	one or all
Interconnect	2^{9} 8-bit registers per agent	Read/write	8 bits	Not supported	one

a) The data width (during the reply phase of the transfer operation) for a memory space operation must be either 8, 16, 24, or 32 bits.
b) The address width (during the request phase of the transfer operation) must always be 32 bits for an access to memory space. See figure 6.3-13.
c) The bus owner that accesses memory space must receive a response from only one replying agent. Initialization should guarantee unique addresses among replying agents in memory space.
d) The replying agent must increment the initial address given by the bus owner to obtain the address for subsequent accesses of data when performing a transfer operation that requires sequential accesses of memory.
e) For sequential data transfers, the address incrementing algorithm varies depending on the data width that is required by the bus owner. For an 8-bit transfer, the address is incremented by one at each access; for a 16-bit transfer, the address is incremented by two at each access, and so on. Refer to figure 6.3-14.
f) The protocol does not support agents that perform sequential memory accesses at the 24-bit data width or 16-bit data width not aligned on 16-bit boundaries (addresses with AD0* = L).

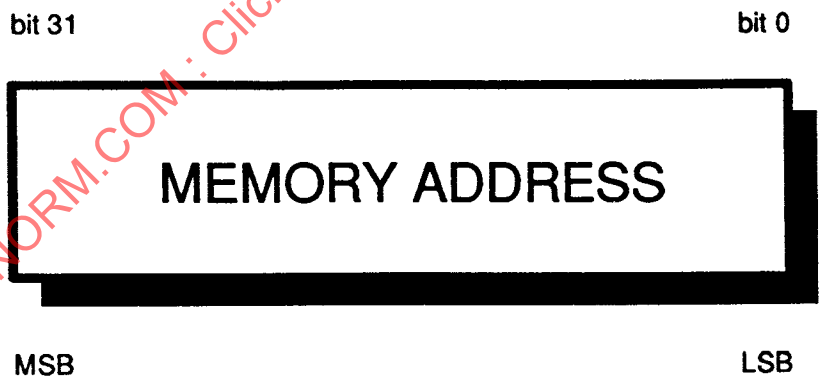


Figure 6.3.13—Address format for sequences using memory address space

6.3.4.2 I/O space attributes

The I/O space defines the I/O devices that are available on the PSB. The IEEE 1296 standard defines the use of the I/O space on the PSB with specific protocol, as follows:

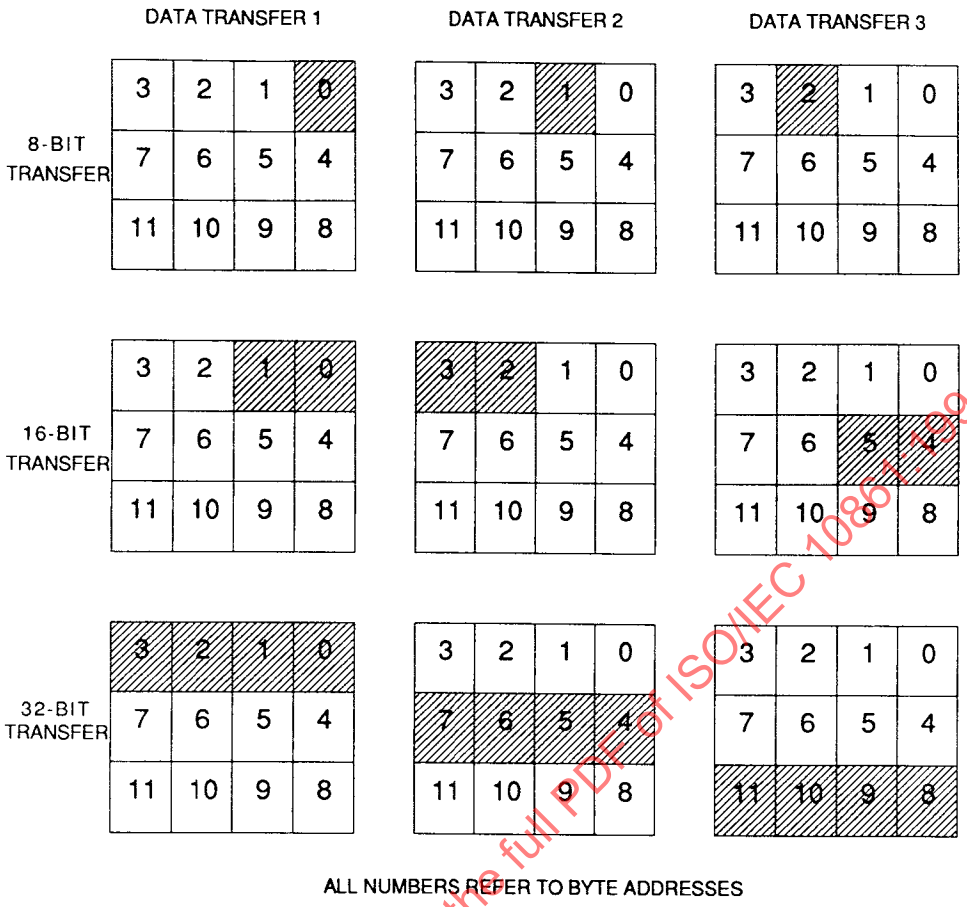


Figure 6.3-14—Block diagram of sequential data transfers in memory space

a) The data width (during the reply phase of the transfer operation) for an I/O space operation must be either 8, 16, 24, or 32 bits.
b) The address width during the request phase of the transfer operation must always be 16 bits for an access to I/O space. See figure 6.3-15.
c) Each I/O address must correspond to one and only one I/O device that can be a replying agent. Initialization should guarantee unique addresses among replying agents in I/O space.
d) Sequential data transfers to I/O space are directed to the same I/O address; the replying agent does not increment in initial address for subsequent I/O operations when performing a transfer operation that requires sequential data transfers to I/O space.
e) The protocol does not support agents that perform sequential data transfers to I/O space at the 24-bit data width or at the nonaligned 16-bit data width (addresses with AD0* = L).

6.3.4.3 Message space attributes

The message space provides interagent communications on the PSB. The message-passing capability, at its simplest, provides an interrupt signalling mechanism. Additionally, message space provides parameter-passing and data transfer mechanisms. The IEEE 1296 standard defines the use of the message space with specific protocol, as follows:

a) The data transfer is one-directional, from the bus owner to the replying agent(s).
b) The data width for a message space operation must be 32 bits.

Figure 6.3-15—Address format for sequences using I/O address space

c) The address width (during the request phase of the transfer operation) must always be 16 bits (8-bit source address and 8-bit destination address) for an access to message space. See figure 6.3-16.
d) The message space supports agents that perform broadcast operations to all replying agents.
e) The message space supports sequential data transfers.
f) Sequential data transfers to message space are directed to the same message address; the replying agent does not increment the initial address for subsequent message space operations when performing a transfer operation that requires sequential data transfers.

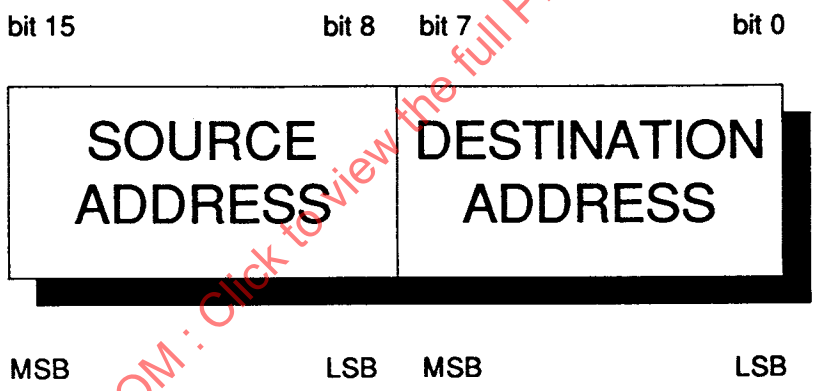


Figure 6.3-16—Address format for sequences using message address space

6.3.4.4 Interconnect space attributes

The interconnect space provides configuration information for the resources on the PSB. The interconnect space is intended to be used for system initialization and diagnostic operations. The IEEE 1296 standard defines the use of the interconnect space with specific protocol, as follows:

a) The data transfer is bidirectional (width is 8-bits only).
b) Each agent on the PSB must have its own unique cardslot address, assigned by the CSM.
c) The address width (during the request phase of the transfer operation) must always be 16 bits, with 9 bits of offset register address and 5 bits of board identification address. During the request phase, AD<1.0>* are deasserted to ensure proper data alignment. See figure 6.3-17.
d) The protocol does not support sequential data transfers or broadcast operations in interconnect space.

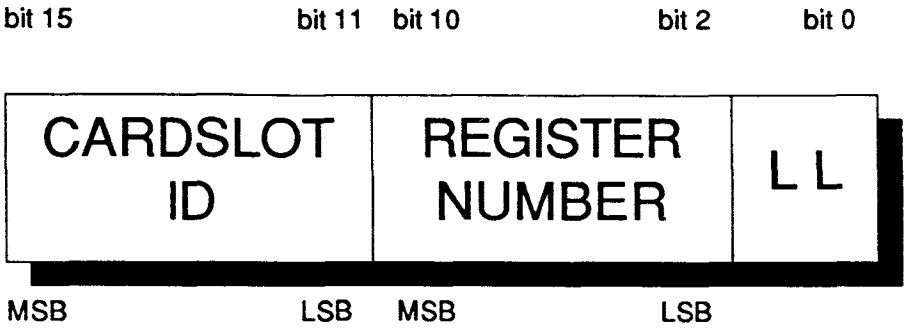


Figure 6.3-17—Address format for interconnect address space

6.3.5 Data width during transfer operations

The data width for an operation is defined via the data-width parameter that the bus owner places onto the SC<3.2>* lines during the request phase of an operation. The selectable options are as follows:

SC3*	SC2*	Data width selected
H	H	8-bit
H	L	16-bit
L	H	24-bit
L	L	32-bit

When the bus owner performs a transfer operation on the PSB, the agents are allowed several data alignment options depending on the nature of the agents and the operation. Data alignment requirements for the four address spaces are described in the following three sections: alignment for agents doing memory and I/O space operations, alignment for agents doing message space operations, and alignment for agents doing interconnect space operations.

6.3.5.1 Data alignment in memory and I/O spaces

The protocol allows the use of ten data alignments for bus owners and replying agents that use the memory or I/O address spaces.

The bus owner controls the alignment of the interface by controlling the condition of address bits 0 and 1 via address lines AD<1..0>* and controls the data width via the SC<3,2>* lines on the interface.

Table 6.3-3 lists the alignment options for the interface. In table 6.3-3, the columns represent the four bytes of the 32-bit address/data bus on the PSB.

The alignment restrictions for sequential data transfers are as follows:

a) For a sequential 8-bit operation in memory space, AD0* may be either H or L, depending on which byte is transferred first, and the data is transferred alternately on either AD<7..0>* or AD<15..8>*, with the initial byte of the transfer specified as in table 6.3-3. For sequential data transfers in I/O space, the initial address, supplied during the request phase, is unchanged for the duration of the transfer operation. The data for sequential 8-bit I/O space operations is always supplied on AD<7..0>* or AD<15..8>*, depending upon AD0* during the request phase.

Table 6.3-3—Data alignments for operations using memory and I/O space

	Request phase bits				Reply phase byte alignments			
	Width		Address					
Transfer width	SC3*	SC2*	AD1*	AD0*	AD<31..24>*	AD<23..16>*	AD<15..8>*	AD<7..0>*
8 bits (Note a)	H	H	H	H	I	I	I	BA0
8 bits (Note a)	H	H	H	L	I	I	BA1	I
8 bits (Note a)	H	H	L	H	I	I	I	BA2
8 bits (Note a)	H	H	L	L	I	I	BA3	I
16 bits (Note a)	H	L	H	H	I	I	BA1	BA0
16 bits (Note a)	H	L	L	H	I	I	BA3	BA2
16 bits (Note d)	H	L	H	L	I	BA2	BA1	I
24 bits (Note d)	L	H	H	H	I	BA2	BA1	BA0
24 bits (Note d)	L	H	H	L	BA3	BA2	BA1	I
32 bits (Notes a,d)	L	L	H	H	BA3	BA2	BA1	BA0

NOTES
a) Identifies those alignments allowed with sequential transfer operations.
b) Abbreviations (values of AD* lines apply to request phase)
BA0 = A data byte with address corresponding to AD1* = H and AD0* = H.
BA1 = A data byte with address corresponding to AD1* = H and AD0* = L.
BA2 = A data byte with address corresponding to AD1* = L and AD0* = H.
BA3 = A data byte with address corresponding to AD1* = L and AD0* = L.
I = Invalid portion of bus; must be ignored by the data recipient, but can be driven by the data supplier.
c) All unlisted configurations in the request phase are not supported in the protocol and reported as transfer not understood agent status.
d) Requires a 32-bit agent. Eight- and 16-bit replying agents must return a transfer-width error.

b) For a sequential 16-bit operation, AD0* must always be H and the data is transferred on AD<15..0>*.
c) No sequential 24-bit operations or 24-bit agents are supported.
d) For a sequential 32-bit operation, AD<1,0>* must both be H and the data is transferred on AD<31..0>*.

6.3.5.2 Data alignment in message space

The protocol allows the use of one data alignment option for bus owners and replying agents that use the message address space. The bus owner sets the data width to 32-bits via the SC <3,2>* lines on the interface. Table 6.3-4 shows the data alignment for message space operations.

An agent uses message space to transfer 32-bit messages or to send/receive interrupts on the PSB.

Table 6.3-4—Data alignment for message space operations

	Request phase bits				Reply phase byte alignments			
	Width		Address					
Transfer width	SC3*	SC2*	AD1*	AD0*	AD<31..24>*	AD<23..16>*	AD<15..8>*	AD<7..0>*
32-bits	L	L	X	X	D	D	D	D

NOTES
a) Abbreviations
D = Valid data on the bus.
X = Either high or low logic level is acceptable.
b) See typical message format for byte ordering details.
c) All unlisted configurations in the request phase are not supported in the protocol and reported as transfer-not-understood agent status.

6.3.5.3 Data alignment in interconnect space

The protocol allows the use of only one data alignment option for bus owners and replying agents that use the interconnect address space. The bus owner controls the alignment of the interface by negating AD<1,0>* during the request phase and controls the data width via the SC<3,2>* lines on the interface.

Table 6.3-5 lists the only valid alignment option for the interface. In table 6.3-5, the columns represent the four bytes of the 32-bit address/data bus on the PSB.

Table 6.3-5—Data alignment for operations using interconnect space

	Request phase bits				Reply phase byte alignments			
	Width		Address					
Transfer width	SC3*	SC2*	AD1*	AD0*	AD<31..24>*	AD<23..16>*	AD<15..8>*	AD<7..0>*
8-bits	H	H	H	H	I	I	I	D

NOTES
a) Sequential transfer operations are not allowed in interconnect space.
b) Abbreviations
D = Valid data on the bus.
I = Invalid portion of bus; must be ignored by replying agent during write sequences, but can be driven by bus owner.
c) All unlisted configurations in the request phase are not supported in the protocol and are reported as transfer-not-understood agent status.

6.3.6 Transfer-width error reporting during transfer operations

All replying agents may check for transfer-width errors, including interface width mismatch and data misalignment problems. Table 6.3-6 shows the data alignment conditions that cause an 8-, 16-, or 32-bit agent to generate transfer-width errors on memory space or I/O space operations.

A width error cannot occur in message space or interconnect space since each has only one allowable transfer width. This error is signalled as a transfer-not-understood error.

6.3.7 Data alignment interface example

The data alignment on the PSB requires that the interface place all 8-bit operations and all 16-bit operations aligned on 16-bit boundaries onto the AD<15..0>* signal lines. This alignment policy forces particular interface constraints for agents. Figure 6.3-18 shows how 32-bit agents perform a 16-bit swap operation. The 16-bit swap logic is not required for 8-bit or 16-bit agents.

Table 6.3-6—Width errors for memory and I/O space operations

Request phase bits				Width error generated		
Width		Address				
SC3*	SC2*	AD1*	AD0*	8-bit agent	16-bit agent	32-bit agent
H	H	X	X	No	No	No
H	L	X	H	Yes	No	No
H	L	H	L	Yes	Yes	No
L	H	H	X	Yes	Yes	No
L	L	H	H	Yes	Yes	No

NOTE—X = Either high or low logic level is acceptable.

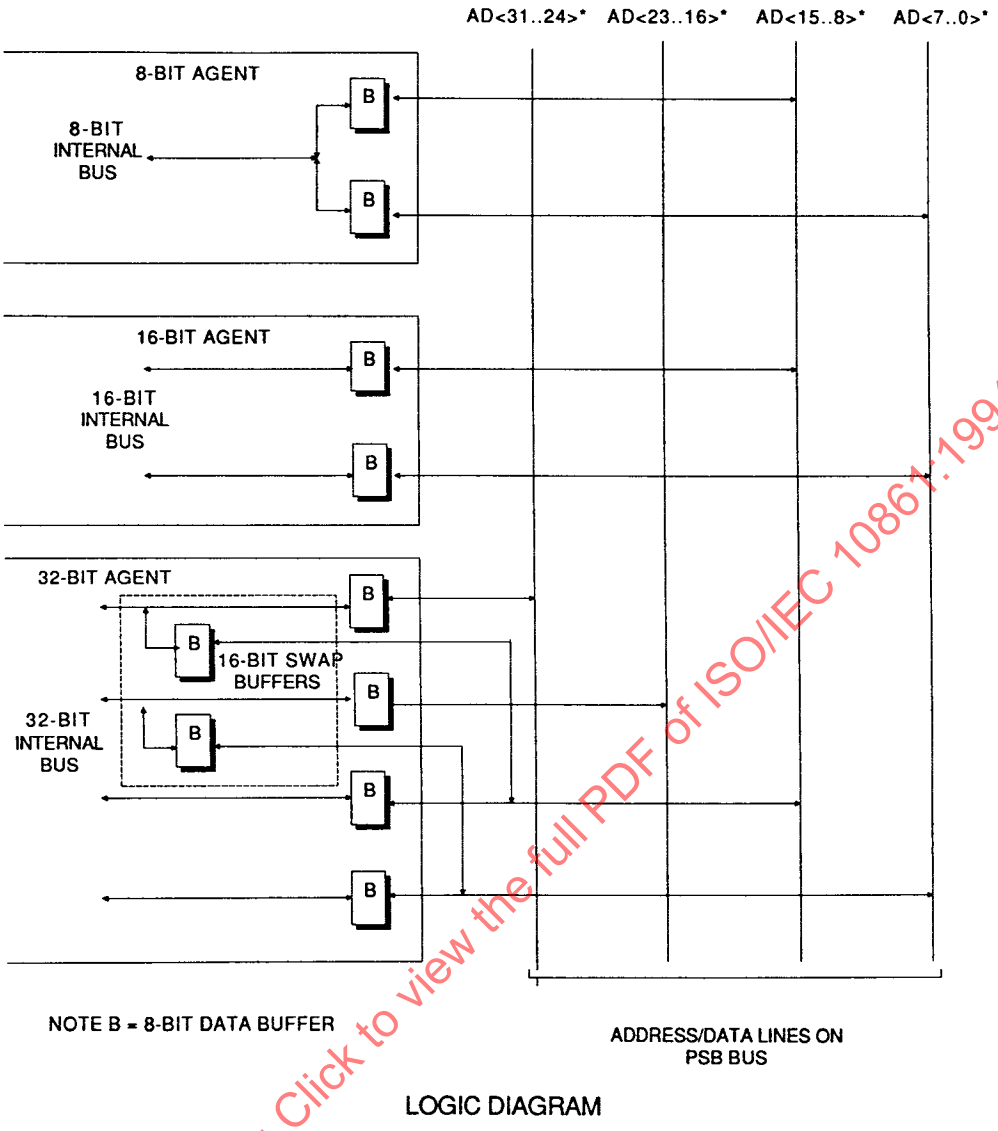


Figure 6.3-18—Interface requirements for data alignment

All agents with 8-bit interfaces must be connected to AD<15..0>* on the address/data bus and must enable either AD<15..8>* or AD<7..0>* depending on the condition of address bit AD0* and the SC<5.4>* (address space) during the request phase of a transfer operation. All 16-bit agents are connected directly to the AD<15..0>*; they do not require additional circuitry on the bus. All 32-bit agents require 16-bit swapping logic to move data as required on the bus for memory and I/O space transfer operations. Nonaligned 16-bit as well as all 24-bit and 32-bit operations do not use the 16-bit swapping logic.

6.4 Exception operation

The exception operation is an error reporting and recovery tool. An agent initiates an exception operation only as a result of sensing an exception. There are two major classes of exceptions. One is the time-out exception and the other is the bus error exception. The time-out exception is caused by a bus

agent taking too long to participate in a bus operation. A bus error exception can be caused by several errors. Both types of exceptions are discussed later in this clause.

The exception operation provides a method for reporting and recovering from an exception. The exception operation has two purposes in the protocol: first, it provides systematic termination of activity on the PSB and second, it provides idle-time on the bus before allowing agents to resume operation. The two purposes correspond directly to the two phases of the exception operation, the signal phase and the recovery phase.

The signal phase of the exception operation begins when an exception-detecting agent senses an exception and asserts either BUSERR* or TIMOUT*, depending on the type of exception that occurred. The assertion of either signal indicates to all agents that an exception has occurred. An exception-detecting agent may be either the control services module (CSM) or any agent capable of detecting an exception.

When either BUSERR* or TIMOUT* is detected active, the current bus owner aborts any transfer operations and inhibits them until the exception operation is completed. Additionally, the current arbitration operation is halted and restarted only after the exception operation is completed. The net effect of the exception operation is to terminate all bus activity and to hold the bus idle for a fixed amount of time. The signal phase continues until the exception signal (BUSERR* OR TIMOUT*) is negated. The signal phase may last one or more bus clock cycles.

The recovery phase of the exception operation begins when the exception signal is negated. Arbitration may begin on the first bus clock cycle after the exception signal is negated. The recovery phase lasts at least three bus clock cycles. Transfer operations may resume at the end of the recovery phase, when an agent has won access rights to the bus.

The CSM or any capable agent indicates the occurrence of a bus error by asserting buserr* onto the bus. A time-out exception can only be asserted by the CSM. The CSM asserts TIMOUT* onto the bus to indicate a time-out exception. To signal an exception, the exception must be asserted for at least one bus clock cycle. There is no maximum period for the assertion of the exception signal; however, asserting the signal for more than one bus clock cycle extends the length of the signal phase.

Figure 6.4-1 shows how an exception affects ongoing bus operations.

