This is an ***Archive IEEE Standard.*** It has been withdrawn or superseded by a later version of the standard.

ARCHIVE STANDARDS MAY HAVE VALUE AS HISTORICAL DOCUMENTS, BUT THEY ARE NOT VALID OR APPROVED IEEE STANDARDS. THE IEEE SHALL NOT BE LIABLE FOR ANY DAMAGES RESULTING FROM THE SUBSCRIBER'S OR USERS' PRACTICE OF THIS UNAPPROVED, ARCHIVE IEEE STANDARD.

For more information about this or other current, revised or withdrawn standards, please consult the IEEE Standards Status report on-line at http://standards.ieee.org/db/status/ .

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

An American National Standard

IEEE Standard for Multiplexed High-Performance Bus Structure: VSB

Sponsor
Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society

Approved June 9, 1988
IEEE Standards Board

Approved May 19, 1989
American National Standards Institute

ISBN 1-55937-007-6

© Copyright 1989 by

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

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

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 free 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
345 East 47th Street
New York, NY 10017
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Foreword

(This Foreword is not a part of ANSI/IEEE Std 1096-1988, IEEE Standard for Multiplexed High-Performance Bus Structure: VSB.)

[The Foreword and Working Group list were not available at the time of publication.]

The following persons were on the balloting committee that approved this document for submission to the IEEE Standards Board:

Andrew Allison
Peter Ashenden
Matt Biewer
John Black
Jack Blevins
Rich Boberg
Paul Borrill
Bradley Brown
Clyde Camp
John Charlton
Steve Cooper
Randy Davis
J. (Bob) Davis
Jim Dunlay
Wayne Fischer
Jim Flournoy
Gordon Force
Martin Freeman
S. Ganesan
David Gustavson
Tom Harkaway
David James
Richard Kalish
Richard Karpinski
Doug Kraft
Glen Langdon
Bill LaPlant
Gerry Laws
Tom Leonard
Rollie Linser
Gary Nelson
Thomas Pittman
Shlomo Pri-Tal
P. Ragu
Richard Rawson
Michael Smolin
Robert Stewart
Michael Teener
Eike Waltz
Carl Warren
George White
Fritz Whittington
Thomas Wicklund
Andrew Wilson

When the IEEE Standards Board approved this standard on June 9, 1988, it had the following membership:

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

Arthur A. Blaisdell
Fletcher J. Buckley
James M. Daly
Stephen R. Dillon
Eugene P. Fogarty
Jay Forster*
Thomas L. Hannan
Kenneth D. Hendrix
Theodore W. Hissey, Jr.
John W. Horch
Jack M. Kinn
Frank D. Kirschner
Frank C. Kitzantides
Joseph L. Koepfinger*
Irving Kolodny
Edward Lohse
John E. May, Jr.
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Richard E. Mosher
L. John Rankine
Gary S. Robinson
Frank L. Rose
Helen M. Wood
Karl H. Zaininger
Donald W. Zipse

*Member Emeritus

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

CLAUSE PAGE

1. Introduction 1

1.1 VSB Specification Objectives 1
1.2 VSB System Elements 1
1.3 Specification Diagrams 6
1.4 Specification Terminology 6
1.5 Protocol Specification 7
1.6 References 8

2. VSB Data Transfer Bus 8

2.1 Introduction 8
2.2 Data Transfer Bus Lines 9
2.3 DTB Modules — Basic Description 13
2.4 Capabilities of MASTERS and SLAVES 15
2.5 Interaction Between MASTERS and SLAVES 27
2.6 Data Transfer Bus Timing Specifications 48

3. VSB Data Transfer Bus Arbitration 69

3.1 Introduction 69
3.2 Arbitration Bus Lines 70
3.3 Arbitration Modules—Basic Description 70
3.4 Capabilities of the REQUESTER 72
3.5 Interaction Between the MASTER, Its Associated REQUESTER and/or Its Associated ARBITER 86
3.6 Arbitration Bus Timing Specifications 88

4. Electrical Characteristics of VSB Boards 95

4.1 Introduction 95
4.2 Power Distribution 96
4.3 Bus Driving and Receiving Requirements 96
4.4 Signal Line Interconnection — Summary 105

5. VSB Backplane Specifications 106

5.1 Introduction 106
5.2 Backplane Physical Characteristics 106
5.3 Power Distribution 107
5.4 Backplane Electrical Characteristics 108
5.5 Signal Line Interconnection 113
5.6 VSB Pin Assignment 114

Annex A (informative) 116

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

IEEE Standard for Multiplexed High-Performance Bus Structure: VSB

1. Introduction

1.1 VSB Specification Objectives

The Subsystem Bus of the IEC 821 BUS (IEC 821-1987 [2],[1] henceforth referred to as the VSB) is a local subsystem extension bus. It allows a processor board to access additional memory and I/O over a local bus, removing traffic from the global bus and improving the total throughput of the system. The system has been conceived with the following objectives:

1) To improve the performance of multiprocessor systems by allowing the design of local subsystems.
2) To specify the electrical characteristics required to design boards that will reliably transfer data over the VSB.
3) To specify the mechanical requirements to be compatible with VSB systems.
4) To specify protocols that precisely define the interaction between the VSB and devices interfaced to it.
5) To provide terminology and definitions that describe VSB protocols.

1.2 VSB System Elements

1.2.1 Basic Definitions

The structure of the VSB can be described from two points of view: its mechanical structure and its functional structure.

Because the primary use of the VSB is as a secondary bus, there are no mechanical specifications of VSB board level, and/or box level products. It is assumed that products that include the VSB have been designed to comply with the mechanical specifications of the global system bus. And therefore, the VSB specification only describes the physical dimensions of the backplane.

The functional specifications of the VSB describe how the bus works, what functional modules participate in its various operations, and the rules that govern their behavior. This section provides informal definitions for the basic terms used to describe both the mechanical and functional structure of the VSB.

[1]The numbers in brackets refer to those of the references listed in 1.6.

Copyright © 1989 IEEE All Rights Reserved

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

1.2.1.1 Physical Structure

BOARD

A printed circuit (PC) board, its collection of electronic components, and at least one 96-pin connector.

VSB BACKPLANE

An assembly that includes a printed circuit (PC) board and 96-pin connectors. The backplane buses the 64 pins on the two outer rows of the VSB connectors, providing the signal paths needed for VSB operation.

SLOT

A position where a board can be inserted into a backplane. Each VSB slot provides at least one 96-pin connector.

SUBRACK

A rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards are not disengaged from the backplane due to vibration.

1.2.1.2 Functional Structure

Figure 1.1 shows a block diagram of the functional modules and subbuses defined by the VSB specification.

BACKPLANE INTERFACE LOGIC

Special interface logic that takes into account the characteristics of the backplane. The VSB specification prescribes certain requirements for the design of this logic, which takes into account the signal line impedance, propagation times, termination values, the maximum length of the backplane, and the number of slots allowed.

FUNCTIONAL MODULE

A collection of electronic circuitry that resides on one board and works to accomplish a specific task. Functional modules are used as a vehicle for discussing bus protocols, and should not be considered to constrain the design of actual logic.

DATA TRANSFER BUS

One of the two subbuses defined in the VSB specification. It allows MASTERS to direct the transfer of binary data to and from SLAVES. (The VSB Data Transfer Bus is often abbreviated DTB.) The DTB contains 32 multiplexed address/data lines and the associated control signals that are required to execute cycles on the VSB.

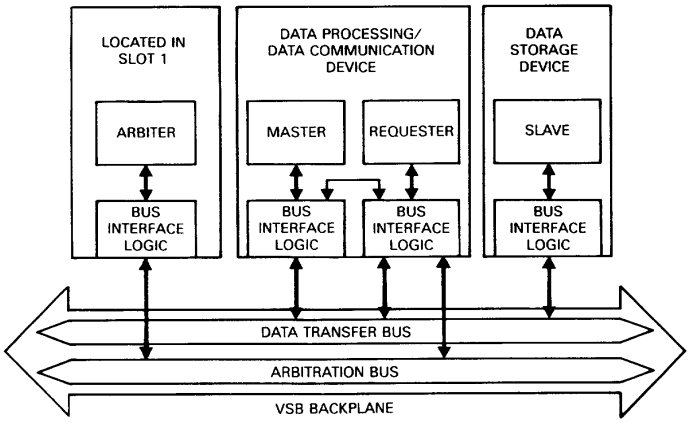


Figure 1.1—Functional Modules and Subbuses Defined by the VSB Specification

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

MASTER

A functional module that initiates bus cycles in order to transfer data between itself and VSB SLAVES. The MASTER that is currently in control of the DTB is referred to as the *active MASTER.*

SLAVE

A functional module that detects bus cycles initiated by the active MASTER and, when those cycles select it, transfers data between itself and the MASTER. The VSB specification defines a mechanism through which any number of SLAVES can participate in a bus cycle. All SLAVES that are selected by the cycle are called *selected SLAVES.* However, only one of the selected SLAVES is allowed to acknowledge the cycle and respond to the active MASTER. This SLAVE is called the *responding SLAVE.* All other selected SLAVES are called *participating SLAVES.* SLAVES that are not selected by the cycle are called *idle SLAVES.*

VSB ARBITRATION BUS

The second of the two subbuses defined in the VSB specification. It allows ARBITER modules and/or REQUESTER modules to coordinate the use of the DTB by VSB MASTERS. The VSB defines two arbitration methods — a Serial arbitration method and a Parallel arbitration method.

REQUESTER

A functional module that resides on the same board as a MASTER and requests use of the DTB whenever its MASTER needs it. When implementing Serial arbitration, after requesting use of the DTB, the REQUESTER waits for the bus to be granted to it by the ARBITER.

In the Parallel arbitration method, the REQUESTER that is associated with the active MASTER initiates an ARBITRATION cycle. This ARBITRATION cycle is used to determine which MASTER will be granted use of the DTB. The VSB specification calls the REQUESTER that is associated with the MASTER the *active REQUESTER.* REQUESTERS that have a bus request pending and that participate in an ARBITRATION cycle are called *contending REQUESTERS.*

ARBITER

When implementing the Serial arbitration method, the ARBITER module accepts requests for the DTB from REQUESTERS and grants control of the DTB to one REQUESTER at a time. There is one and only one active ARBITER in the Serial arbitration scheme, and it is always located in slot 1. An ARBITER is not required in the Parallel arbitration method.

DAISY CHAIN

A special type of signal line that is used to propagate bus grants from board to board, starting with the board installed in the first slot and ending with the one installed in the last slot.

GEOGRAPHICAL ADDRESSING

A scheme wherein each slot in the backplane is assigned a unique ad dress. This address can be read by the board that is installed in the slot. The VSB specification defines the use of the geographical address for two purposes: (1) it forms part of the INTERRUPT ID used during an INTERRUPT-ACKNOWLEDGE cycle and, (2) it forms part of the ARBITRATION ID used during a Parallel ARBITRATION cycle. The geographical address can also be used to set global board variables such as the base address of a memory board.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

1.2.1.3 VSB Cycles

VSB BUS CYCLE

A sequence of level transitions on the signal lines of the DTB that results in the transfer of an address and (in most cases) data between the active MASTER and selected SLAVES. The protocols of the VSB are fully asynchronous. The active MASTER asserts a strobe signal indicating that a cycle is in progress. The responding SLAVE acknowledges the MASTER's signal However, the responding SLAVE can delay its acknowledgment for as long as it needs. The DTB cycle is generally divided into three phases: an address broadcast, zero or more data transfers, and then cycle termination.

ADDRESS BROADCAST

The phase of a bus cycle that selects one SLAVE as the responding SLAVE and zero or more SLAVES as participating SLAVES. During the address broadcast the active MASTER broadcasts the addressing information and then asserts an address strobe. After the SLAVES acknowledge the address broadcast, the MASTER terminates the address broadcast.

