VMEbus Specification Manual
Revision C.1
October 1 985
Third Printing

INTRODUCTION

VMEbus is the most popular 16/32-bit backplane bus. Its use of the Eurocard format, its high performance, and its versatility are some of the reasons that it appeals to a wide range of users. The designer friendly and user friendly style of its specification offers useful advice and helps ensure compatibility between VMEbus products.

The following VMEbus specification is a result of both IEEE P1014 Standard Committee and the IEC 47b Standards Committee work. The version of the VMEbus specification presented here (Revision C.1) has been approved by the IEEE P1014 Standard Committee as draft 1.2. It is presently undergoing the final stages of approval in both the IEEE and the IEC. While the final IEEE 1014 and IEC 821 BUS standards are not expected to differ significantly from Revision C.1, the publisher of this document does not guarantee that those standards will not differ at all. This document is being published to provide the VMEbus community with an up-to-date professional and readable document in the hope of promoting compatibility between VMEbus products and encouraging the widespread use of VMEbus.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION TO THE VMEbus SPECIFICATION

1.1 VMEbus SPECIFICATION OBJECTIVES

The VMEbus specification defines an interfacing system used to interconnect data processing, data storage, and peripheral control devices in a closely coupled hardware configuration. The system has been conceived with the following objectives:

a. To allow communication between devices on the VMEbus without disturbing the internal activities of other devices interfaced to the VMEbus.

b. To specify the electrical and mechanical system characteristics required to design devices that will reliably and unambiguously communicate with other devices interfaced to the VMEbus.

c. To specify protocols that precisely define the interaction between the VMEbus and devices interfaced to it.

d. To provide terminology and definitions that describe system protocols.

e. To allow a broad range of design latitude so that the designer can optimize cost and/or performance without affecting system compatibility.

f. To provide a system where performance is primarily device limited, rather than system interface limited.

1.2 VMEbus INTERFACE SYSTEM ELEMENTS

1.2.1 Basic Definitions

The VMEbus structure can be described from two points of view: its mechanical structure and its functional structure. The mechanical specification describes the physical dimensions of subracks, backplanes, front panels, plug-in boards, etc. The VMEbus functional specification describes how the bus works, what functional modules are involved in each transaction, and the rules which govern their behavior. This section provides informal definitions for some basic terms used to describe both the mechanical and the functional structure of the VMEbus.

1.2.1.1 Terms Used To Describe The VMEbus Mechanical Structure

VMEbus BACKPLANE -- A printed circuit (PC) board with 96 pin connectors and signal paths that bus the connector pins. Some VMEbus systems have a single PC board, called the J1 backplane. It provides the signal paths needed for basic operation. Other VMEbus systems also have an optional second PC board, called a J2 backplane. It provides the additional 96 pin connectors and signal paths needed for wider data and address transfers. Still others have

a single PC board that provides the signal conductors and connectors of both the J1 and J2 backplanes.

BOARD -- A printed circuit (PC) board, its collection of electronic components, and either one or two 96 pin connectors that can be plugged into VMEbus backplane connectors.

SLOT -- A position where a board can be inserted into a VMEbus backplane. If the VMEbus system has both a J1 and a J2 backplane (or a combination J1/J2 backplane) each slot provides a pair of 96 pin connectors. If the system has only a J1 backplane, then each slot provides a single 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 do not disengage themselves from the backplane due to vibration or shock.

1.2.1.2 Terms Used To Describe The VMEbus Functional Structure

Figure 1.1 shows a simplified block diagram of the functional structure, including the VMEbus signal lines, backplane interface logic, and functional modules.

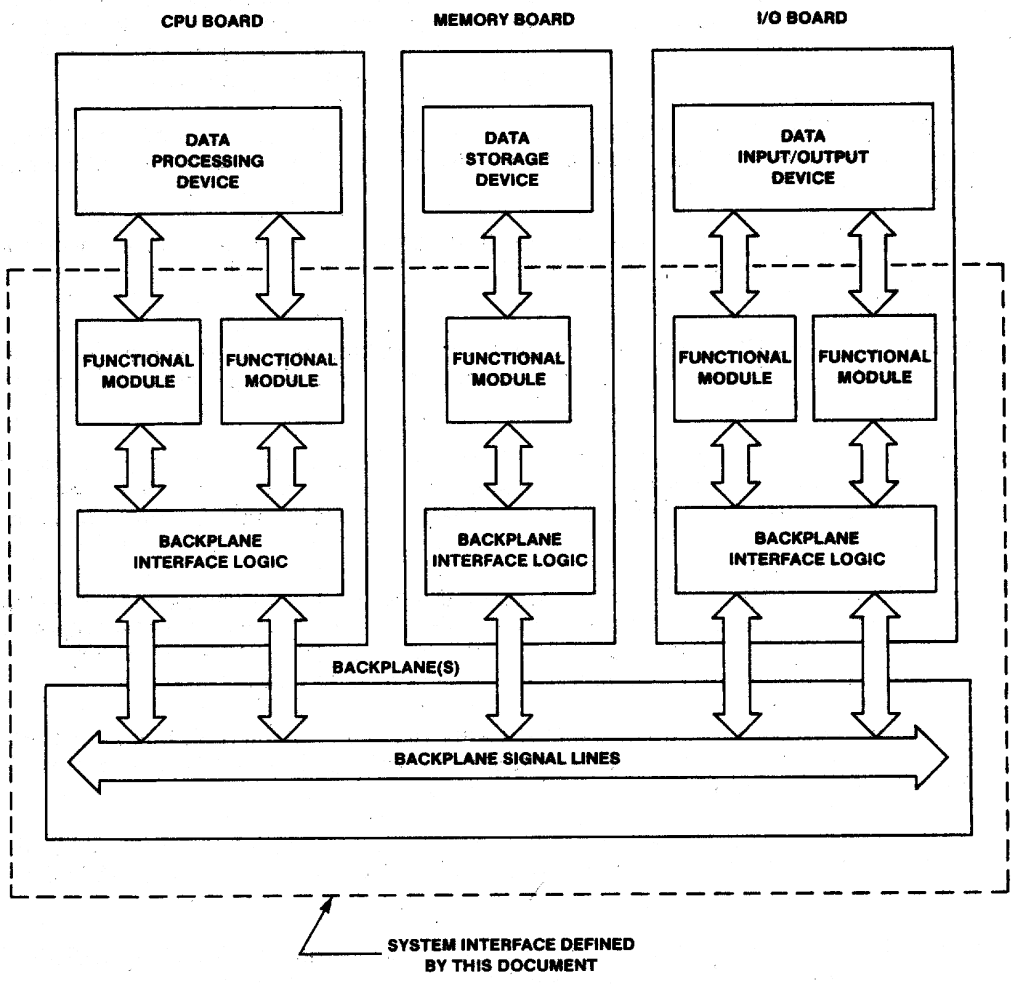


Figure 1-1. System Elements Defined by this Document

BACKPLANE INTERFACE LOGIC -- Special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc. The VMEbus specification prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.

FUNCTIONAL MODULE -- A collection of electronic circuitry that resides on one VMEbus board and works together to accomplish a task.

