

An American National Standard

IEEE Standard Microcomputer System Bus

Sponsor
**Standards Committee
of the
IEEE Computer Society**

Approved December 9, 1982
Reaffirmed June 9, 1988

IEEE Standards Board

Approved February 17, 1984
Reaffirmed January 23, 1989

American National Standards Institute

© Copyright 1983 by

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

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

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

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

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

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

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

Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

Foreword

(This Foreword is not a part of IEEE Std 796-1983, IEEE Standard Microcomputer System Bus.)

The IEEE Std 796 bus is a commercial-quality bus for use in a microcomputer-based system. The board modules that connect to the bus may be slaves (memory or input/output, or both), masters (CPU or controllers, or both), or both.

The IEEE Std 796 bus provides an electrical and mechanical specification to allow many different manufacturers to produce varied but compatible microcomputer modules. This offers the user the flexibility to choose the modules that solve his microcomputer system problem in the most complete and cost-effective manner.

The bus is a departure from the minicomputer market, where each major manufacturer defines its bus without regard for its potential use by other systems or manufacturers. The IEEE Std 796 bus, though initiated by Intel as their Multibus, is a cooperative industry effort toward establishing a standard for a large number of manufacturers and users of microcomputer modules.

The original development of the Multibus was done at Intel by their initial systems engineering group—Bob Garrow, Rich Boberg, Hap Walker, and Mike Yen—with significant input by Fred Coury, a consultant. The bus was designated the Multibus, an Intel trademark, and was used as the basis for Intel's Intellec Microcomputer Development System introduced in 1975. Intel decided to use the same bus for its subsequent iSBC product line. The success of these reliable OEM board products produced hundreds of competitive and complementary products, all made to be compatible with this commercial-quality bus. This diverse offering of modules compatible at the board level has provided customers with a comprehensive and cost-effective product selection—even with second-sourcing at the board level. The customers of these boards are companies whose value-added consists of their own special-purpose hardware and software; they use the standard CPU, memory, I/O, and peripheral controller modules as the basis for their systems. The version of the bus standardized by the IEEE will help assure the user community that the compatibility and high commercial quality of the bus will be maintained from one manufacturer to another.

This edition of IEEE Std 796 is complete and accurate, however, as expected, the IEEE Std 796 bus will continue to evolve with enhancements and refinements. The general philosophy being followed for these changes is to maintain upward and downward compatibility, where possible, while creating a clean, professional standard. In some cases, the IEEE Std 796 Working Group has elected to specify the bus in a way that will cause technical incompatibility with existing products. The removal of the –5 volt power supply from the bus pins is an example. In these cases, it is the opinion of the committee that there is minimal impact since the products can still be used (by making exceptions to the standard) until the products are changed or their lifetimes expire. This temporary inconvenience permits the deliberate focusing of all new or redesigned products onto the preferred standard.

The IEEE Standards Board calls attention to the fact that it is claimed that there is a patent on the common bus request owned by Intel Corporation and that Intel has patents pending on the byte-swap aspects of the bus. These patents appear to cover the subject, as partially described in 2.1.3.2.2 and 2.2.2.4. IEEE takes no position with respect to patent validity. Intel Corporation has assured the IEEE that it is willing to grant a license on this aspect of the bus on reasonable and nondiscriminatory terms to anyone wishing to obtain such a license. Intel Corporation's undertakings in this respect are on file with the IEEE Standards Office, and the license details may be obtained from the legal department of Intel Corporation whose address is Intel Corporation, 5200 N.E. Elam Young Parkway, Hillsboro, Oregon 97123.

Although the original work in creating the IEEE Std 796 bus was done at Intel Corporation, the current specification represents the work of individuals from many different companies. These individuals include the members of the IEEE Std 796 Working Group of the IEEE Computer Society's Microprocessor Standards Committee.

At the time this standard was approved the membership of the working group was as follows.