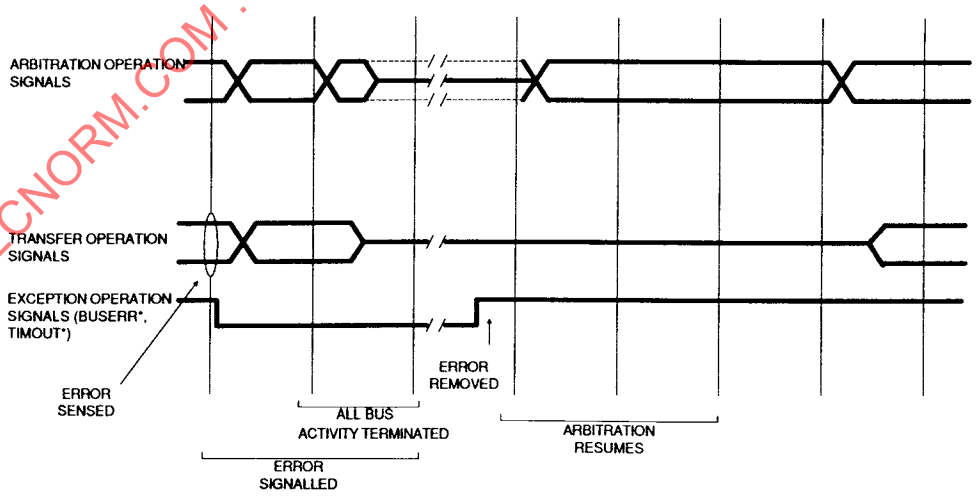


Figure 6.4-1—Exception cycle signal relationships

6.4.1 Causes of exception operations

Exception operations are initiated when an agent detects the assertion of either BUSERR* or TIMOUT*. Each of the exceptions serves a different purpose, and each is defined in the following subclauses.

6.4.1.1 Time-out exception

The CSM indicates a time-out exception on the PSB when it detects that the delay between handshakes for data transfers within a single transfer operation exceeds the specified duration. The time limit is the same for all types of transfer operations and for operations to all address spaces. The PSB provides a dedicated line (TIMOUT*) on the bus for passing the time-out exception among all agents.

Figure 6.4-2 shows the timing for signalling a TIMOUT* exception. As the figure shows, the CSM asserts TIMOUT* when the time between successive handshakes or the request phase and the first handshake has been exceeded. The CSM responsibilities for the generation and timing of the TIMOUT* signal are discussed in 6.5.4.

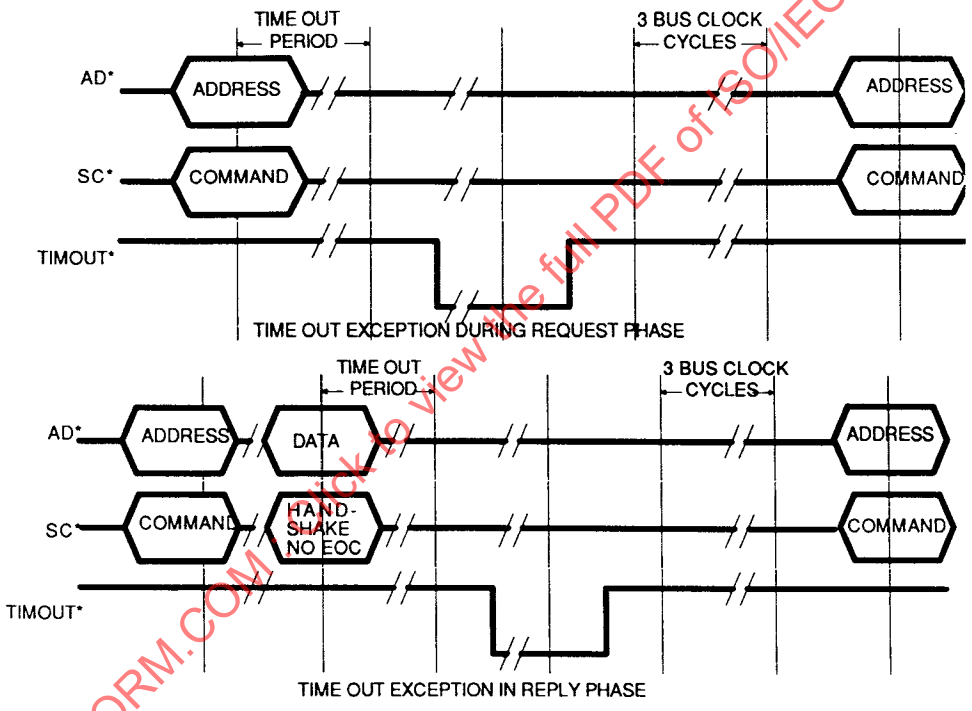


Figure 6.4-2—Signalling a TIMOUT* exception

6.4.1.2 Bus error exception

Agents assert the buserr* line whenever they detect a problem with data, address, or control information. All agents must continually monitor this line to ensure that information on the bus is valid with respect to the interface logic.

The bus error exception is generated whenever there is a parity error or protocol violation on the SC* lines. It is also generated by parity errors on the AD* lines when specified as valid by the SC* lines, or

bus owner reply phase termination. Parity is not checked during the assignment of cardslot and arbitration IDs.

In response to detecting BUSERR* asserted, an agent may assert its buserr* line. This is required for some implementations that may require multiple bus clock cycles to establish a known state after a bus error exception. Each agent asserting its buserr* line must negate buserr* when a known state has been established.

6.4.1.2.1 Conditions

The conditions under which parity must be valid at the falling edge of BCLK* are as follows:

a) SC* lines at all times.
b) Valid bytes of the AD* lines during reply phase of read transfers when SC4* is asserted and SC<7..5>* not asserted.
c) Valid bytes of the AD* lines during reply phase of write transfers when SC3* is asserted.

6.4.1.2.2 Minimum parity checking requirements

The minimum parity checking requirements are as follows:

a) SC* lines at all times.
b) AD* lines by all replying agents supporting the requested address space during request phase.
c) Valid bytes of the AD* lines by bus owner during reply phase of its read data transfers.
d) Valid bytes of the AD* lines by replying agent during reply phase of its write data transfers.

6.4.1.2.3 Protocol violations

Protocol violations that may result in the assertion of BUSERR* by an agent at the falling edge of BCLK* and can result in a bus error exception are as follows:

a) SC0* is asserted while a transfer operation is in progress.
b) SC2* is asserted while SC3* is negated during a reply phase.
c) SC5*, SC6*, or SC7* are asserted while SC4* is negated during a reply phase.
d) After being asserted during a reply phase, SC3* is negated before a handshake occurs.
e) During a reply phase, SC2* changes state before a handshake occurs while SC3* is asserted.
f) After being asserted during a reply phase, SC4* is negated before a handshake occurs.
g) During a reply phase, SC5*, SC6*, or SC7* changes state before a handshake occurs while SC4* is asserted.

Detection occurs during the bus clock cycle that the error is present on the bus and is signalled on the bus clock cycle after detection occurs. Agents may hold the BUSERR* signal asserted to complete an internal operation or perform any recovery.

Figure 6.4-3 shows the timing sequence for two cases of detection and signalling of a bus error exception. The first BUSERR* exception occurs after the request phase and terminates the transfer operation during the request phase. The second terminates the transfer operation during the reply phase. An agent asserts BUSERR* one bus clock cycle after it detects the exception. One bus clock cycle later, the exception operation terminates both transfer operations and arbitration operations.

6.4.2 Priority of errors

If multiple errors occur, they are reported as follows:

a) The first error detected is reported and subsequent errors are ignored except in the following cases:

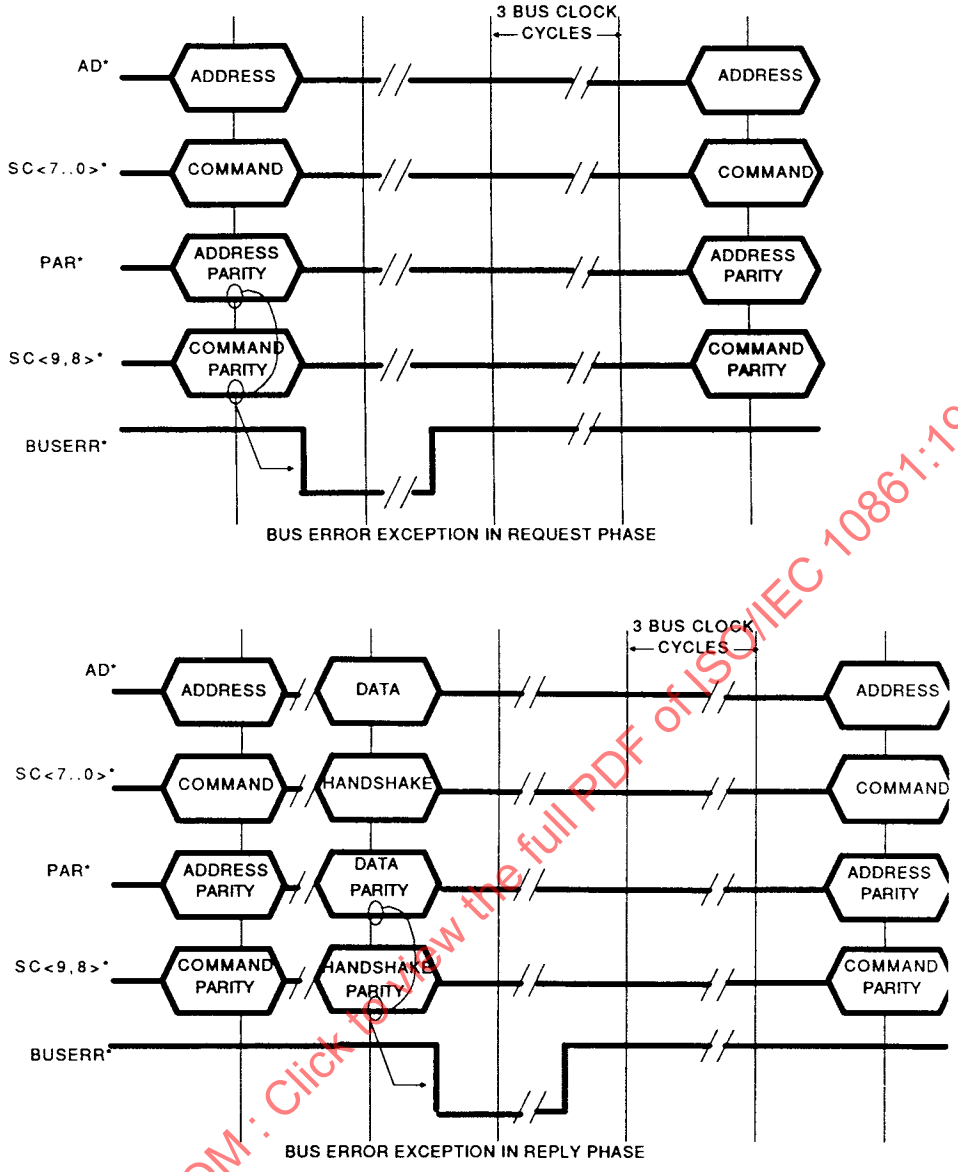


Figure 6.4-3—Signalling a BUSERR* exception

1) BUSERR* is monitored for one bus clock cycle after the detection of an agent error. If a bus error is detected during that bus clock cycle, it is reported and the agent error is considered to be invalid.
2) Whenever an agent error and a bus error are reported at the same time, in light of 1) above, the agent error is considered to be invalid.

b) When detected at the same time, exceptions override agent errors. When TIMOUT* and BUSERR* are reported at the same time, both are considered valid.

6.5 Central control functions

All central control functions for the PSB are performed by the CSM. The CSM must be installed in cardslot 0 in the PSB. The CSM provides certain system-level services and functions common to all bus

agents. The system services provided by the CSM ensure uniform system operation; that is, the CSM provides the coordination required among the three types of bus operation. The CSM performs the following system functions:

a) It provides a central source for BCLK* and CCLK*.
b) It generates RST*, DCLOW*, and PROT* at power-up and during power-fail.
c) It assigns arbitration and cardslot IDs to each agent during reset of the system. The CSM provides a cardslot ID number for each agent in the system based on the geographical location within the backplane. The CSM assigns a separate and unique arbitration ID to the other 20 agents on the PSB. This initialization occurs during every reset operation, when DCLOW* and PROT* are inactive while RST* is active. At least eight bus clock cycles must occur, however, before the CSM sends the information onto the bus. This allows agents time to prepare for the operation.
d) It monitors for time-out exceptions and asserts the TIMOUT* exception signal when detected.

6.5.1 Clock source

The CSM generates the two clock signals for the PSB: bus clock (BCLK*) and control clock (CCLK*). Timing relationships exist between the two clock signals.

The CSM is installed in cardslot 0. The clock signals can only be driven from cardslot 0. Some systems may employ more than one CSM circuit board, but only the CSM installed in cardslot 0 is permitted to drive the clock signals for the bus.

At power-up, each CSM circuit board present in the system may determine whether it is located in cardslot 0 by sampling the voltage level present at pin 4A of the P1 connector. For a given CSM circuit board, if pin 4A is at a high voltage level, the cardslot is slot 0 and the CSM is required to drive the bus clock signals (BCLK* and CCLK*). If 4A is at a low voltage level, the cardslot is not slot 0, and the CSM must not drive the bus clock signals. Refer to 7.5.1 for information about the P1 connector pinout.

6.5.2 System level resets

The PSB provides three system-level functions via the central control signal group. These functions include power-up sequence control (cold-start), initialization sequence control (warm-start), and power-fail-recovery control. Each is described in the following subclauses. All timing specifications for system level resets are supplied in clause 7.

6.5.2.1 Power-up function (cold-start)

The power-up reset, sometimes referred to as a "cold-start," initializes all agents in the system. The sequence of events for a cold-start is shown in figure 6.5-1. The cold-start provides two benefits in the system: It ensures a uniform initialization period for all agents on application of power, and it gives all agents in the system the opportunity to begin operation from a known state.

The coincident assertion of DCLOW* and RST* along with the negation of PROT* indicates to all agents that a bus-wide power-on reset is in progress. Each agent must then execute its local power-on reset sequence. The power-on system reset sequence is performed automatically at power-up and may also be performed during system operation.

At power-up, the CSM asserts DCLOW* and RST*. The CSM holds DCLOW* active for a minimum duration to guarantee detection by all agents. The CSM holds RST* active for minimum specified time after deactivating the DCLOW* signal. If one or more agents require additional time to complete initialization operations, those agents may assert rstnc* to extend the duration of the initialization period. RSTNC* inhibits bus operations until all agents have completed initialization and are ready to proceed.

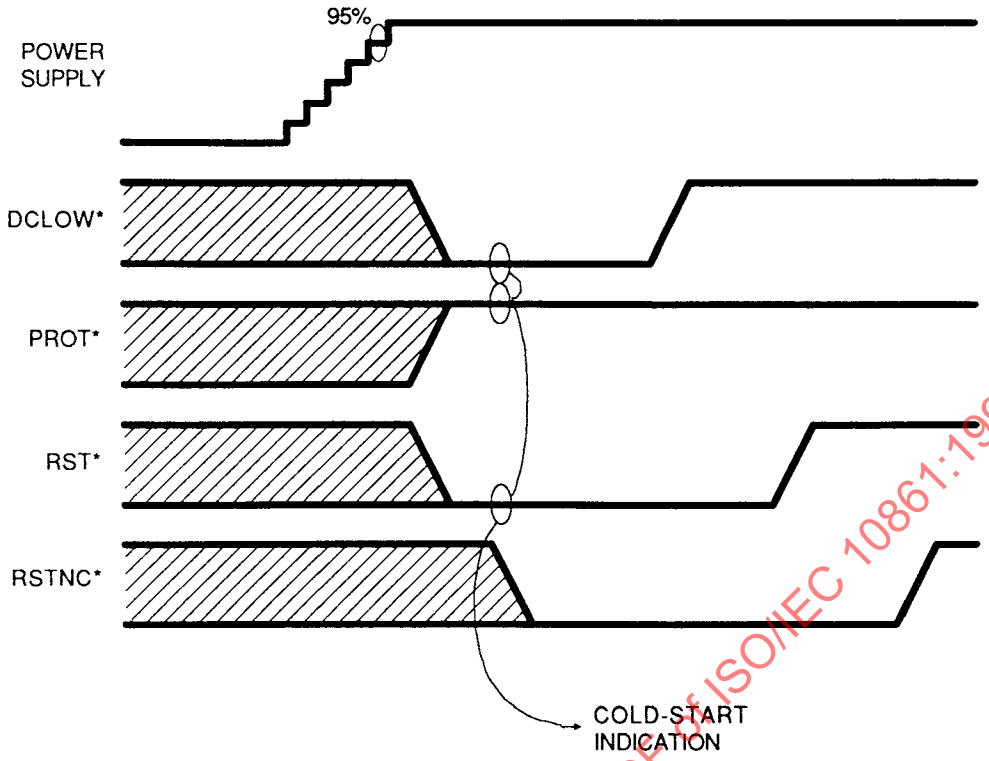


Figure 6.5-1—Power-on system reset sequence

DCLOW* and PROT* are asserted by the CSM in cardslot 0 and are asynchronous to the bus clock. Both reset signals (RST* and RSTNC*), however, are synchronous to the bus clock.

6.5.2.2 Initialization sequence (warm-start)

The sequence of events for resetting a running system, sometimes referred to as a "warm-start," is shown in figure 6.5-2. This type of reset is typically the result of pushing a front panel switch. The figure does not show bus clock cycles because of the extended time-span of the sequence. However, both reset signals (RST* and RSTNC*) are synchronous to the bus clock.

The CSM initiates a warm-start sequence by activating RST* for a minimum specified duration. Agents may extend the initialization period by asserting rstnc*. As with the cold-start sequence, RSTNC* prevents all agents from starting bus operations until all agents are reset.

Unlike the cold-start, this "warm-start" reset sequence does not assert DCLOW*. Agents that must differentiate between the two types of reset sequence may do so by examining the condition of the DCLOW* line on the PSB while RST* is active.

6.5.2.3 Power failure and recovery sequence

The CSM uses the power-fail and recovery sequence to provide orderly control of system shut-down and start-up during a power failure. The system power supply is expected to provide a power-fail signal to the CSM if the CSM is to provide power-fail and recovery sequences. When the ac input power drops below an acceptable value, the power supply asserts signals that inform the CSM of the power failure condition. Figure 6.5-3 shows a power failure and recovery sequence on the PSB.

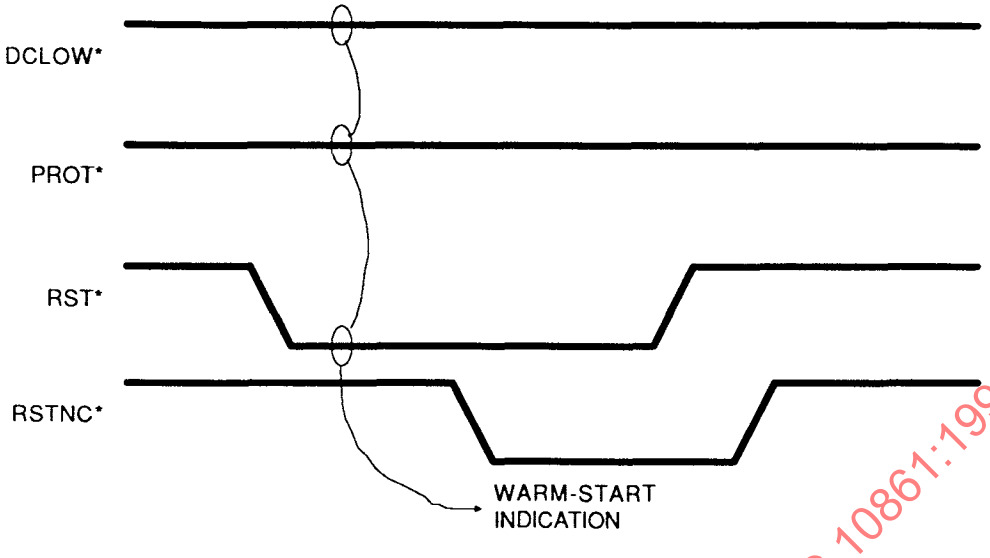


Figure 6.5-2—Warm-start reset sequence

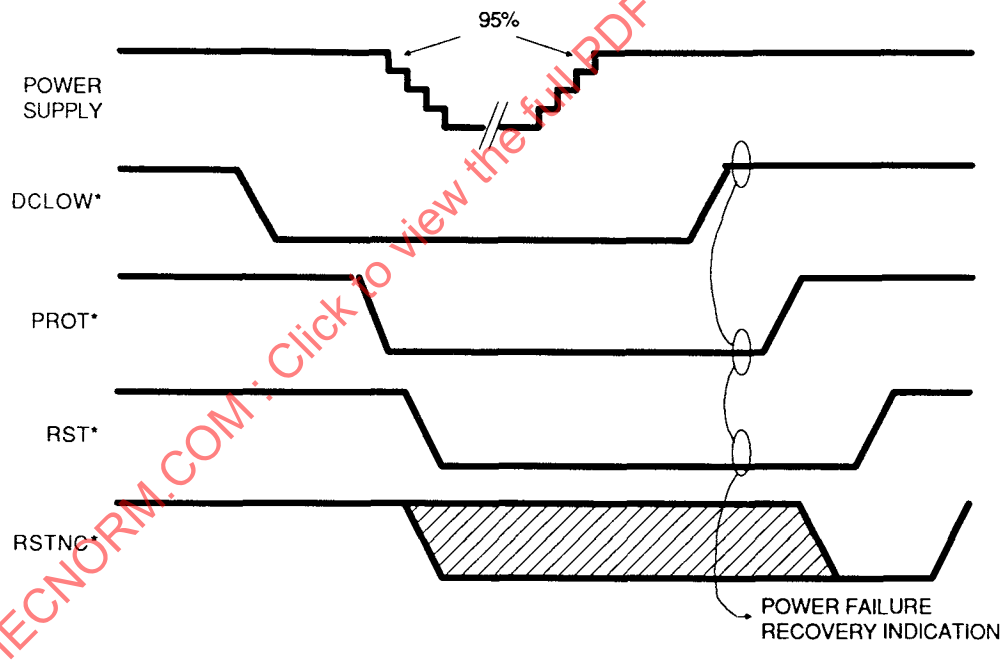


Figure 6.5-3—Power failure recovery sequence

This sequence assumes that the system supports battery back-up and allows a reasonable period for the back-up and recovery sequences to execute.

On learning of an imminent power failure (via an early warning signal from the power supply), the CSM asserts DCLOW* onto the PSB. The assertion of DCLOW* informs all agents of the impending power failure. Safe power is guaranteed for a minimum duration from the assertion of DCLOW*.

The power-down sequence allows this specified minimum time for the software to save system status. The CSM then asserts the protect signal (PROT*) to prevent any further bus activity. Additional time is provided from the assertion PROT* for the hardware to configure itself for power-down.