DATA TRANSFER BUS -- One of the four buses provided by the VMEbus backplane. The Data Transfer Bus allows MASTERS to direct the transfer of binary data between themselves and SLAVES. (Data Transfer Bus is often abbreviated DTB.)

DATA TRANSFER BUS CYCLE -- A sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a MASTER and a SLAVE. The Data Transfer Bus cycle is divided into two portions, the address broadcast, and then zero or more data transfers. There are 34 types of Data Transfer Bus cycles. They are defined later in this chapter.

MASTER -- A functional module that initiates DTB cycles in order to transfer data between itself and a SLAVE module.

SLAVE -- A functional module that detects DTB cycles initiated by a MASTER and, when those cycles specify its participation, transfers data between itself and the

LOCATION MONITOR -- A functional module that monitors data transfers over the DTB in order to detect accesses to the locations it has been assigned to watch. When an access to one of these assigned locations occurs, the LOCATION MONITOR generates an on-board signal.

BUS TIMER -- A functional module that measures how long each data transfer takes on the DTB and terminates the DTB cycle if a transfer takes too long. Without this module, if the MASTER tries to transfer data to or from a non-existent SLAVE location it might waft forever. The BUS TIMER prevents this by terminating the cycle.

PRIORITY INTERRUPT BUS -- One of the four buses provided by the VMEbus backplane. The Priority Interrupt Bus allows INTERRUPTER modules to send interrupt requests to INTERRUPT HANDLERS.

INTERRUPTER -- A functional module that generates an interrupt request on the Priority Interrupt Bus and then provides STATUS/ID information when the INTERRUPT HANDLER requests it.

INTERRUPT HANDLER -- A functional module that detects interrupt requests generated by INTERRUPTERS and responds to those requests by asking for STATUS/ID information.

DAISY-CHAIN -- A special type of VMEbus signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot. There are four bus grant daisy-chains and one interrupt acknowledge daisy-chain on the VMEbus.

JACK DAISY-CHAIN DRIVER -- A functional module which activates the interrupt acknowledge daisy-chain whenever an INTERRUPT HANDLER acknowledges an interrupt request. This daisy-chain ensures that only one INTERRUPTER will respond with its STATUS/ID when more than one has generated an interrupt request.

ARBITRATION BUS -- One of the four buses provided by the VMEbus backplane. This bus allows an ARBITER module and several REQUESTER modules to coordinate use of the DTB.

REQUESTER -- A functional module that resides on the same board as a MASTER or INTERRUPT HANDLER and requests use of the DTB whenever its MASTER or INTERRUPT HANDLER needs it.

ARBITER -- A functional module that accepts bus requests from REQUESTER modules and grants control of the DTB to one REQUESTER at a time.

UTILITY BUS -- One of the four buses provided by the VMEbus backplane. This bus includes signals that provide periodic timing and coordinate the power-up and powerdown of VMEbus systems.

SYSTEM CLOCK DRIVER -- A functional module that provides a 16 MHz timing signal on the Utility Bus.

SERIAL CLOCK DRIVER -- A functional module that provides a periodic timing signal that synchronizes operation of the VMSbus. (Although the VMEbus specification defines a SERIAL CLOCK DRIVER for use with the VMSbus, and although it reserves two backplane signal lines for use by that bus, the VMSbus protocol is completely independent of the VMEbus). Timing specifications for the SERIAL CLOCK DRIVER are given in Appendix C.

POWER MONITOR MODULE -- A functional module that monitors the status of the primary power source to the VMEbus system, and signals when that power has strayed outside the limits required for reliable system operation. Since most systems are powered by an AC source, the POWER MONITOR is typically designed to detect drop-out or brown-out conditions on AC lines.

SYSTEM CONTROLLER BOARD -- A board which resides in slot 1 of a VMEbus backplane and has a SYSTEM CLOCK DRIVER, an ARBITER, an JACK DAISY-CHAIN DRIVER, and a BUS TIMER. Some also have a SERIAL CLOCK DRIVER, a POWER MONITOR or both.

1.2.1.3 Types Of Cycles On The VMEbus.

READ CYCLE --A DTB cycle used to transfer 1, 2, 3, or 4 bytes from a SLAVE to a MASTER. The cycle begins when the MASTER broadcasts an address and an address modifier. Each SLAVE captures the address modifier and address, and checks to see if it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus and acknowledges the transfer. The MASTER then terminates the cycle.

WRITE CYCLE -- A DTB cycle used to transfer 1, 2, 3, or 4 bytes from a MASTER to a SLAVE. The cycle begins when the MASTER broadcasts an address and address modifier and places data on the DTB. Each SLAVE captures the address and address modifier and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The MASTER then terminates the cycle.

BLOCK READ CYCLE -- A DTB cycle used to transfer a block of 1 to 256 bytes from a SLAVE to a MASTER. This transfer is done using a string of 1, 2, or 4-byte data transfers. Once the block transfer is started, the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data for the next transfer is retrieved from the next higher location.

BLOCK WRITE CYCLE -- A DTB cycle used to transfer a block of 1 to 256 bytes from a MASTER to a SLAVE. The block write cycle is very similar to the block read cycle. It uses a

string of 1, 2, or 4-byte data transfers and the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data from the next transfer is stored in the next higher location.

READ-MODIFY-WRITE CYCLE -- A DTB cycle that is used to both read from, and write to a SLAVE location without permitting any other MASTER to access that location. This cycle is most useful in multiprocessing systems where certain memory locations are used to provide semaphore functions.

ADDRESS-ONLY CYCLE -- A DTB cycle that consists of an address broadcast, but no data transfer. SLAVES do not acknowledge ADDRESS-ONLY cycles and MASTERS terminate the cycle without waiting for an acknowledgment.

INTERRUPT ACKNOWLEDGE CYCLE -- A DTB cycle, initiated by an INTERRUPT HANDLER, that reads a STATUS/ID from an INTERRUPTER. An INTERRUPT HANDLER generates this cycle whenever it detects an interrupt request from an INTERRUPTER and it has control of the DTB.

1.2.2 Basic VMEbus Structure

The VMEbus interface system consists of backplane interface logic, four groups of signal lines called "buses", and a collection of "functional modules" which can be configured as required. The functional modules communicate with each other using the backplane signal lines.

The "functional modules" defined in this document are used as vehicles for discussion of the bus protocol, and need not be considered a constraint to logic design. For example, the designer might choose to design logic which interacts with the VMEbus in the manner described, but uses different on-board signals, or monitors other VMEbus signals. VMEbus boards might be designed to include any combination of the functional modules defined by this document.

The VMEbus functional structure can be divided into four categories. Each consists of a bus and its associated functional modules which work together to perform specific duties. Figure 1-2 shows the VMEbus functional modules and buses. Each category is briefly summarized below.