DATA TRANSFER

The phase of a cycle during which data are transferred between the MASTER and the selected SLAVES. It starts when the active MASTER asserts the data strobe and ends after the responding SLAVE acknowledges the transfer and all participating SLAVES indicate that they are ready to participate in a new cycle.

CYCLE TERMINATION

The phase of a cycle during which the MASTER terminates the cycle and SLAVES acknowledge this termination by establishing the intercycle state of bus signals.

DYNAMIC BUS SIZING

The ability of some microprocessors to adjust the number and the size of data transfers to the amount of data that the responding board can access in one transfer. During the address broadcast portion of the cycle, the SLAVE informs the MASTER how many data lines it actually drives or receives. This information is made available to on-board logic that can then adjust the amount of data that it accesses during the data transfer to the capabilities of the SLAVE.

DATA BROADCAST

A broadcast operation is one wherein participating SLAVES capture the data that are placed on the data lines by the active MASTER during a write cycle.

DATA BROADCALL

A broadcall operation is one wherein participating SLAVES capture the data that are placed on the data lines by the responding SLAVE during a read cycle.

SINGLE-TRANSFER READ CYCLE

A cycle that is used to transfer 1, 2, 3, or 4 bytes from the responding SLAVE to the active MASTER, and possibly to participating SLAVES. The cycle begins when the active MASTER broadcasts the addressing information on the address/data lines. Each SLAVE checks the address to see if it is to respond to the cycle. If so, it acknowledges the address and retrieves the data from its internal storage. When the MASTER releases the address/data lines, the responding SLAVE places its data on them and acknowledges the transfer. The MASTER as well as participating SLAVES capture the data. After all selected SLAVES signal their agreement the MASTER terminates the cycle.

Copyright © 1989 IEEE All Rights Reserved
Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

SINGLE-TRANSFER WRITE CYCLE

A cycle that is used to transfer 1, 2, 3, or 4 bytes from the active MASTER to the selected SLAVES(S). The cycle begins when the MASTER broadcasts the addressing information on the address/data lines. Each SLAVE checks the address to see if it is to participate in the cycle. The responding SLAVE acknowledges the address broadcast. The MASTER then switches the address/ data lines to carry data, and places its data on the bus. The selected SLAVES can then store the data. The responding SLAVE acknowledges the transfer. After all selected SLAVES signal their agreement the MASTER terminates the cycle.

BLOCK-TRANSFER READ CYCLE

A DTB cycle that is used to transfer a block of bytes from the responding SLAVE to the active MASTER, and possibly to participating SLAVES. This transfer is done using a number of 1, 2, or 4-byte data transfers. It differs from a series of SINGLE-TRANSFER read cycles in that the MASTER broadcasts the address only once, at the beginning of the cycle. It is the responsibility of the selected SLAVES to control the address for each subsequent data transfer.

BLOCK-TRANSFER WRITE CYCLE

A DTB cycle that is used to transfer a block of bytes from the active MASTER to the selected SLAVES. This transfer is done using a series of 1, 2, or 4-byte data transfers. It differs from a series of SINGLE-TRANSFER write cycles in that the MASTER broadcasts the address only once, at the beginning of the cycle. It is the responsibility of the selected SLAVES to control the address for each subsequent data transfer.

INDIVISIBLE-ACCESS CYCLE

A DTB cycle that is used to access SLAVE locations indivisibly and without permitting any other MASTER to access these locations until the operation is complete.

ADDRESS-ONLY CYCLE

The DTB cycle that consists of an address broadcast, but no data transfer. The active MASTER terminates the cycle after the SLAVES acknowledge the address broadcast.

INTERRUPT-ACKNOWLEDGE CYCLE

A DTB cycle that is initiated by a MASTER in response to an interrupt request from a SLAVE. An INTERRUPT-ACKNOWLEDGE cycle involves two types of SLAVES. *Contending SLAVES* have an interrupt request pending and participate in the cycle. The *responding SLAVE* is the one that transfers its STATUS/ ID information to the MASTER. During the INTERRUPT-ACKNOWLEDGE cycle, all contending SLAVES drive an INTERRUPT ID on the bus. This ID is a combination of the geographical address of the board that is supplied by the backplane slot, and a priority code that is supplied by user defined on-board logic. The INTERRUPT ID is used to determine which of the contending SLAVES will respond to the cycle.

ARBITRATION CYCLE

A cycle that is initiated by the active REQUESTER in response to a bus request, after its associated active MASTER no longer needs the bus. This cycle is used to select the MASTER that will be granted use of the DTB. If the active REQUESTER detects a request for the bus, and if its associated MASTER no longer needs the bus, it initiates an ARBITRATION cycle. During the ARBITRATION cycle, all contending REQUESTERS drive an ARBITRATION ID on the bus. This ID is a combination of the geographical address of the board that is supplied by the backplane slot, and a priority code that is supplied by user defined on-board logic. At the end of the ARBITRATION cycle one of the contending REQUESTERS becomes the active REQUESTER.

Copyright © 1989 IEEE All Rights Reserved

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

1.3 Specification Diagrams

Three types of diagrams are used to help define and describe the operation of the VSB:

1) *Block diagrams* show the signal line interconnect requirements of the functional modules defined by the VSB specification.
2) *Flow diagrams* show the stream of events as they would occur during a VSB operation. The events are stated in words and sequentially describe the interaction between two or more functional modules. The VSB specification describes in detail the behavior of the various functional modules. It discusses how a module responds to a signal without saying where the signal came from. Because of this, a protocol specification does not give the reader a complete picture of what happens over the bus. Flow diagrams are used to help the reader overcome this difficulty.
3) *Timing diagrams* show the timing relationships between signal transitions. The timing parameters have minimum and/or maximum limits associated with them. Some of these timing parameters specify the behavior of the backplane interface logic, while others specify the behavior of the functional modules.

1.4 Specification Terminology

To avoid confusion, and to make very clear what the requirements for compliance are, many of the paragraphs in the VSB specification are labeled with sequentially numbered keywords that indicate the type of information they contain. The keywords are:

RULE
RECOMMENDATION
SUGGESTION
PERMISSION
OBSERVATION

RULE chapter.number:

Rules form the basic framework of the VSB specification. They are sometimes expressed in text form and sometimes in the form of figures or tables. Rules are characterized by an imperative style. The uppercase words MUST and MUST NOT are reserved exclusively for stating rules in this document and are not used for any other purose.

RECOMMENDATION chapter. number:

Wherever a recommendation appears, designers would be wise to take the advice given. Doing otherwise might result in some awkward problem or poor performance. While the VSB has been designed to support high performance systems, it is possible to design a VSB system that complies with all the rules, but has poor performance. In many cases, a designer needs a certain level of experience with VSB in order to design boards that deliver top performance. Recommendations found in this document are based on this kind of experience, and are provided to designers to speed their traversal of the learning curve.

SUGGESTION chapter.number:

A suggestion in the VSB specification contains advice that is helpful but not vital. The reader is encouraged to consider the advice before discarding it. Some decisions made while designing VSB boards are difficult until experience has been gained with the VSB. Suggestions are included to help a designer who has not yet gained this experience. Some suggestions have to do with designing boards that can be easily reconfigured for operation with other boards, or with designing boards that are easier to debug.

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

PERMISSION chapter.number:

In some cases a VSB rule does not specifically prohibit a certain design approach. However, the reader might be left wondering whether or not that approach violates the spirit of the rule, or whether or not it might lead to some subtle problem. Permissions reassure the reader that a certain approach is acceptable, and will cause no problems. The upper case word MAY is reserved exclusively for stating permissions in this document and is not used for any other purpose.

OBSERVATION chapter.number:

Observations do not offer any specific advice. They usually follow naturally from what has just been discussed. They spell out the implications of certain VSB requirements and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain requirements, so that the reader understands why they are needed.

1.4.1 Signal Line States

The protocols of the VSB are described in terms of levels and transitions on bus lines. A signal line is always assumed to be in one of two levels, high or low, or in transition between these two levels.

On *level significant* signal lines, the TTL, voltage level represents meaningful information. Signal lines that are described as level significant can be either *high* or *low*. Whenever the term high is used, it refers to a high TTL voltage level. The term low refers to a low TTL voltage level.

Edge significant signal lines mark an event when they make a transition between the two TTL voltage levels. There are two possible transitions that can appear on a signal line. A *rising edge* is the time during which a signal makes its transition from a low level to a high level. The *falling edge* is the time during which a signal makes its transition from a high level to a low level.

The rise and fall times of bus drivers are the result of a complex set of interactions involving the impedance of the backplane's signal lines, the terminations and length of the signal lines, the source impedance of the drivers, and the capacitive loading of the signal lines. The VSB specification does not specify rise and fall times. Instead, it specifies the electrical characteristics for drivers and receivers. It also specifies all signal timing requirements, taking into account the worst-case bus loading and the affect it has on the propagation delay times of these drivers. If VSB designers follow these timing requirements, then their boards will operate reliably with other VSB compatible boards under worst-case conditions.

1.4.2 Use of the Asterisk (*)

Some signal names have an asterisk suffix (*) to help define their usage. The meaning of the asterisk is as follows:

1) An asterisk (*) following the name of a signal that is level significant denotes that the signal is true or valid when the signal is low.
2) An asterisk (*) following the name of a signal that is edge significant denotes that the falling edge is of greater significance in the protocol than the rising edge.

1.5 Protocol Specification

The primary protocol used in the VSB system is a closed loop protocol on interlocked bus lines. Each interlocked signal has a source module and is one or more destination modules. It is sent from a specific module and is acknowledged by the receiving module(s). An interlocked relationship exists between the sending and the receiving modules until the signal is properly acknowledged. For example, a MASTER asserts the data strobe that is handshaked later with an acknowledge signal (no time limit is prescribed by the VSB specification). The MASTER does not remove the data strobe until all SLAVES have acknowledged the data transfer.

The signal lines used on the VSB may be driven by different modules at different times. They are driven with drivers that can be turned on and off at each board. It is very important that their turn-on and turn-off times be carefully controlled to prevent two drivers from attempting to drive the same signal line to different levels. A special notation in the timing diagrams is used to specify their turn-on and turn-off times. It is shown in Fig 1.2.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

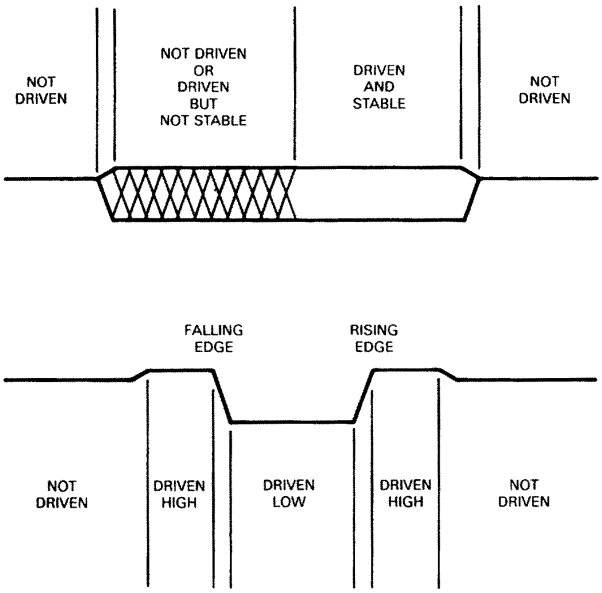


Figure 1.2—Signal Timing Notation

1.6 References

The following publications shall be used in conjunction with this standard.

[1] IEC603-2-1980, Part 2: Two-Part Connectors for Printed Boards, for Basic Grid of 2.54 mm (0.1 in), with Common Mounting Features.[2]

[2] IEC821-1987, IEC821 Bus — Microprocessor System for 1 to 4 Byte Data.