At this point, the power may drop out completely or return to normal operating voltages. Note that the logic that drives the PROT*, DCLOW* and RST* signals from the CSM must be powered from a battery back-up source if the system supports power-fail recovery.

In a case where transient faults cause a power failure, the power supply may recover during the shutdown sequence. The CSM guarantees a DCLOW* active period so that the sequence goes to completion before recovery is allowed to occur.

During the power-down period, the CSM is not required to drive the BCLK* and CCLK* signals. Typically the RST* and RSTNC* signals are not connected to battery back-up power. The CSM must continue to assert PROT* and DCLOW* during the power-down period. Both PROT* and DCLOW* must be driven from devices powered by the battery back-up power source.

After power returns to its operating level, the CSM asserts RST*. After the power has been maintained at the operating level for a specified time, the CSM negates DCLOW*. The CSM then negates PROT*, after allowing a minimum time from the DCLOW* transition.

An agent determines whether a power failure occurred by examining the PROT*, DCLOW*, and RST* signals. A specific condition of the signals (DCLOW* inactive, RST* active, and PROT* active) indicates that a power failure has occurred.

Both DCLOW* and PROT* are asynchronous to the bus clock.

A power-off sequence is defined for these systems that do not support battery back-up. This power-off sequence is initiated when all supply voltages, including the +5 battery, fail. For systems that do not support battery back-up, DCLOW* must be asserted prior to the loss of safe voltage levels. PROT* and RST* are held inactive during this period. The CSM may optionally attempt the power-fail sequence until the power actually fails.

Power failure during a cold-start or during a warm-start causes the CSM to not regulate the power shutdown. Either condition should be followed by a cold-start sequence. If a warm-start occurs during a power shutdown, the CSM has the option of either executing the power shutdown and ignoring the reset command, or aborting the shutdown and initiating cold-start on power recovery.

6.5.3 Arbitration ID and cardslot ID assignment

During reset, the CSM in cardslot 0 assigns IDs (1 through 20) and arbitration IDs to each agent on the bus. The sequence involves the use of the arbitration ID lines (ARB*), the address/data bus lines (AD<20..1>*), the ID latch signal (LACHn*), and the reset signal (RST*). When ARB5* is asserted, the CSM places cardslot IDs on ARB<4..0>*; when ARB5* is negated, the CSM places arbitration IDs on ARB<4..0>*. Only the CSM is allowed to drive the ARB* and AD* lines while RST* is asserted.

The default assignment of arbitration IDs and cardslot IDs is listed in table 6.5-1.

Figure 6.5-4 shows the timing sequence for a CSM assigning IDs to each agent and cardslot in the system. When RST* and ARB5* are asserted, the CSM is assigning a cardslot ID to an agent on the bus. When RST* is asserted and ARB5* is negated, the CSM assigns an arbitration ID to an agent on the bus. The CSM must wait at least eight bus clock cycles from its asserting of RST* to the assignment of any arbitration or cardslot ID.

Table 6.5-1—Cardslot and default arbitration ID assignment

Cardslot ID	Cardslot ID ARB<4..0>*	Arbitration ID	Arbitration ID ARB<4..0>*	AD(n)*
0	HHHHH	31	LLLLL	—
1	HHHHL	30	LLLLH	1
2	HHHLH	29	LLLHL	2
3	HHHLL	28	LLLHH	3
4	HHLHH	27	LLHLL	4
5	HHLHL	25	LLHHL	5
6	HHLLH	24	LLHHH	6
7	HHLLL	23	LHLLL	7
8	HLHHH	19	LHHLL	8
9	HLHHL	17	LHHHL	9
10	HLHLH	16	LHHHH	10
11	HLHLL	15	HLLLL	11
12	HLLHH	14	HLLLH	12
13	HLLHL	12	HLLHH	13
14	HLLLH	08	HLHHH	14
15	HLLLL	07	HHLLL	15
16	LHHHH	06	HHLLH	16
17	LHHHL	04	HHLHH	17
18	LHHLH	03	HHHLL	18
19	LHHLL	02	HHHLH	19
20	LHLHH	01	HHHHL	20

NOTES
a) ARB5* asserted for slot ID initialization, deasserted for arbitration ID assignment.
b) All legal arbitration ID numbers are shown, all others are illegal. The arbitration ID is not bound to the cardslot. If the arbitration ID numbers are changed, only the values shown in the table may be used.
c) The CSM must initialize its own ID differently; refer to 6.5.1 for information about CSM determination.
d) An agent recognizes an interconnect access when the value on AD<15..11>* equals that assigned on ARB<4..0>* during cardslot ID initialization.
e) The cardslot ID must be assigned exactly as shown.
f) An agent recognizes its own ID during arbitration when the value of ARB<4..0>* equals the value of ARB<4..0>* assigned during arbitration ID initialization.

With each arbitration/cardslot ID, the CSM drives a corresponding address/data line (connected to the LACHn* signal for each cardslot) to identify the destination for the ID. The CSM asserts one and only one of the AD* lines at any time during the assignment sequence. The other AD* lines may be negated or not driven by the CSM. Each agent uses its LACHn* line (connected to one of the address/data lines on the backplane) to identify the ID from the ARB<4..0>* lines and latch it.

Figure 6.5-5 details the relationship among RST*, ARB*, and LACHn*. The LACHn* signal must stay asserted for at least one bus clock cycle. The arbitration or slot ID is guaranteed to be stable for at least one clock cycle before and after the LACHn* signal is active.

The order of assignment is implementation dependent and therefore not specified in this International Standard.

6.5.4 Time-out monitoring

The CSM initiates a time-out exception operation by asserting TIMOUT*. Transfer operations are monitored by the CSM, and an exception operation is initiated whenever the specified maximum time limit is exceeded. The maximum time limit is from 10 000 to 12 500 bus clock cycles. The timing of transfer operations begins whenever one of the following conditions is met at the falling edge of BCLK*:

a) SC0* is asserted.
b) SC3* and SC4* are asserted, and SC2* is not asserted.

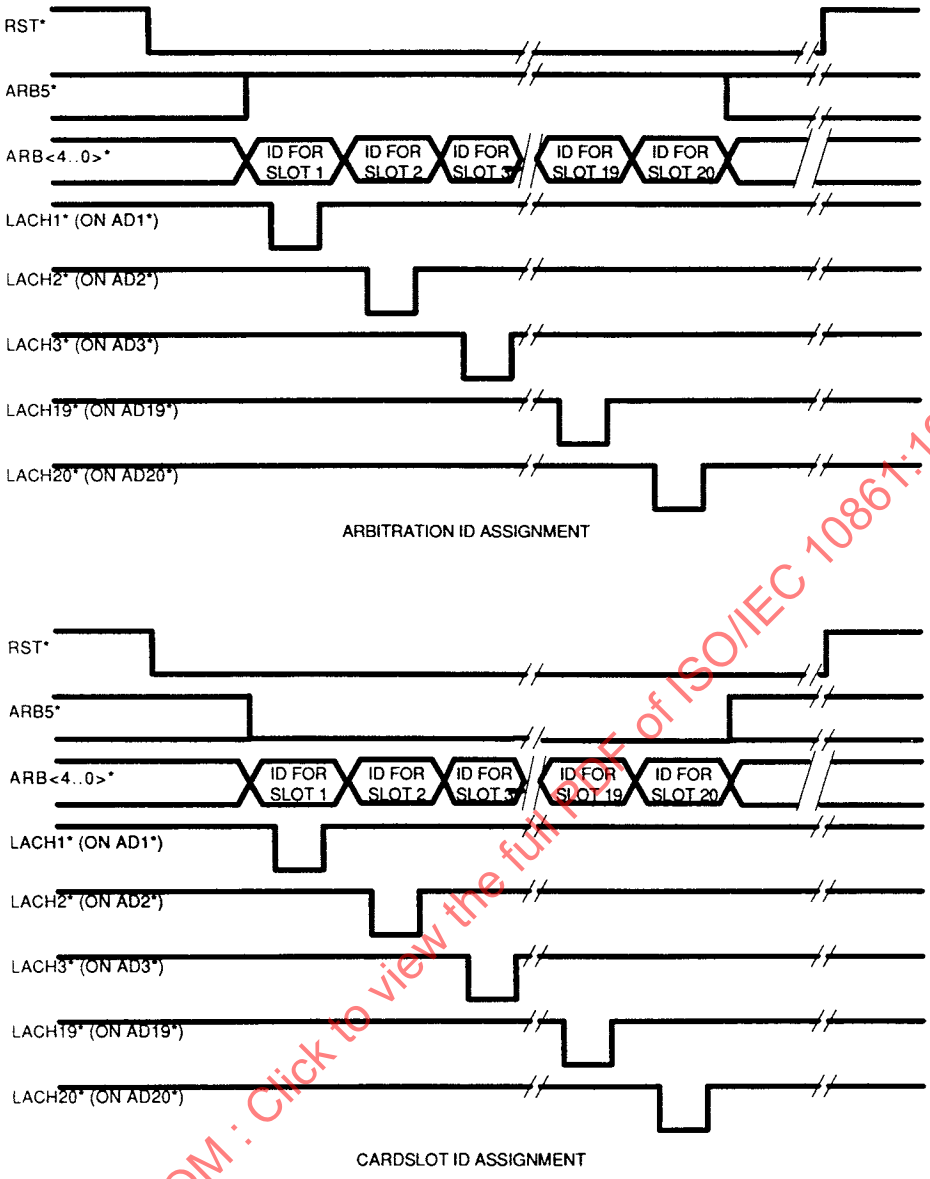


Figure 6.5-4 — Example arbitration/cardslot ID assignment timing sequence

The CSM may also assert the TIMOUT* signal in response to a bus owner failing to release ownership. If a request from another agent(s) is asserted on the bus, and the current bus owner does not release the bus in a specified amount of time, the CSM may assert the TIMOUT* signal on the bus. The time limit for ownership is implementation dependent and must be specified by the CSM documentation.

6.6 State-flow diagrams

This discussion of the PSB uses state-flow diagrams to describe the operation of agents on the bus. The state-flow diagrams separate the operation of an agent into several steps. The state-flow diagrams for requesting and replying agents are as follows:

a) State-flow for requesting agents monitoring transfer operations
b) State-flow for requesting agents monitoring artibration operations

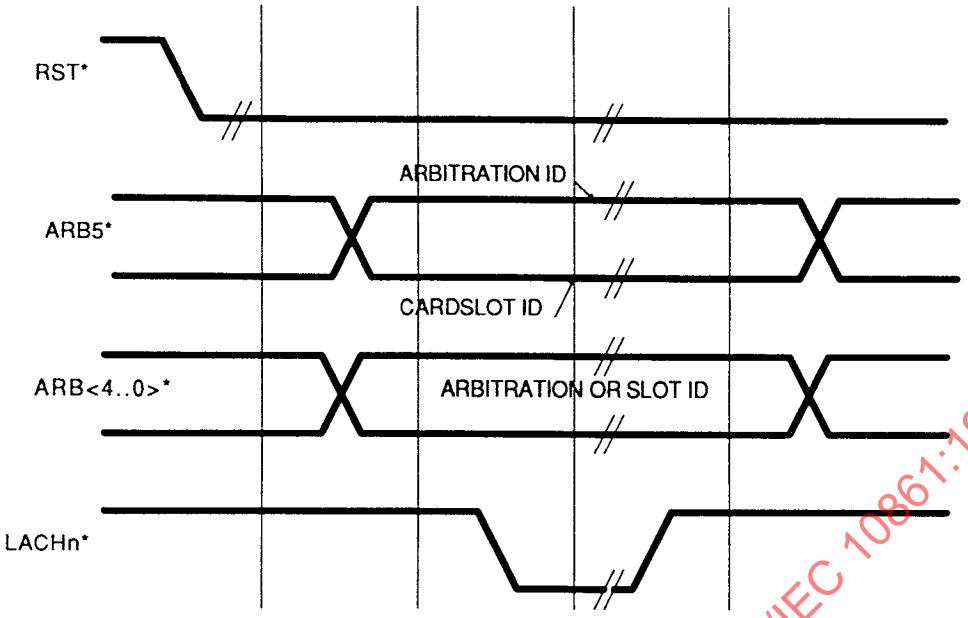


Figure 6.5-5—Reset sequence

c) State-flow for requesting agents in an arbitration operation
d) State-flow for bus owners in a transfer operation
e) State-flow for replying agents in a transfer operation

The exception operation does not have a separate state-flow diagram; however, it causes transitions in each of the five state-flows.

The initial transition from the CLEAR condition overrides all other conditions and transitions. If none of the conditions required for a transition from a given state occur then the agent remains in that state.

6.6.1 State-flow sequence for an agent monitoring the bus

Figure 6.6-1 represents the state machines for agents monitoring the bus. All active agents must continually monitor the bus to determine if a transfer operation is in progress.

The following is an explanation of each condition and its various components.

EXCEPTION: BUSERR*=L OR TIMOUT*=L
This condition occurs whenever the BUSERR* or TIMOUT* line is asserted. See 6.4 for details on exception operations.

CLEAR: RST*=L OR RSTNC*=L OR **EXCEPTION**
This condition initializes the state machine and has the effect of synchronizing all agents on the bus.

EOT HANDSHAKE: SC2*=L AND SC4*=L
This condition indicates that the transfer operation in progress is completing. This condition is meaningful only if there is a transfer operation in progress (agent is in the TRANSFER OPERATION state of the Monitor State Machine, shown in figure 6.6-1). Note that the bus owner state machine guarantees that SC3*=L whenever SC2*=L, allowing SC3* to be disregarded for this condition.

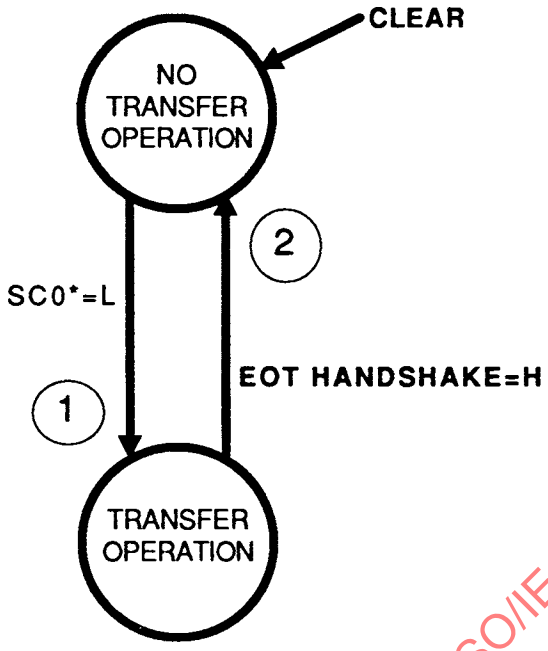


Figure 6.6-1—State-flow diagram for monitoring transfer operations

Below is a description of each state and the transitions that are possible. Outputs are a description of agent activity on the bus while the agent is in the state being described.

INITIAL TRANSITION
Transitions: The **CLEAR** condition aborts any operation currently in progress.

NO TRANSFER OPERATION
Description: The bus is idle and requesting agents are monitoring SC0*.
Outputs: None; all agents monitoring.
Transitions: (1) The state machines of all agents enter the TRANSFER OPERATION state on an agent asserting SC0* (request phase). SC0* = L indicates the start of a transfer operation, therefore all state machines make the transition to the TRANSFER OPERATION state to track the transfer operation.

TRANSFER OPERATION
Description: Bus is busy with a transfer operation. Agents are monitoring for end-of-transfer indication.
Outputs: None.
Transitions: (2) The state machines of all agents enter the NO TRANSFER OPERATION state on **EOT HANDSHAKE** condition since the **EOT HANDSHAKE** condition signals the end of the transfer.

6.6.2 State-flow sequence for an arbitration operation

Figure 6.6-2 represents the state-flow for agents monitoring arbitration operations and figure 6.6-3 the state-flow for requesting agents in an arbitration operation. Only requesting agents will implement the state flow since only they may become the bus owner.

Figure 6.6-2 represents the state-flow that agents use to monitor the resolution phase of an arbitration operation. Since three bus clock cycles are required to resolve arbitration, this state machine monitors the

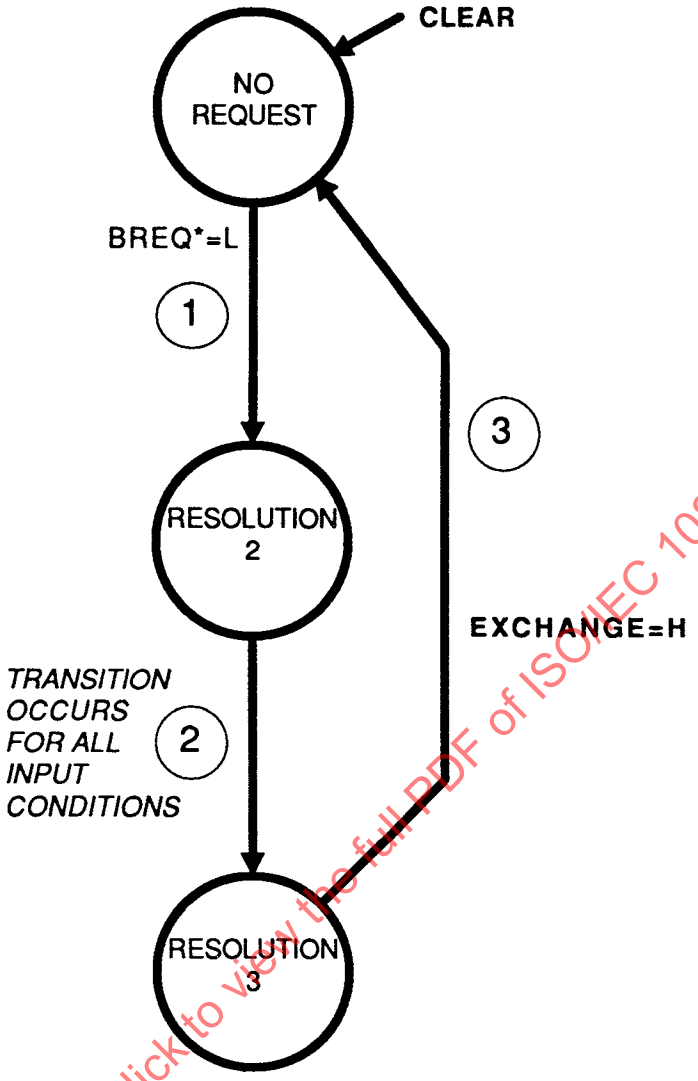


Figure 6.6-2—State-flow diagram for agents monitoring arbitration operations

bus and counts the bus clock cycles in the resolution phase for the agent. The following explains conditions that are relevant to the monitoring process.

EXCEPTION: BUSERR*=L OR TIMOUT*=L
This condition occurs whenever the BUSERR* or TIMOUT* line is asserted. See 6.4 for details on exception operations.

CLEAR: RST*=L OR RSTNC*=L OR **EXCEPTION**
This condition initializes the state machine and has the effect of synchronizing all agents on the bus.

EOT HANDSHAKE: SC2*=L AND SC4*=L
This condition indicates that the transfer operation in progress is completing. This condition is meaningful only if there is a transfer operation in progress (agent is in the TRANSFER OPERATION state of the Monitor State Machine, shown in figure 6.6-1). Note that the owner state machine guarantees that SC3*=L whenever SC2*=L, allowing SC3* to be disregarded for this condition.

EXCHANGE: RESOLUTION-3 state AND SC1*=H AND ((NO TRANSFER OPERATION state AND SC0*=H) OR (TRANSFER OPERATION state AND **EOT HANDSHAKE**))
This condition indicates that exchange of the bus ownership is possible. RESOLUTION-3 state is the final state of a resolution phase. SC1*=H means that the bus is not locked. NO TRANSFER OPERATION state and SC0*=H means there is no transfer operation in progress and none starting. TRANSFER OPERATION state and **EOT HANDSHAKE** means that the current transfer operation is completing.

INITIAL TRANSITION

Transitions: The CLEAR condition aborts any operation currently in progress.

NO REQUEST state

Description: In this state either no resolution is in progress or it is the first state of the resolution.
Outputs: None.
Transitions: (1) An agent has asserted breq* and begun arbitration. Therefore the state machine makes the transition because the previous bus clock cycle was the first bus clock cycle of the resolution phase.

RESOLUTION-2

Description: This state represents the second bus clock cycle of the resolution phase; ARB* lines are settling.
Outputs: None.
Transitions: (2) For all input conditions, the state machines of all agents progress to RESOLUTION-3 state.

RESOLUTION-3

Description: At the falling edge of BCLK* in this state, the arbitration lines have settled with the ID of the highest priority requesting agent. All state machines remain in this state until an exchange of bus ownership is possible.
Outputs: None.
Transitions: (3) When the **EXCHANGE** condition is met, bus ownership exchanges and the resolution phase is ready to begin again; therefore, the state machine returns to the NO REQUEST state.

The state-flow for agents in an arbitration operation is implemented to follow activities related to bus ownership. The following is an explanation of the conditions and their components that are used for this purpose. Figure 6.6-3 shows the diagram for the state-flow.

EXCEPTION: BUSERR*=L OR TIMOUT*=L
This condition occurs whenever the BUSERR* or TIMOUT* line is asserted. See 6.4 for details on exception operations.

CLEAR: RST*=L OR RSTNC*=L OR **EXCEPTION**
This condition initializes the state machine and has the effect of synchronizing all agents on the bus.

EOT HANDSHAKE: SC2*=L AND SC4*=L
This condition indicates that the transfer operation in progress is completing. This condition is meaningful only if there is a transfer operation in progress (agent is in the TRANSFER OPERATION state of the Monitor State Machine, shown in figure 6.6-1). Note that the owner state machine guarantees that SC3*=L whenever SC2*=L, allowing SC3* to be disregarded for this condition.