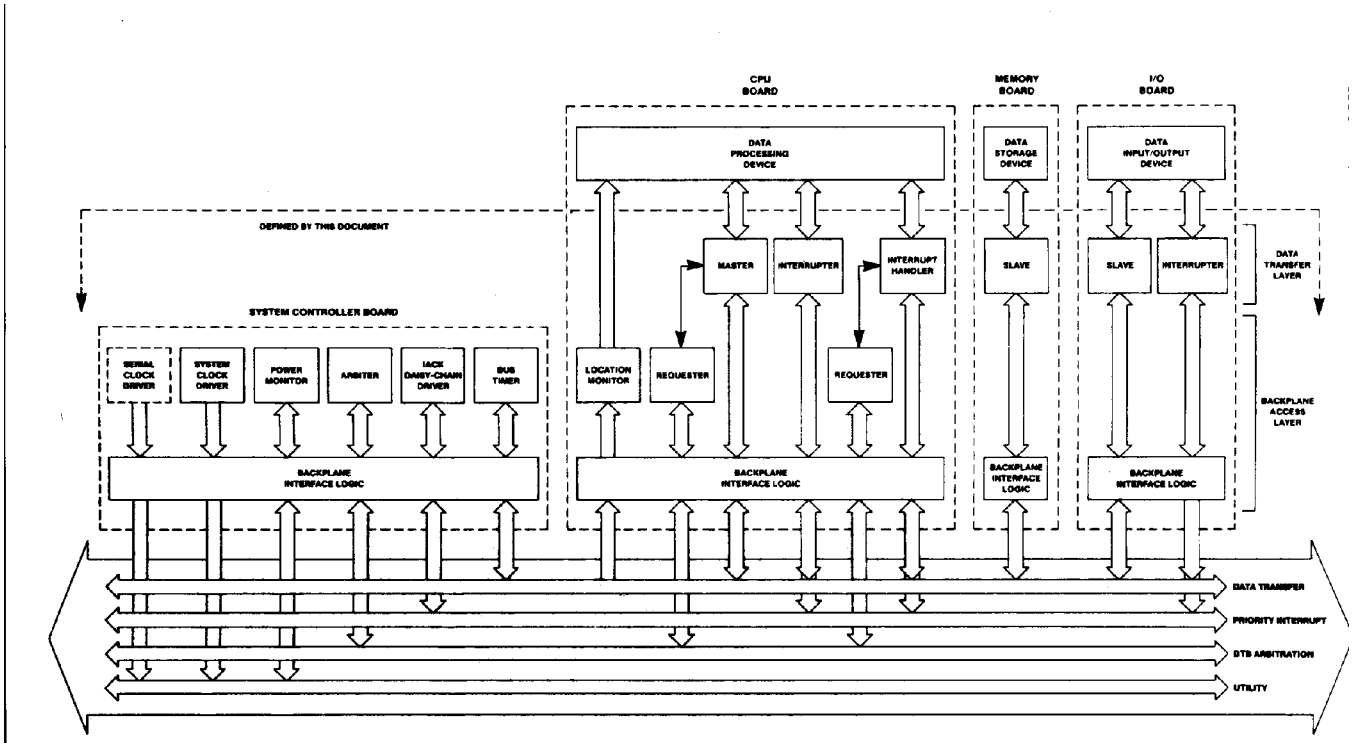


Figure 1-2. Functional Modules and Busses Defined by this Document

Data Transfer -- Devices transfer data over the Data Transfer Bus (DTB), which contains data and address pathways and associated control signals. Functional modules called MASTERS, SLAVES, INTERRUPTERS, and INTERRUPT HANDLERS use the DTB to transfer data between each other. Two other modules, called a BUS TIMER and an JACK DAISY-CHAIN DRIVER also assist them in this process.

DTB Arbitration -- Since a VMEbus system can be configured with more than one MASTER or INTERRUPT HANDLER, a means is provided to transfer control of the DTB between them in an orderly manner and to guarantee that only one controls the DTB at a given time. The Arbitration Bus modules (REQUESTERS and ARBITER) coordinate the control transfer.

Priority Interrupt -- The priority interrupt capability of the VMEbus provides a means by which devices can request service from an INTERRUPT HANDLER. These interrupt requests can be prioritized into a maximum of seven levels. INTERRUPTERS and INTERRUPT HANDLERS use the Priority Interrupt Bus signal lines.

Utilities -- Periodic clocks, initialization, and failure detection are provided by the Utility Bus. It includes two clock lines, a system reset line, a system fail line, an AC fail line, and a serial data line.

1.3 VMEbus SPECIFICATIONS DIAGRAMS

As aids to defining or describing VMEbus operation, several types of diagrams are used, including:

a. Timing diagrams that show the timing relationships between signal transitions. The times involved will have minimum and/or maximum limits associated with them. Some of the times specified on these diagrams specify the behavior of the backplane interface logic, while other times specify the interlocked behavior of the functional modules.
b. Sequence diagrams that are similar to a timing diagram but show only the interlocked timing relationships of the functional modules. This diagram is intended to show a sequence of events, rather than to specify the times involved. For example, a sequence diagram might indicate that module A cannot generate signal transition B until it detects module C's generation of signal transition D.
c. Flow diagrams that show a stream of events as they would occur during a VMEbus operation. The events are stated in words and result from interaction of two or more functional modules. The flow diagram describes VMEbus operations in a sequential manner and, at the same time, shows interaction 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 this document are labeled with keywords that indicate the type of information they contain. The keywords are listed below:

RULE
RECOMMENDATION
SUGGESTION
PERMISSION
OBSERVATION

Any text not labeled with one of these keywords describes the VMEbus structure or operation. It is written in either a descriptive or a narrative style. These keywords are used as follows:

RULE chapter.number:
Rules form the basic framework of the VMEbus specification They are sometimes expressed in text form and sometimes in the form of figures, tables, or drawings. All VMEbus rules MUST be followed to ensure compatibility between VMEbus designs. Rules are characterized by an imperative style. The upper-case words MUST and MUST NOT are reserved exclusively for stating rules in this document and are not used for any other purpose.

RECOMMENDATION chapter.number:
Wherever a recommendation appears, designers would be wise to take the advice given. Doing otherwise might result in some awkward problems or poor performance. While VMEbus has been designed to support high performance systems, it is possible to design a VMEbus system that complies with all the rules, but has abysmal performance. In many cases, a designer needs a certain level of experience with VMEbus 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:

In the VMEbus specification, a suggestion contains advice which is helpful but not vital. The reader is encouraged to consider the advice before discarding it. Some design decisions that need to be made in designing VMEbus boards are difficult until experience has been gained with the VMEbus. 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 compatibility with other boards, or with designing the board to make the job of system debugging easier.

PERMISSION chapter.number:
In some cases a VMEbus rule does not specifically prohibit a certain design approach, but the reader might be left wondering whether that approach might violate the spirit of the rule, or whether it might lead to some subtle problem. Permissions reassure the reader that a certain approach is acceptable, and will cause no problems. The uppercase 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 VMEbus rules and bring attention to things that might otherwise be overlooked. They also give the rationale behind certain rules, so that the reader understands why the rule must be followed.

1.4.1 Signal Line States

The VMEbus specification describes its protocol in terms of levels and transitions on bus lines.

A signal line is always assumed to be in one of two levels or in transition between these levels. Whenever the term "high" is used, it refers to a high TTL voltage level. The term "low"" refers to a low TTL voltage level. A signal line is ""in transition"' when its voltage is moving between these levels. (See Chapter 6 for voltage thresholds used on the VMEbus.)