2. VSB Data Transfer Bus

2.1 Introduction

The VSB includes a high speed asynchronous multiplexed Data Transfer Bus (DTB). Figure 2.1 shows a typical VSB system including all the functional modules of the DTB. MASTERS use the DTB to select byte locations provided by SLAVES and to transfer data to or from those locations. Some MASTERS and SLAVES use all of the DTB lines, while others use only a subset.

After a MASTER initiates a data transfer cycle, it waits for the responding SLAVE to acknowledge the transfer before finishing the cycle. The asynchronous transfer protocols of the VSB allow a SLAVE to take as long as it needs to respond.

[2]IEC publications are available in the US from the American National Standards Institute, Sales Department, 1430 Broadway, New York, NY 10018.

Copyright © 1989 IEEE All Rights Reserved
Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

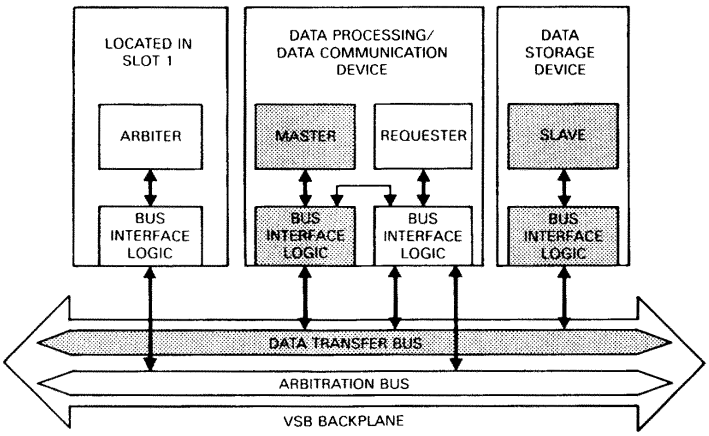


Figure 2.1—Data Transfer Bus Functional Block Diagram

2.2 Data Transfer Bus Lines

The VSB Data Transfer Bus lines can be grouped into three catgories:

(1) Addressing lines:	AD00-AD31	— Address/Data 00-31
	SPACE0	— SPACE select 0
	SPACE1	— SPACE select 1
	SIZE0	— SIZE request 0
	SIZE1	— SIZE request 1
	ASACK0*	— Address/Size ACKnowledge 0
	ASACK1*	— Address/Size ACKnowledge 1
	GA0-GA2	— Geographical Addressing
(2) Data lines:	AD00-AD31	— Address/Data 00-31
(3) Control lines:	PAS*	— Physical Address Strobe
	AC	— Address decode Complete
	WR*	— WRite
	LOCK*	— LOCK
	WAIT*	— WAIT
	ASACK0*	— Address/Size ACKnowledge 0
	ASACK1*	— Address/Size ACKnowledge 1
	DS*	— Data Strobe
	ACK*	— data ACKnowledge
	ERR*	— data ERRor
	IRQ*	— Interrupt ReQuest
	CACHE*	— CACHEable

This section provides a general description of the Data Transfer Bus lines to familiarize the reader with the function. The function of some of the lines might vary during INTERRUPT-ACKNOWLEDGE cycles, described in section 2.5.4. In addition, some the the DTB lines are used in the parallel arbitration method, as described in Section 3.

2.2.1 Addressing Lines

The active MASTER drives the addressing lines during the address broadcast phase of the cycle to select the SLAVE that will respond to, as well as SLAVES that will participate in, the data transfer phase.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

2.2.1.1 AD00-AD31

During the address broadcast phase of a cycle, the active MASTER drives address lines AD00-AD31 with the address of the byte location(s) that are to be accessed during the data transfer phase of the cycle.

The address of a 4-byte group is determined by the level of address lines AD02-AD31. The address of byte locations within a 4-byte group differs only by the level of the two least significant bits of the address lines, that is, address lines AD00 and AD01. Byte locations are categorized as BYTE(0), BYTE(l), BYTE(2), and BYTE(3) according to the state of AD00 and AD01 in their address, as defined in section 2.5.1.2, Table 2.11.

OBSERVATION 2.1:

When access to more than one byte location is requested in a cycle, the address on address lines AD00-AD31 is the lowest among the addresses of the byte locations.

2.2.1.2 SPACE0-SPACE1

The active MASTER drives SPACE0 and SPACE1 with a space code. It is used to select one of three address spaces, or to initiate an INTERRUPT-ACKNOWLEDGE or an ARBITRATION cycle. The VSB defines three separate address spaces of 4 Gbyte each. These three address spaces are the System Address Space, the I/O Address Space, and the Alternate Address Space.

2.2.1.3 SIZE0-SIZE1

During the address broadcast phase, the active MASTER drives SIZE0-SIZE1 with a size code. This size code indicates the number of byte location(s) that the MASTER wishes to access during the data transfer phase. The VSB specification defines four sizes of data transfer: Single-Byte, Double-Byte, Triple-Byte, and Quad-Byte.

During *Single-Byte transfers*, the MASTER requests access to one byte location.

During *Double-Byte transfers*, the MASTER requests access to two consecutive byte locations.

During *Triple-Byte transfers*, the MASTER requests access to three consecutive byte locations.

During *Quad-Byte transfers*, the MASTER requests access to four consecutive byte locations.

2.2.1.4 ASACK0*-ASACK1*

ASACK0* and ASACK1* serve a dual function:

1) The responding SLAVE drives its size code on these lines. This SLAVE size code informs the active MASTER what is the maximum number of byte locations that the responding SLAVE can access in a single transfer. The VSB specification defines three SLAVE sizes: Single-Byte, Double-Byte, and Quad-Byte SLAVES.
   A *Single-Byte SLAVE* responds to data transfer requests by accessing only one byte location during each transfer.
   A *Double-Byte SLAVE* responds to data transfer requests by accessing either one or two byte locations during each data transfer.
   A *Quad-Byte SLAVE* responds to data transfer requests by_accessing either one, two, three, or four byte locations during each data transfer.
2) The first falling edge of either ASACK0* or ASACK1* informs the active MASTER that the responding SLAVE has finished decoding the address and that it is ready to start the data transfer phase.

Copyright © 1989 IEEE All Rights Reserved
Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

2.2.1.5 GA0-GA2

In the VSB system, each slot in the backplane is assigned a unique address. This address is set by the voltage levels on GA0-GA2. When the board is installed in the backplane slot, it uses the levels of GA0-GA2 to set part of the INTERRUPT ID that it uses during INTERRUPT-ACKNOWLEDGE cycles. (In addition, as described in Section 3, GA0-GA2 set part of the ARBITRATION ID used during parallel ARBITRATION cycles.) Section 5, 5.5.3, describes how the voltage levels of the geographical addressing bits in each slot are set.

2.2.2 Data Lines AD00-AD31

Data lines AD00-AD31 are used to carry the data between the active MASTER and VSB SLAVES. During read cycles, the responding SLAVE retrieves data from its internal storage and drives it on the data lines, and the active MASTER captures the data from data lines AD00-AD31. During write cycles, the active MASTER drives data lines AD00-AD31 with valid data. The responding SLAVE captures this data and stores it in its internal storage. Participating SLAVES might also capture the data transferred during read or write cycles.

2.2.3 Control Lines

The control lines coordinate the transfer of the addressing information from the active MASTER to VSB SLAVES and of the data between the active MASTER and the selected SLAVES.

2.2.3.1 PAS*

The active MASTER drives PAS* low to start the address broadcast phase of VSB cycles and maintains it low for the duration of the cycle. VSB SLAVES start the address decode sequence when they detect a falling edge on PAS*. The responding SLAVE drives either ASACK0* and/or ASACK1* low. An interlocked relationship exists between the active MASTER and the responding SLAVE for as long as the MASTER drives PAS* low and the SLAVE drives either ASACK0* and/or ASACK1* low. This interlocked relationship ends when the active MASTER allows PAS* to go high, and after the responding SLAVE releases ASACK0*-ASACK1* to high.

2.2.3.2 AC

The AC line serves a dual function:

1) In conjunction with WAIT*, it provides VSB SLAVES with a mechanism to prolong the address broadcast phase to fit their address decoding requirements. When the active MASTER detects a low level on WAIT*, it is not allowed to complete the address broadcast phase and start the data transfer phase until it detects a high level on AC.
2) All SLAVES release AC to high in response to the address broadcast phase. The responding SLAVE drives AC low after it has accessed the last byte location it can access without requiring a new address from the active MASTER. When the active MASTER detects a low level on AC before it receives ACK* low, it terminates the cycle.

2.2.3.3 WR*

The active MASTER drives the WR* line during the address broadcast phase to inform SLAVES of the direction of the data transfer. The active MASTER drives WR* high when it retrieves data from byte locations provided by the responding SLAVE. The active MASTER drives WR* low to store data in byte locations provided by the responding SLAVE.

2.2.3.4 LOCK*

MASTERS drive the LOCK* line low to initiate INDIVISIBLE-ACCESS cycles. Some VSB SLAVES allow access to the byte locations they provide from more than one port; for example, their memory might be accessible over the

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

global system bus. Such SLAVES will typically lock out access from other ports to the byte locations that they provide for the duration of the INDIVISIBLE-ACCESS cycle. This ensures the indivisibility of the VSB cycle.

2.2.3.5 DS*

The active MASTER drives DS* low to indicate to the selected SLAVES that the address broadcast phase is complete and that the data transfer phase is in progress.

2.2.3.6 WAIT*

WAIT* serves a dual function:

1) In conjunction with AC, it provides VSB SLAVES with a mechanism to prolong the address broadcast phase to fit their address decoding requirements. When the active MASTER detects a low level on WAIT*, it is not allowed to complete the address broadcast phase and start the data transfer phase until it detects a high level on AC.
2) Participating SLAVES release WAIT* to high to indicate that they are done with the current data transfer and are ready to participate in a new one. This allows the active MASTER to perform broadcast and broadcall operations. During a broadcast operation, the data that the active MASTER drives on the data lines are captured by participating SLAVES. During a broadcall operation, the data that the responding SLAVE drives on the data lines are captured by participating SLAVES.
   For example, after caching a data item, the system contains two copies of it, one in global VSB memory and one in private cache memory. To ensure that these two copies are identical, the cache controller can monitor all VSB cycles, and either invalidate the cached copy or update it when it detects a write cycle to the global copy of the data. However, this operation should be completed before the transfer that caused it is terminated, so that the cache controller is ready to monitor a new one. This requirement is handled by the WAIT* line, which can be maintained low by such a cache controller, to prevent the active MASTER from terminating the current transfer until the cache has been updated.

2.2.3.7 ACK*

During write cycles, the responding SLAVE drives the ACK* line low to inform the active MASTER that it has captured the data on AD00-AD31. During read cycles the responding SLAVE drives ACK* low to inform the MASTER that valid data has been placed on some or all of AD00-AD31. The active MASTER interprets the falling edge of ACK* as a signal that the responding SLAVE has completed its participation in the data transfer portion of the cycle.

2.2.3.8 ERR*

The responding SLAVE drives ERR* low to inform the active MASTER that an error (for example, a parity error) has been encountered during the data transfer cycle. The active MASTER interprets the falling edge of ERR* as a signal that the responding SLAVE has completed its participation in the data transfer portion of the cycle.

2.2.3.9 IRQ*

The IRQ* line is used by SLAVES to request an interrupt from a MASTER. The MASTER responds to an interrupt request by one of two ways:

1) It might initiate an INTERRUPT-ACKNOWLEDGE cycle. SLAVES that have an interrupt request pending participate in the INTERRUPT-ACKNOWLEDGE cycle to determine the one SLAVE whose interrupt request will be serviced. The active MASTER then reads STATUS/ID information from this SLAVE.
2) It might initiate a VSB read cycle, polling the various devices for STATUS/ID information until a device that has an interrupt request pending is found.