EXCHANGE: RESOLUTION-3 state AND SC1*=H AND ((NO TRANSFER OPERATION state and SC0*=H) OR (TRANSFER OPERATION state AND **EOT HANDSHAKE**))
This condition indicates that exchange of the bus ownership is possible. RESOLUTION-3 state is the final state of a resolution phase. SC1*=H means that the bus is not locked. NO TRANSFER OPERATION state and SC0*=H

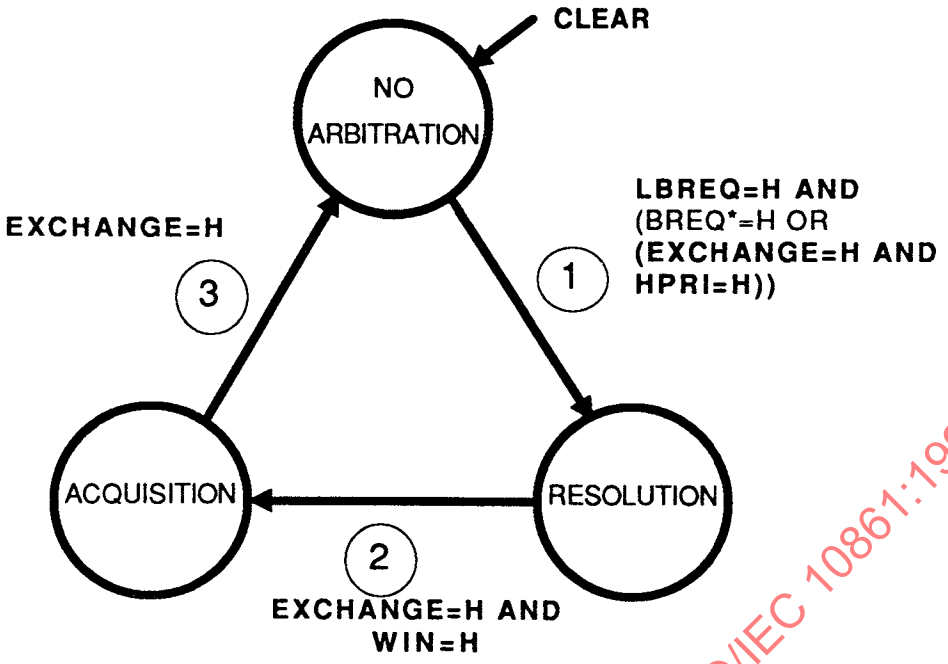


Figure 6.6-3—State-flow diagram for arbitration operations

means there is not a transfer operation in progress and none starting. TRANSFER OPERATION state and **EOT HANDSHAKE** means that the current transfer operation is completing.

WIN: ARB* signals match the agent's arbitration ID and requesting priority
This condition occurs when, after entering arbitration resolution, the agent has the highest priority.

LBREQ: This condition indicates that the agent desires to use the bus.

HPRI: This condition indicates that the agent desires to make a high priority request of the bus.

The following are descriptions of each state and the transitions that are possible. Outputs are a description of agent activity on the bus during the time that the agent is in the state being described.

INITIAL TRANSITION
Transitions: The CLEAR condition aborts any operation currently in progress.

NO ARBITRATION state
Description: The agent does not require the bus.
Outputs: None; arbitration not required.
Transitions: (1) State machines make the transition to the RESOLUTION state. **LBREQ**=H indicates that the agent desires access to the bus. The agent enters an arbitration operation in one of two ways depending on **HPRI**. If the agent has a high priority request, it may enter the arbitration operation at the start of the next resolution phase (**EXCHANGE**=H). If there is no bus request sequence (BREQ*=H), any requesting agent may enter the resolution phase.

RESOLUTION state
Description: The agent is involved in resolving priority with other requesting agents as shown in figure 6.6-4. The state machine will remain in the RESOLUTION state until it obtains ownership of the bus.
Outputs: The agent asserts breq* and places its arbitration ID onto the arb* lines.

Transitions: (2) On both the **EXCHANGE** and **WIN** being met, the state machine of the agent moves to ACQUISITION state. If the agent loses in the resolution phase (**WIN**=L) or the bus cannot be exchanged (**EXCHANGE**=L), the state machine remains in RESOLUTION state to resolve priority for the next bus exchange.

ACQUISITION state

Description: Agent is the bus owner and performs transfer operations.

Outputs: Agent can perform transfer operations. The agent negates breq* and arb*.

Transitions: (3) With the **EXCHANGE**=H, the state machine enters the NO ARBITRATION state. Since the **EXCHANGE** condition can be met only if the transfer operation is complete, the agent cannot lose ownership during a transfer operation. Please note that the standard ensures that, after entering the OWNERSHIP state, at least three bus clock cycles will occur before the agent can lose ownership of the bus. If the agent has not started a transfer by then he may lose the bus before starting the transfer. Further, if no other agents are requesting the bus (BREQ*=H) the RESOLUTION-3 state cannot be entered and, therefore, the **EXCHANGE** condition will not be met. If the **EXCHANGE** condition is not met the current owner will remain the bus owner (referred to as parked).

6.6.3 State-flow sequence for bus owners in a transfer operation

Figure 6.6-4 is the state-flow diagram for a bus owner performing a transfer operation. It represents the conditions of both the agent performing the transfer and on the bus. The following explains the various conditions.

EXCEPTION: BUSERR*=L OR TIMOUT*=L
This condition occurs whenever the BUSERR* or TIMOUT* line is asserted. See 6.4 for details on exception operations.

CLEAR: RST*=L OR RSTNC*=L OR **EXCEPTION**
This condition initializes the state machine and has the effect of synchronizing all agents on the bus.

ADDRDY: An internal condition that indicates that the request information can be driven valid in the next bus clock cycle. Some implementations require time to drive the request phase information.

OWNER NEXT CLK: (**EXCHANGE**=L AND ACQUISITION state) OR (**EXCHANGE**=H AND **WIN**=H AND RESOLUTION state)
This condition indicates that the agent will be the bus owner in the next state. ACQUISITION state and **EXCHANGE**=L means that the bus is not being exchanged; therefore, the current owner will also be the next owner. **EXCHANGE**=H and **WIN**=H means that the agent will become the owner in the next state.

EOT HANDSHAKE: SC2*=L AND SC4*=L
This condition indicates that the transfer operation in progress is completing. This condition is meaningful only if there is a transfer operation in progress (agent is in the TRANSFER OPERATION state of the Monitor State Machine, shown in figure 6.6-1). Note that the owner state machine guarantees that SC3*=L whenever SC2*=L, allowing SC3* to be disregarded for this condition.

OWNRDY: Internal data valid timing for read and write data transfers on the bus. In the case of a read data transfer, it means the bus owner will be able to finish taking in the data in the next bus clock cycle. For read data transfers, **OWNRDY** may depend on SC4*=L (i.e., that the owner can wait until the replying agent has provided the data and had some time to take it in before signalling the handshake). In the case of write data transfers, it indicates that the data will be driven on the bus in the next state. For write data transfers, this cannot depend

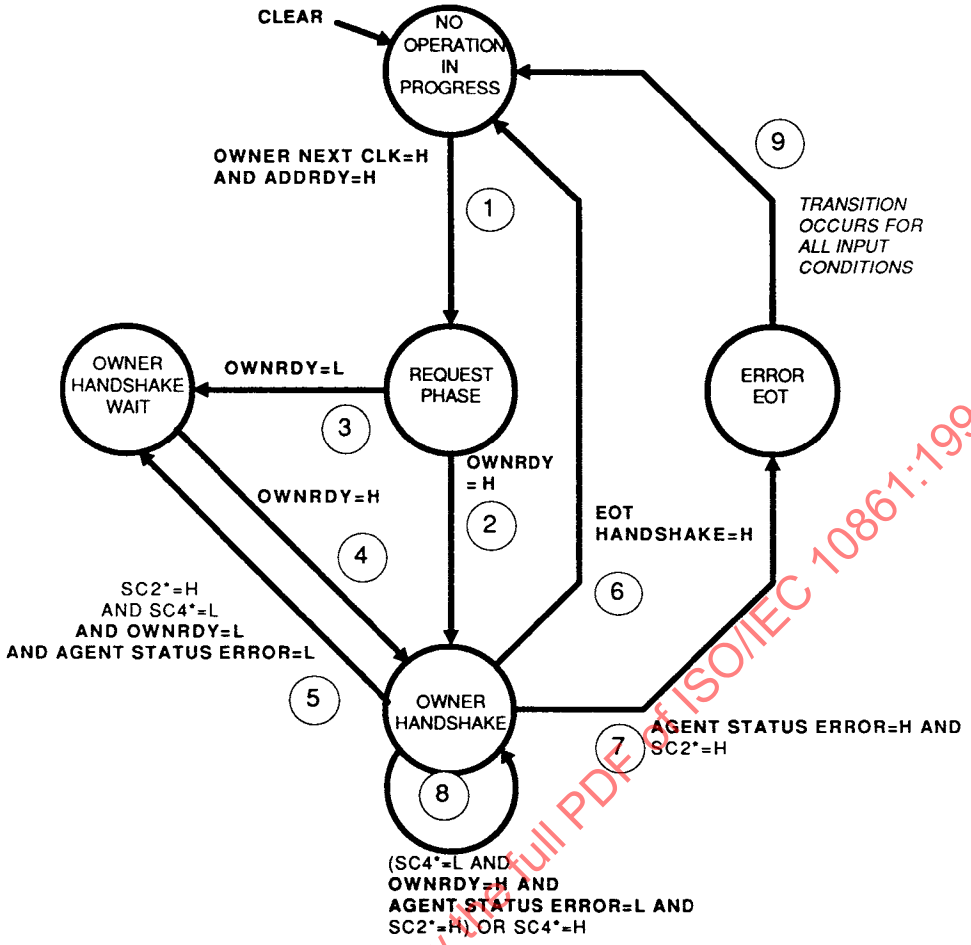


Figure 6.6-4—State-flow diagram for requesting agents in a transfer operation

on SC4*=L. The owner must supply data without waiting for the replying agent to be ready.

EXCHANGE: Resolution-3 state AND SC1*=H AND ((NO TRANSFER OPERATION state AND SC0*=H) OR (TRANSFER OPERATION state AND **EOT HANDSHAKE**))

This condition indicates that exchange of the bus ownership is possible. RESOLUTION-3 state is the final state of a resolution phase. SC1*=H means that the bus is not locked. NO TRANSFER OPERATION state and SC0*=H means there is no transfer operation in progress and none starting. TRANSFER OPERATION state and **EOT HANDSHAKE** mean that the current transfer operation is completing.

WIN: ARB* signals match the agent's arbitration ID and priority.

This condition occurs when, after entering arbitration resolution, the agent has the highest priority.

AGENT STATUS ERROR:

SC5*=L OR SC6*=L OR SC7*=L

The replying agent is signalling an agent status error. This condition is meaningful only when there is a transfer operation in progress (agent is in the TRANSFER operation state of the Monitor State Machine, shown in figure

6.6-1). Note that the replying agent state machine guarantees that SC4* = L whenever SC5* = L, SC6* = L, or SC7* = L; allowing SC4* to be disregarded for this condition.

Below is a listing of the states and the transitions. The numbers for each transition correspond to the number of the transition in the diagram.

INITIAL TRANSITION

Transitions: The **CLEAR** condition aborts any operation currently in progress.

NO OPERATION IN PROGRESS state

Description: Agent may be participating in the resolution phase of arbitration and is not performing a transfer operation.

Outputs: None.

Transitions: (1) On **OWNER NEXT CLOCK** and **ADDRDY** conditions, the state machine progresses to REQUEST PHASE state.

REQUEST PHASE state

Description: Agent performs request phase of transfer operation.

Outputs: Agent places address information on bus lines AD* and control information on SC<7..1>*, SC0* = L (Request Phase), SC8* (even parity on SC<7..4>*), SC9* (even parity on SC<3..0>*).

Transitions: (2) If **OWNRDY** = H, the state machine moves to the OWNER HANDSHAKE state or,
(3) If **OWNRDY** = L, the state machine moves to the OWNER HANDSHAKE WAIT state.

OWNER HANDSHAKE WAIT state

Description: Bus owner is not ready to perform the bus operation or the handshake.

Outputs: SC0* = H (reply phase), SC1* (lock), SC2* = H (not end-of-transfer), SC3* = H (bus owner not ready), SC9* (even parity on SC<3..0>*).

Transitions: (4) On **OWNRDY** = H, the state machine moves to the OWNER HANDSHAKE state.
Other: Some implementations may be able to preempt bus operations in progress. Such implementations may make a transition from the OWNER HANDSHAKE WAIT state to the ERROR EOT state on detecting an agent status error condition.

OWNER HANDSHAKE state

Description: The bus owner is ready to perform the data transfer and handshake.

Outputs: SC0* = H (reply phase), SC1* (lock), SC2* (end-of-transfer), SC3* = L (bus owner ready). SC2* may be asserted only in this state or ERROR EOT state. This state is entered once per data transfer period. On entering this state, the bus owner sets SC2* at the proper level indicating whether this is the last data transfer or not. If the transfer operation is a write data transfer, the data must be valid the entire time the bus owner is in this state.

Transitions: (5) With SC4* = L (replying agent ready), SC2* = H (not end of operation), **OWNRDY** = L (owner not ready), and **AGENT STATUS ERROR** = L, the state machine moves to the OWNER HANDSHAKE WAIT state. This means that the current data transfer has finished, that this is not the last transfer, and that no error occurred on the current data transfer.
(6) Or, with **EOT HANDSHAKE** = H, the agent completes the operation and its state machine returns to the NO OPERATION IN PROGRESS state.
(7) Or, with SC2* = H (not end-of-transfer) and **AGENT STATUS ERROR** = H, the state machine moves to the ERROR EOT state. This represents terminating a block transfer if an agent status error occurs.
(8) Transitions to the same state are ordinarily not shown; however, this transition represents handshaking on the previous data transfer and being immediately ready for the next one.

Other: Some implementations may be able to do back-to-back transfer operations. In that case, the state machine of the bus owner may make a transition from the OWNER HANDSHAKE state to the REQUEST PHASE state.

ERROR EOT state

Description: Agent has sensed an agent status error or data width error from the replying agent.

Outputs: SC0*=H (reply phase), SC1* (lock), SC2*=L (end of operation for current operation), SC3*=L (bus owner ready), and SC9* (even parity on SC<3..0>*).

Transitions: (9) The state machine returns to NO OPERATION IN PROGRESS state.
Other: Some implementations may be able to do back-to-back transfer operations. In that case, the state machine of the bus owner may make a transition from the ERROR EOT state to the REQUEST PHASE state.

6.6.4 State-flow diagram for replying agents in a transfer operation

Figure 6.6-5 represents the state-flow for replying agents in a transfer operation. The following explains the conditions associated with this state-flow.

EXCEPTION: BUSERR*=L OR TIMOUT*=L
This condition occurs whenever the BUSERR* or TIMOUT* line is asserted. See 6.4 for details on exception operations.

CLEAR: RST*=L OR RSTNC*=L OR **EXCEPTION**
This condition initializes the state machine and has the effect of synchronizing all agents on the bus.

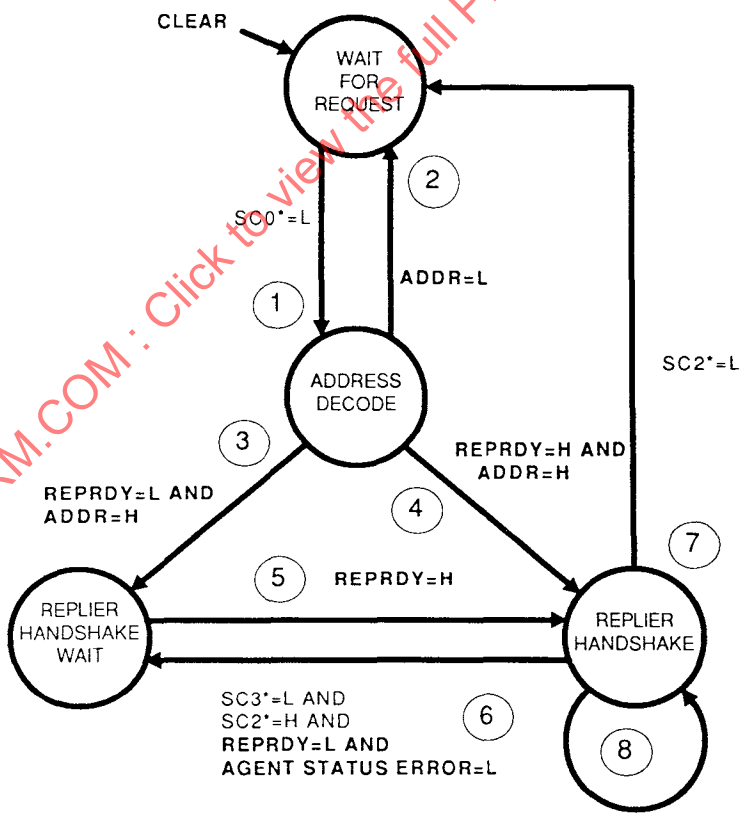


Figure 6.6-5—State-flow diagram for replying agents

ADDR: This condition indicates that the address on the bus during the request phase matches the address of the agent in the selected address space.

REPRDY: Internal data-valid timing for read and write data transfers on the bus. In the case of a read data transfer, it indicates that valid data will be driven on the bus in the next bus clock cycle. In the case of a read data transfer, there can be no dependence on SC3* or SC2* being asserted as indicators. The replying agent must supply data without waiting for the owner. In a write data transfer it indicates that the replying agent will be able to consume the data in the next bus clock cycle. **REPRDY** may depend on SC3* = L (i.e., the replying agent may wait until after the data has been driven onto the bus and had time to take it in before signalling it is ready).

AGENT STATUS ERROR:

SC5* = L OR SC6* = L OR SC7* = L

The replying agent is signalling an agent status error. This condition is meaningful only if there is a transfer operation in progress (agent is in the TRANSFER operation state of the Monitor State Machine, shown in figure 6.6-1). Note that the replying agent state machine guarantees that SC4* = L whenever SC5* = L, SC6* = L, or SC7* = L; allowing SC4* to be disregarded for this condition.

INITIAL TRANSITION

Transitions: The **CLEAR** condition aborts any operation currently in progress.

WAIT FOR REQUEST state

Description: Agent is idle.

Outputs: None.

Transitions: (1) On sensing SC0* = L, the state machine moves to the ADDRESS DECODE state.
Other: Some implementations do not require a state to perform the decode, and may have transitions to either REPLIER HANDSHAKE WAIT state or REPLIER HANDSHAKE state directly from the WAIT FOR REQUEST state.

ADDRESS DECODE state

Description: Agent checks to determine if the address and address space on the bus matches its address.

Outputs: None.

Transitions: (2) If **ADDR** = L (address does not match), the state machine returns to the WAIT FOR REQUEST state.
(3) Or, if **ADDR** = H (address matches) and **REPRDY** = L (replying agent not ready), state machine moves to the REPLIER HANDSHAKE WAIT state.
(4) Or, if both **ADDR** = H and **REPRDY** = H, the state machine moves to the REPLIER HANDSHAKE state.
Other: Some implementations may require more than one state to decode. Such implementations must be able to preempt a decode and return to the WAIT FOR REQUEST state on sensing **EOT HANDSHAKE**.

REPLIER HANDSHAKE WAIT state

Description: The replying agent is not ready to perform its part of the data transfer and does not complete its portion of the handshake.

Outputs: SC4* = H, SC5* = H, SC6* = H, SC7* = H, and SC8* = H.

Transitions: (5) On **REPRDY** = H, the state machine moves to the REPLIER HANDSHAKE state.

REPLIER HANDSHAKE state

Description: The agent waits in this state until the owner completes its portion of the handshake. When this occurs, both agents have completed the data transfer.

Outputs: SC4 = L (replying agent ready); AD* (in the case of a read data transfer); SC<7..5>* (status reporting); and SC8* (parity).
Agent status (indicating successful or unsuccessful transfer operations) may only be signalled in this state. This state is entered once per data transfer period. Read data and SC<7..5>* must be valid the entire time the agent is in this state. Once an agent error is reported the replying agent remains in this state until SC2* = L.

Transitions: (6) On SC3* = L (bus owner ready) and SC2* = H (not EOT) and **REPRDY** = L and **AGENT STATUS ERROR** = L, the state machine moves to the REPLIER HAND-SHAKE WAIT state.
(7) Or, SC2* = L (EOT) the state machine moves to the WAIT FOR REQUEST state.
(8) Normally transitions to the same state have been omitted; however, this transition represents handshaking the current data being immediately ready for the next one.

6.6.5 Effects of exception operations on state-flow diagrams

As figures 6.6-1 through 6.6-5 show, a system exception always returns an agent to the initial state of the state-flow as indicated by the **CLEAR** condition. During the exception operation, the requesting agents are held in the initial state of each state-flow diagram until the exception indication is removed from the bus. At the bus clock cycle after the exception has been removed the agents begin rearbitrating for the bus. Three bus clock cycles after that transfer operations may begin.

The exception always forces the requesting agents, replying agents, and the bus owner into the initial state of the arbitration state-flow and the transfer operation state-flow.

7. Electrical characteristics

7.1 General

This clause provides the electrical characteristics and connector pin assignments for the signals on the PSB. Abbreviations used to describe the electrical characteristics of the PSB are listed in table 7.1-1.

Table 7.1-1—Abbreviations used for the electrical specification

Name	Description	Notes
t_{on}	Time from falling edge of BCLK* until the signal is driven from tri-state to a valid logic state	
$t_{Clock\text{-}to\text{-}Data}$	Time from the falling edge of BCLK* until the signal is driven to a steady logic state	
t_{Off}	Time from the falling edge of BCLK* until the signal is no longer driven, i.e., to tri-state	
t_{Hold}	Time from the falling edge of BCLK* that signal must maintain its previous logic level	
t_{Setup}	Time from signal driven to its valid logic level until the falling edge of BCLK*	
I_{ih}	High input current load measured at 2.0 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
I_{il}	Low input current load measured at 0.8 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
I_{oh}	High output current drive measured at 2.4 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
I_{ol}	Low output current drive measured at 0.55 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
I_{ozh}	Leakage current for three-state high impedance output measured at 2.0 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
I_{ozl}	Leakage current for three-state high impedance output measured at 0.8 V	Positive values indicate current from the PSB to the agent. Values shown are measured in milliamperes.
C_i	Capacitive load presented by drivers/receiver	Values listed are in picofarads.
C_o	Maximum distributed capacitive load distributed over the length of a 21 slot backplane	Values listed are in picofarads.
Tri	3-state driver	
O.C.	Open collector driver	
TTL	Totem-pole driver	

7.2 AC timing specifications

All agents on the PSB must adhere to two general categories of timing requirements: requirements for boards driving signals and requirements for boards receiving signals. In both cases, the timing requirements are given with respect to the systemwide bus (BCLK*) received at the agent. The timing diagrams previously presented have been amended in this clause to show relevant timing parameters. Figures 7.2-1 through 7.2-7 and tables 7.2-1 through 7.2-7 provide timing specifications for the PSB.

The maximum bus trace length between any two agents must be less than or equal to 406.40 mm (16 in). The maximum trace length on the bus is 426.72 mm (16.8 in). The maximum stub length on agents interfacing to the bus must be less than 63.50 mm (2.5 in).