There are two possible transitions which can appear on a signal line, and these are called “edges”. A rising edge is the time during which a signal level makes its transition from a low level to a high level. The falling edge is the time during which a signal level makes its transition from a high level to a low level.

Some bus specifications prescribe maximum or minimum rise and fall times for these edges. The problem with doing this is that board designers have very little control over these times. If the backplane is heavily loaded, the rise and fall times will be long. If it is lightly loaded" these times might be short. Even if designers know what the maximum and minimum loading will be, they still need to spend time in the lab, experimenting to find out which drivers will provide the needed rise and fall times.

In fact, rise and fall times are the result of a complex set of interactions involving the signal line impedances of the backplane, its terminations, the source impedance of the drivers, and the capacitive loading of the signal line. In order to trade off all of these factors the board designer would have to study transmission line theory, as well as certain specific parameters of drivers and receivers which are not normally found in most manufacturers data sheets.

Recognizing all of this, the VMEbus standard doesn't specify rise and fall times. Instead, it specifies the electrical characteristics for drivers and receivers and suggests how to design the backplane. It also tells designers how the worst case bus loading will affect the propagation delay of these drivers so that they can ensure that the VMEbus timing is met before building a board. If VMEbus designers follow these propagation delay guidelines, their boards will operate reliably with other VMEbus compatible boards under worst case conditions.

1.4.2 Use Of The Asterisk (*)

To help define usage, signal mnemonics have an asterisk suffix where required:

a. An asterisk (*) following the signal name of signals which are level significant denotes that the signal is true or valid when the signal is low.
b. An asterisk (*) following the signal name of signals which are edge significant denotes that the actions initiated by that signal occur on a high to low transition.

OBSERVATION 1.1:
The asterisk is inappropriate for the asynchronously running clock lines SYSCLK and SERCLK. There is no fixed phase relationship between these clock lines and other VMEbus timing.

1.5 PROTOCOL SPECIFICATION

There are two layers of VMEbus protocol. The lowest VMEbus layer called the backplane access layer, is composed of the backplane interface logic, the Utility Bus modules, and the Arbitration Bus modules. The VMEbus's data transfer layer, is composed of the Data Transfer Bus and Priority Interrupt Bus modules. Figure 1-2 shows this layering.

OBSERVATION 1.2:
The signal lines used by the data transfer layer modules form a special class because they are driven by different modules at different times. They are driven with line drivers that can be turned on and off at each board, based upon signals generated in the backplane access layer. 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. Special timing diagram notation is used in this document to specify their turn-on and turn-off times. It is shown in Figure 1-3.

There are two basic kinds of protocol used on the VMEbus: closed loop protocols and open loop protocols. Closed loop protocols use interlocked bus signals while open loop protocols use broadcast bus signals.

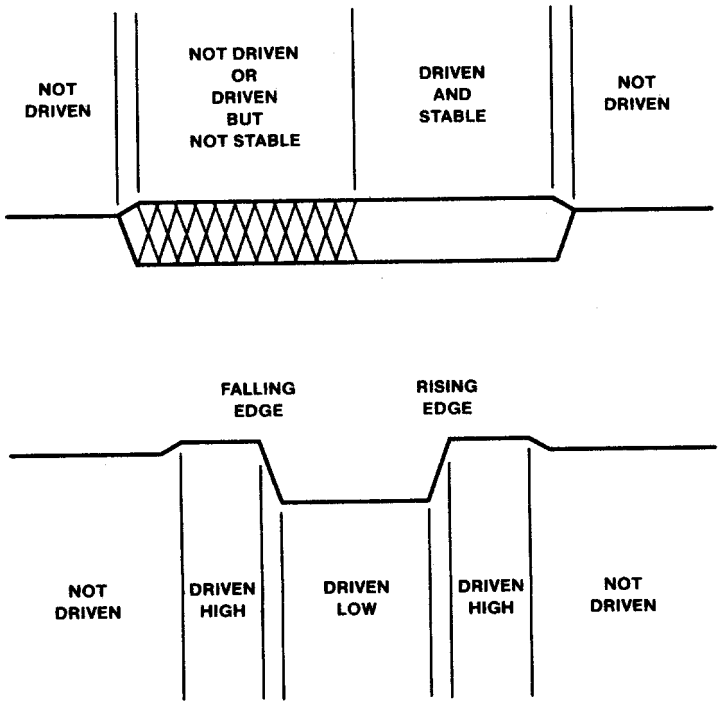


Figure 1-3. Signal Timing Notation

1.5.1 Interlocked Bus Signals

An interlocked bus signal is sent from a specific module to another specific module. The signal is acknowledged by the receiving module. An interlocked relationship exists between the two modules until the signal is acknowledged.

For example, an INTERRUPTER can send an interrupt request which is answered later with an interrupt acknowledge signal (no time limit is prescribed by the VMEbus specification). The INTERRUPTER doesn't remove the interrupt request until the INTERRUPT HANDLER acknowledges it.

Interlocked bus signals coordinate internal functions of the VMEbus system, as opposed to interacting with external stimuli. Each interlocked signal has a source module and a destination module within the VMEbus system.

The address strobe and data strobes are especially important interlocking signals. They are interlocked with the data transfer acknowledge and bus error signals and coordinate the transfer of addresses and data which are the basis for all information flow between modules in the data transfer layer.

1.5.2 Broadcast Bus Signal

A module generates a broadcast signal in response to an event. There is no protocol for acknowledging a broadcast signal. Instead, the broadcast is maintained for a minimum specified time, long enough to assure that all appropriate modules detect the signal. Broadcast

signals might be activated at any time, irrespective of any other activity taking place on the bus. They are each sent over a dedicated signal line. Some examples are the system reset and AC failure lines. These signal lines are not sent to any specific module, but announce special conditions to all modules.

1.6 SYSTEM EXAMPLES AND EXPLANATIONS

A protocol 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 is going on over the bus. To help the reader, the VMEbus specification provides examples of typical VMEbus operations. Each example shows one possible sequence of events: other sequences are also possible. In providing these examples, there is the danger that readers will assume that the sequence shown in the example is the only legal one. To help readers avoid this trap, all examples are given in a narrative style, using the present tense. This is in contrast to the imperative style used when giving rules for compliance with the VMEbus specification.

CHAPTER 2

DATA TRANSFER BUS

2.1 INTRODUCTION

The VMEbus includes a high speed asynchronous parallel Data Transfer Bus (DTB). Figure 2-1 shows a typical VMEbus system, including all of the DTB functional module types. MASTERS use the DTB to select storage 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.

LOCATION MONITORS monitor data transfers between MASTERS and SLAVES. When an access is done to one of the byte location(s) that it monitors, a LOCATION MONITOR generates an on-board signal. For example, it might signal its on-board processor by means of an interrupt request. In such a configuration, if processor board A writes into a location of the global VMEbus memory that is monitored by processor B's LOCATION MONITOR, processor B will be interrupted.

After a MASTER initiates a data transfer cycle it waits for the responding SLAVE to respond before finishing the cycle. The asynchronous definition of the VMEbus allows a SLAVE to take as long as it needs to respond. If a SLAVE fails to respond because of some malfunction, or if the MASTER accidentally addresses a location where there is no SLAVE, the BUS TIMER intervenes, allowing the cycle to be terminated.