Copyright © 1989 IEEE All Rights Reserved
IHS Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

2.2.3.10 CACHE*

The responding SLAVE drives CACHE* to indicate whether the data transferred are cacheable. For example, a memory board might drive the CACHE* line low when it responds to a cycle. The active MASTER, as well as participating SLAVES can then cache the data. On the other hand, an I/O controller does not drive CACHE* low during the cycle, preventing the MASTER from caching data that are read from a status register.

2.3 DTB Modules — Basic Description

Figures 2.2 and 2.3 provide block diagrams for the two types of DTB functional modules: MASTER and SLAVE.

RULE 2.1:

Output signal lines shown with solid lines in Figs 2.2 and 2.3 MUST be driven by the module, unless it would always drive them high.

OBSERVATION 2.2:

IF an output signal line is not driven,

THEN terminators ensure that it is high.

RULE 2.2:

Input signal lines shown with solid lines in Figs 2.2 and 2.3 MUST be monitored and responded to in the appropriate fashion.

OBSERVATION 2.3:

RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figs 2.2 and 2.3 are given in Tables 2.1 and 2.2.

2.3.1 MASTER

The block diagram of the MASTER is shown in Fig 2.2. The dotted lines in the diagram show signals whose use varies among the various types of MASTERS. Table 2.1 specifies how the various types of MASTERS use these lines.

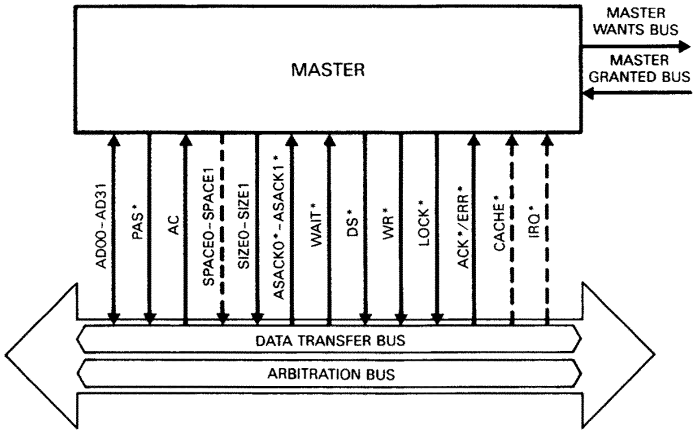


Figure 2.2—Block Diagram: MASTER

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.1—RULES and PERMISSIONS That Specify the Use of the Dotted Lines by the Various Types of MASTERS

Type of MASTER	Use of Dotted Lines
IHP	MUST monitor IRQ*
IHV	MUST monitor IRQ*
ALL MASTERS	MAY or MAY not monitor CACHE*

NOTE — The mnemonics IHP and IHV are defined in section 2.4.3.1, Table 2.7.

2.3.2 SLAVE

The block diagram of the SLAVE is shown in Fig 2.3. The dotted lines in the diagram show signals whose use varies among the various types of SLAVES. Table 2.2 shows how the various types of SLAVES use these lines.

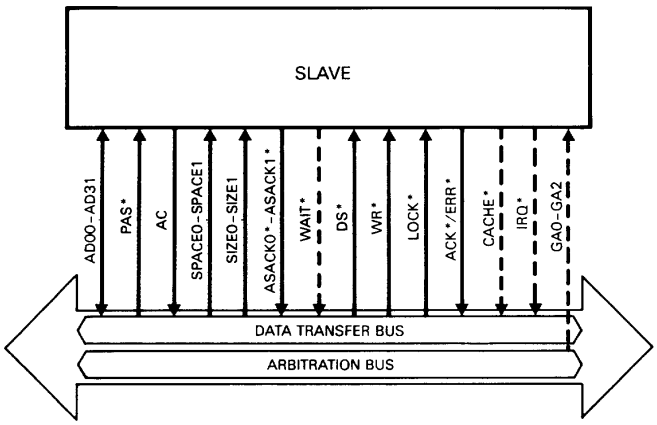


Figure 2.3—Block Diagram: SLAVE

OBSERVATION 2.4:

All VSB SLAVES monitor each address broadcast. Some SLAVES decode the address, determine that they do not have to act upon it, and release AC and WAIT* to high immediately after they finish decoding the address. These SLAVES are called *idle SLAVES.*

OBSERVATION 2.5:

Some SLAVES decode the address and determine that they need to act upon it. These SLAVES release AC to high after they finish decoding the address. However, they maintain WAIT* low, releasing it to high only after they have acted upon the data transfer phase and are ready to participate in a new data transfer phase or a new cycle. These SLAVES are called *participating SLAVES.*

OBSERVATION 2.6:

No more than one SLAVE responds to the cycle. It does so by acknowledging the address broadcast phase on ASACK0* and/or ASACK1*, and the data transfer phase on ACK* and/or ERR*. This SLAVE is called the *responding SLAVE.*

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.2—RULES and PERMISSIONS That Specify the Use of the Dotted Lines by the Various Types of SLAVES

Type of SLAVE	Use of Dotted Lines
D08	MUST monitor and drive AD24-AD31
	MAY or MAY not monitor or drive AD00-AD23
	MAY or MAY not drive WAIT*
D16	MUST monitor and drive AD16-AD31
	MAY or MAY not monitor or drive AD00-AD15
	MAY or MAY not drive WAIT*
D32	MUST monitor and drive AD00-AD31
	MAY or MAY not drive WAIT*
INTP	MUST drive IRQ*
	MAY or MAY not drive WAIT*
	MAY or MAY not monitor GA0-GA2
INTV	MUST drive IRQ*
	MUST drive WAIT*
	MUST monitor GA0-GA2
ALL SLAVES	MAY or MAY not drive ERR*
	MAY or MAY not drive CACHE*

NOTES:
1 — The menmonics D08, D16, and D32 are defined in section 2.4.2.2, Table 2.5.
2 — The mnemonics INTP and INTV are defined in section 2.4.3.1, Table 2.7.

2.4 Capabilities of MASTERS and SLAVES

The protocols of the VSB define two basic types of cycles that are used to transfer data: the SINGLE-TRANSFER cycle and the BLOCK-TRANSFER cycle. Another type of cycle, the ADDRESS-ONLY cycle, is used to extend the capabilities of MASTERS during the address broadcast phase. Each of these types of cycles can be used to access byte locations in any of the three address spaces: System Address Space, Alternate Address Space, and I*O Address Space. In addition, each of these cycles can be made indivisible by driving the LOCK* line low. An additional type of cycle is the INTERRUPT-ACKNOWLEDGE cycle, which is used by MASTERS and SLAVES to coordinate the service of interrupts. This section describes the capabilities of MASTERS and SLAVES to execute the above cycles and establishes mnemonics that define those capabilities.

An interlocked relationship exists between MASTERS and SLAVES during the course of VSB cycles. As shown in Fig 2.4, the cycle begins after the MASTER has been granted exclusive use of the Data Transfer Bus (as described in Section 3).

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

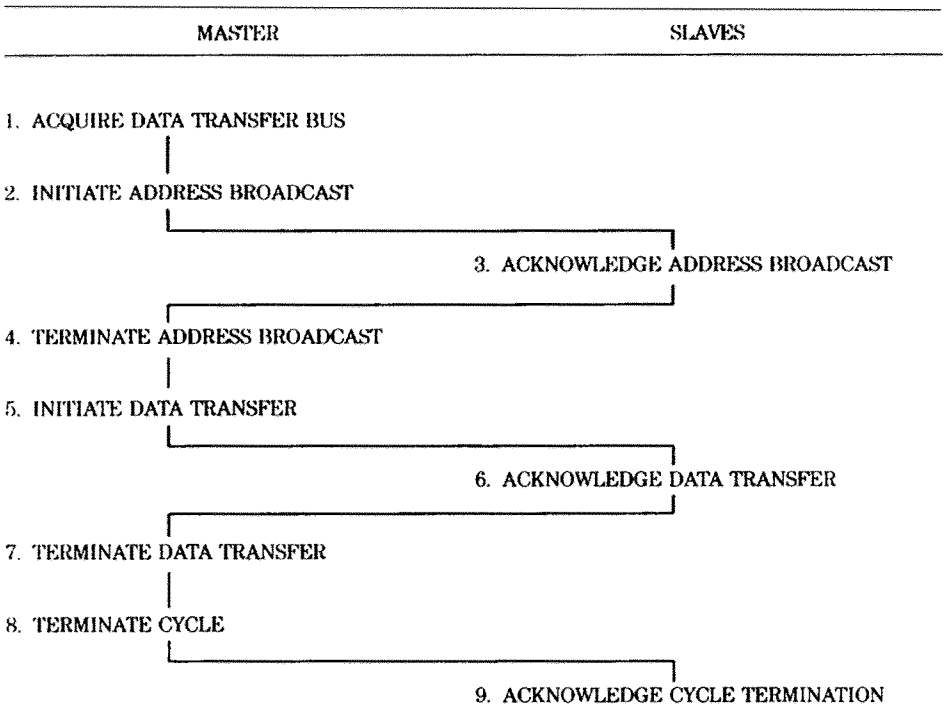


Figure 2.4—General Flow of a VSB Cycle

The active MASTER initiates the cycle by broadcasting addressing information to VSB SLAVES, which acknowledge and respond to the address broadcast. The active MASTER maintains the address broadcast until it detects this acknowledge and then terminates the address broadcast.

The active MASTER might then initiate one or more data transfers, or it might terminate the cycle without performing any data transfers. When SLAVES detect a data transfer they acknowledge it. The MASTER maintains the data transfer until it detects this acknowledgment. After the MASTER terminates the data transfer it might execute another data transfer, or it might terminate the cycle. When SLAVES detect a cycle termination they acknowledge it. This concludes the cycle.

The capabilities of MASTERS and SLAVES that pertain to the address broadcast phase are described in section 2.4.1.

The capabilities of MASTERS and SLAVES that pertain to the data transfer phase are described in section 2.4.2.

The capabilities of MASTERS and SLAVES that pertain to INTERRUPT-ACKNOWLEDGE cycles are described in section 2.4.3.

2.4.1 Addressing Capabilities

The smallest addressable unit of storage that can be provided by SLAVES is the byte location. A group of 4 byte locations that can be accessed simultaneously in a single data transfer is called a 4-byte group. During the course of VSB cycles a MASTER might request access to either one, two, three, or four byte locations provided by SLAVES.

The notation *BYTE(x) cycle* is used to describe a cycle that requests access to byte location *x*, where *x* can assume the value 0, 1, 2, or 3. For example, a BYTE(0) Cycle requests access to byte location BYTE (0) of the 4-byte group.

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

The notation *BYTE(x-y) cycle* is used to describe a cycle that requests access to a set of consecutive byte locations in the same 4-byte group, where *x-y* define a range between 0 and 3. For example, a BYTE(1-2) Cycle requests access to byte locations BYTE(l) and BYTE(2) of the 4-byte group.

A special notation *BYTE(x-y)(z) cycle* is used to describe cycles that specify access to byte locations in 2. consecutive 4-byte groups. For example, a BYTE (1-3)(0) Cycle requests access to byte locations BYTE(1-3) of 4-byte group *n* and byte location BYTE(0) of 4-byte group *n*+1. Multiple cycles are always required to complete this type of data access. If this example selects a D32 SLAVE, then byte locations BYTE(1-3) of 4-byte group *n* will be accessed during the first cycle. Byte location BYTE(0) of 4-byte group *n*+1 will be accessed in a subsequent cycle initiated by on-board logic that supports dynamic bus sizing. (The mnemonics D08, D16, and D32 are defined in section 2.4.2.2, Table 2.5.)