Richard W. Boberg, *Chair*

Rod H. Allen
Mark Bagula
Ron Dilbeck
Gary Fieland

Robert Garrow
Bill Halloway
Jim Johnson
Jim Kelley
Craig Kinnie

Jim Konsevich
Larry Michel
Martin A. Newman
Tung-sun Tung

When the IEEE Standards Board approved this standard on December 9, 1982, it had the following membership:

I. N. Howell, Jr, *Chair*
Edward Chelotti, *Vice Chair*
Sava I. Sherr, *Secretary*

G. Y. R. Allen
J. J. Archambault
James H. Beall
John T. Boettger
J. V. Bonucchi
Edward J. Cohen
Len S. Corey

Donald C. Fleckenstein
Jay Forster
Kurt Greene
Joseph L. Koepfinger
Irving Kolodny
John E. May
Donald T. Michael*

A. R. Parsons
J. P. Riganati
Frank L. Rose
Robert W. Seelbach
Jay A. Stewart
Clifford O. Swanson
Robert E. Weiler

*Member emeritus

CLAUSE	PAGE
1. General	1
1.1 Scope.....	1
1.2 Object	2
1.3 Definitions.....	2
2. Functional Description	3
2.1 IEEE Std 796 Bus Elements.....	4
2.2 Data Transfer Operation.....	8
2.3 Interrupt Operations	16
2.4 IEEE Std 796 Bus Exchange.....	19
3. Electrical Specifications.....	22
3.1 General Bus Considerations.....	22
3.2 Timing.....	27
3.3 Receiver Modules, Driver Modules and Terminations	34
4. Mechanical Specifications	36
4.1 Backplane Considerations.....	37
4.2 IEEE Std 796 Bus Board Form Factors	41
5. Levels of Compliance	43
5.1 Variable Elements of Capability	43
5.2 Masters and Slaves.....	44
5.3 Compliance Level Notation	44
6. Bibliography	45

IEEE Standard Microcomputer System Bus

1. General

1.1 Scope

One of the most important elements in a computer system is the bus structure that supplies the interface for all the hardware components. This bus structure contains the necessary signals to allow the various system components to interact with each other. It allows memory and I/O data transfers, direct memory accesses, generation of interrupts, etc. This standard provides a detailed description of all the elements and features that make up the IEEE Std 796 bus.

The bus supports two independent address spaces: memory and I/O. During memory cycles, the bus allows direct addressability of up to 16 megabytes using 24-bit addressing. During I/O bus cycles, the bus allows addressing of up to 64 kilobyte I/O ports using 16-bit addressing. Memory and I/O cycles can support 8-bit or 16-bit data transfers.

The bus structure is built upon the masterslave concept where the master device in the system takes control of the bus and the slave device, upon decoding its address, and acts upon the command provided by the master. This handshake (master-slave relationship) between the master and slave devices allows modules of different speeds to be interfaced by way of the bus. It also allows data rates up to five million transfers per second (bytes or words) to take place across the bus.

Another important feature of the bus is the ability to connect multiple master modules for multiprocessing configurations. The bus provides control signals for connecting multiple masters in either a serial or parallel priority fashion. With either of these two arrangements, more than one master may share bus resources.

This standard has been prepared for those users who evaluate or design products that will be compatible with the IEEE Std 796 system bus structure. To this end, the necessary signal definitions and timing and electrical specifications have been covered in detail.

This standard deals only with the interface characteristics of microcomputer devices, not with design specifications, performance requirements, and safety requirements of modules.

Throughout this standard, the term *system* denotes the byte or word interface system that, in general, includes all the circuits, connectors, and control protocol to effect unambiguous data transfer between devices. The term *device* or

module denotes any product connected to the interface system that communicates information by way of the bus, and that conforms to the interface system definition.

1.2 Object

The objective of this standard is to

- 1) Define a general purpose microcomputer system bus.
- 2) Specify the device-independent electrical and functional interface requirements that a module shall meet in order to interconnect and communicate unambiguously by way of the system.
- 3) Specify the terminology and definitions related to the system.
- 4) Enable the interconnection of independently manufactured devices into a single functional system.
- 5) Permit products with a wide range of capabilities to be introduced to the system simultaneously.
- 6) Define a system with a minimum of restrictions on the performance characteristics of devices connected to the system.