2.2 DATA TRANSFER BUS LINES

The Data Transfer Bus lines can be grouped into three categories:

Addressing Lines	Data Lines	Control Lines
A01 -A31	D00-D31	AS*
AM0-AM5		DS0*
DS0*		DS1 *
DS1 *		BERR*
LWORD*		DTAC K*
		WRITE*

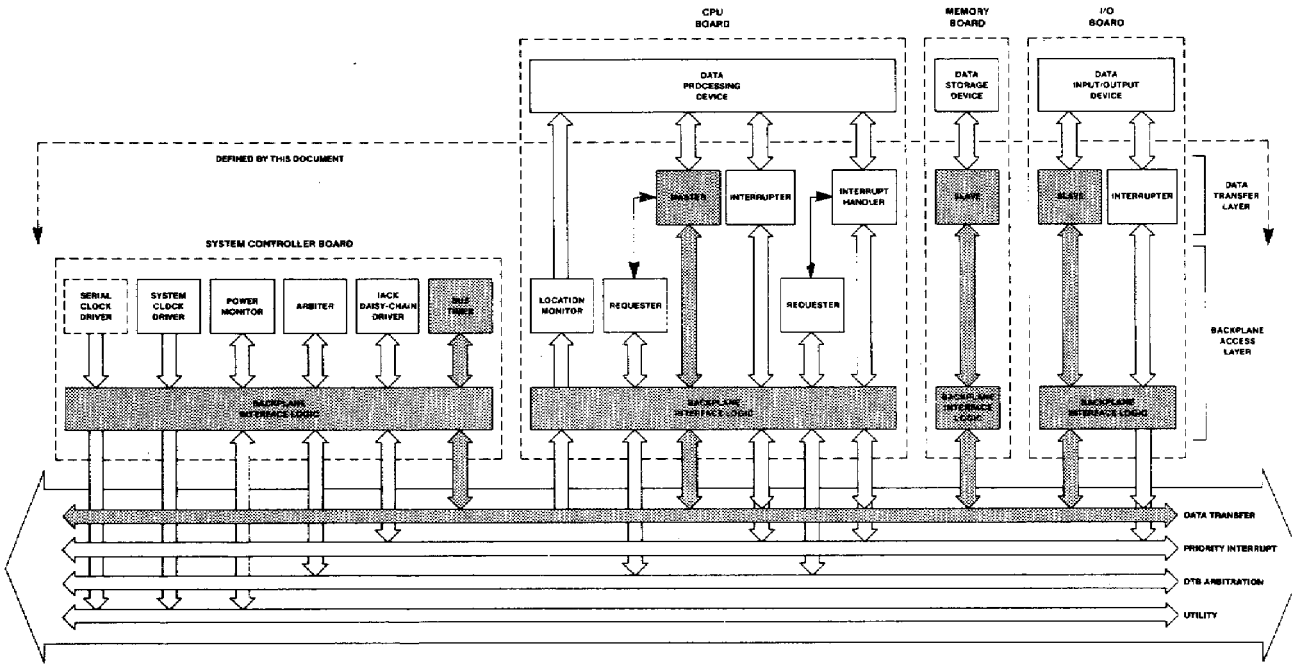


Figure 2-1. Data Transfer Bus Functional Block Diagram

OBSERVATION 2.1:
The two data strobes (DS0* and DS1*) serve a dual function:

a. The levels of these two data strobe lines are used to select which byte(s) are accessed.

b. The edges of the data strobes are also used as timing signals which coordinate the transfer of the data between the MASTER and SLAVE.

2.2.1 Addressing Lines

The smallest addressable unit of storage is the byte location. Each byte location is assigned a unique binary address. Each byte location can be assigned to one of four categories, according to the the least significant two bits of its address. (See Table 2-1)

Table 2-1. The Four Categories Of Byte Locations

Category	Byte address
BYTE(0)	XXXXXX…XXXXXX00
BYTE(1)	XXXXXX…XXXXXX01
BYTE(2)	XXXXXX…XXXXXX10
BYTE(3)	XXXXXX…XXXXXX11

A set of byte locations whose addresses differs only in the two least significant bits, is called a 4-byte group or a BYTE(0-3) group. Some, or all of the bytes in a 4-byte group can be accessed simultaneously by a single DTB cycle.

MASTERS use address lines A02-A31 to select which 4-byte group will be accessed. Four additional lines (DS1*, DS0*, A01, and LWORD*) are then used to select which byte location(s) within the 4-byte group are accessed during the data transfer. Using these four lines, a MASTER can access 1, 2, 3, or 4-byte locations simultaneously, depending upon the type of cycle. The 34 possible cycle types with the corresponding levels of these four lines are listed in Table 2-2 .

OBSERVATION 2.2:
In cycles where both data strobes are driven low, one data strobe might go low slightly after the other. In this case the signal levels indicated in Table 2-2 are the final levels.

OBSERVATION 2.3:
Given the 4 signal line levels shown in Table 2-2, there are 16 possible combinations of levels. Of these 16, there are two illegal combinations that are not used (see Table in RULE 2.1).

RULE 2.1:
MASTERS **MUST NOT** generate DTB cycles where the final levels of DS0*, DS1*, A01, and LWORD* are either of the following illegal combinations:

DS1*	DS0*	A01	LWORD*
high	low	high	low
low	high	high	low

PERMISSION 2.1:
MASTERS that generate BYTE(1-2) READ or BYTE(1-2) WRITE cycles **MAY** generate either of the two combinations described in RULE 2.1 briefly as transition states (i.e. while one data strobe has fallen, but the other has not).

OBSERVATION 2.4:
Whenever a MASTER drives LWORD* low and A01 high it drives both data strobes low. (Any other combination is illegal.) VMEbus board designers can take advantage of this to simplify the logic on SLAVES.

PERMISSION 2.2:
To simplify the required logic, SLAVES which respond to BYTE(1-2) READ and BYTE(1-2) WRITE cycles **MAY** be designed without logic to distinguish between these cycles and the two illegal cycles described in RULE 2.1.