Four types of cycles are defined:

1) During *Quad-Byte cycles*, the active MASTER requests access to four consecutive byte locations. There are four types of Quad-Byte cycles: BYTE(0-3), BYTE(1-3)(0), BYTE(2-3)(0-1), and BYTE(3)(0-2). The only Quad-Byte cycle that can be completed in a single data transfer is a BYTE(0-3) cycle to a D32 SLAVE.
2) During *Triple-Byte cycles*, the active MASTER requests access to three consecutive byte locations. Triple-byte cycles that can be completed in a single data transfer to a D32 SLAVE are: BYTE(0-2) and BYTE(1-3). BYTE (2-3)(0) and BYTE(3)(0-1) Triple-Byte cycles, as well as any Triple-Byte cycle to D16 or D08 SLAVES, require multiple data transfers.
3) During *Double-Byte cycles*, the active MASTER requests access to two consecutive byte locations. The four types of Double-Byte cycles are: BYTE(0-1), BYTE(1-2), BYTE(2-3), and BYTE(3)(0). BYTE(0-1) and BYTE(2-3) cycles can be completed in a single data transfer to both D16 and D32 SLAVES. In addition, a BYTE(1-2) cycle can be completed in a single data transfer to a D32 SLAVE. All other types of Double-Byte cycles require multiple data transfers.
4) During *Single-Byte cycles*, the active MASTER accesses one byte location. The four types of Single-Byte cycles are: BYTE(0), BYTE(1), BYTE(2), and BYTE(3). All types of Single-Byte cycles complete in a single data transfer.

2.4.1.1 Basic Addressing Capabilities

MASTERS broadcast an address over the VSB at the beginning of each cycle. This address might select byte locations in the System Address Space, in the Alternate Address Space, or in the I/O Address Space. Table 2.3 shows the mnemonics that are used to describe the addressing capabilities of MASTERS and SLAVES.

The active MASTER drives a space code on SPACE0-SPACE1 during the address broadcast phase. This space code informs the SLAVES whether the broadcasted address selects byte locations in the System Address Space, in the Alternate Address Space, or in the I/O Address Space, as defined in section 2.5.1.2, Table 2.9.

OBSERVATION 2.7:

In addition to the three addressing modes described here, there is a fourth mode that is used to select INTERRUPT-ACKNOWLEDGE and ARBITRATION cycles. They can be distinguished from other VSB cycles by the levels of SPACE0-SPACE1, as described in section 2.5.4.2, Table 2.18.

2.4.1.2 ADDRESS-ONLY Capability

The ADDRESS-ONLY cycle is the only cycle on the Data Transfer Bus that is not used to transfer data. The active MASTER starts the ADDRESS-ONLY cycle by executing the address broadcast phase. After its address broadcast is acknowledged in the proper manner, the MASTER terminates the cycle without executing any data transfers. (See Fig 2.5.)

Table 2.4 shows how the ADDRESS-ONLY mnemonic ADO is used to describe MASTERS.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

Table 2.3—Mnemonics That Specify Addressing Capabilities

The Following Mnemonic	When Applied to a	Means That It Can
SAS	MASTER SLAVE	Generate cycles to the System Address Space Accept cycles to the System Address Space
ALTAS	MASTER SLAVE	Generate cycles to the Alternate Address Space Accept cycles to the Alternate Address Space
IOAS	MASTER SLAVE	Generate cycles to the I/O Address Space Accept cycles to the I/O Address Space

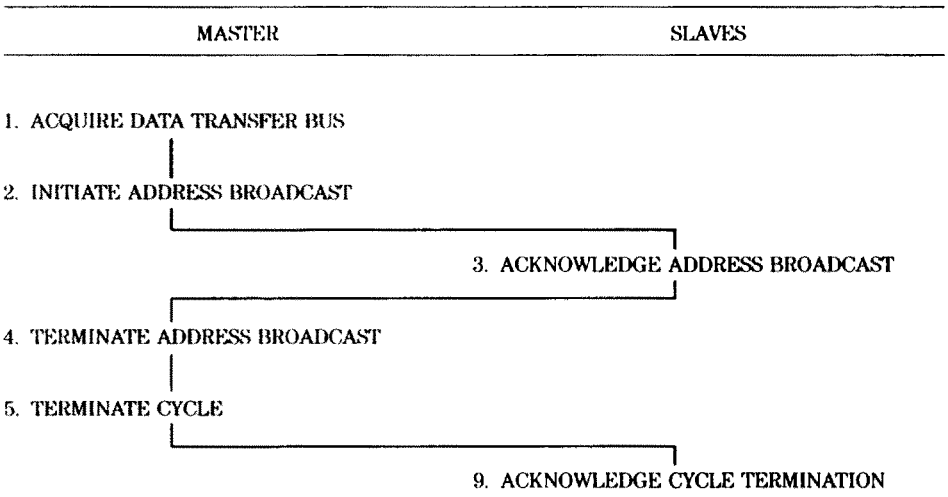


Figure 2.5—General Flow of an ADDRESS-ONLY Cycle

Table 2.4—Mnemonic That Specifies ADDRESS-ONLY Capability

The Following Mnemonic	When Applied to a	Means That It Can
ADO	MASTER	Generate ADDRESS-ONLY cycles

OBSERVATION 2.8:

ADDRESS-ONLY cycles can be used to enhance performance by allowing a CPU board to broadcast an address before it has determined whether that address selects an on-board device or a VSB device. Broadcasting the address in this fashion allows the CPU board and VSB SLAVES to decode the address at the same time.

RULE 2.3:

VSB SLAVES MUST be able to tolerate ADDRESS-ONLY cycles without loss of data or incorrect operation.

2.4.2 Data Transfer Capabilities

During VSB data transfer cycles, the active MASTER directs the transfer of data to and from byte locations provided by VSB SLAVES.

Copyright © 1989 IEEE All Rights Reserved
IEEE Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

A special notation is used to describe the data that on-board logic (for example, a microprocessor) outputs. These data might be either 8, 16, 24, or 32 bits wide. Each 8-bits of the data are referred to as a data byte.

1) *Quad-byte data* are data items that contain four bytes of data called DATA(0), DATA(1), DATA(2), and DATA(3).
2) *Triple-byte data* are data items that contain three bytes of data called DATA(1), DATA(2), and DATA(3).
3) *Double-byte data* are data items that contain two bytes of data called DATA(2) and DATA(3).
4) *Single-byte data* are data items that contain only one byte of data called DATA(3).

The organization of the four sizes of data items are shown in Fig 2.6.

OBSERVATION 2.9:

The relationship of the information among the bytes of a data item is beyond the scope of this specification. Different processors might organize information in different ways. When systems are assembled using a mixture of processors, the responsibility for correct data interchange among these processors should be borne by system software.

2.4.2.1 The Basic Data Transfer Capability of MASTERS

The VSB specification defines only one basic data transfer capability for MASTERS.

RULE 2.4:

All VSB MASTERS MUST drive and monitor data lines AD00-AD31.

OBSERVATION 2.10:

The requirement to drive all the data lines allows VSB MASTERS to generate the following types of cycles:

Quad-Byte Data	DATA(0)	DATA(1)	DATA(2)	DATA(3)
Triple-Byte Data		DATA(1)	DATA(2)	DATA(3)
Double-Byte Data			DATA(2)	DATA(3)
Single-Byte Data				DATA(3)

Figure 2.6—Organization of Data

1) Single-Byte cycles to byte locations: BYTE(0), BYTE(1), BYTE(2), and BYTE(3)
2) Double-Byte cycles to byte locations: BYTE(0-1), BYTE(1-2), BYTE(2-3), and BYTE(3)(0)
3) Triple-Byte cycles to byte locations: BYTE(0-2), BYTE(1-3), BYTE(2-3)(0), and BYTE(3)(0-1)
4) Quad-Byte cycles to byte locations: BYTE(0-3), BYTE(1-3)(0), BYTE(2-3) (0-1), and BYTE(3)(0-2).

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

2.4.2.2 Basic Data Transfer Capabilities of SLAVES

The VSB specification defines three basic data transfer capabilities for SLAVES. Table 2.5 shows how the mnemonics D08, D16, and D32 are used to describe the basic data transfer capabilities of SLAVES.

2.4.2.3 Dynamic Bus Sizing

Central to the data transfer capabilities of the VSB is the support of dynamic bus sizing. Dynamic bus sizing describes the ability of on-board logic to automatically adjust the size of the data transfer and the number of cycles that are required to transfer a data item to the basic data transfer capabilities of the responding board. The ability to perform dynamic bus sizing allows programmers to store information without regard for the memory organization of the system on which the software is run. The system hardware itself is then given the task of adjusting the size of the transfer to the physical constraints of the system.

The VSB protocols support dynamic bus sizing as follows.

The responding SLAVE acknowledges the address broadcast phase by identifying itself as a D08, D16, or a D32 SLAVE on ASACK0*-ASACK1* (see section 2.5.1.2, Table 2.13). This allows the active MASTER to determine which data lines are driven with valid data during the data transfer phase of read cycles (see section 2.5.2.3, Tables 2.15, 2.16, and 2.17). In addition, the SLAVE size information is made available to on-board logic, allowing it to perform additional cycles until all the required byte locations have been accessed.

During some write cycles, the active MASTER drives multiple data lines with the same data bytes to accommodate the three allowed SLAVE sizes. This operation is sometimes referred to as data replication. When the active MASTER initiates a write cycle it has no way of knowing whether the SLAVE that will respond to the cycle is a D08, D16, or a D32 SLAVE. When appropriate, the active MASTER places the same data bytes on multiple data lines to accommodate the various types of SLAVES (see section 2.5.2.3, Table 2.14). This ensures that some data will be transferred during the first data transfer even if multiple transfers are required to complete the cycle.

For example, one could have designed a processor board so that Double-Byte data for a BYTE(1-2) write cycle were only placed on AD16-AD23. However, a D08 SLAVE only monitors data lines AD24-AD31, preventing it from capturing any data during the first data transfer. Therefore, when executing a BYTE(1-2) write cycle, DATA(2) is placed on AD24-AD31 as well as on AD16-AD23. DATA(3) is placed on AD08-AD15 only. Therefore, if a D32 SLAVE is responding, all the required byte locations will be accessed in the course of a single data transfer. The multiple placement of DATA(2) ensures that if either a D08 or a D16 SLAVE is responding, they can both write DATA(2) during the first data transfer. This allows microprocessors that support dynamic bus sizing to always complete the data transfer of DATA(3) in only one additional cycle.

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.5—Memonics That Specify Basic Data Transfer Capabilities of SLAVES

The Following Mnemonic	When Applied to a	Means That It Can Accommodate the Following Cycles in a Single Data Transfer
D08	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3)
D16	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3) and Double-Byte data transfers to: BYTE(0–1) BYTE(2–3)
D32	SLAVE	Single-Byte data transfers to: BYTE(0) BYTE(1) BYTE(2) BYTE(3) and Double-Byte data transfers to BYTE(0–1) BYTE(1–2) BYTE(2–3) and Triple-Byte data transfers to: BYTE(0–2) BYTE(1–3) and Quad-Byte data transfers to: BYTE(0–3)

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

2.4.2.4 SINGLE-TRANSFER Capability

A SINGLE-TRANSFER cycle consists of an address broadcast followed by one data transfer. There are two types of SINGLE-TRANSFER cycles:

1) SINGLE-TRANSFER Write Cycles, during which the MASTER stores data in byte locations provided by SLAVES.
2) SINGLE-TRANSFER Read Cycles, during which the MASTER retrieves data from byte locations provided by SLAVES.

RULE 2.5:

All VSB SLAVES MUST include SINGLE-TRANSFER capability.