1.3 Definitions

The following general definitions apply throughout this standard. More detailed definitions can be found in the appropriate section.

1.3.1 General System Terms

compatibility: The degree to which devices may be interconnected and used without modification, when designed as defined in Sections 2 and 3 of this standard. Section 5 introduces the notion of levels of compliance and the corresponding notation.

bus cycle: The process whereby digital signals effect the transfer of data bytes or words across the interface by means of an interlocked sequence of control signals. *Interlocked* denotes a fixed sequence of events in which one event must occur before the next event can occur.

interface: A shared boundary between two systems, or between parts of systems, through which information is conveyed.

interface system: The device-dependent electrical and functional interface elements necessary for communication between devices. Typical elements are: driver and receiver circuits, signal line descriptions, timing and control conventions, and functional logic circuits.

override: A bus master overrides the bus control logic when it is necessary to guarantee itself back-to-back bus cycles. This is called *overriding* the bus, temporarily preventing other masters from using the bus.

system: A set of interconnected elements which achieve a given objective through the performance of a specified function.

1.3.2 Signals and Paths

bus: A signal line or a set of lines used by an interface system to connect a number of devices and to transfer information.

byte: A group of eight adjacent bits operated on as a unit.

word: Two bytes or sixteen bits operated on as a unit.

high state: The more positive voltage level used to represent one of two logical binary states.

low state: The more negative voltage level used to represent one of two logical binary states.

signal: The physical representation of data.

signal level: The relative magnitude of a signal when compared to an arbitrary reference. Signal levels in this standard are specified in volts.

signal line: One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.

signal parameter: That element of an electrical quantity whose values or sequence of values convey information.

2. Functional Description

This section provides an overall understanding of how the IEEE Std 796 bus functions and describes the elements that connect to the bus, the signals that provide the interface to the bus, and the different types of operations performed on the bus.

In this section, as well as throughout the standard, a clear and consistent notation for signals has been used. The memory write command (MWTC) is used to explain this notation. The terms one:zero and true:false can be ambiguous, so their use is avoided. In their place, the terms electrical high and low (H and L) are used. A nathan (asterisk) following the signal name, MWTC*, indicates that the signal is active low as shown:

MWTC* = asserted at 0 volts

The signal (MWTC*) driven by a three-state driver will be pulled up to V(cc) when not asserted. To further explain the notation used in this standard see Table 1.

Table 1— Electrical and Logic State Definition

Function	Definition		
	Electrical	Logic	State
MWTC	H	1 True	Active, asserted
	L	0 False	
MWTC*	L	1 True	Active, asserted
	H	0 False	