Table 2-2. Signal Levels Used To Select Which Byte Location(s) Are Accessed During A Data Transfer

Type of cycle	DS1 *	DS0*	A01	LWORD*
ADDRESS-ONLY	high	high	<----Notel----->	
Single even byte transfers				
BYTE(0) READ or WRITE	low	high	low	high
BYTE(2) READ or WRITE	low	high	high	high
Single odd byte transfers				

BYTE(1) READ or WRITE	high	low	low	high
BYTE(3) READ or WRITE	high	low	high	high
Double byte transfers				
BYTE(0-1) READ or WRITE	low	low	low	high
BYTE(2-3) READ or WRITE	low	low	high	high
Quad byte transfers				
BYTE(0-3) READ or WRITE	low	low	low	low
Single byte block transfers				
SINGLE BYTE BLOCK READ or WRITE	<------Note 2------>			high
Double byte block transfers				
DOUBLE BYTE BLOCK READ or WRITE	low	low	Note 3	high
Quad byte block transfers				
QUAD BYTE BLOCK READ or WRITE	low	low	low	low
Single byte RMW transfers				
BYTE(0) READ-MODIFY-WRITE	low	high	low	high
BYTE(1) READ-MODIFY-WRITE	high	low	low	high
BYTE(2) READ-MODIFY-WRITE	low	high	high	high
BYTE(3) READ-MODIFY-WRITE	high	low	high	high
Double byte RMW transfers				
BYTE(0-1) READ-MODIFY-WRITE	low	low	low	high
BYTE(2-3) READ-MODIFY-WRITE	low	low	high	high
Quad byte RMW transfers				
BYTE(0-3) READ-MODIFY-WRITE	low	low	low	low
Unaligned transfers				
BYTE(0-2) READ or WRITE	low	high	low	low
BYTE(1-3) READ or WRITE	high	low	low	low
BYTE(1-2) READ or WRITE	low	low	high	low

Table 2-2. Signal Levels Used To Select Which Byte Location(s) Are Accessed During A Data Transfer

Notes:

1. During ADDRESS-ONLY cycles, both data strobes are maintained high, but the A01 and LWORD* lines might be either high or low.

2. During single byte block transfers, the two data strobes are alternately driven low. Either data strobe might be driven low on the first transfer. If the first accessed byte location is BYTE(0) or BYTE(2)" then DS1* is driven low first. If the first accessed byte location is BYTE(1) or BYTE(3), then DS0* is driven low first. A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and might be either high or low depending upon which byte the single byte block transfer begins with. If the first byte location is BYTE(0) or BYTE(1), then A01 is low. If the first byte location is BYTE(2) or BYTE(3), then A01 is high.

An example of a Single byte block transfer cycle which starts with BYTE(2) is given below:

	DS1*	DS0*	A01	LWORD*
First data transfer: BYTE(2)	low	high	high	high
BYTE(3)	high	low	X	X
BYTE(0)	low	high	X	X
BYTE(1)	high	low	X	X
Last data transfer: BYTE(2)	low	high	X	X

X=high or low.

3. During a Double byte block transfer, the two data strobes are both driven low on each data transfer. A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and might be either high or low depending upon what double byte group the double byte block transfer begins with. If the first double byte group is BYTE(0-1), then A01 is low. If the first double byte group is BYTE(2-3), then A01 is high.

An example of a Double byte block transfer cycle which starts with BYTE(2-3) is given below:

	DS1*	DS0*	A01	LWORD*
First data transfer BYTE(2-3)	low	low	high	high
BYTE(0-1)	low	low	X	X
BYTE(2-3)	low	low	X	X
Last data transfer BYTE(0-1)	low	low	X	X

X = high or low.

2.2.2 Address Modifier Lines

There are 6 address modifier lines. They allow the MASTER to pass additional binary information to the SLAVE during data transfers. Table 2-3 lists all of the 64 possible address modifier (AM) codes and classifies each into one of three categories:

Defined
Reserved
User defined

The defined address modifier codes can be further classified into three categories:

Short addressing AM codes indicate that address lines A02-A15 are being used to select a BYTE(0-3) group.

Standard addressing AM codes ,indicate that address lines A02-A23 are being used to select a BYTE(0-3) group.

Extended addressing AM codes indicate that address lines A02-A31 are being used to select a BYTE(0-3) group.

RULE 2.2:
Except for the User defined codes" the codes defined in Table 2-3 **MUST NOT** be used for purposes other than those specified.

RULE 2.3:
VMEbus SLAVE boards **MUST NOT** respond to reserved address modifier codes.

OBSERVATION 2.5:
Reserved address modifier codes are for future enhancements. If SLAVE boards respond to these codes" incompatibilities might result at some future date" when usage of these codes is defined.

PERMISSION 2.3:
User defined codes **MAY** be used for any purpose which board vendors or board users deem appropriate (page switching" memory protection" MASTER or task identification, privileged access to resources, etc.)

Table 2-3. Address Modifier Codes

HEX CODE	ADDRESS MODIFIER 5 4 3 2 1 0	FUNCTION
3F	H H H H H H	Standard Supervisory Block Transfer
3E	H H H H H L	Standard Supervisory Program Access
3D	H H H H L H	Standard Supervisory Data Access
3C	H H H H L L	Reserved
3B	H H H L H H	Standard Non-Privileged Block Transfer
3A	H H H L H L	Standard Non-Privileged Program Access
39	H H H L L H	Standard Non-Privileged Data Access
38	H H H L L L	Reserved
37	H H L H H H	Reserved
36	H H L H H L	Reserved
35	H H L H L H	Reserved
34	H H L H L L	Reserved
33	H H L L H H	Reserved
32	H H L L H L	Reserved
31	H H L L L H	Reserved
30	H H L L L L	Reserved
2F	H L H H H H	Reserved
2E	H L H H H L	Reserved
2D	H L H H L H	Short Supervisory Access
2C	H L H H L L	Reserved
2B	H L H L H H	Reserved

2A	H L H L H L	Reserved
29	H L H L L H	Short Non-Privileged Access
28	H L H L L L	Reserved
27	H L L H H H	Reserved
26	H L L H H L	Reserved
25	H L L H L H	Reserved
24	H L L H L L	Reserved
23	H L L L H H	Reserved
22	H L L L H L	Reserved
21	H L L L L H	Reserved
20	H L L L L L	Reserved
1F	L H H H H H	User defined
1E	L H H H H L	User defined
1D	L H H H L H	User defined
1C	L H H H L L	User defined
1B	L H H L H H	User defined
1A	L H H L H L	User defined
19	L H H L L H	User defined
18	L H H L L L	User defined
17	L H L H H H	User defined
16	L H L H H L	User defined
15	L H L H L H	User defined
14	L H L H L L	User defined
13	L H L L H H	User defined
12	L H L L H L	User defined
11	L H L L L H	User defined
10	L H L L L L	User defined
OF	L L H H H H	Extended Supervisory Block Transfer
OE	L L H H H L	Extended Supervisory Program Access
OD	L L H H L H	Extended Supervisory Data Access
OC	L L H H L L	Reserved
OB	L L H L H H	Extended Non-Privileged Block Transfer
OA	L L H L H L	Extended Non-Privileged Program Access
09	L L H L L H	Extended Non-Privileged Data Access
08	L L H L L L	Reserved
07	L L L H H H	Reserved
06	L L L H H L	Reserved
05	L L L H L H	Reserved
04	L L L H L L	Reserved

03	L L L L H H	Reserved
02	L L L L H L	Reserved
01	L L L L L H	Reserved
00	L L L L L L	Reserved

L= low signal level H = high signal level

RECOMMENDATION 2.1:
To allow VMEbus users to tailor the use of the user defined address modifier codes to their own needs, decode them in a flexible way on SLAVE boards. Users can then configure the board to give any decoding required by their system.

OBSERVATION 2.6:
Socketed PROMS and FPLAS provide a flexible way for decoding the address modifier codes.