Table 7.2-1—Clock specification for the CSM

Refer to figure 7.2-1						
Parameter number	Parameter description	BCLK*		CCLK* (Note a)		Units
		Minimum	Maximum	Minimum	Maximum	
t_1	Rise time		2		2	nanoseconds
t_2	High time	48	52.1	23	27.0	nanoseconds
t_3	Fall time		2		2	nanoseconds
t_4	Low time	48	52.1	23	27.05	nanoseconds
t_5	Period	99.9	100.1	49.95	50.05	nanoseconds
t_6	Clock-to-clock	0	+10			nanoseconds

NOTES
a) The frequency of CCLK* is twice that of BCLK*.
b) All of the parameters are specified at the driver for a 50 pF capacitive load over the temperature and voltage range of the driver. The parameters are measured at the connector of the CSM without backplane loading. The effects of backplane loading have already been taken into account in the ac specifications.
c) The frequencies of BCLK* and CCLK* may be reduced to dc for testing purposes only.

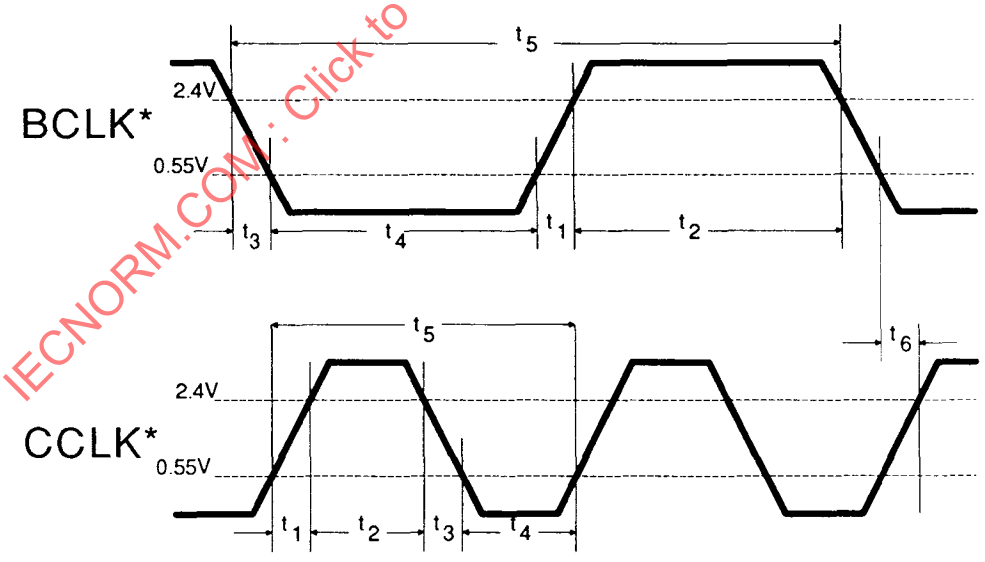


Figure 7.2-1—BCLK* and CCLK* timing relationships at the CSM

Table 7.2-2—Clock specifications for receivers

Refer to figure 7.2-2						
Parameter number	Parameter description	BCLK*		CCLK* (Note a)		Units
		Minimum	Maximum	Minimum	Maximum	
t_1	Rise time	2	10	2	10	nanoseconds
t_2	High time	40	60.1	15	35.05	nanoseconds
t_3	Fall time		2		2	nanoseconds
t_4	Low time	40	60.1	15	35.05	nanoseconds
t_5	Period	99.9	100.1	49.95	50.05	nanoseconds
t_6	Clock-to-clock	−5	+10			nanoseconds
t_7	BCLK* to CCLK*	20	39			nanoseconds

NOTES
a) The frequency of CCLK* is twice that of BCLK*.
b) Clock skew between BCLK* and CCLK*, because of differences in backplane propagation, is ±5 ns (CCLK* may be faster because of its lighter loading; some agents may not electrically load this signal).
c) The frequencies of BCLK* and CCLK* may be reduced to dc for testing purposes only.

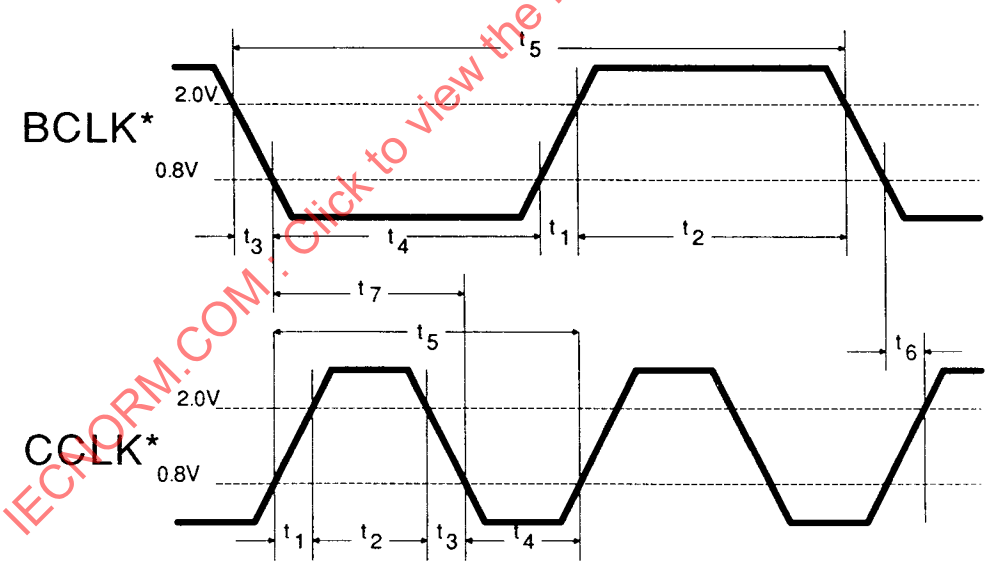


Figure 7.2-2—BCLK* and CCLK* timing relationships for receivers

Table 7.2-3—Timing parameters for signal driver

Refer to figure 7.2-3				
Signal names	$t_{Clock\text{-}to\text{-}Data}$ **Maximum**	t_{Hold} **Minimum**	t_{Off} **Maximum**	**Units**
AD*	40	7	40	nanoseconds
PAR*	40	7	40	nanoseconds
SC*	40	6	31	nanoseconds
BREQ*	40	6	40	nanoseconds
ARB* (Note c)	40	6	40	nanoseconds
TIMOUT*	40	6	40	nanoseconds
BUSERR*	40	6	40	nanoseconds
RST*	40	6	40	nanoseconds
RSTNC*	40	6	40	nanoseconds
LACHn	40	7	40	nanoseconds

NOTES

a) The minimum t_{on} must be greater than or equal to the minimum t_{Hold}.

b) The minimum t_{Off} must be greater than or equal to the minimum t_{Hold}.

c) All of the parameters are specified at the driver for a 50 pF capacitive load over the temperature and voltage range of the driver without backplane loading. The effects of backplane loading have already been taken into acount in the ac specifications.

d) During the resolution phase of arbitration the maximum propagation delays are ARBn* to ARBn-1* is 38 ns (e.g., ARB4* to ARB3*), ARBn* to ARBn-2* is 40 ns (e.g., ARB3* to ARB1*) and ARBn* to ARBn-3* is 42 ns (e.g., ARB3* to ARB0*).

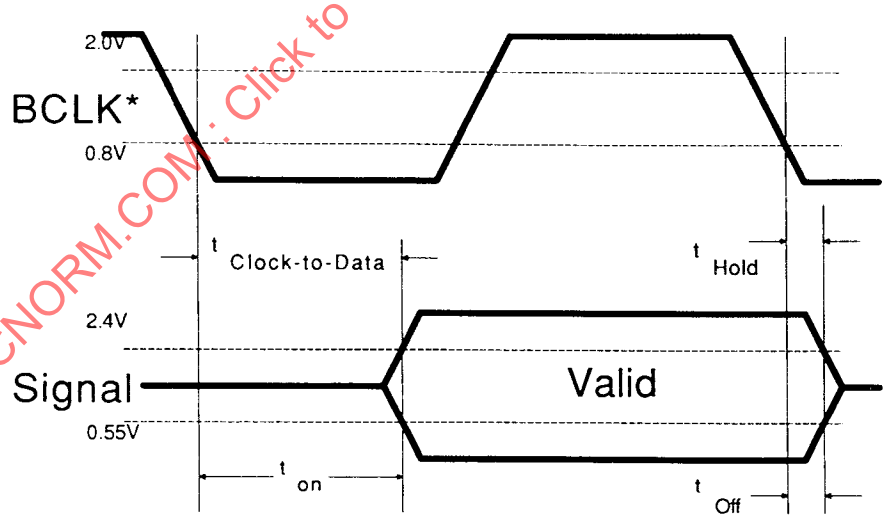


Figure 7.2-3—Driver timing parameters

Table 7.2-4—Timing parameters for signal receivers

Refer to figure 7.2-4			
Signal names	**t_{Setup} Minimum**	**t_{Hold} Minimum**	**Units**
AD*	30	5	nanoseconds
PAR*	30	5	nanoseconds
SC*	30	4	nanoseconds
BREQ*	21	4	nanoseconds
ARB*	40	4	nanoseconds
TIMOUT*	30	4	nanoseconds
BUSERR*	21	4	nanoseconds
RST*	30	4	nanoseconds
RSTNC*	21	4	nanoseconds
LACHn*	30	5	nanoseconds

NOTES

a) The amount of time lost due to backplane transmission line requirements (reflection and cross-talk noise) has been taken into account in calculating the t_{Setup} times from the $t_{Clock\text{-}to\text{-}Data}$ times. The time delay is the sum of two bus propagation delays (25 ns) plus the maximum clock skew (5 ns) for all signals except the open-collector and SC* lines. These add to a total bus loss of 30 ns for non-open-collector signals and for the SC* signals. The bus loss for open-collector signals is 39 ns. The SC* signals have a bus loss of 30 ns when driven and have a bus loss of 39 ns when not driven.

b) The maximum clock skew between any two agents is less than or equal to 5 ns.

c) t_{Setup} and t_{Hold} for ARB* are only valid at the end of the RESOLUTION-3 state.

d) The BCLK* fall time from 2.0 V to 0.8 V is less than or equal to 2 ns.

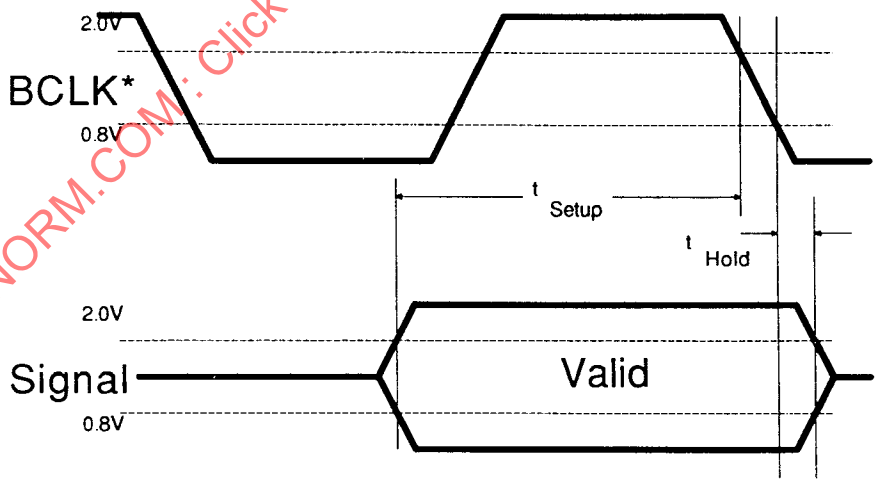


Figure 7.2-4—Timing parameters for signal receivers

Table 7.2-5—Cold-start control timing

Refer to figure 7.2-5				
Parameter	**Description**	**Minimum**	**Maximum**	**Units**
t_1	DC power set-up to DCLOW*	—	1	milliseconds
t_2	Cold-reset duration	2.5	—	milliseconds
t_3	Warm-reset duration	50	—	milliseconds
t_4	RSTNC* set-up to RST* (inactive)	1	—	Bus clock cycles

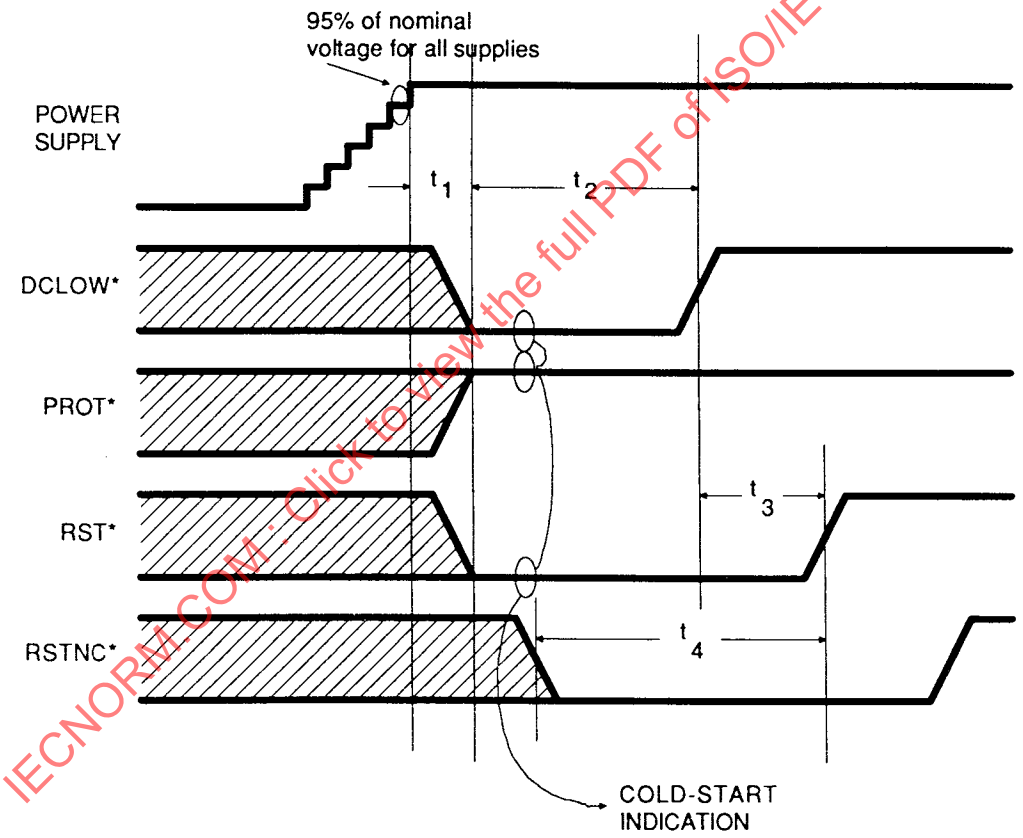


Figure 7.2-5—Cold-start timing parameters

Table 7.2-6—Warm-start control timing

Refer to figure 7.2-6				
Parameter	**Description**	**Minimum**	**Maximum**	**Units**
t_1	RST* pulse width (CSM)	50	—	milliseconds
t_2	RSTNC* set-up to RST* (inactive)	1	—	Bus clock cycles

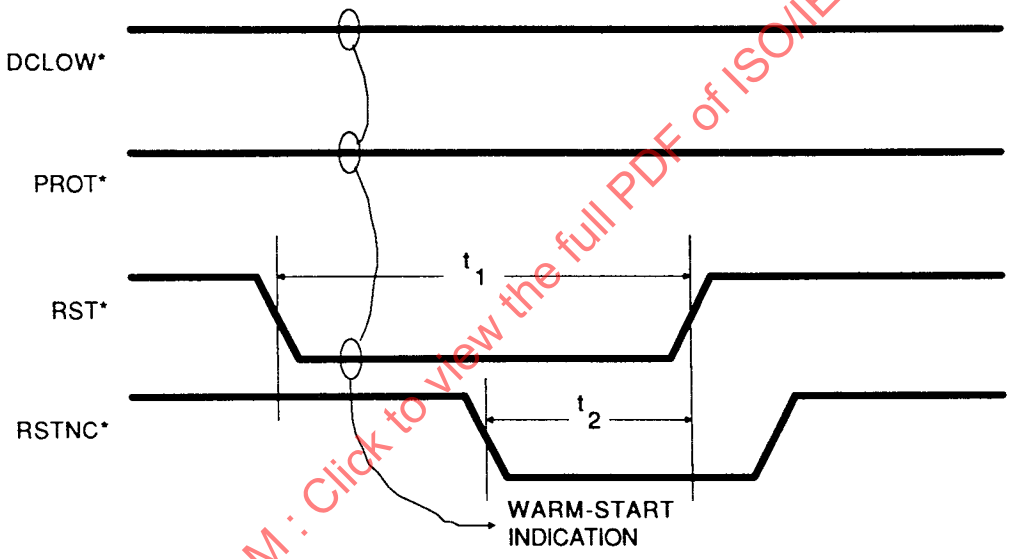


Figure 7.2-6—Warm-start timing parameters

Table 7.2-7—Power failure and recovery timing

Refer to figure 7.2-7				
Parameter	**Description**	**Minimum**	**Maximum**	**Units**
t_1	DC power hold from DCLOW*	6.5	—	milliseconds
t_2	PROT* delay from DCLOW*	6.0	6.25	milliseconds
t_3	DC power set-up to DCLOW*	1	—	milliseconds
t_4	RST* delay from DCLOW*	6.5	7.0	milliseconds
t_5	RST* set-up from DCLOW*	0.5	—	milliseconds
t_6	RST* active from PROT*	50	—	milliseconds
t_7	DCLOW* pulse width	7.5	—	milliseconds
t_8	PROT* hold from DCLOW*	2.0	2.5	milliseconds
t_9	RSTNC* set-up to RST*	1		Bus clock cycles

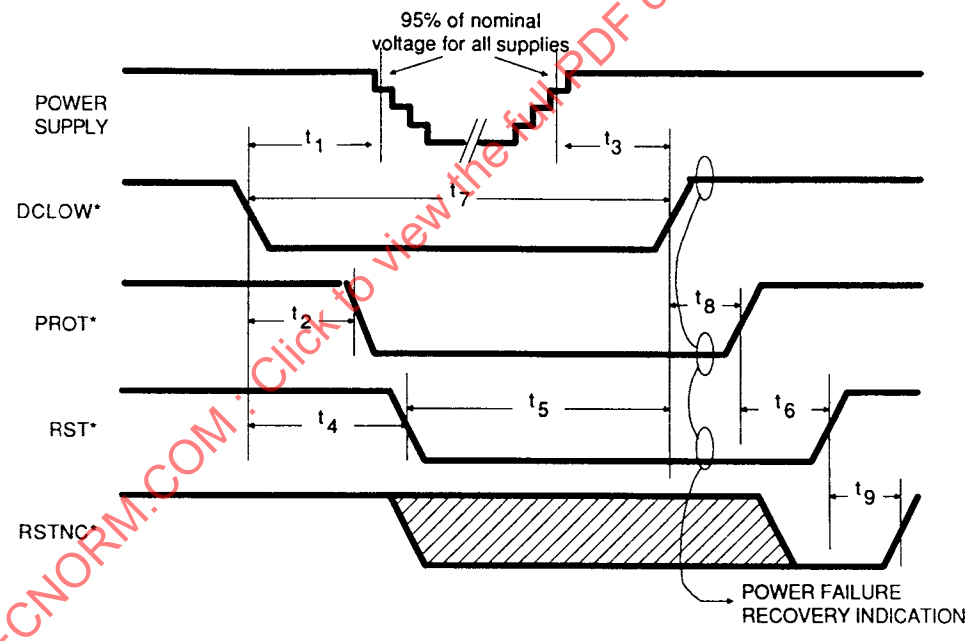


Figure 7.2-7—Power Failure and recovery timing parameters

7.3 DC specifications for signals

All agents on the PSB must adhere to the dc requirements presented in tables 7.3-1 through 7.3-3.

Table 7.3-1—DC specifications for signal drivers

Signal names	Driver locations	Driver type	I_{ol} minimum (milliamperes)	I_{oh} minimum (milliamperes)	C_o (Note d) (picofarads)
AD* (Note c)	Requesting and replying agents	Tri	48	−3	500
PAR*	Requesting and replying agents	Tri	48	−3	500
SC*	Requesting and replying agents	Tri	64	−3	500
BREQ*	Requesting agents	O.C.	60	(Note b)	500
ARB*	Requesting agents	O.C.	60	(Note b)	500
BUSERR*	Requesting and replying agents	O.C.	60	(Note b)	500
TIMOUT*	CSM	TTL	48	−3	300
LACHn*	Connected to AD* lines		(refer to the AD* lines)	(refer to the AD* lines)	(refer to the AD* lines)
RST*	CSM	TTL	18	−3	300
RSTNC*	All bus agents	O.C.	60	(Note b)	500
DCLOW*	CSM	TTL	48	−3	300
PROT*	CSM	TTL	48	−3	300
BCLK* (Note a)	CSM	TTL	60	−3	120
CCLK* (Note a)	CSM	TTL	60	−3	120

NOTES
a) If the number of cardslots in the backplane exceeds twelve, BCLK* and CCLK* signals must be driven from the middle of the backplane by two sets of drivers, each driving half of the backplane. The maximum skew between the two halves of either BCLK* or CCLK* must not exceed 1 ns at the CSM connector.
b) The high level output leakage (I_{oh}) of each open-collector driver must be less than or equal to 400 μA at 2.4 V.
c) The I_{ol} for AD* lines must be guaranteed at the connector. If a driver/receiver pair are used instead of a transceiver, then the I_{il} of the receiver on the driving agent must be subtracted from the I_{ol} of the driver.
d) C_o shown for reference purposes only.

Table 7.3-2—DC specifications for signal receivers

Signal names	Signal receiver locations	$I_{il} = I_{ozl}$ maximum (milliamperes)	$I_{ih} = I_{ozh}$ maximum (Note c) (microamperes)	C_i maximum (picofarads)
AD*	Requesting and replying agents	−1	100	20
PAR*	Requesting and replying agents	−1	100	20
SC*	Requesting and replying agents	−1	100	20
BREQ*	Requesting agents	−0.9	100 (Note b)	20
ARB*	All agents	−0.9	100 (Note b)	20
BUSERR*	Requesting and replying agents	−0.9	100 (Note b)	20
TIMOUT*	All bus agents	−1	100	20
LACHn*	On AD* lines for agents	−1	100	10
RST*	All bus agents	−1	100	12
RSTNC*	All bus agents	−0.9	100 (Note b)	20
DCLOW*	All bus agents	−1	100	12
PROT*	All bus agents	−1	100	12
BCLK* (Note a)	All bus agents	−1.5	100	8
CCLK* (Note a)	All bus agents	−1.5	100	8

NOTES
a) If the number of cardslots in the backplane exceeds twelve, BCLK* and CCLK* signals must be driven from the middle of the backplane by two sets of drivers, each driving half of the backplane. The maximum skew between the two halves of either BCLK* or CCLK* must not exceed 1 ns.
b) The high level output leakage (I_{oh}) of each open-collector driver must be less than or equal to 400 μA at 2.4 V.
c) The open-collector drivers' high-level output current (I_{oh}, Note b above) must not be confused with I_{ozh}. The value specified here is the maximum allowed for all receivers (on an agent) of a signal driven by open-collector drivers.