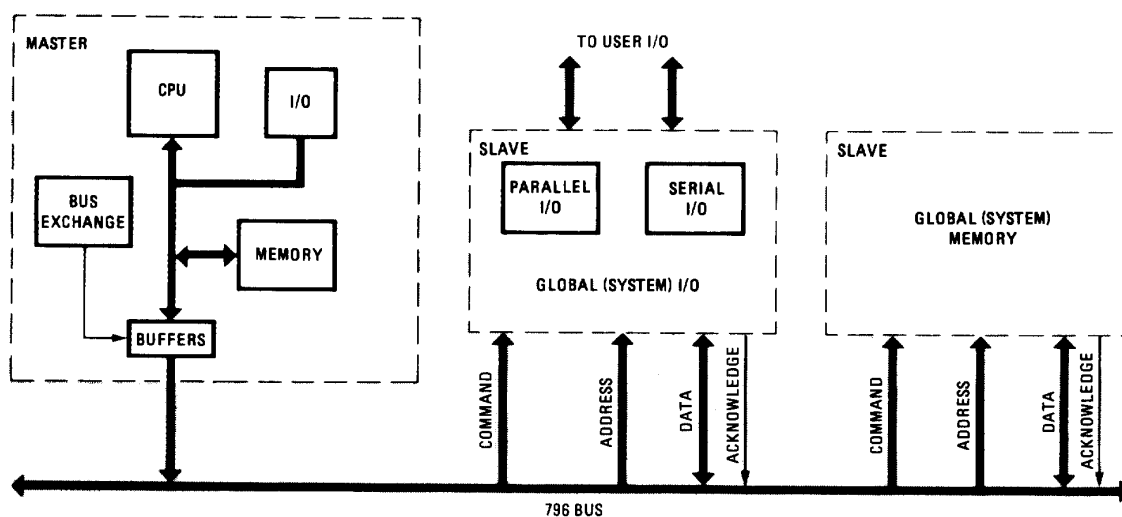


Figure 1— IEEE Std 796 Bus Master and Slave Example

2.1 IEEE Std 796 Bus Elements

This subsection describes the elements, masters and slaves, that interface to the bus and the IEEE Std 796 bus signal lines that comprise this interface.

2.1.1 Masters

A master is any module having the ability to control the bus. The master exercises this control by acquiring the bus through bus exchange logic and then generating command signals, address signals, and memory or I/O addresses. To perform these tasks, the master is equipped with either a central processing unit or logic dedicated to transferring data over to the bus and to and from other destinations. Figure 1 depicts a system that includes a master and two slave models.

The IEEE Std 796 bus architecture can support more than one master in the same system, but to do this, there should be a means for each master to gain control of the bus. This is accomplished through the bus exchange logic (see 2.4).

Masters may operate in one of two modes of operation. Modes 1 and 2 are defined as follows:

Mode 1: Masters are limited to single bus transfers per bus connect. If all masters are Mode 1, system timing is rendered deterministic by conformance with a maximum bus busy period. That period is limited by the parameter t_{BYSO} maximum (see 3.2.5).

Mode 2: Masters are unlimited in this bus control. They may invoke bus override. Bus timeouts are allowed. Conformance with the maximum busy period is not required. The last classification is included to allow for a very broad class of operations, giving users maximum flexibility in meeting these applications' needs. The first mode of operation is defined to allow system designers to predict the overall performance of their systems without concern for uncontrolled timing parameters such as bus timeout. For masters which can only operate in Mode 2, their specification shall state "Mode 2 master only."

2.1.2 Slaves

Another type of module that can interface to the bus is the slave. Slave modules decode the address lines and act upon the command signals from the masters. The slaves are not capable of controlling the bus. Some examples of bus slaves are shown in Fig 1.

Table 2— Control Line Signals

Class	Function	Signal
Clocks	Constant clock	CCLK*
	Bus clock	BCLK*
Commands	Memory write	MWTC*
	Memory read	MRDC*
	I/O write	IOWC*
	I/O read	IORC*
Acknowledge	Transfer acknowledge	XACK*
Initialize		INIT*
Lock		LOCK*

2.1.3 IEEE Std 796 Bus Signals

Signals transferred over the bus can be grouped into several classes based on the functions they perform.

The classes are as follows:

- 1) Control lines
- 2) Address and inhibit lines
- 3) Data lines
- 4) Interrupt lines
- 5) Bus exchange lines

The following subsections explain the different classes of IEEE Std 796 bus signals.

2.1.3.1 Control Lines

Signals classified as control lines are listed in Table 2.

2.1.3.1.1 Clock Lines

- 1) *Bus Clock (BCLK*)*. A periodic signal used to synchronize the bus contention logic; it may be slowed, stopped, or single stepped. The bus clock shall be generated by one and only one source within the system. This means that each standalone bus master must have the capability of generating an acceptable clock that can optionally be connected to, or disconnected from, the bus. In a multimaster system, only one of the masters shall have its clock connected to the bus.
- 2) *Constant Clock (CCLK*)*. A periodic signal of constant frequency, which may be used by masters or slaves as a master clock. The constant clock shall be generated by one and only one source within the system. This means that each bus master should have the capability of generating an acceptable clock that can optionally be connected to, or disconnected from, the bus. In a multimaster system, only one of the masters shall have its clock connected to the bus.