Figure 2.7 shows the general flow of a SINGLE-TRANSFER cycle.

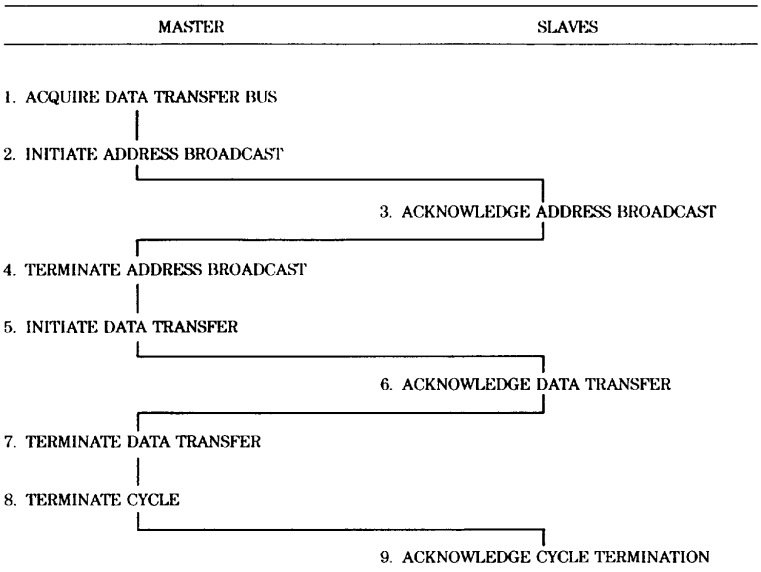


Figure 2.7—General Flow of a SINGLE-TRANSFER Cycle

2.4.2.5 BLOCK-TRANSFER Capability

MASTERS often transfer blocks of data. When this is the case, BLOCK-TRANSFER cycles are very useful. They allow the MASTER to provide a single address, and then transfer a block of data without providing additional addresses.

There are two primary types of BLOCK-TRANSFER cycles:

1) BLOCK-TRANSFER Write Cycles, during which the MASTER stores data in byte locations provided by SLAVES.
2) BLOCK-TRANSFER Read Cycles, during which the MASTER retrieves data that was stored in byte locations provided by SLAVES.

The BLOCK-TRANSFER read cycle is similar to a string of SINGLE-TRANSFER read cycles. Likewise, the BLOCK-TRANSFER write cycle is similar to a string of SINGLE-TRANSFER write cycles. The difference is that the MASTER only broadcasts the initial address once, at the beginning of the cycle, and then proceeds to execute as many data transfers as are required.

OBSERVATION 2.11:

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Control of the DTB cannot be transferred during BLOCK-TRANSFER cycles because the address strobe is held low through all of the data transfers, and control of the DTE can only be transferred while the address strobe is high.

Each of these cycles can be one of the following:

1) Quad-byte BLOCK-TRANSFER cycles, in which each data transfer accesses four bytes of data in byte locations provided by D32 SLAVES.
2) Double-byte BLOCK-TRANSFER cycles, in which each data transfer accesses two bytes of data in byte locations provided by D16 or D32 SLAVES.
3) Single-byte BLOCK-TRANSFER cycles, in which each data transfer accesses one byte of data in byte locations provided by D08, D16, or D32 SLAVES.

When a MASTER initiates a BLOCK-TRANSFER cycle, the responding SLAVE latches (at least the low order part of) the address into an on-board address counter and acknowledges the address broadcast. The MASTER then starts the data transfer. After the responding SLAVE acknowledges the data transfer the MASTER terminates it (that is, it drives the data strobe high). However, the MASTER does not allow the address strobe to go high. Instead, it repeatedly drives the data strobe low, interlocked with data transfer acknowledgments from the responding SLAVE. The responding SLAVE uses its on-board address counter to generate the address for each data transfer. Figure 2.8 shows the general flow of a BLOCK-TRANSFER cycle.

The VSB does not limit the number of bytes that a MASTER can transfer during a BLOCK-TRANSFER cycle. However, to simplify the design of BLOCK-TRANSFER SLAVES and to protect against loss of data, a BLOCK-TRANSFER cycle cannot cross a board boundary. Therefore, BLOCK-TRANSFER SLAVES are required to signal the active MASTER when they can no longer perform data transfers without receiving another address. For example, a BLOCK-TRANSFER SLAVE that only latches the lower eight address lines (requiring only an 8-bit counter) will signal the end of its self-addressing range when a 256-byte boundary is crossed. On the other hand, a 32-bit address counter on a BLOCK-TRANSFER SLAVE imposes no limit on the size of the transfer. However, such a SLAVE is still required to signal the end of its self-addressing range after it has accessed the last byte location it provides on-board.

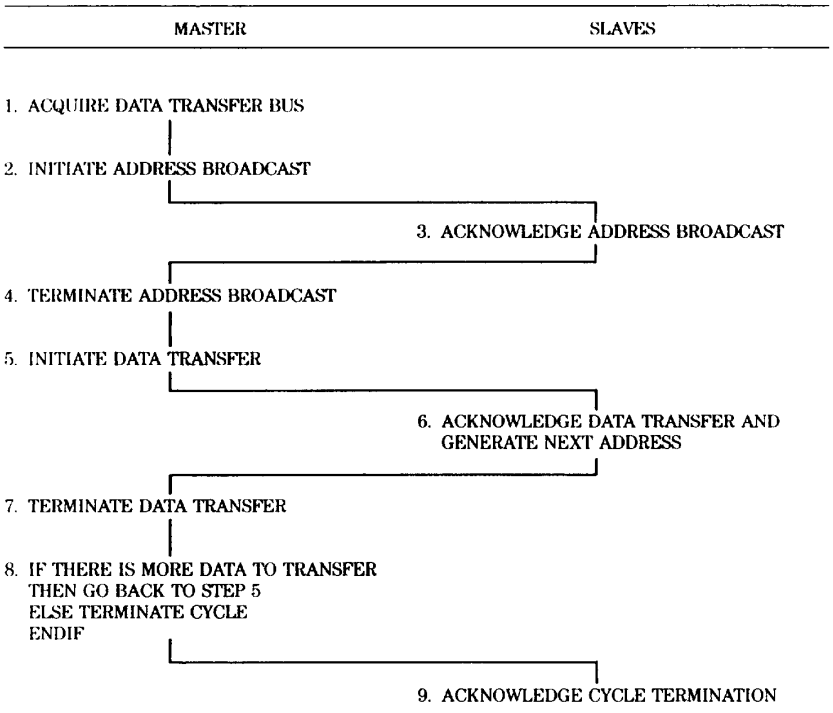


Figure 2.8—General Flow of a BLOCK-TRANSFER Cycle

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.6 lists the mnemonic that is used to describe the BLOCK-TRANSFER capability of MASTERS and SLAVES.

Table 2.6—Mnemonic That Specifies BLOCK-TRANSFER Capability

The Following Mnemonic	When Applied to a	Means That It
BLT	MASTER	Can generate Single-Byte BLOCK-TRANSFER cycles, and Double-Byte BLOCK-TRANSFER cycles, and Quad-Byte BLOCK TRANSFER cycles
	D08 SLAVE	Responds to all BLOCK-TRANSFER cycles by executing Single-Byte data transfers
	D16 SLAVE	Responds to Single-Byte BLOCK-TRANSFER cycles by executing Single-Byte data transfers Responds to Double-Byte and Quad-Byte BLOCK-TRANSFER cycles by executing Double-Byte data transfers
	D32 SLAVE	Responds to Single-Byte BLOCK-TRANSFER cycles by executing Single-Byte data transfers Responds to Double-Byte BLOCK-TRANSFER cycles by executing Double-Byte data transfers Responds to Quad-Byte BLOCK-TRANSFER cycles by executing Quad-Byte data transfers

2.4.2.6 INDIVISIBLE-ACCESS Capability

Many modern processors can perform indivisible sequences, that is, sequences of memory accesses that have to proceed one after the other, without allowing any other processor or MASTER to access any of the same byte locations before the sequence is complete. Such processors signal indivisible sequences differently from normal bus cycles, so that external hardware can guarantee the needed indivisibility.

The LOCK* line in the VSB can be used to signal such indivisibility, but it has to be used with caution to avoid system deadlocks. This section provides a description of two types of indivisible accesses and offers suggestions as to how to avoid system deadlocks.

There are two classes of indivisible sequences:

1) An *Intradomain Indivisible Sequence* (INTRASEQ) is one in which access to all the byte locations involved in the sequence is controlled by the same arbitration mechanism, that is, access to all of them is granted simultaneously. An example of an INTRASEQ is a Test and Set instruction, in which a single byte location is accessed twice in succession. Another example is when all of the byte locations that are accessed during the sequence are on the same board.
2) An *Interdomain Indivisible Sequence* (INTERSEQ) is one in which access to the byte locations involved in the sequence is controlled by two or more arbitration mechanisms. In such a case, two or more processors might each be granted access to some of the byte locations involved but not all of them. An example of an INTERSEQ is a sequence that involves access to a byte location on the VSB and also a byte location on the global system bus.

Use of the LOCK* line is sufficient to guarantee indivisibility for INTRASEQS but not for INTERSEQS. This is best demonstrated by an example as follows:

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Two CPU boards, CPU A and CPU B, each have dual-ported memory, which allows access both locally and from the VSB. CPU A's memory is accessed starting at address 100000, while CPU B's memory is accessed starting at address 200000.

CPU A starts an INTERSEQ that involves its local memory at address 100000 (accessed first) and global memory at address 200000 (accessed second).

At the same time CPU B starts an INTERSEQ that involves its local memory at address 200000 (accessed first) and global memory at address 100000 (accessed second).

In this example, without a control mechanism for INTERSEQS, each CPU will first access the byte location in its respective local memory, and then request control of the VSB to access the second, global byte location. Suppose that CPU A is granted use of the VSB first. It assumes control of the VSB and attempts to access byte location 200000 in CPU B’s memory. However, this access cannot be granted since CPU B is in the middle of its own INTERSEQ, which it cannot complete until it is granted use of the VSB, which cannot be granted until CPU A completes its INTERSEQ. The system is in a deadlock.

One way to prevent deadlock in such a case is to constrain the system software from executing INTERSEQS. This can be achieved by placing all the variables that might ever be accessed in an indivisible manner in global memory, that is, memory that can only be accessed over the global system bus. Imposing such a constraint requires that all software adhere to it.

This strategy is fine for information that is accessed more or less equally by all the processors in a system. But, for information that is accessed frequently by one processor and only occasionally by others, placing the information in global memory increases bus traffic and can reduce system performance, compared to placing it in memory local to the predominant processor.

A more general solution to potential system deadlocks is to establish a protocol for executing INTERSEQS. CPU boards that are capable of generating INTERSEQS have to request and be granted control of an agreed upon central resource BEFORE executing ANY PART of an indivisible sequence. It does not much matter what this central resource is, just that all processor boards “play by the same rule.” An obvious choice for such a central resource is the global system bus. It is a better choice than the subsystem bus because processor boards on different subsystem buses might need to perform INTERSEQS that involve byte locations that are local to another processor board.

2.4.3 Interrupt Capability

The VSB defines a protocol that allows SLAVES to request interrupt service from a MASTER which, in turn, responds to these requests. Any system that has interrupt capability includes software routines that are called interrupt service routines. The interrupt protocol of the VSB defines how the SLAVE transfers STATUS*ID information to the MASTER. This information is used by on-board logic to invoke an interrupt service routine.

OBSERVATION 2.12:

The VSB protocols do not limit the amount of the STATUS*ID information that can be transferred in the course of a single cycle.