Table 7.3-3—Backplane termination requirements

Signal names	Location of termination	Termination to 5 V (ohms) (3%)	Termination to 0 V (ohms) (3%)
AD*	Both ends of backplane	330	470
PAR*	Both ends of backplane	330	470
SC*	Both ends of backplane	220	330
BREQ*	Both ends of backplane	220	330
ARB*	Both ends of backplane	220	330
BUSERR*	Both ends of backplane	220	330
TIMOUT*	Both ends of backplane	330	470
LACHn*	Not required (see AD* lines)	none	none
RST*	Both ends of backplane	330	470
RSTNC*	Both ends of backplane	220	330
DCLOW*	Both ends of backplane	330	470
PROT*	Both ends of backplane	330	470
BCLK*	Farthest point from driver	110	120
CCLK*	Farthest point from driver	110	120

7.4 Current limitations per connector

Tables 7.4-1 and 7.4-2 give the voltage and current requirements for a system using only the PSB and for a system using both the PSB and the recommended P2 power connections (shown in 7.5). The voltage specifications at the connector are measured over the full current range.

Table 7.4-1—Power limitations for a one-connector agent

Minimum voltage	Nominal voltage	Maximum voltage	Maximum amperes
+4.90	+5.00	+5.25	9.0
+4.90	+5.00	+5.25	2.0 (Battery)
+11.40	+12.00	+12.60	2.0
−11.40	−12.00	−12.60	2.0
—	0 V	—	15.0 (Note a)

NOTES
a) On all cardslots except cardslot 0 which has a maximum of 14 A at 0 V.
b) Voltages are measured at the connector of each board.
c) Refer to DIN 41612[5] for power capabilities per connector pin.

Table 7.4-2—Power limitations for a two-connector agent

Minimum voltage	Nominal voltage	Maximum voltage	Maximum amperes
+4.90	+5.00	+5.25	13.0
+4.90	+5.00	+5.25	2.0 (Battery)
+11.40	+12.00	+12.60	2.0
−11.40	−12.00	−12.60	2.0
—	0	—	19.0 (Note a)

NOTES
a) On all cardslots except cardslot 0 which has a maximum of 18 A at 0 V.
b) Voltages are measured at the connector of each board.
c) Refer to DIN 41612 for power capabilities per connector pin.

[5] Information on references can be found in 1.2.

7.5 Pin assignments

In this subclause, the pin assignments for the PSB on the P1 connector and the recommended power pin assignments for the P2 connector are provided.

7.5.1 PSB pin assignments

The pin assignment for the 96-pin connector to the PSB is listed in table 7.5-1.

Table 7.5-1—PSB connector pinout

Connector pin number	Row A	Row B	Row C
1	0 V	PROT*	0 V
2	+5 V	DCLOW*	+5 V
3	+12 V	+5 Battery	+12 V
4	0 V (Note a)	SDA (Note c)	BCLK*
5	TIMOUT*	SDB (Note c)	0 V
6	LACHn* (Note b)	0 V	CCLK*
7	AD0*	AD1*	0 V
8	AD2*	0 V	AD3*
9	AD4*	AD5*	AD6*
10	AD7*	+5 V	PAR0*
11	AD8*	AD9*	AD10*
12	AD11*	+5 V	AD12*
13	AD13*	AD14*	AD15*
14	PAR1*	0 V	AD16*
15	AD17*	AD18*	AD19*
16	AD20*	0 V	AD21*
17	AD22*	AD23*	PAR2*
18	AD24*	0 V	AD25*
19	AD26*	AD27*	AD28*
20	AD29*	0 V	AD30*
21	AD31*	Reserved	PAR3*
22	+5 V	+5 V	Reserved
23	BREQ*	RST*	BUSERR*
24	ARB5*	+5 V	ARB4*
25	ARB3*	RSTNC*	ARB2*
26	ARB1*	0 V	ARB0*
27	SC9*	SC8*	SC7*
28	SC6*	0 V	SC5*
29	SC4*	SC3*	SC2*
30	−12 V	+5 Battery	−12 V
31	+5 V	SC1*	+5 V
32	0 V	SC0*	0 V

NOTES
a) In slots 1–20 pin 4A is a 0 V pin. This pin signals to a CSM module whether it must perform its CSM function (if not 0 V). In slot 0 pin 4A is used for the second BCLK* driver on the CSM module in systems that contain more than twelve slots. BCLK* is then routed through pin 4C to the left half of backplane.
b) Slot 0: Pin 6A is the second CCLK* driver for systems containing more than 12 slots; CCLK* is then routed to pin 6C to the left half of the backplane. Slot 1–20: Pin 6A is the LACHn* signal.
c) Signal lines SDA and SDB are reserved for a backplane serial bus.

7.5.2 P2 recommended power pin assignments

Table 7.5-2 shows the recommended power connections for the P2 connector of IEEE 1296 systems.

Table 7.5-2—P2 connector recommended power pinout

Connector pin number	Row A	Row B	Row C
1	0 V		0 V
2	+5 V		+5 V
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31	+5 V		+5 V
32	0 V		0 V

NOTE—All other connections to the P2 connector are available for user-defined functions.

8. Mechanical specifications

8.1 General

The mechanical specifications for IEEE 1296 are based upon IEEE Std 1101-1987.[6] This clause contains the limitations, additions, and restrictions required by IEEE 1296 to the IEEE Std 1101-1987 specification; please refer to that document for any details not included herein.

This clause lists and defines the mechanical specifications that designers must be concerned with to ensure that an IEEE 1296 system is dimensionally compatible with International Electrotechnical Commission (IEC) standards. The dimensions and drawings within this specification are based on IEC Pubs 297-1 and 297-3, and IEEE Std 1101-1987, where applicable.

The four main parts of this clause provide mechanical specification for connectors, IEEE 1296 boards, front panels and backplanes.

The IEEE 1296 boards, board accessories, and backplanes must be designed and manufactured to the following specifications:

a) IEC Pub 603-2 Class 2 connectors
b) IEC Pub 297-3 for subrack, board design, and connector mounting
c) IEEE Std 1101-1987, Core Mechanical Specification

If there are discrepancies between these specifications, IEEE Std 1101-1987 should overrule.

Figure 8.1-1 shows the conventional, double-high boards defined for the IEEE 1296 specification and shows the assignment of functions and buses to the connectors. Figure 8.1-2 shows the allowable

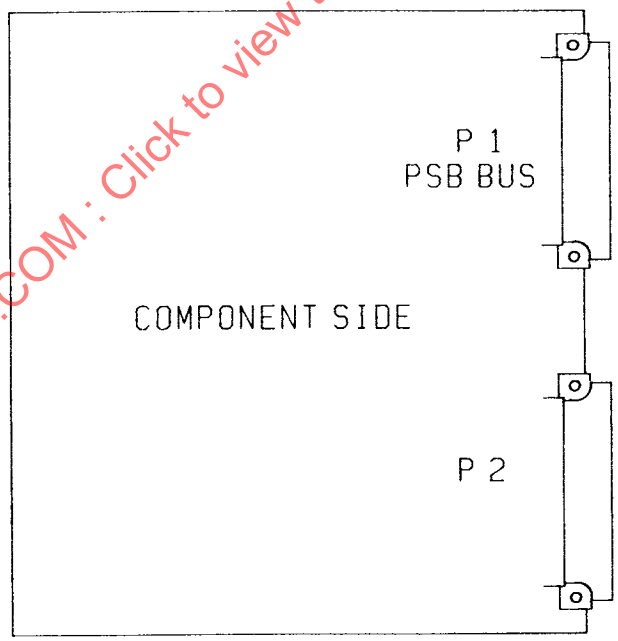


Figure 8.1-1—Double eurocard connector definitions

[6] Information on references can be found in 1.2.

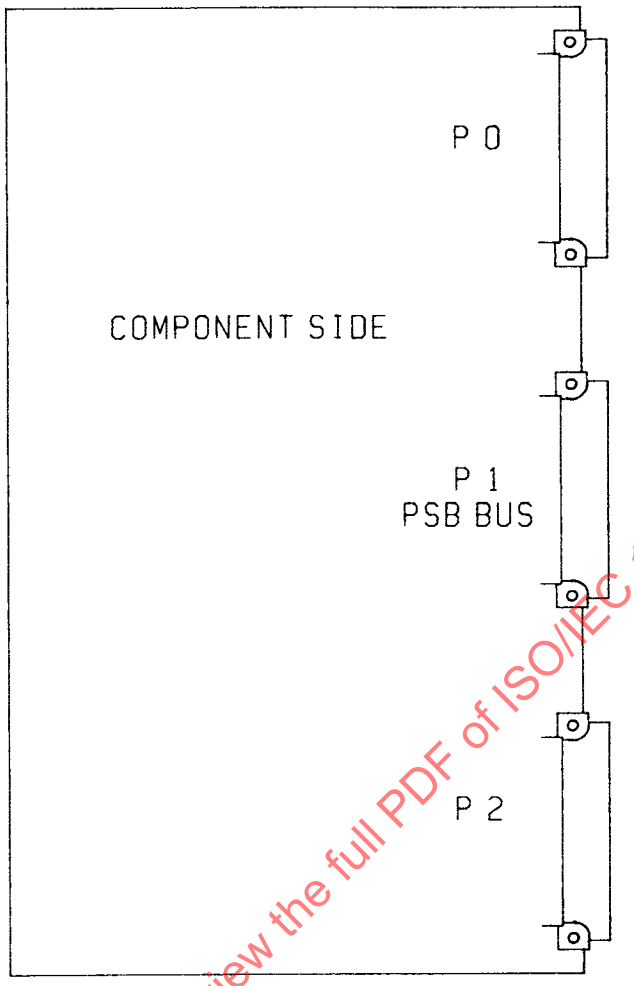


Figure 8.1-2—Triple eurocard connector definitions

triple-high boards and the connector assignment. The PSB is always defined on the P1 connector, as depicted in the figures.

The example system in figure 8.1-3 illustrates the physical relationships between subracks, backplanes, printed boards, and accessories. Boards reside vertically in the subrack with the component side facing the right and the P1 connector (used for the PSB backplane) above the P2 locations. The PSB backplane is mounted onto the subrack. The double-high board is connected to both the PSB and P2 backplanes.

8.2 Board sizes and dimensions

The physical dimensions for the boards that connect to the bus are described in this subclause. The conventional module size is 233.35 mm (9.19 in) (two connectors high) by 220 mm (8.66 in) deep. In that this International Standard follows the standard IEEE microcomputer board size specification (as in IEEE Std 1101-1987), this International Standard also permits usage of all board sizes allowed by that specification.

If a board size larger than the conventional 233.35 mm by 220 mm is required, this International Standard makes particular reference to the following larger board sizes: 366.7 mm (14.44 in) (3 connector) by 220 mm, and 366.7 mm (3 connector) by 280 mm (11.02 in). The smallest size that will satisfy the

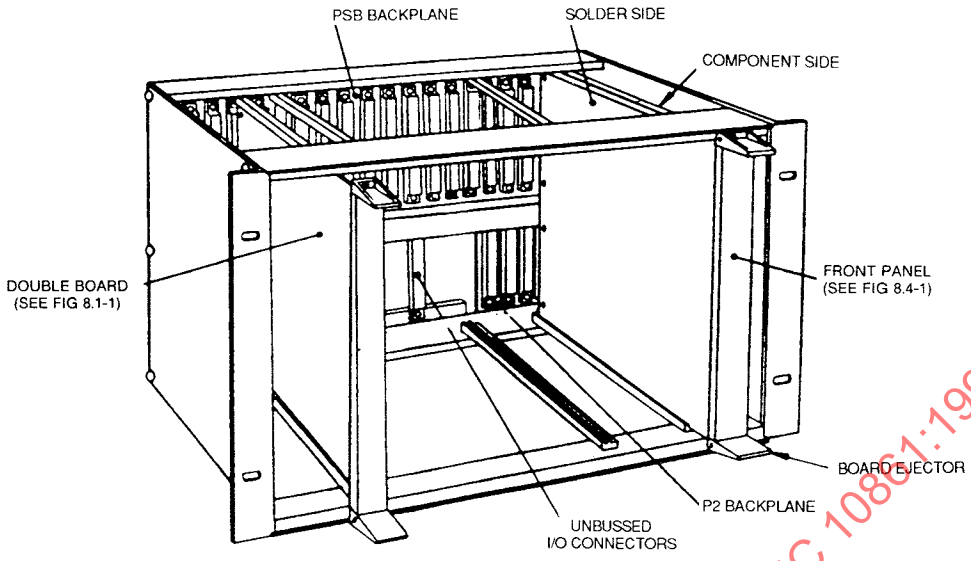


Figure 8.1-3—Typical subrack

system requirements should be used with the conventional 233.35 mm by 220 mm (8.66 in) size being the smallest.

IEEE 1296 boards are designed for compatibility with subracks designed according to IEC Pub 297-3. Figure 8.2-1 shows the double-high form factor defined for an IEEE 1296 system. As defined in IEC Pub 297-3, the double-high board corresponds to the 6U board height (233.35 mm) and the triple-high board corresponds to the 9U (366.7 mm) board height.

8.3 Printed board layout considerations

Hole positions and optional center mounting bracket are specified in accordance with figure 8.3-1.

Explanation: On the double-high board, a central mounting bracket is required to provide lateral support and reduce the effect of mechanical shock and vibration.

8.4 Front panel

Dimensional information for front panels and ejectors is provided in figures 8.4-1 and 8.4-2.

The standard spacing between boards is 20.32 mm (0.8 in). If a system requires a board spacing that deviates from the 20.32 mm dimension, a spacing increment of 5.08 mm shall be used.

Explanation: The 5.08 mm (0.2 in) dimension is the spacing increment of the cards specified in IEC Pub 297-3.

8.5 Connectors

IEEE 1296 boards and backplanes use two-piece, 96 pin connectors for interconnection to the PSB and P2 buses. The right-angle connectors on the printed circuit board are C096 board mounted connectors;

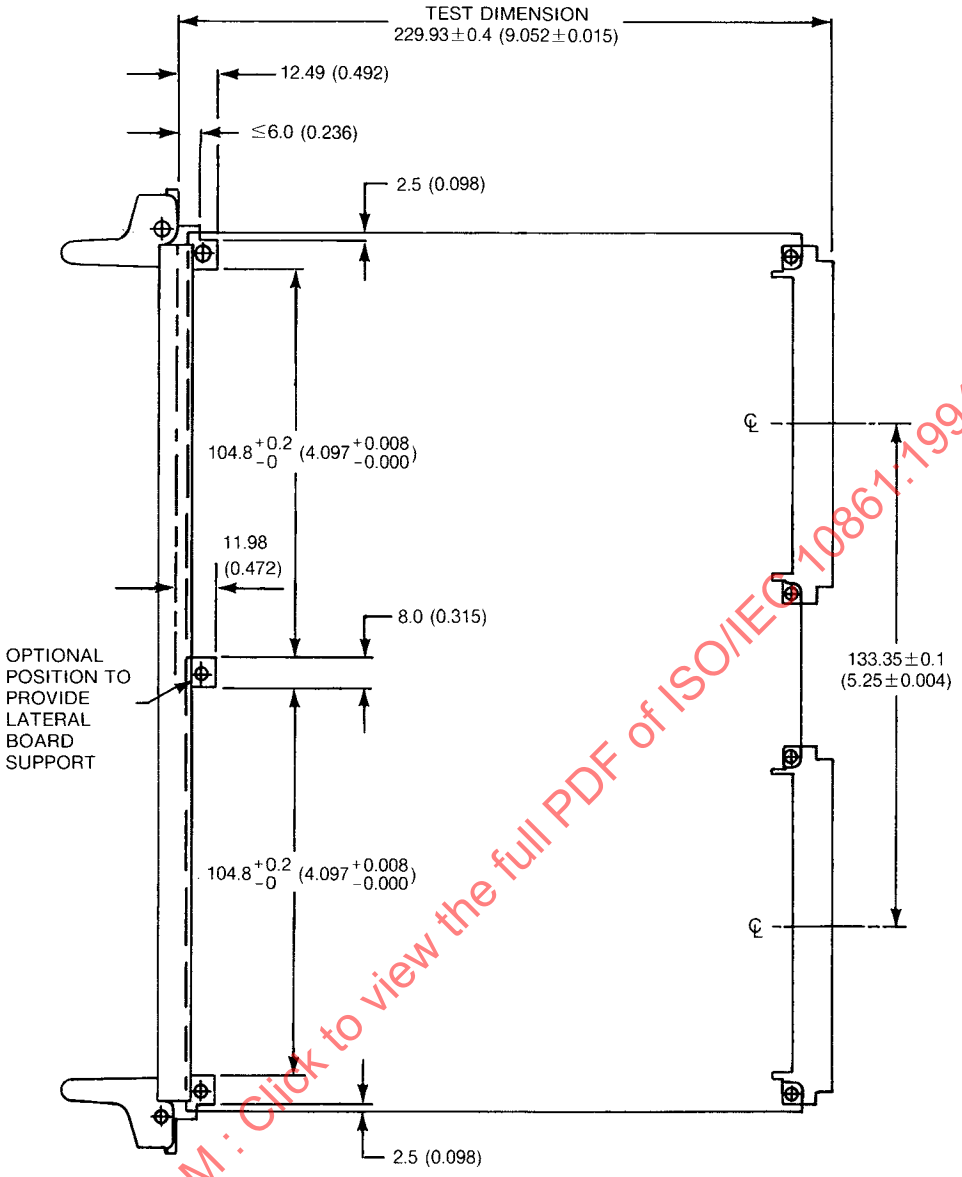


Figure 8.2-1—Double-high board dimensions

the receptacle connectors on the backplanes are C096 fixed connectors, both as specified in IEC Pub 603-2. All connectors should be Class II. Figure 8.5-1 illustrates the mechanical relationship between connectors and boards in the subrack.

8.6 Backplanes

This subclause specifies the layout and overall dimensioning of IEEE 1296 backplanes.

8.6.1 Mounting

Backplanes are mounted to the subrack with M2.5 × 0.45 pitch screws, so that adequate chassis ground is maintained. Electrical continuity is mandatory between backplane and mounting rails. Figure 8.6-1 shows the relationship between the connector mounting lug and the backplane mounting position.

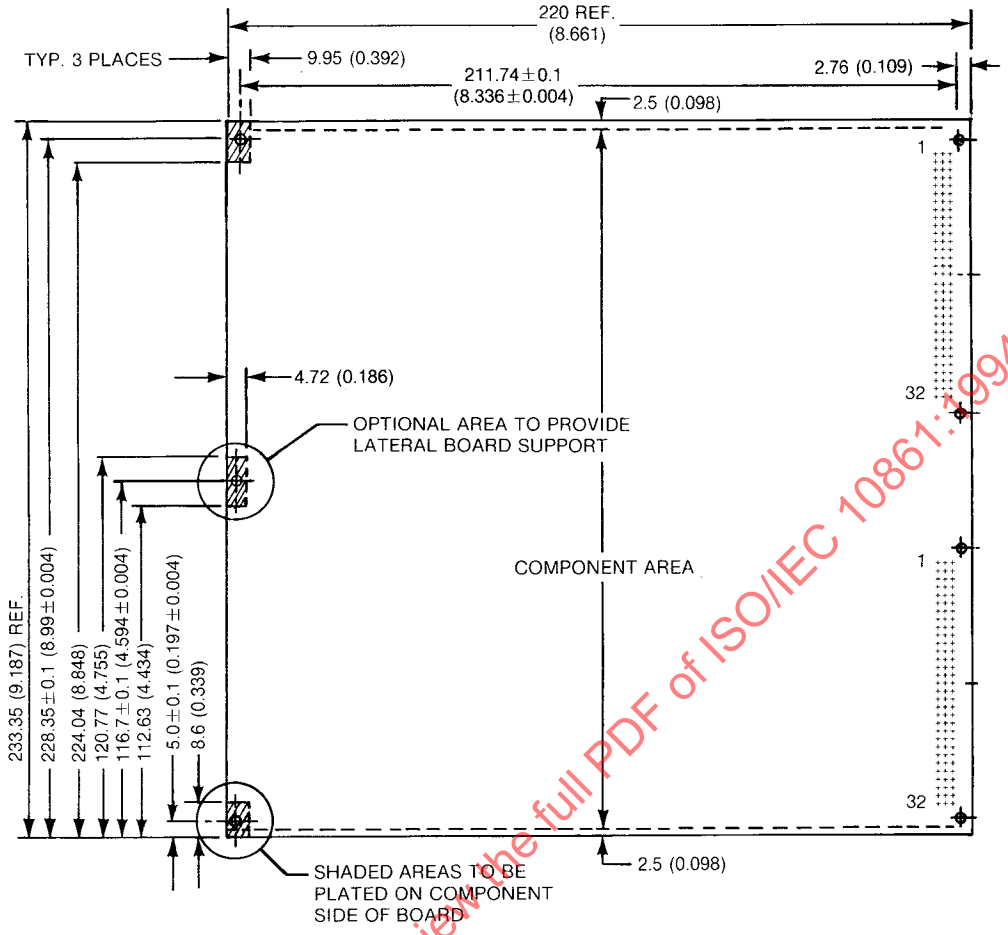


Figure 8.3-1—Board layout considerations for front panels

8.6.2 Backplane design

This subclause describes the recommended design of the backplane(s). Physical and electrical parameters are provided to enable backplane designers to more easily design to IEEE 1296 electrical specifications. Figure 8.6-2 shows detailed dimensional requirements for the layout of connectors on the backplane.

System designers should allow additional depth behind the plane of the backplane for input/output. The recommended minimum depth is 38 mm (1.5 in).

8.6.3 PSB backplane specifications

This subclause describes the electrical and physical parameters of the PSB backplane. Table 8.6-1 summarizes these parameters. This subclause also includes recommended PSB backplane design guidelines.

In 12-cardslot subrack systems, the leftmost cardslot is numbered 0. Cardslot numbers increment to the right. This manner of numbering applies to systems with a vertical board orientation.