2.1.3.1.2 Command Lines (MWTC*, MRDC*, IOWC*, IORC*)

The command lines are elements of a communication link between the masters and slaves. There are two command lines for memory and two command lines for I/O. An active command line indicates to the slave that the address lines are carrying a valid address and that the slave is to perform the specified operation. In a data-write cycle, the active command line (MWTC* or IOWC*) additionally indicates that the data is valid on the bus. In a data-read cycle, the transition of the command (MRDC* or IORC*) from active to inactive indicates that the master has received the data from the slave.

2.1.3.1.3 Transfer Acknowledge Line (XACK*)

This line is used by the slaves to acknowledge commands from the master. XACK*, indicates to the master that the requested action is complete and that data has been placed on, or accepted from, the data lines.

2.1.3.1.4 Initialize (INIT*)

The INIT* signal is generated to reset the entire system to a known internal state. This signal is usually generated prior to starting any operations on the system. INIT* may be generated by any or all of the bus masters or by an external source such as a buffered and debounced front panel switch.

2.1.3.1.5 Lock (LOCK*)

The LOCK signal is generated by the master in control of the bus to indicate the slave is locked. LOCK* is used to extend mutual exclusion to multiple port RAM designs.

2.1.3.2 Address and Inhibit Times

The address and inhibit lines are used for the following signals:

Function	Signal
Address lines	A0*–A23*
Byte high enable	BHEN*
Inhibit lines	INH1* and INH2*

2.1.3.2.1 Address Lines (A0*–A23*)

These lines, which specify the address of the referenced memory location or I/O device, allow a maximum of 16 megabytes (16 777 216 bytes) of memory to be accessed. When addressing an I/O device, a maximum of 16 address lines (A0*–A15*) is used, thus allowing the addressing of a maximum of 64 kilobyte devices. An I/O module should also be able to be configured to decode only eight address lines (A0*–A7*) and ignore the upper eight lines (see 2.2.2.3).

2.1.3.2.2 Byte High Enable Line (BHEN*)

This byte control line is used to enable the upper byte (bits 8-15) of a 16-bit word to drive the bus. The signal is used only on systems that incorporate 16-bit data transfers.

2.1.3.2.3 Inhibit Lines (INH1* and INH2*)

The inhibit lines can be invoked for any memory read or memory write operation (MRDC* or MWTC*). An inhibit line is asserted by a slave to inhibit another slave's bus activity during a memory read or write operation. The inhibit signal generated by the inhibiting slave is derived from decoding the memory address lines. The inhibiting slave can decode a single address, a block of addresses, or any combination of single and block addresses.

When it detects the specific address during an actual command (MRDC* or MWTC*), the inhibiting slave generates an inhibit signal, which is sensed by the inhibited slave. When so inhibited, this slave module disables its drivers from all data, address, and acknowledge bus lines, although it may actually perform internal operations. [All modules that may be inhibited should have completed internal operations within 1.5 μ s from the start of the command line. This interval (1.5 μ s) is also the minimum acknowledge timing for modules issuing inhibits. This guarantees that inhibited modules have sufficient time to return to their normal state before the current bus command is completed.]

2.1.3.3 Data Lines (D0*–D15*)

These 16 bidirectional data lines transmit and receive information to and from a memory location or an I/O port. (D15* is the most significant bit and D0* is the least significant bit). In 8-bit transfer, only lines D0*–D7* are valid.

2.1.3.4 Interrupt Lines

The interrupt lines consist of the following signals:

Function	Signal
Interrupt requests	INT0*–INT7*
Interrupt acknowledge	INTA*

2.1.3.4.1 Interrupt Request Lines (INT0*–INT7*)

Interrupts are requested by activating one of the eight interrupt request lines. INT0* has the highest priority and INT7* has the lowest priority.

2.1.3.4.2 Interrupt Acknowledge (INTA*)

In response to an interrupt request signal, an interrupt acknowledge signal can be generated by a bus master with bus vectored interrupt capability. The interrupt acknowledge signal is used to freeze the interrupt status and request the placement of the interrupt vector address on the bus data lines.

2.1.3.5 Bus Exchange Lines

The bus exchange lines are used by the following signals:

Function	Signal
Bus clock	BCLK*
Bus request	BREQ*
Bus priority	BPRN*, BPRO*
Bus busy	BUSY*
Common bus request	CBRQ*

A master gains control of the bus through the manipulation of these signals.