SUGGESTION 2.1:
Where SLAVES are manufactured with a programmed device (e.g. a PROM or a FPLA) installed in the socket, program the device so that the SLAVE responds to the following AM codes:

A16 SLAVES with D08(0), D08(EO), D16, or D32 capability: 29, 2D
A24 SLAVES with D08(0), D08(EO), D16, or D32 capability: 39, 3A, 3D, and 3E
A32 SLAVES with D08(0), D08(EO), D16, or D32 capability: 09, OA, OD, and OE
A24 SLAVES with BLT capability: 3B, 3F
A32 SLAVES with BLT capability: OB, OF

The mnemonics A16, A24, and A32 are defined in Table 2-9. The mnemonics D08(0), D08(EO), D16, D32, and BLT are defined in Tables 2-10, and 2-11.

2.2.3 Data Lines

VMEbus systems can be built with a backplane configuration that provides either 16 data lines (D00-D15), or 32 data lines (D00-D31). Backplane configurations that provide 16 data lines allow a MASTER to access only two byte locations simultaneously, while those with 32 data lines allow up to four byte locations to be accessed. When the MASTER has selected 1, 2, 3, or 4 byte locations, using the method described in Section 2.2.1, it can transfer binary data between itself and those locations over the data bus. Table 2-4 shows how the data lines are used to move data during each of the 34 cycle types.

PERMISSION 2.4:
The data sender (MASTER for a write cycle; SLAVE for a read cycle) may drive data lines which are not used to transfer data.

Table 2-4. Use Of The Data Lines To Move Data During Each Of The 34 Cycle Types

During the following types of cycles...	the data lines are used to transfer data as shown below:

	D24-D31	**D16-D23**	**D08-D15**	**D00-D07**
ADDRESS-ONLY	<---------- no bytes transferred-------->			
Single even byte transfers				
BYTE(0) READ or WRITE			BYTE(0)	
BYTE(2) READ or WRITE			BYTE(2)	
Single odd byte transfers				
BYTE(1) READ or WRITE				BYTE(1)
BYTE(3) READ or WRITE				BYTE(3)
Double byte transfers				
BYTE(0-1) READ or WRITE			BYTE(0)	BYTE(1)
BYTE(2-3) READ or WRITE			BYTE(2)	BYTE(3)
Quad byte transfers				
BYTE(0-3) READ or WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)
Single byte block transfers				
SINGLE BYTE BLOCK READ or WRITE			<---- Note 1 ----->	
Double byte block transfers				
DOUBLE BYTE BLOCK READ or WRITE			<----- Note 2 ----->	
Quad byte block transfers				
QUAD BYTE BLOCK READ or WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)
Single byte RMW transfers				
BYTE(0) READ-MODIFY-WRITE			BYTE(0)	
BYTE(1) READ-MODIFY-WRITE				BYTE(1)
BYTE(2) READ-MODIFY-WRITE			BYTE(2)	
BYTE(3) READ-MODIFY-WRITE				BYTE(3)
Double byte RMW transfers				
BYTE(0-1) READ-MODIFY-WRITE			BYTE(0)	BYTE(1)
BYTE(2-3) READ-MODIFY-WRITE			BYTE(2)	BYTE(3)
Quad byte RMW transfers				
BYTE(0-3) READ-MODIFY-WRITE	BYTE(0)	BYTE(1)	BYTE(2)	BYTE(3)
Unaligned transfers				
BYTE(0-2) READ or WRITE	BYTE(0)	BYTE(1)	BYTE(2)	
BYTE(1-3) READ or WRITE		BYTE(1)	BYTE(2)	BYTE(3)
BYTE(1-2) READ or WRITE		BYTE(1)	BYTE(2)	

Notes:

1. During Single byte block transfers, data is transferred 8 bits at a time over D00-D07 or D08-D15. One example of this is given below:

	D24-D31	**D1 6-D23**	**D08-D15**	**D00-D07**
First data transfer				BYTE(1)

			BYTE(2)	
				BYTE(3)
			BYTE(0)	
				BYTE(1)
			BYTE(2)	
Last data transfer				BYTE(3)

2. During a Double byte block transfer, data is transferred 16 bits at a time over D00-D15. One example of this is given below:

	D24-D31	**D16-D23**	**D08-D15**	**D00-D07**
First data transfer			BYTE(2)	BYTE(3)
			BYTE(0)	BYTE(1)
			BYTE(2)	BYTE(3)
			BYTE(0)	BYTE(1)
			BYTE(2)	BYTE(3)
Last data transfer			BYTE(0)	BYTE(1)

2.2.4 Data Transfer Bus Control Lines

The following signal lines are used to control the movement of data over the data transfer lines:

AS*	Address Strobe
DS0*	Data Strobe Zero
DS1*	Data Strobe One
BERR*	Bus Error
DTACK*	Data Transfer Acknowledge
WRITE*	Read/Write

2.2.4.1 AS*

A falling edge on this line informs all SLAVE modules that the address is stable and can be captured.

2.2.4.2 DS0* And DS1*

In addition to their function in selecting byte locations for data transfer, as described in Section 2.2.1, the data strobes also serve additional functions. On write cycles, the first data strobe failing edge indicates when the MASTER has placed valid data on the data bus. On read cycles, the first rising edge tells the SLAVE when it can remove valid data from the data bus.

OBSERVATION 2.7:
VMEbus MASTERS are not permitted to drive either of the data strobes low before driving AS* low. However, due to the fact that AS* might be more heavily loaded on the backplane than the

data strobes, SLAVES and LOCATION MONITORS might detect a falling edge on a data strobe, before they detect the falling edge on AS*.

PERMISSION 2.5:
VMEbus SLAVES and LOCATION MONITORS **MAY** be designed to capture the address when they detect a falling edge on a data strobe instead of on the falling edge

OBSERVATION 2.8:
VMEbus SLAVES and LOCATION MONITORS that capture the address on the falling edge of the data strobes) need not monitor AS*.

OBSERVATION 2.9:
In order to take full advantage of address pipelining as described in Section 2.4.2, or to perform block read and write cycles, a SLAVE should capture the address on the falling edge of AS*.

2.2.4.3 DTACK*

The SLAVE drives DTACK* low to indicate that it has successfully received the data on a write cycle. On a read cycle, the SLAVE drives DTACK* low to indicate that it has placed data on the data bus.

2.2.4.4 BERR*

BERR* is driven low by the SLAVE or by the BUS TIMER to indicate to the MASTER that the data transfer was unsuccessful. For example, if a MASTER tries to write to a location which contains Read-Only Memory, the responding SLAVE might drive BERR* low. If the MASTER tries to access a location that is not provided by any SLAVE, the BUS TIMER would drive BERR* low after waiting a specified period.

OBSERVATION 2.10:
The BERR* line is a convenience which is useful when debugging VMEbus systems. It also allows system failures to be detected quickly during normal operation. Not all VMEbus systems will need this capability.

PERMISSION 2.6:
VMEbus SLAVES **MAY** be designed without a driver for BERR*.

SUGGESTION 2.2:
Design SLAVES to respond with a falling edge on BERR* in the following situations:

a. When a D08(0), D08(EO), or D16 SLAVE is requested to do a quad byte cycle.
b. When a D08(0) or D08(EO) SLAVE is requested to do a double byte cycle.
c. When a non-UAT SLAVE is requested to do an unaligned transfer. (i.e. A triple byte transfer or a double byte BYTE(1-2) transfer.)
d. When a SLAVE detects an uncorrectable error in the data it retrieves from its internal storage during a read cycle.