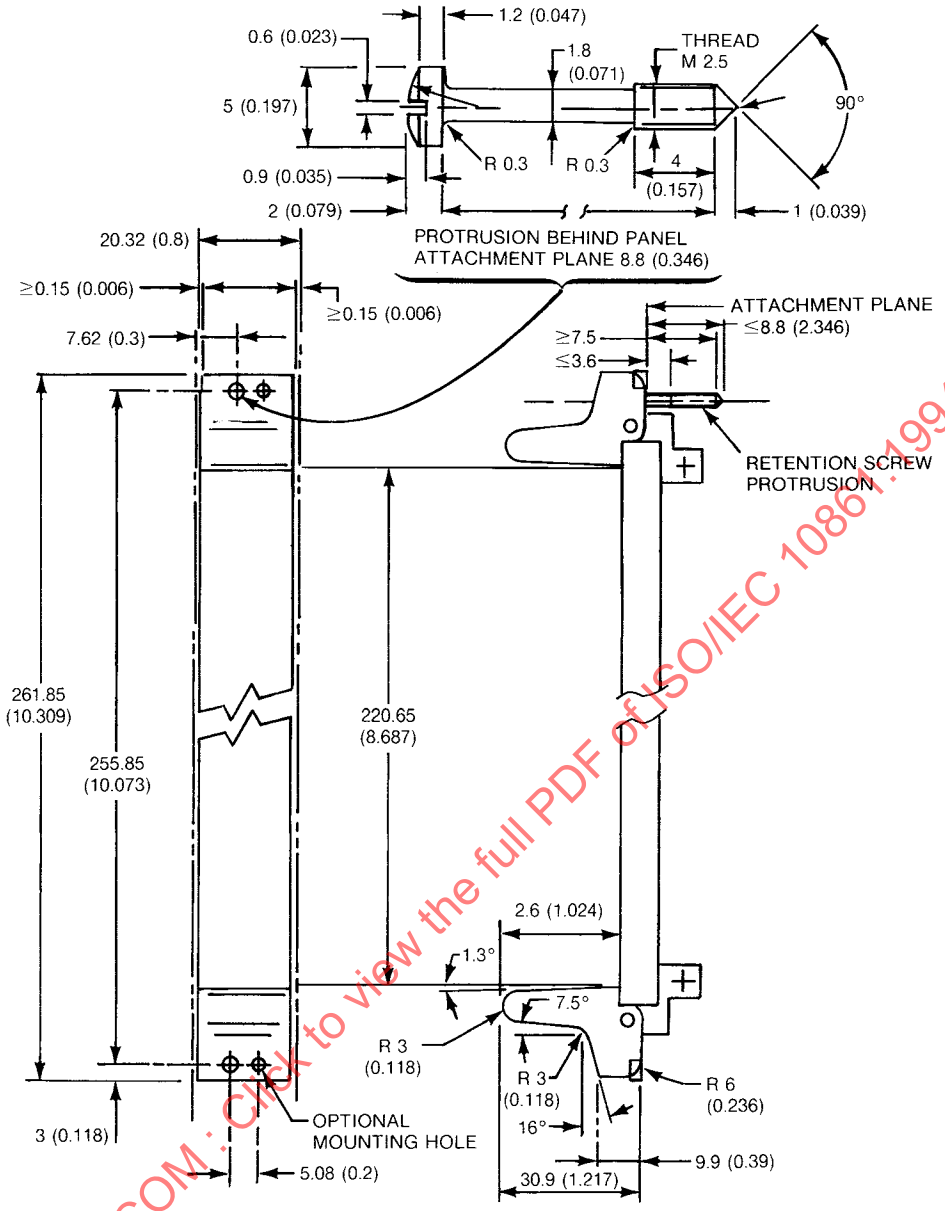


Figure 8.4-1—Front panel dimensions

In 13- through 21-cardslot subrack systems, cardslot 0 is located in the center of the backplane. The cardslot numbering is shown in table 8.6-2.

Cardslot 0 should be clearly identified on both backplane and subrack.

Termination is included at each end of the backplane and may be "off-board" (figure 8.6-3) or "on-board" (figure 8.6-4).

8.6.4 PSB backplane characteristics

Figure 8.6-5 is an example of the layer make-up of a backplane that will achieve the necessary electrical characteristics.

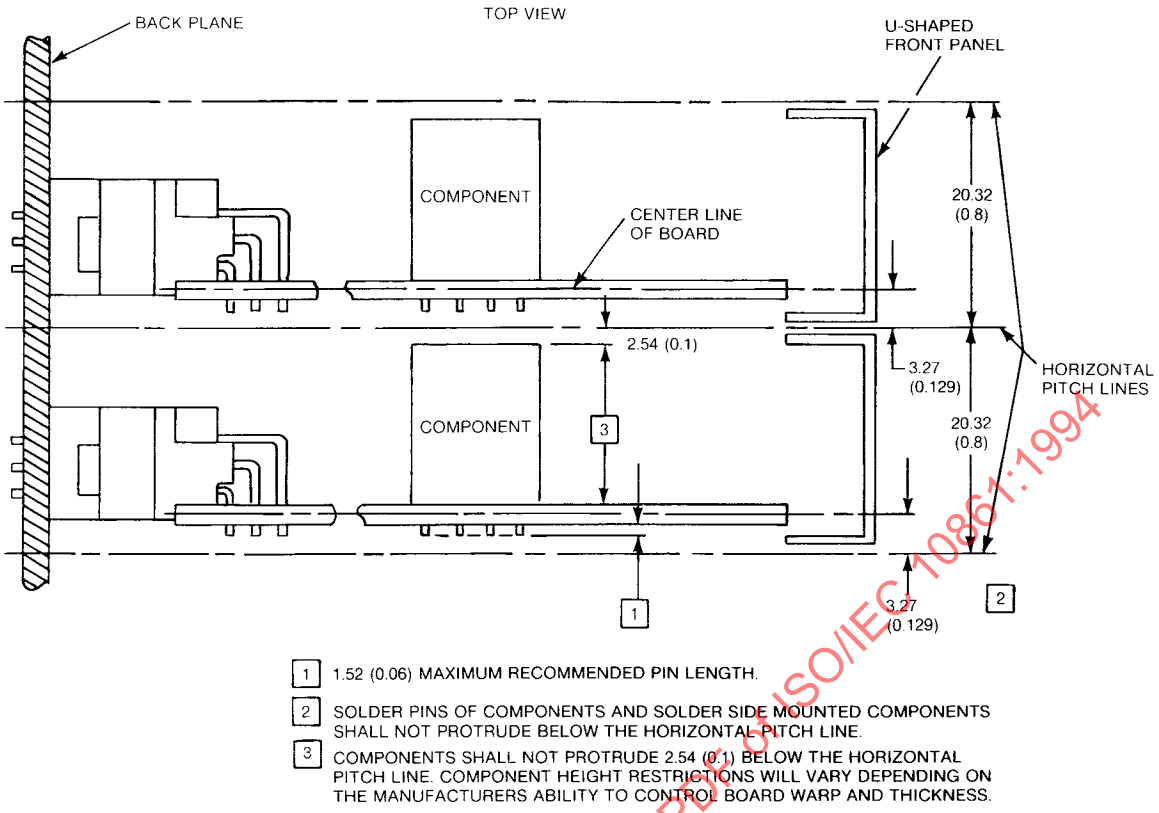


Figure 8.4-2—Board spacing and front panel relationships

Table 8.6-1—PSB backplane characteristics

Characteristic	Parameter
Center-to-center trace spacing	2.5 mm (0.1 in)
Trace width	0.25 mm (0.010 in)
Number of planes	6
Unloaded impedance	60–85 Ω
Copper thickness	0.61 kg/m² (2 oz/ft²)

Table 8.6-2—Cardslot numbering sequence for large backplanes

Number of cardslots	Numbering sequence
13	7 8 9 10 11 12 0 1 2 3 4 5 6
14	7 8 9 10 11 12 13 0 1 2 3 4 5 6
15	8 9 10 11 12 13 14 0 1 2 3 4 5 6 7
16	8 9 10 11 12 13 14 15 0 1 2 3 4 5 6 7
17	9 10 11 12 13 14 15 16 0 1 2 3 4 5 6 7 8
18	9 10 11 12 13 14 15 16 17 0 1 2 3 4 5 6 7 8
19	10 11 12 13 14 15 16 17 18 0 1 2 3 4 5 6 7 8 9
20	10 11 12 13 14 15 16 17 18 19 0 1 2 3 4 5 6 7 8 9
21	11 12 13 14 15 16 17 18 19 20 0 1 2 3 4 5 6 7 8 9 10

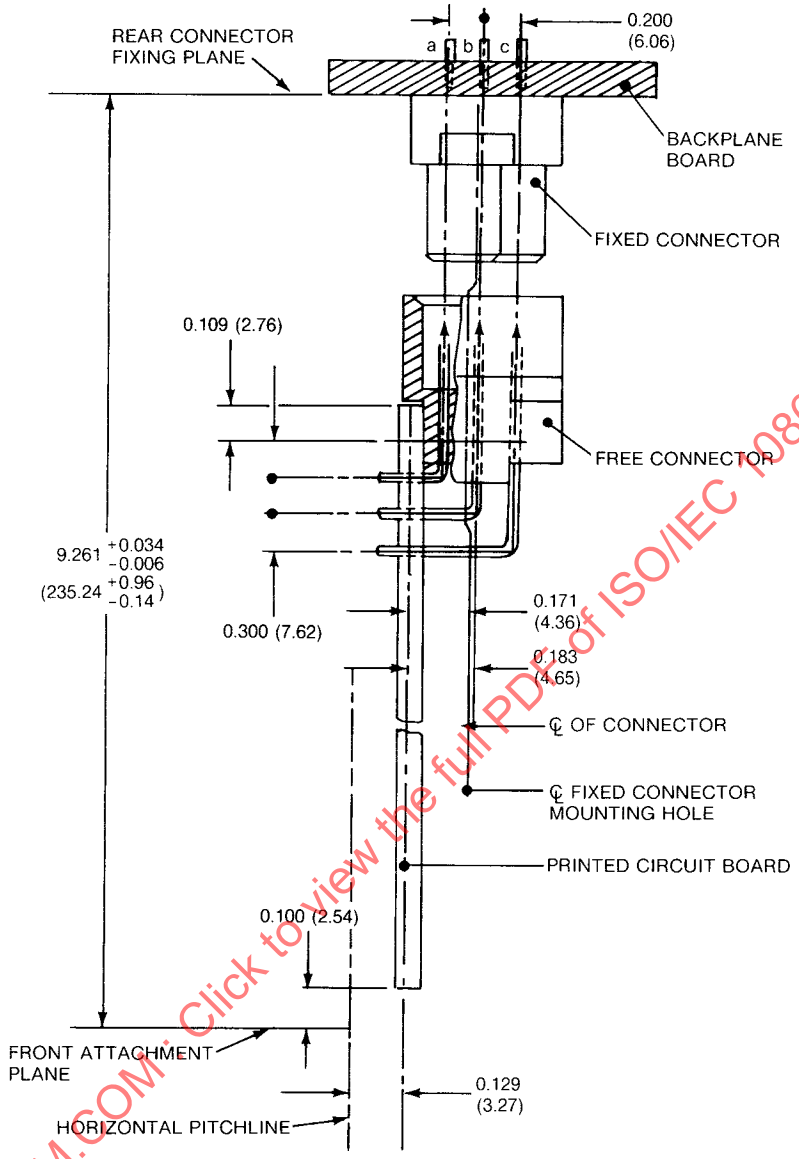


Figure 8.5-1 — Relationship of boards and connectors

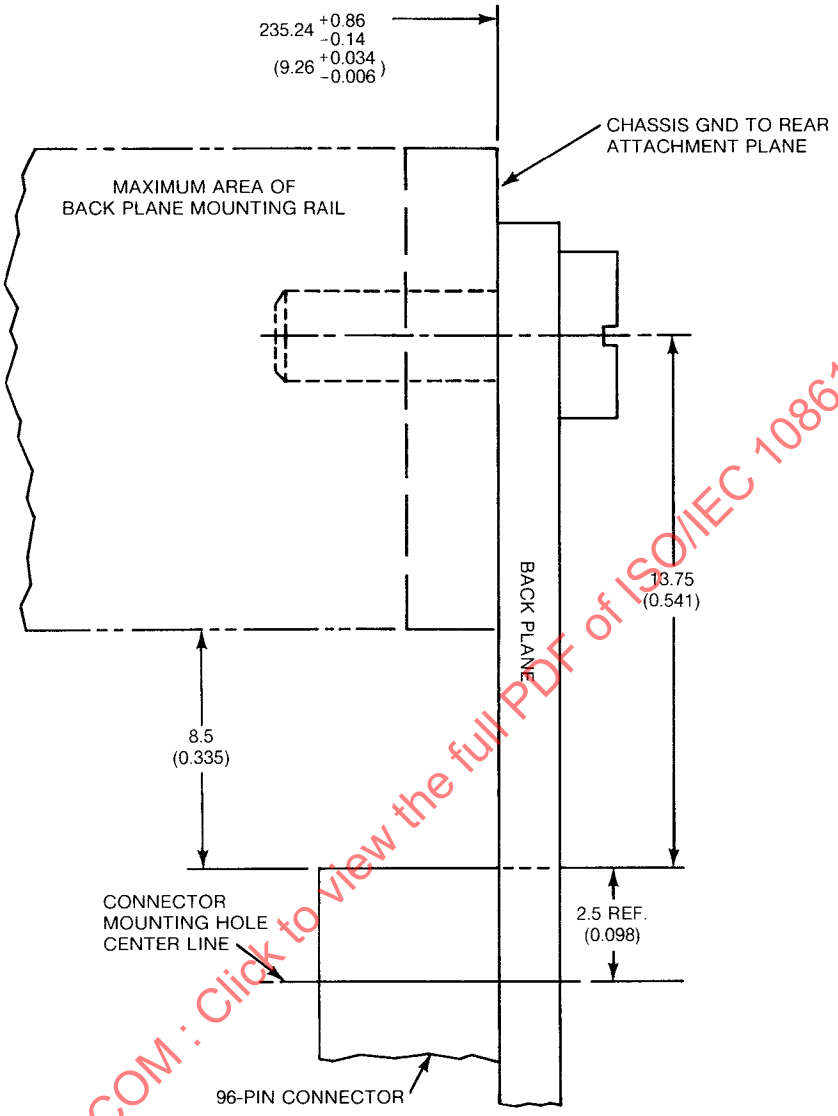


Figure 8.6-1 — Backplane and mounting rail relationship

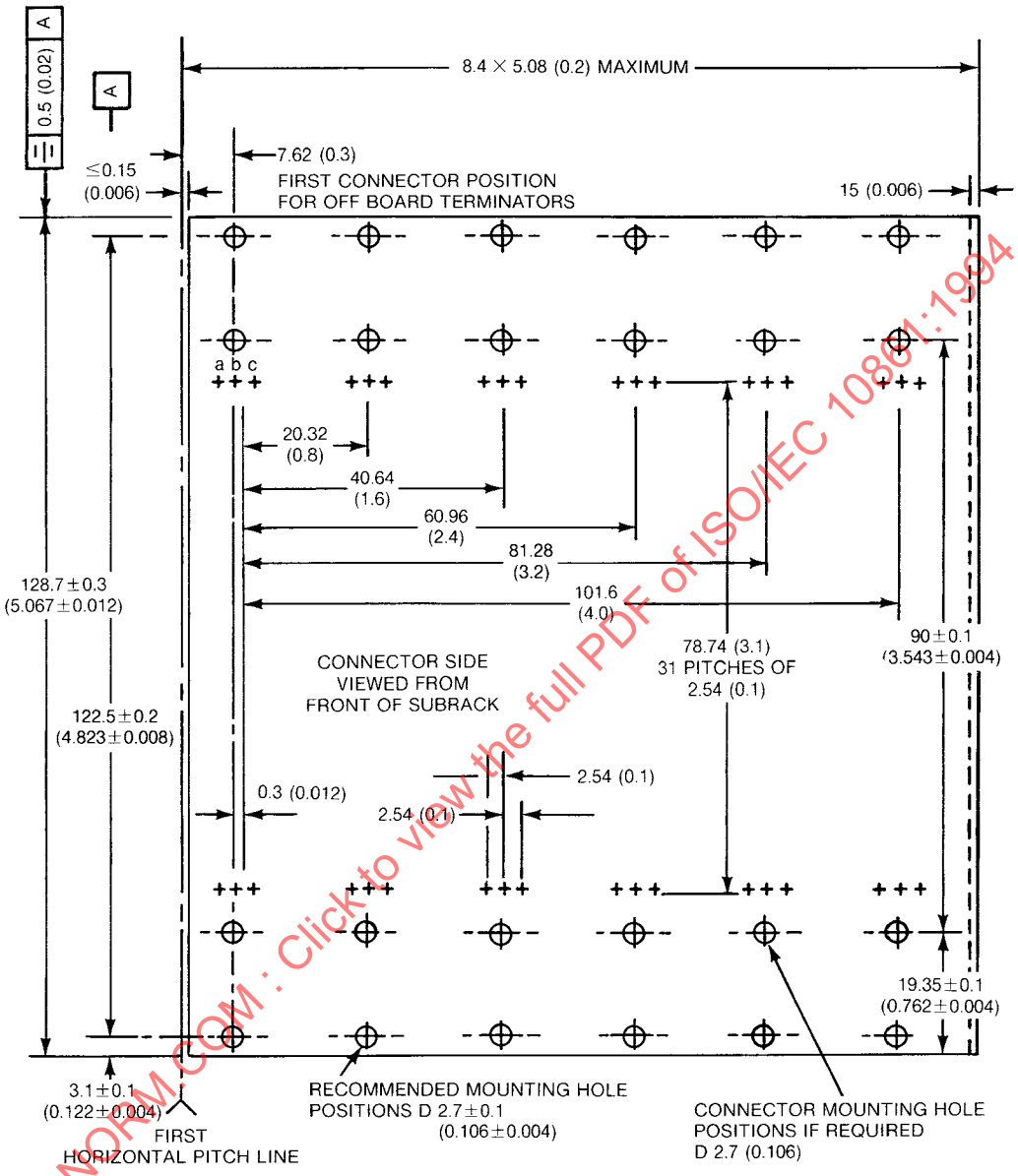


Figure 8.6-2—Backplane connector layout

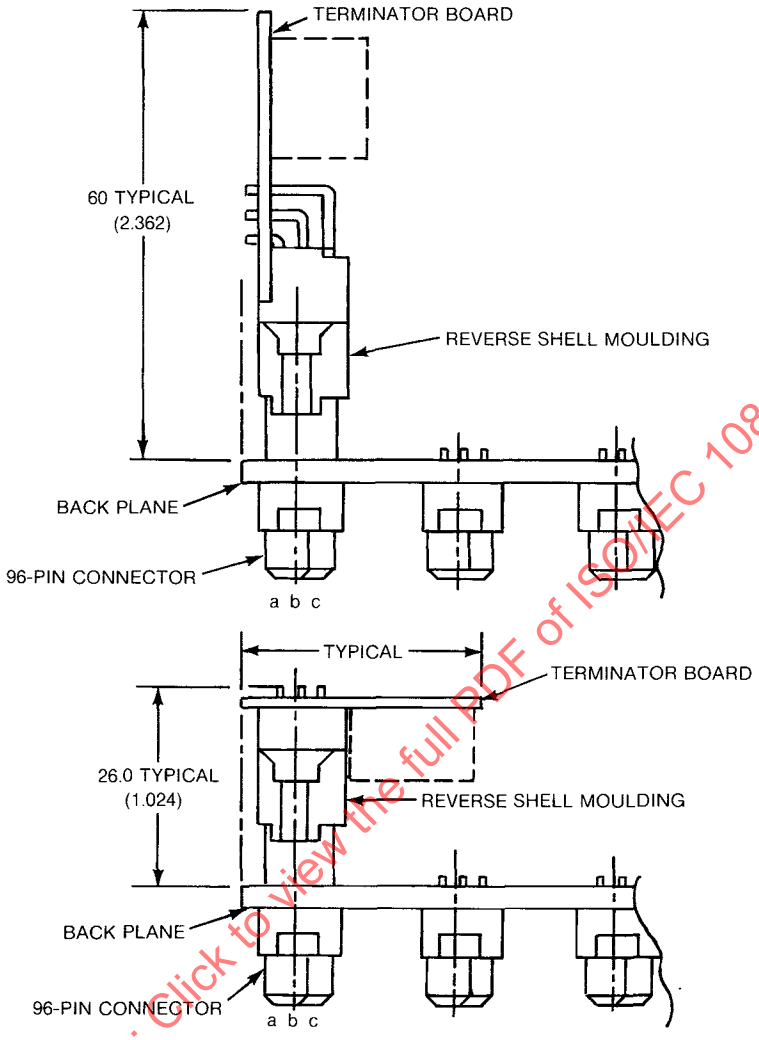


Figure 8.6-3—Off-board termination

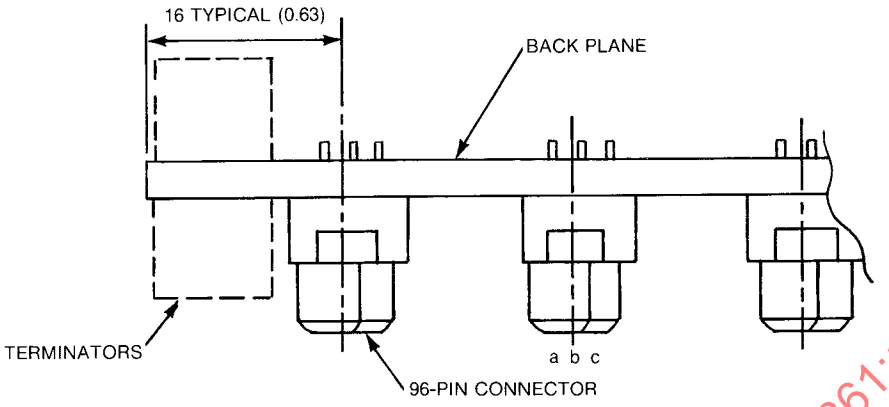
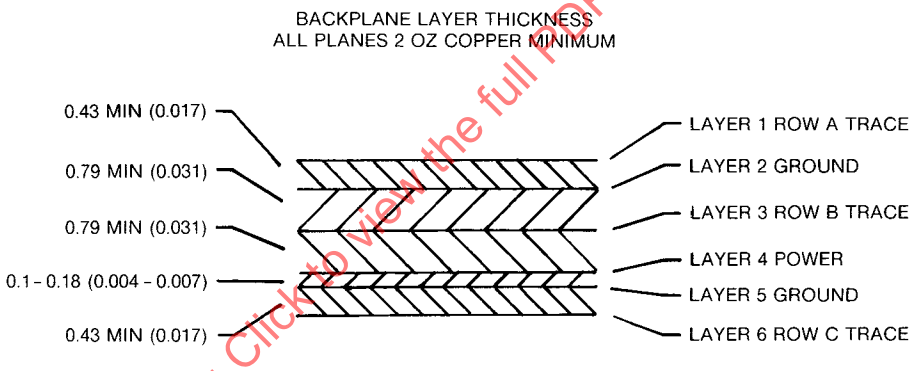


Figure 8.6-4 — On-board termination



NOTE: Slanted lines indicate dielectric material.

Figure 8.6-5 — PSB backplane layer thickness

9. IEEE 1296 System Interface specification

9.1 Overview

9.1.1 Scope

The IEEE 1296 System Interface specification provides information for system integrators and board implementors to ensure interoperability in a system constructed of implementations which select the various address spaces available on the PSB.