2.1.3.5.1 Bus Request (BREQ*)

A signal used by the bus masters in a priority resolution circuit to indicate a request for control of the bus.

2.1.3.5.2 Bus Priority (BPRN* and BPRO*)

The priority functions allow masters to break deadlocks that occur when more than one master concurrently requests the bus. The bus priority in (BPRN*) signal indicates to a particular master that no higher priority master is requesting use of the bus. The bus priority out (BPRO*) signal is used in serial (daisy chain) bus priority resolution schemes. In such a scheme, BPRO* is passed by one master to the BPRN* input of the master with the next lower bus priority; when active, the BPRO* signal indicates that the higher priority master does not require control of the bus.

2.1.3.5.3 Bus Busy (BUSY*)

A signal activated by the master in control of the bus to indicate that the bus is in use. This prevents other masters from gaining control of the bus.

2.1.3.5.4 Common Bus Request (CBRQ*)

A signal that maximizes a master's data transfer rate to the bus by sensing the absence of other bus requests. The CBEQ* signal does this by serving two functions. It indicates to the master controlling the bus whether or not another master needs to gain control of the bus. For the other masters, it is a means of notifying the controlling bus master that it must relinquish control of the bus if it is not using the bus.

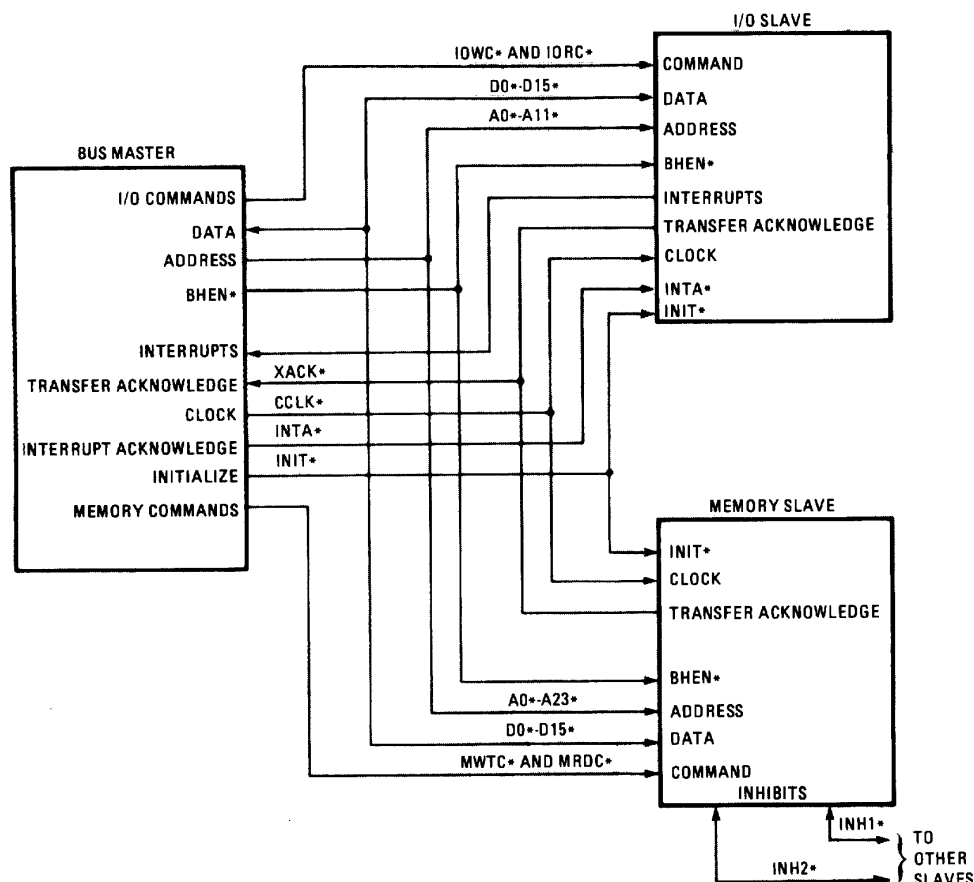


Figure 2— IEEE Std 796 Bus Interface Lines

2.2 Data Transfer Operation

The primary function of the IEEE Std 796 bus architecture is to provide a path for the transfer of data between modules on the bus. The following subsections describe the different types of data transfers and the means by which they are implemented using the signals previously described. Figure 2 can be referenced during the following discussion.