2.4.3.1 Basic Interrupt Capabilities of MASTERS and SLAVES

The VSB defines two basic interrupt capabilities for MASTERS and SLAVES:

1) In response to an interrupt request, the MASTER initiates, and SLAVES participate in an INTERRUPT-ACKNOWLEDGE cycle. Each SLAVE is assigned a unique INTERRUPT ID, which is used to prioritize multiple interrupts. This INTERRUPT ID is a 7-bit ID that is a combination of the 3-bit slot address and a 4-bit user defined priority code. It is used during the selection phase of an INTERRUPT-ACKNOWLEDGE

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

cycle to select the SLAVE whose interrupt request will be serviced next. This SLAVE transfers the STATUS*ID information to the MASTER, and releases its contribution to the IRQ* line to high. When executing an INTERRUPT-ACKNOWLEDGE cycle, this read operation is referred to as a STATUS*ID transfer.

2) In response to an interrupt request, the MASTER INITIATES read cycles to predefined addresses to determine which SLAVE has requested an interrupt. These SLAVES include a STATUS*ID register, and release their contribution to the IRQ* line to high during the read cycle that accesses their STATUS*ID register. The MASTER might then execute additional cycles to read additional STATUS*ID information from the SLAVE. The STATUS*ID information that the MASTER reads from the SLAVE is transferred in the course of a read data transfer, as defined in section 2.5.2.2. When responding to an interrupt request by polling, a MASTER executes either a SINGLE-TRANSFER or a BLOCK-TRANSFER read cycle, depending on its capabilities and on the capabilities of the responding SLAVE.

Table 2.7 shows the mnemonics that are used to describe the basic interrupt capabilities of MASTERS and SLAVES.

Table 2.7—Mnemonics That Specify Interrupt Capability

The Following Mnemonic	When Applied to a	Means That It
IHV	MASTER	Responds to interrupt requests by initiating an INTERRUPT-ACKNOWLEDGE cycle
IHP	MASTER	Responds to interrupt requests by initiating a read cycle to determine their cause
INTV	SLAVE	Can generate interrupt requests and participate in INTERRUPT-ACKNOWLEDGE cycles
INTP	SLAVE	Can generate interrupt requests and identify itself in the course of a read cycle

RULE 2.6:

INTV SLAVES MUST include INTP capability.

2.4.3.2 INTERRUPT-ACKNOWLEDGE Cycle Capabilities

When a MASTER responds to an interrupt request it reads STATUS*ID information from the SLAVE. As described above, the MASTER reads this STATUS*ID in the course of an INTERRUPT-ACKNOWLEDGE cycle or by polling the SLAVE. The general flow of the INTERRUPT-ACKNOWLEDGE cycle is shown in Fig 2.9. The cycle starts when a SLAVE requests an interrupt service. The IHV MASTER that services interrupts acquires the bus and initiates an INTERRUPT-ACKNOWLEDGE cycle that is comprised of three distinct phases:

1) The first phase is the *selection phase*. It starts when the IHV MASTER initiates an INTERRUPT-ACKNOWLEDGE cycle and ends after the responding INTV SLAVE is determined (steps 3-6 in Fig 2.9). The interaction between the INTV MASTER and INTV SLAVES during this phase is described in section 2.5.4.1.
2) The second phase is the *STATUS*ID transfer phase*. It starts after the responding INTV SLAVE is selected, and ends after all of the STATUS*ID information is transferred (steps 7-9 in Fig 2.9). The interaction between the IHV MASTER and the responding INTV SLAVE during this phase is described in section 2.5.4.1.
3) The third phase is the *termination phase* of the INTERRUPT-ACKNOWLEDGE cycle (steps 9-11 in Fig 2.9), described in section 2.5.3.

Copyright © 1989 IEEE. All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

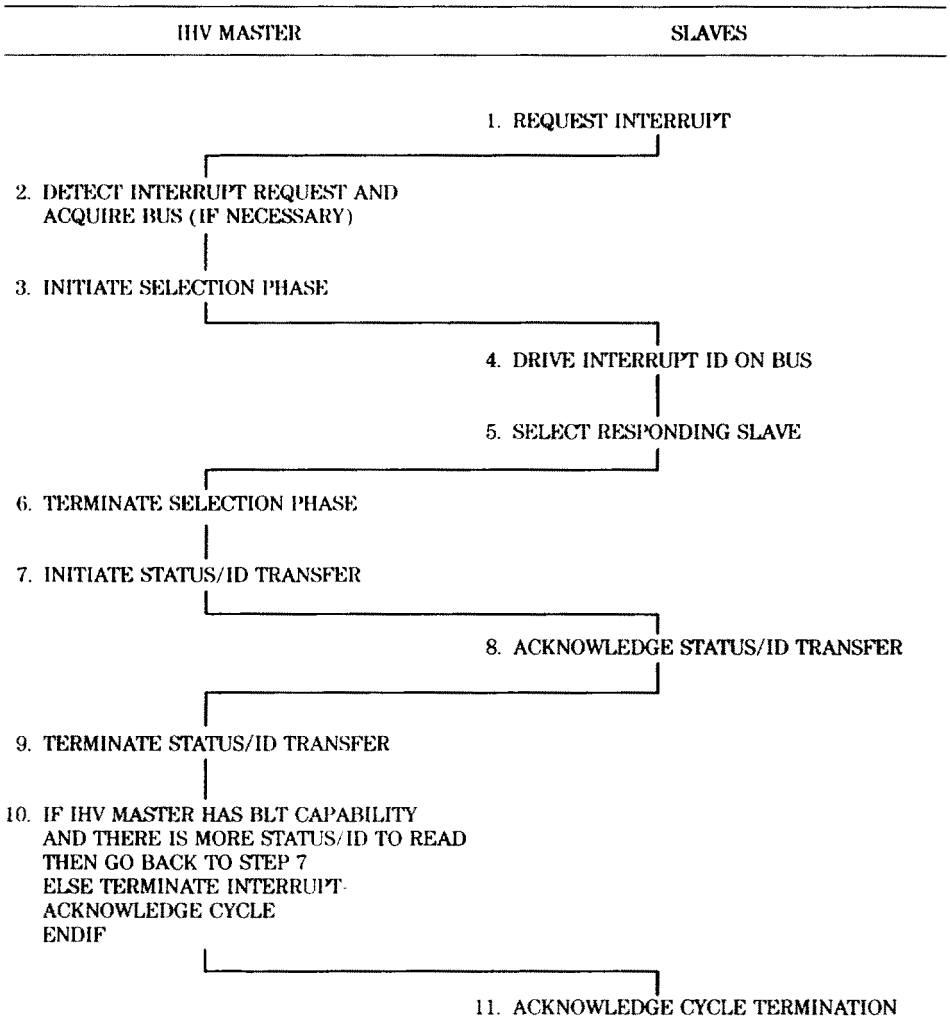


Figure 2.9—General Flow of an INTERRUPT-ACKNOWLEDGE Cycle

There are three types of INTERRUPT-ACKNOWLEDGE cycles:

1) Quad-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS*ID transfer accesses four consecutive bytes of the available STATUS*ID information.
2) Double-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS*ID transfer accesses two consecutive bytes of the available STATUS*ID information.
3) Single-byte INTERRUPT-ACKNOWLEDGE cycles, in which each STATUS*ID transfer accesses one byte of the available STATUS*ID information.

Table 2.8 shows how the mnemonics D08, D16, D32, and BLT are used to describe the STATUS*ID transfer capabilities of IHV MASTERS and INTV SLAVES.

2.5 Interaction Between MASTERS and SLAVES

The interaction between MASTERS and SLAVES is described with the aid of flow diagrams. Section 2.5.1 describes the interaction between MASTERS and SLAVES during the address broadcast phase.

Section 2.5.2 describes the interaction between MASTERS and SLAVES during the data transfer phase.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Section 2.5.3 describes the interaction between MASTERS and SLAVES during cycle termination.

Table 2.8—Mnemonics That Specify STATUS/ID Transfer Capabilities of IHV MASTERS and INTV SLAVES

The Following Mnemonic	When Applied to a	Means That It
D08	INTV SLAVE	Responds to all STATUS/ID requests by executing a Single-Byte STATUS/ID transfer
D16	INTV SLAVE	Responds to Single-Byte STATUS/ID transfer requests by executing a Single-Byte STATUS/ID transfer Responds to Double-Byte and Quad-Byte STATUS/ID transfer requests by executing a Double-Byte STATUS/ID transfer
D32	INTV SLAVE	Responds to Single-Byte STATUS/ID transfer requests by executing a Single-Byte STATUS/ID transfer Responds to Double-Byte STATUS/ID transfer requests by executing a Double-Byte STATUS/ID transfer Responds to Quad-Byte STATUS/ID transfer requests by executing a Quad-Byte STATUS/ID transfer
BLT	INTV MASTER	Can request multiple STATUS/ID transfers in the course of an INTERRUPT-ACKNOWLEDGE cycle
BLT	INTV SLAVE	Can properly respond to multiple STATUS/ID transfers in the course of an INTERRUPT-ACKNOWLEDGE cycle

The interaction between MASTERS and SLAVES during INTERRUPT-ACKNOWLEDGE cycles is described in section 2.5.4.

2.5.1 Interaction Between MASTERS and SLAVES During Address Broadcast

An interlocked relationship exists between MASTERS and SLAVES during the address broadcast phase of the VSB cycle. Once the active MASTER has started the address broadcast it is required to maintain all the relevant signal lines valid until its address broadcast is fully acknowledged. VSB SLAVES might acknowledge the address broadcast in one of three different ways, becoming either idle, participating, or responding SLAVES.

Section 2.5.1.1 and Fig 2.10 show the flow of the address broadcast, and how MASTERS and SLAVES interact during this phase. This section only provides an informal description of the protocol so as to familiarize the reader with the flow of the address broadcast. Section 2.5.1.2 constitutes a formal specification of this protocol, while section 2.6 specifies the timing requirements.

2.5.1.1 Flow of the Address Broadcast Phase

Figure 2.10 shows the interaction between MASTERS and SLAVES during the address broadcast phase. Step 1 describes the intercycle state of VSB SLAVES. After they detect a high level on PAS*, SLAVES release ASACK0*-ASACK1* to high, and drive AC low. Some SLAVES might, in addition, drive WAIT* to low.

The address broadcast begins in step 2 after the MASTER has been granted use of the DTB, becoming the active MASTER. (Section 3 describes how the MASTER is granted use of the DTB.) After receiving PAS* high, signifying that the previous MASTER stopped driving bus lines, the active MASTER drives SPACE0-SPACE1 to select an address space and SIZE0-SIZE1 to define the number of byte locations that it wishes to access. It drives WR* either high for a read cycle, or low for a write cycle. If the cycle is to be locked, it drives LOCK* low. After detecting ASACK0*ASACK1 * high, signifying that the SLAVE that responded to the previous read cycle has stopped driving the data lines, the active MASTER drives AD02-AD31 to select a 4-byte group and AD00-AD01 to select the lowest addressed byte location to be accessed within the 4-byte group. It then drives PAS* low to signal the transfer of the addressing information to the SLAVES.

In step 3 SLAVES acknowledge the address broadcast. After they receive a low level on PAS* they decode the address, the space code, and the size code to determine if they are to respond to or participate in the data transfer. SLAVES that have BLOCK-TRANSFER capability latch the required address lines into an on-board address counter, using it to generate the address for subsequent data transfers.

Copyright © 1989 IEEE All Rights Reserved
IEEE Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

The SLAVE that will be responding to the data transfer drives ASACK0* ASACK1* to indicate its size, ensures that the CACHE* line is valid, and releases AC and WAIT* to high. All idle SLAVES release AC and WAIT* to high. SLAVES that wish to participate in the data transfer do not release WAIT* to high, maintaining it low until after they are ready to participate in a new data transfer.