9.1.2 Object

The IEEE 1296 architecture defines four separate address spaces on the PSB. These are Interconnect space, I/O space, Memory space, and Message space. This System Interface specification describes the address spaces and defines how an agent can use an address space as a bus owner or replying agent.

9.1.3 Architecture overview

On the PSB, two system architectures are supported. The first, a memory-mode system, has all communication between agents via memory space or I/O space. The second architecture, a message-mode system, consists of agents that communicate only in the message space. Both architectures may be simultaneously supported in a system, and this is referred to as a dual-mode architecture.

Both architectures require support of interconnect space.

9.1.4 System model

The PSB is optimized for a system which is partitioned into modules with each module supporting a specific function. This is called functional partitioning. A functionally partitioned system is characterized by agents which have local intelligence and are optimized for some specific function.

In a functionally partitioned system, the role of an agent (or a collection of agents) is either that of a client or a server. A client issues commands to servers, and servers respond with information based on the request. Both clients and servers are functional units characterized by local intelligence and optimized for some specific function.

9.1.4.1 Memory-mode architecture

In a memory-mode system, the memory space provides the communication mechanism for clients and servers. This memory must be accessible from the PSB. Also clients and servers must support unsolicited messages without data in message space as both the bus owner and replying agent.

In the I/O space, the client must be able to perform I/O space operations as the bus owner. In memory space, the client must either be able to access global memory as a bus owner or contain memory that can be accessed by both the processing element on the client and other agents.

A server in the memory-mode system must support one or more of these three access interfaces: the I/O space replying agent interface; the message space interrupt message for both bus owner and replying agent; and memory space bus owner and/or replying agent interface. For some implementations of clients and servers, it may be necessary to install an agent to support memory space transfer operations for the clients and servers.

9.1.4.2 Message-mode architecture

For message-mode systems, clients and servers must support unsolicited messages with data in the message space as both bus owner and replying agent. Further, the clients and servers must be capable of solicited messages (as opposed to unsolicited) using a protocol between the agents which binds solicited messages to allow higher performance data transfer. The implementor of the agent(s) may extend the defined protocol.

9.1.4.3 Dual-mode architecture

Clients and servers support the required interfaces of the memory-mode and/or message-mode architectures. At least one client or server supports the required interfaces for both architectures to provide a communication path, if necessary, among all agents.

9.2 Interconnect space operation

9.2.1 Introduction

The purpose of interconnect space on the PSB is to allow flexible system configuration and diagnostic capability. Agents may use interconnect space for initialization and configuration. All agents must support interconnect address space on the PSB. Since interconnect space is provided for configuration and diagnostics, data transfer performance may be quite low. Therefore, it should not be used for other purposes by system integrators.

Table 9.2-1 gives an overview of the attributes available for an agent operating in interconnect address space.

Table 9.2-1 — Interconnect address space summary

- Supports only 8-bit data transfers.
- Supports both read and write data transfers.
- Supports selection of one of 512 registers within the agent.
- Supports only single transfer operations.
- Supports cardslot ID of 0 to 20 depending on the physical cardslot of the agent.
- Supports 16-bit interconnect addresses with 5-bits for cardslot ID, 9-bits for register number, and 2-bits that are always zero.
- Supports only a single replying agent for any transfer operation.

9.2.2 Interface model

Interconnect space support is required for memory-mode, dual-mode, and message-mode architectures. Each agent in the PSB must be able to decode interconnect operations on the bus. Agents compare the interconnect cardslot address to that assigned to the agent by the CSM.

9.2.3 Interconnect address format

The interconnect address consists of three fields: a cardslot ID, a sequence of up to 512 register numbers at that cardslot, and two bits that are always zero. Figure 9.2-1 shows the bit positions of these fields.

Each agent on the PSB is assigned a unique interconnect address by the CSM during system reset based on physical cardslot position. The PSB cardslot IDs are 0 through 20. The PSB cardslot ID of 0 always contains the agent with the active CSM functions.

Each register is an 8-bit quantity that is individually addressed by an interconnect space operation.

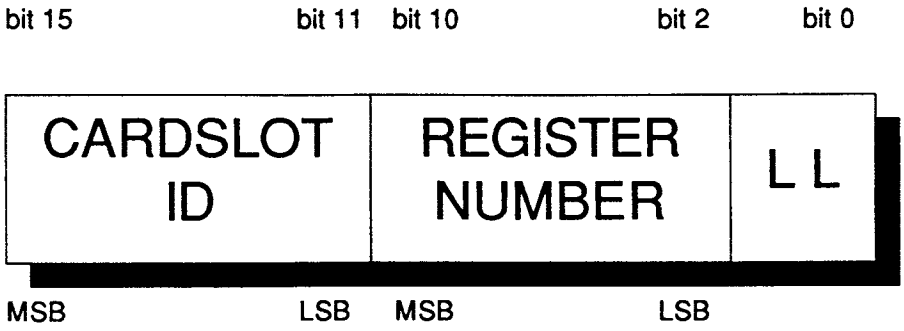


Figure 9.2-1—Interconnect address format

Support of registers 0 and 1 (the Vendor ID Registers) is required for all agents.

Figure 9.2-2 shows the allocation of registers within the interconnect address for an agent. Each agent in a IEEE 1296 system has a separate interconnect address and a separate set of interconnect registers. The format of interconnect registers of an agent is called the agent's interconnect template.

Register number	Bit position 7	6	5	4	3	2	1	0
Register 0	Vendor identification number (bits 7..0)							
Register 1	Vendor identification number (bits 15..8)							
Registers 2 through 511	Board-specific attributes							

Figure 9.2-2—Interconnect space format

9.2.4 Vendor identification registers

Minimum interconnect space support requires that each agent respond to accesses to registers 0 and 1. In these registers, the vendor ID is contained. Register number 0 contains the bits 7 through 0 of the vendor ID, and register number 1 contains bits 15 through 8.

All even vendor ID numbers indicate that the interconnect template structure of the agent does not comply with this document. Additional documentation is required to determine the interconnect template of the agent.

All odd vendor ID numbers indicate that the interconnect template structure of the agent complies with this document.

9.2.5 Compliant interconnect template structure

While the vendor ID does provide limited identification information, broader and more useful functions can be implemented. This document describes the use of interconnect space to provide support for agent identification, configuration, and diagnostics.

The structure of the compliant interconnect template is shown in figure 9.2-3. The header record provides information regarding the agent's product name and vendor-specific identification information beyond the minimal vendor ID requirements. Following the header record are some number of function records. Each function record is used for configuration of a special feature of the agent, for example, setting memory addresses for decoding on a memory agent. Each function record contains registers which have a defined format. Function records are added until all features have been described. A special function record (the end-of-template record) signals the end of the interconnect template of the agent.

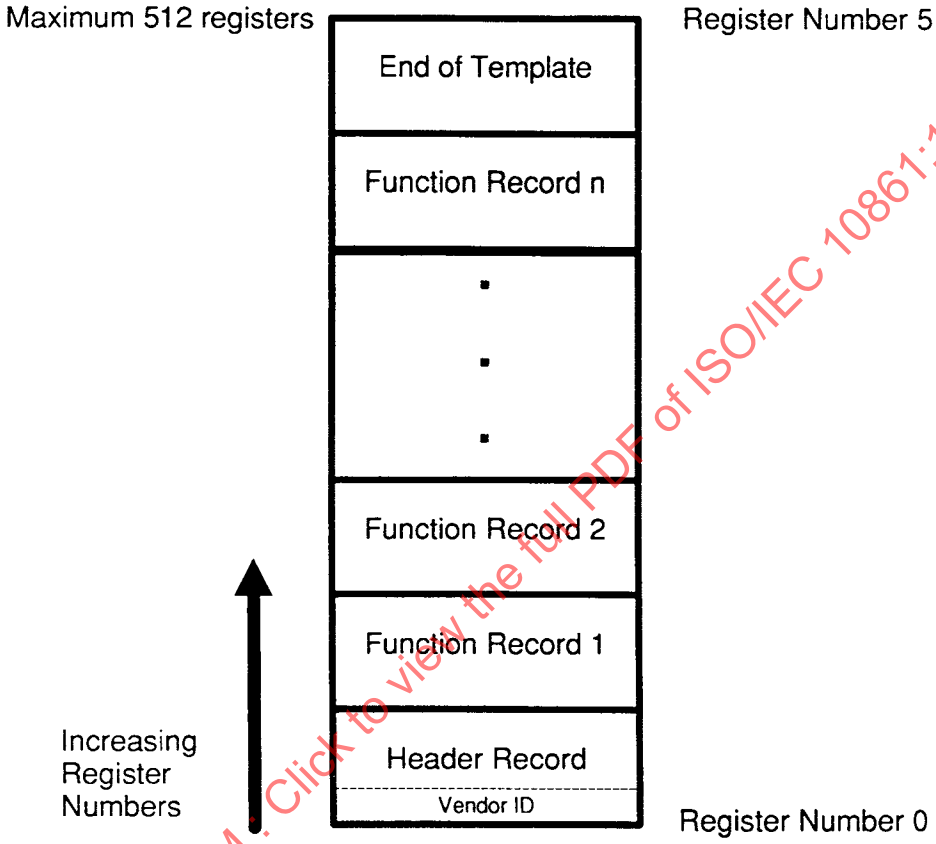


Figure 9.2-3 — Structure of the compliant interconnect template

Identical function records can be used many times within a system, on different agents or a single agent. For example, all agents with addressable memory in the system could use the same function record to set the address range for the memory operations. The registers within the function record support the various configuration options. Since these records can be used many times and their meaning is consistent, simple configuration software is possible. By providing software capability for configuration, the task of hardware configuration (jumper options, slot dependencies, and device reprogramming) is minimized.

Built into the definition of function records is a simple mechanism for searching for records. The "chained" structure of the function records is shown in figure 9.2-4. Within each function record, a register contains the length, in number of registers of the record. Searching for a particular function

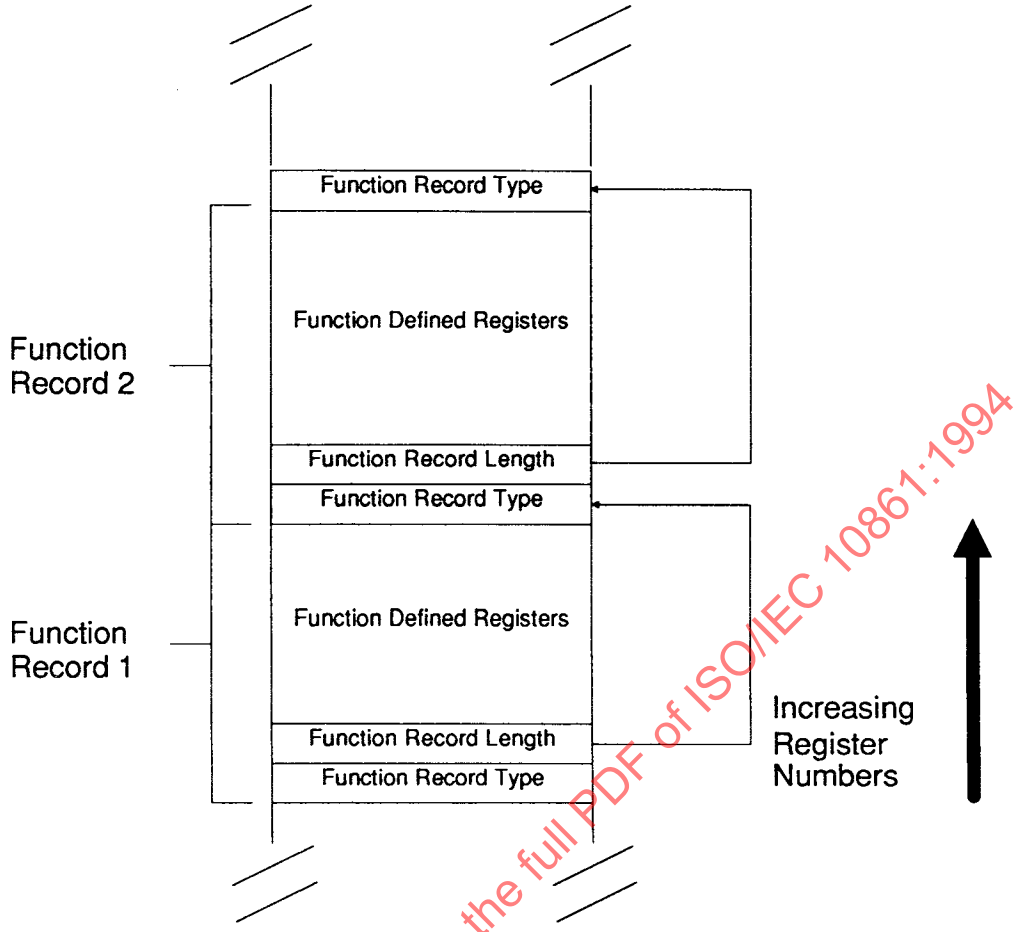


Figure 9.2-4—Function record chaining

record is accomplished by "chaining" through the records. This structure makes individual function records relocatable within the interconnect template.

9.2.5.1 Description of the header record

The header record of the interconnect template combines the identification and minimal support requirements. The format of the header record is shown in table 9.2-2.

Table 9.2-2—Header record format

Interconnect register number	Description
0..1	Vendor ID
2..11	Product code registers
12..31	Vendor defined

9.2.5.1.1 Product code registers

The product code registers in the header record contain the "name" of the agent. The registers contain up to 10 characters in the ASCII. If the product code contains less than 10 characters, the remaining

registers are filled with the ASCII null character (0). For example, if the product code of the agent is XYZ/123ES, then the product code register contents would be represented as:

PRODUCT Code = XYZ/123ES,0

9.2.5.1.2 Vendor-defined registers

The interconnect registers from 12 through 31 are reserved for additional identification requirements of the vendor.

9.2.5.2 Description of function records

Each function controlled by interconnect space is represented by a group of consecutive registers. This group of registers is called a function record.

In general, function records are represented as shown in figure 9.2-5. There are two dedicated registers in each function record, the record type and record length. The record type corresponds to the record function. Each different function record has a different record type. The record type indicates the format or template to be applied to the registers that follow. The second dedicated register in the function record is the record length. This register contains the number of registers used to describe this function not including the record type and record length. The record length register allows a simple method of searching for individual function records with the list.

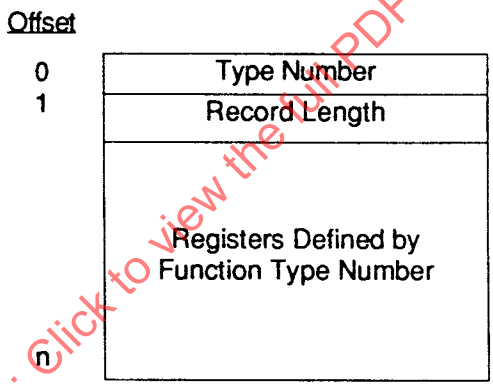


Figure 9.2-5—Function record format

There is one function record that does not have the two dedicated registers. This record is called the end-of-template record and consists of a single record type register. This record is placed at the end of the agent's interconnect template and indicates that interconnect accesses to registers beyond this point should not be performed.

Figure 9.2-6 shows the organization of an agent's interconnect template. Following the fixed length header record are function records. Any number of function records follow (up to the maximum number of 512 registers), with the record length register used for searching. At the end of the interconnect template is the special end-of-template record.

Within the interconnect template, the position of records is not order dependent. If multiple copies of the same function record exist to describe multiple identical resources on a single agent, the vendor must supply information to distinguish the function records. Individual function records may move within the interconnect template of an agent so long as the chained structure is preserved.

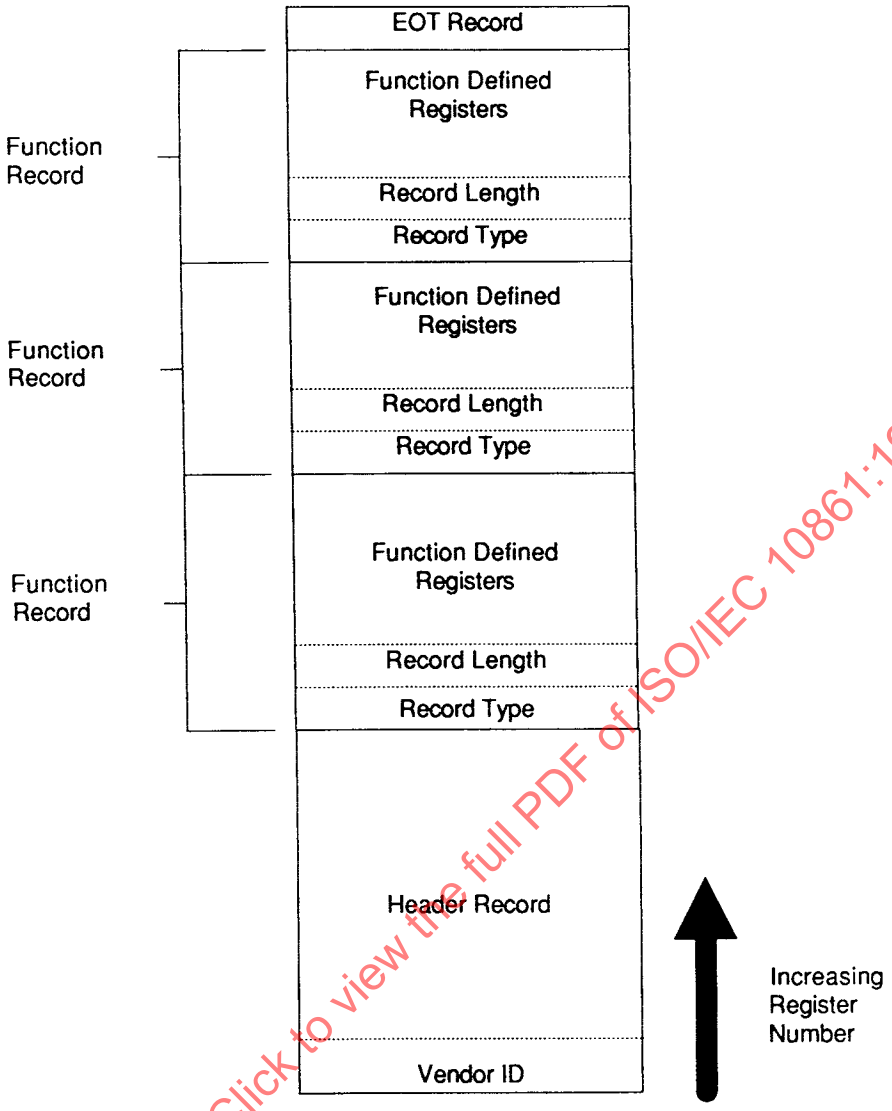


Figure 9.2-6—Interconnect template organization

To search for a particular function record, the first register after the header record is read. If the record type read from the agent matches the record type of the desired record, the record has been found. If the record type does not match, the next register, the record length, is read. When the current number is added to the record length, then incremented, it provides the register number for the record type of the next function record. This searching process continues until the record is found or the end-of-template record is reached.

9.2.5.3 Function record definitions

Table 9.2-3 shows the predefined function records. Except for the end-of-template record, agents are not required to support the listed function records. These definitions are suggested ways of implementing function records for common functions in order to provide a more consistent software interface across all modules.

Table 9.2-3—Predefined function records

Type number	Description
0	Extended record
8	CSM record
9	Time/date record
128	Memory space record
129	I/O space record
130	Message space record
131	Initialization record
255	End of template

All other function records are available for vendor definition. Each of the function records is discussed in the subsequent subclauses.

9.2.5.3.1 Extended record

The extended record can be used to increase the number of record types available in interconnect space. Instead of being limited to 256 record types, the use of the extended record allows the definition of an additional 65 536 interconnect records.

9.2.5.3.1.1 Recommended uses of the extended record

All of the 65 536 extended record types are reserved for use by individual implementations. This specification makes no attempt to standardize the use of extended record types.

9.2.5.3.1.2 Register format of extended record

The register format of the extended record is shown in table 9.2-4.

Table 9.2-4—Format of the extended record

Offset	Description	Value
0	Extended record type	0
1	Record length	*
2	Extended type (bits 7..0)	*
3	Extended type (bits 15..8)	*
4...	Defined by extended type	*

*Defined by implementor

9.2.5.3.1.3 Record length

The value of record length register in the extended record is determined by the implementor. The value is determined by the number of registers used for the extended record not including the record type and the record length registers.

9.2.5.3.1.4 Extended type

The extended type registers, at offset 2 and 3, are used to determine the record type for the extended record. The register at offset 2 is used for bits 7 through 0 and the register at offset 3 is used for bits 15 through 8 of the 16-bit extended type. This 16-bit number is used to determine the use of the remaining registers in the record.

9.2.5.3.2 CSM record

The CSM record allows control of those functions performed by the CSM for the PSB. Control of these functions permits testing of these features for increased system confidence.

9.2.5.3.2.1 Recommended uses of the CSM record

This record can be used on every agent that may become the CSM on the PSB bus. Only the agent installed in PSB cardslot 0 has the CSM functions enabled.

9.2.5.3.2.2 Register format of the CSM record

The format of the CSM record is shown in table 9.2-5.

Table 9.2-5—CSM record format

Offset	Description	Initial value
0	CSM record type	8
1	Record length	2
2	CSM command	0
3	Reserved	0

9.2.5.3.2.3 CSM command register

The CSM command register can be used to invoke functions of the CSM for the PSB. The bit fields in the CSM command register are described in the following list:

a) **Time out disable (bit 7).** Bit 7 of the CSM command register is used to disable the generation of the TIMOUT* signal on the PSB. The CSM asserts the TIMOUT* signal when a maximum time has expired for a data transfer operation on the PSB. When the value of bit 7 is 0, the generation of TIMOUT* by the CSM is enabled. When the value of bit 7 is 1, the generation of TIMOUT* is disabled.

b) **Reserved bits (bits 6..2).** Bits 6 through 2 of the CSM command register are reserved for future definition. When read from the register, the bits should be masked by the software. When written, the value of these bits must be 0.

c) **Reset generation (bits 1..0).** Bits 1 through 0 of the CSM command register are used to force a particular type of reset operation to occur on the PSB. Table 9.2-6 shows the programmed values and the corresponding reset operation performed.

d) **Reserved register.** The register at offset 3 within the CSM record is reserved for future definition. When read, a value of 0 is returned. If written, the contents to be written must have a value of 0.

Table 9.2-6—Reset codes for the CSM command register

Value	Description
0	No reset operation performed
1	Perform a cold reset operation
2	Perform a warm reset operation
3	Perform a recovery reset operation

9.2.5.3.3 Time/date record

The time/date record can be used for system time-keeping functions.