The discussion of the data transfer operation of the bus is covered in three parts

- 1) An overview of the operation
- 2) A detailed description of the signals used in the transfer
- 3) A discussion of the specifics pertaining to the different transfers

It is assumed in this discussion that there is only one master on the bus, and therefore no bus contention exists. The bus exchange logic is discussed in 2.4.

2.2.1 Data Transfer Overview

A data transfer is accomplished as follows: the bus master places the memory address or I/O port address on the address lines. If the operation is a write, the data is also placed on the data lines at this time. The bus master then generates a command (I/O read or write, or memory read or write), which activates the appropriate bus slave. The slave accepts the

data if it is a write operation, or places the data on the data lines if it is a read operation. A transfer acknowledge signal is then sent to the bus master by the bus slave, allowing the bus master to complete its cycle by removing the command from the command line and then clearing the address and data lines. Figures 3 and 4 show the basic timing for a read and write data transfer operation.

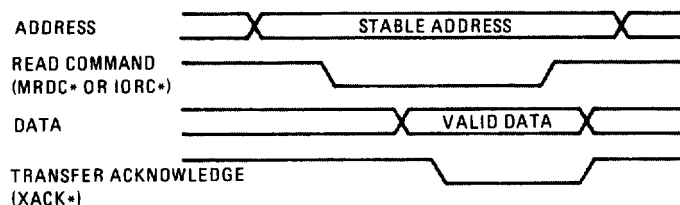


Figure 3— IEEE Std 796 Bus Read Operation

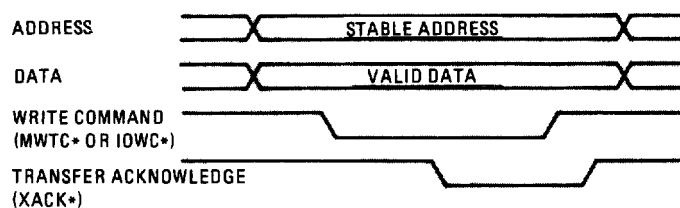


Figure 4— IEEE Std 796 Bus Write Operation

2.2.2 Signal Descriptions

This subsection provides a detailed description of the IEEE Std 796 bus signals. Included are timing, signal origination, and other information pertaining to the specific function that each signal performs in the data transfer operation.

2.2.2.1 Initialize (INIT*)

Prior to any operation of the bus, all system modules should be reset to a known internal state. This can be accomplished by an INIT* signal initiated by one of the following three sources:

- 1) A power-on clear circuit (RC network), which holds INIT* low until the power supplies reach their specific voltage outputs.
- 2) A reset button, which is sometimes provided on the system front panel for operator use. Note that this button should be debounced.
- 3) A software command that can be implemented to pull down the INIT* line.

The INIT* line is driven by open-collector gates and requires signal conditioning to meet the electrical specifications of the bus.

2.2.2.2 Constant Clock (CCLK*)

The constant clock signal, which is driven by only one source, provides a timing source for any or all modules on the bus. CCLK* is a periodic signal with a specified frequency and is driven by a clock driver circuit.

2.2.2.3 Address Lines (A0*–A23*)

The address lines are used to specify the address of the memory location or the I/O device that is being referenced by the command. There are 24 address lines, binary coded, to allow up to 16 777 216 bytes of memory to be referenced. These lines are driven by three-state drivers and are always controlled by the master using the bus.

For I/O bus cycles, master modules have the option of gathering 8-bit or 16-bit addresses. Because of this, all I/O slaves should be capable of being configured to decode eight address bits (A0*–A7*) and ignore the upper address bits or to decode all 16 bits of address (A0*–A15*). Note that in a system using 8-bit I/O addresses, the value of the upper 16 bits of address is unknown. A master generating only 8-bit address may set the upper 16 address bits to any arbitrary value.