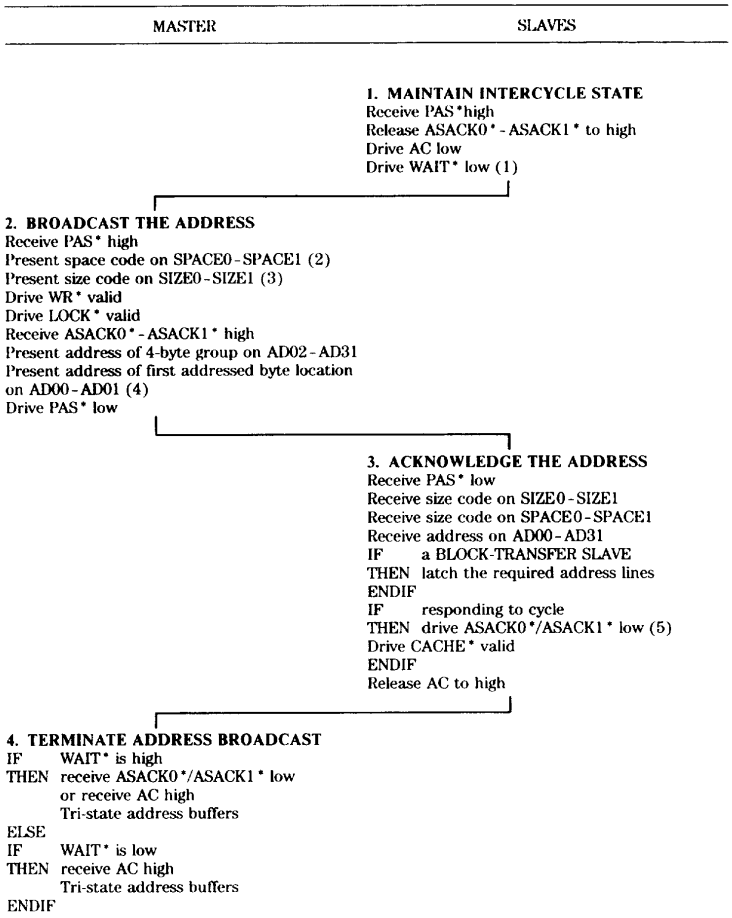


Figure 2.10—Flow of the Address Broadcast Phase

NOTES: (1) In step 1, only those SLAVES that wish to extend either the address broadcast phase or the data transfer phase drive WAIT* to low.

(2) In step 2, the active MASTER drives SPACE0-SPACE1 to select one of the three address spaces, as defined in section 2.5.1.2, Table 2-9.

(3) In step 2, the active MASTER drives SIZE0-SIZE1 to select the size of the data transfer, as defined in section 2.5.1.2, Table 2-10.

(4) In step 2, the active MASTER drives AD00-AD01 to select the lowest addressed byte location, as defined in section 2.5.1.2, Table 2-11.

(5) In step 3, the responding SLAVE drives ASACK0*-ASACK1* to inform the MASTER of its size, as defined in section 2.5.1.2, Table 2-13.

Figure 2-10 —*(continued)*

In step 4 the MASTER terminates the address broadcast. It does so after the address broadcast is acknowledged by the SLAVES. If the active MASTER detects a low level on WAIT*, then it is required to maintain the address broadcast valid until after it detects a high level on AC. On the other hand, if WAIT* is not driven low, then the active MASTER is allowed to terminate the address broadcast after it detects a falling edge on either ASACK0* or on ASACK1*, or a rising edge on AC, whichever comes first.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

The way VSB SLAVES acknowledge the address broadcast phase might specify one of the following conditions:

1) A responding SLAVE has been selected and all SLAVES are ready to participate in the data transfer. This condition is determined by a low level on ASACK0* and/or ASACK1*, and a high level on AC.
2) At least one participating SLAVE is ready to participate in a data transfer, but no SLAVE has been configured to respond to the cycle. This condition is determined by a high level on ASACK0*, ASACK1 * and AC, and a low level on WAIT*.
3) No SLAVES have been configured to either respond to or participate in the cycle. This condition is determined by a high level on ASACK0*, ASACK1*, AC, and WAIT*.

In any case the active MASTER terminates the address broadcast. It does so by turning its address buffers off. This terminates the interaction between MASTERS and SLAVES during the address broadcast portion of the cycle.

Depending on the MASTER'S design, on the type of address acknowledge it receives, and on the type of cycle it executes, the MASTER might then either terminate the cycle, as described in section 2.5.3, or proceed to execute one or more data transfers, as described in section 2.5.2.

2.5.1.2 Signaling During the Address Broadcast Phase

This section provides the formal specifications for driving and releasing VSB signal lines during the address broadcast phase of VSB cycles. The signaling specifications during the selection phase of an INTERRUPT-ACKNOWLEDGE cycle are covered in section 2.5.4.

RULE 2.7:

All VSB SLAVES MUST drive AC low whenever they detect a high level on PAS*.

RULE 2.8:

VSB SLAVES that are capable of driving WAIT* low MUST do so whenever they detect a high level on PAS*.

RULE 2.9:

Before driving any of AD00-AD31, SPACE0-SPACE1, SIZE0-SIZE1, WR*, LOCK*, or PAS*, the active MASTER MUST be granted use of the Data Transfer Bus as specified in Section 3..

RULE 2.10:

The active MASTER MUST provide the space code on SPACE0-SPACE1 to select an address space, as shown in Table 2.9.

RULE 2.11:

VSB SLAVES MUST decode the space code on SPACE0-SPACE1 to select an address space, as shown in Table 2.9.

RULE 2.12:

During the address broadcast phase, the active MASTER MUST place the size code on SIZE0-SIZE1 to request the size of the data transfer, as shown in Table 2.10.

RULE 2.13:

VSB SLAVES MUST decode the size code on SIZE0-SIZE1 to determine the requested size of the requested size of the data transfer, as shown in Table 2.10.

RULE 2.14:

During the address broadcast phase, the active MASTER MUST drive address lines AD00-AD01 to select the offset of the lowest addressed byte location that is accessed within a 4-byte group, as shown in Table 2.11.

Copyright © 1989 IEEE All Rights Reserved
IEEE Documents Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.9—Use of SPACE0 and SPACE1 to Select the Address Space

Address Space	SPACE1	SPACE0
Alternate Address Space	low	high
I/O Address Space	high	low
System Address Space	high	high

Table 2.10—Encoding of SIZE0 and SIZE1 for Requested Size of the Transfer

Requested Size of Data Transfer	SIZE1	SIZE0
Quad-Byte Data Transfer	low	low
Single-Byte Data Transfer	low	high
Double-Byte Data Transfer	high	low
Triple-Byte Data Transfer	high	high

Table 2.11—Use of AD00 and AD01 to Select the Lowest Addressed Byte Location To Be Accessed

Lowest Address Byte Accessed	AD01	AD00
BYTE(0)	low	low
BYTE (1)	low	high
BYTE(2)	high	low
BYTE(3)	high	high

RULE 2.15:

During the address broadcast portion of VSB cycles, SLAVES MUST decode address lines AD00-AD01 to select the offset of the lowest addressed byte location that is accessed within a 4-byte group, as shown in Table 2.11.

RULE 2.16:

During the address broadcast phase, the active MASTERS MUST encode SIZE0, SIZE1, AD00, and AD01 to define the byte locations that are to be accessed, as shown in Table 2.12.

RULE 2.17:

SLAVES MUST decode SIZE0, SIZE1, AD00, and AD01 to determine which byte locations are to be accessed, as shown in Table 2.12.

RULE 2.18:

IF the active MASTER wants to execute a write cycle, that is, store data in byte locations provided by SLAVES,

THEN it MUST drive the WR* line to a low level.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988

Table 2.12—Encoding of SIZE0, SIZE1, AD00, and AD01 to Define the Byte Locations To Be Accessed

Byte Locations to be Accessed	SIZE1	SIZE0	AD01	AD00
BYTE(0)	low	high	low	low
BYTE(1)	low	high	low	high
BYTE(2)	low	high	high	low
BYTE(3)	low	high	high	high
BYTE (0–1)	high	low	low	low
BYTE(1–2)	high	low	low	high
BYTE(2–3)	high	low	high	low
BYTE (3)(0)	high	low	high	high
BYTE(0–2)	high	high	low	low
BYTE(1–3)	high	high	low	high
BYTE(2–3)(0)	high	high	high	low
BYTE(3)(0–1)	high	high	high	high
BYTE(0–3)	low	low	low	low
BYTE(1–3)(0)	low	low	low	high
BYTE(2–3)(0–1)	low	low	high	low
BYTE (3)(0–2)	low	low	high	high

RULE 2.19:

IF the active MASTER wants to execute a read cycle, that is, retrieve data from byte locations provided by SLAVES,

THEN it MUST ensure that the WR* line is high.

RULE 2.20:

IF the active MASTER wants to execute an INDIVISIBLE-ACCESS cycle,

THEN it MUST drive the LOCK* line to a low level.

RULE 2.21:

All SLAVES MUST decode AD00-AD31 and SPACE0-SPACE1, and MUST NOT drive either ASACK0* or ASACK1* to low if they do not provide the requested byte locations in the requested address space.

RULE 2.22:

During the address broadcast phase, the responding SLAVE MUST drive and the active MASTER MUST decode the code on ASACK0*-ASACK1 * to define the size of the SLAVE, as shown in Table 2.13.

Copyright © 1989 IEEE All Rights Reserved
Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1096-1988

Table 2.13—Encoding of ASACK0* and ASACK1* to Define the Size of the SLAVE

Size of SLAVE	ASACK1*	ASACK0*
D32 SLAVE responding	low	low
D16 SLAVE responding	low	high
D08 SLAVE responding	high	low
No SLAVE responding	high	high

PERMISSION 2.1:

The responding SLAVE MAY drive CACHE* to a valid level during the cycle.

SUGGESTION 2.1:

For optimum performance, design SLAVES so that they release AC to high as soon as possible.

OBSERVATION 2.13:

Participating SLAVES maintain WAIT* low during the address broadcast phase.

RULE 2.23:

IF a participating SLAVE does not release WAIT* to high during the address broadcast phase,

THEN it MUST release it to high during the data transfer phase of the same cycle.

OBSERVATION 2.14:

A bus timer module is not required in the VSB system since:

1) All VSB SLAVES are required to release AC to high after they finish decoding the address.
2) Participating SLAVES maintain a low level on WAIT* until after they have finished the data transfer.
3) The responding SLAVE is required to drive ASACK0* and/or ASACK1* low when it finishes decoding the address.

Therefore, if the active MASTER detects a high level on ASACK0*-ASACK1*, AC, and WAIT* after it has driven PAS* low, then it is assured that no VSB SLAVES are configured to either respond to or participate in an access to the requested address. At such a time, the active MASTER can terminate the cycle.

2.5.2 Interaction Between MASTERS and SLAVES During the Data Transfer

An interlocked relationship exists between MASTERS and SLAVES during the data transfer phase of VSB cycles. Once the active MASTER has started a data transfer, it is required to maintain all the relevant signal lines valid until it is acknowledged in the proper manner. VSB SLAVES acknowledge the data transfer in two different ways, depending on whether they are participating in, or responding to the cycle.

Section 2.5.2.1 and Fig 2-13 describe the flow and the interaction between the active MASTER and the selected SLAVES during a write data transfer. Section 2.5.2.2 and Fig 2-12 describe the flow and the interaction between the active MASTER and the selected SLAVES during a read data transfer. These sections only provide an informal description of the protocol to familiarize the reader with the flow of the data transfer phase. Section 2.5.2.3 constitutes a formal specification of this protocol, while section 2.6 specifies the timing requirements.

Copyright © 1989 IEEE All Rights Reserved

Documents Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on March 06,2026 at 05:19:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1096-1988