Refer to Fig 5 for an example of address line usage.

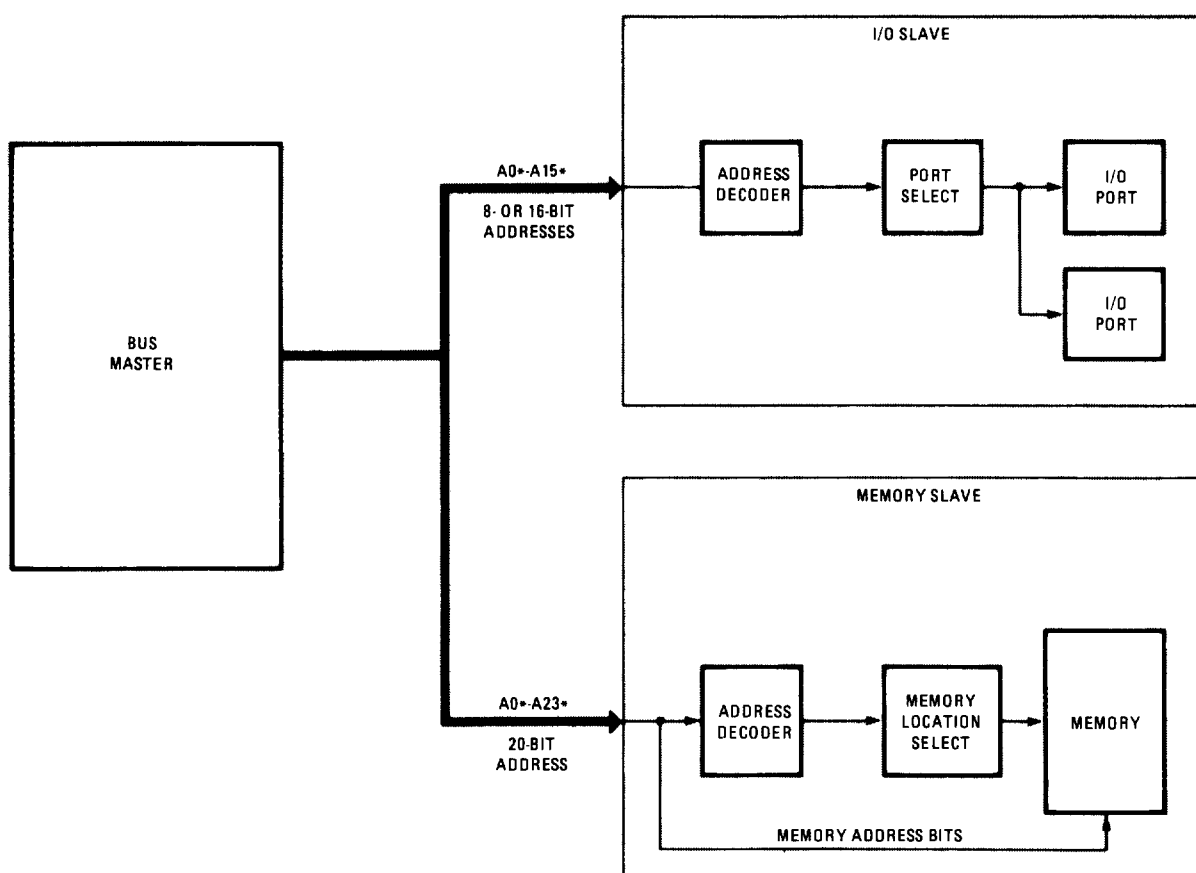


Figure 5— IEEE Std 796 Bus Address Line Usage

2.2.2.4 Data Lines (D0*–D15*)

These are 16 bidirectional data lines used to transmit and receive information to and from a memory location or I/O port. The 16 lines are driven by the master on write operations and by the addressed memory or I/O, on read operations. Both 16-bit and 8-bit transfers can be accomplished by using only lines D0*–D7*, with D0* being the least significant bit.