The mnemonics D08(0), D08(EO), D16, and UAT are defined in Tables 2-10 and 2-15.

RULE 2.4:
D08(0), D08(EO), and D16 SLAVES **MUST NOT** respond with a falling edge on DTACK* during a quad byte cycle if they do not have quad byte capability.

RULE 2.5:
D08(0) and D08(EO) SLAVES **MUST NOT** respond with a falling edge on DTACK* during a double byte cycle if it does not have double byte capability.

RULE 2.6:
A SLAVE **MUST NOT** respond with a falling edge on DTACK* during an unaligned transfer cycle, if it does not have UAT capability.

2.2.4.5 WRITE*

WRITE* is a level significant signal line strobed by the falling edge of the first data strobe. It is used by the MASTER to indicate the direction of data transfer operations. When WRITE* is driven low, the data transfer direction is from the MASTER to the SLAVE. When WRITE* is driven high, the data transfer direction is from the SLAVE to the MASTER.

2.3 DTB MODULES - BASIC DESCRIPTION

In addition to the ADDRESS-ONLY cycle, the DTB protocol defines 33 different cycle types that can be used to transfer data. Each of these 34 cycle types can be used in any of three addressing modes: short (16-bit), standard (24- bit), and extended (32-bit). The capabilities of MASTERS, SLAVES and LOCATION MONITORS are described by a list of mnemonics that show what cycle types they can generate, accept, or monitor respectively. (This will be described in more detail later in this chapter.)

Sections 2.3.1 through 2.3.4 provide block diagrams for the four types of DTB functional modules: MASTER, SLAVE, LOCATION MONITOR, and BUS TIMER.

RULE 2.7:
Output signal lines shown with solid lines in Figures 2-2 through 2-5 **MUST** be driven by the module, unless it would always drive them high.

OBSERVATION 2.11:
IF an output signal line is not driven,
THEN terminators on the backplane ensure that it is high.

RULE 2.8:
Input signal lines shown with solid lines in Figures 2-2 through 2-5 **MUST** be monitored and responded to in the appropriate fashion.

OBSERVATION 2.1 2:

RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figures 2-2 through 2-5 are given in Tables 2-5, 2-6, and 2-8.

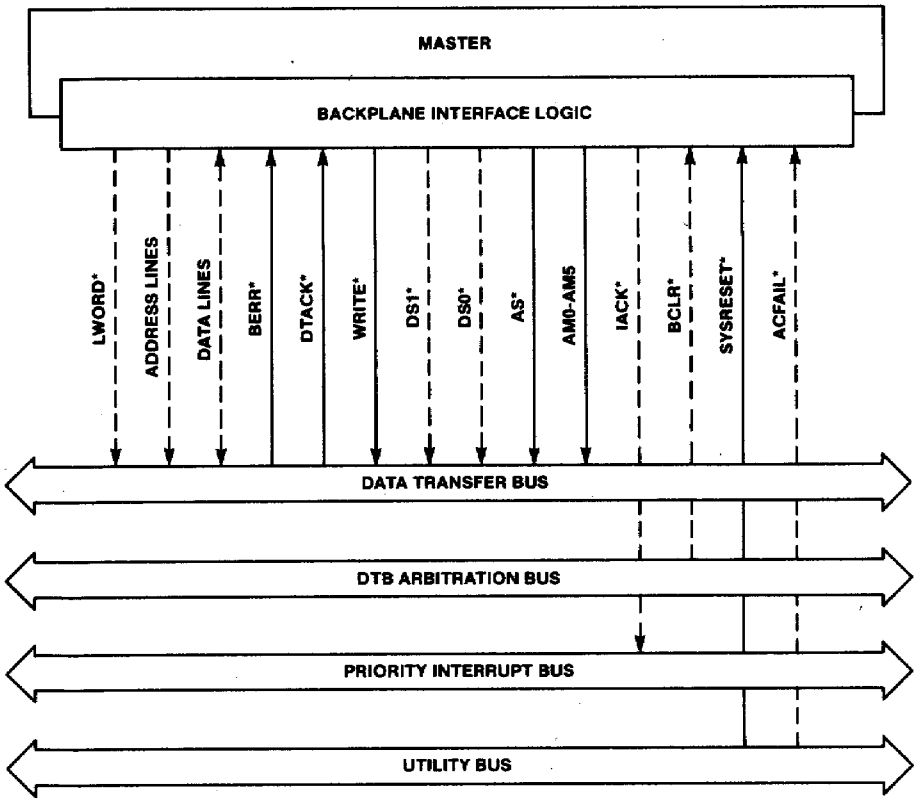


Figure 2-2. Block Diagram: MASTER

2.3.1 MASTER

The block diagram of the MASTER is shown in Figure 2-2. The dotted lines in the diagram show signals whose use varies among the various types of MASTERS. Table 2-5 specifies how the various types of MASTERS use these lines. Further information about how the various types of MASTERS drive the address lines, the data lines, LWORD*, DS0*, and DS1* is given in Tables 2-19, 2-20, 2-21.

Table 2-5. RULES And PERMISSIONS That Spesify The Use of The Dotted Lines By The Various Types of MASTERS

Type of MASTER	Use of dotted lines
D08(EO)	**MUST** drive DS0* and DS1*, but not both low on the same data transfer. **MUST** monitor and drive D00-D15. **MUST NOT** drive IACK* low. **MAY** or **MAY** not drive LWORD*. **MAY** or **MAY** not drive or monitor D16-D31.

D16	**MUST** drive DS0* and DS1*. **MUST** monitor and drive D00-D15. **MUST NOT** drive IACK* low. **MAY** or **MAY** not drive LWORD*. **MAY** or **MAY** not drive or monitor D16-D31.
D32	**MUST** drive DS0*" DS1*, and LWORD*. **MUST** monitor and drive D00-D31. **MUST NOT** drive IACK* low.
A16	**MUST** drive A01 -A15. **MAY** or **MAY** not drive A16-A31
A24	**MUST** drive A01-A23. **MAY** or **MAY** not drive A24-A31.
A32	**MUST** drive A01-A31.
ALL	**MAY** or **MAY** not monitor BCLR*" or ACFAIL* (see Chapters 3 and 5).

Notes:
1. The mnemonics D08(EO), D16, and D32 are defined in Table 2-10.
2. The mnemonics A16, A24, and A32 are defined in Table 2-9.

2.3.2 SLAVE

The block diagram of the SLAVE is shown in Figure 2-3. The dotted lines in the diagram show signals whose use varies among the various types of SLAVES. Table 2-6 shows how the various types of SLAVES use these lines. Further information about how the various types of SLAVES drive the data lines is given in Table 2-22.

Table 2-6. 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(0)	**MUST** monitor and drive D00-D07. **MAY** or **MAY** not monitor AS*. **MAY** or **MAY** not monitor or drive D08-D31.
D08(EO)	**MUST** monitor and drive D00-D07. **MAY** or **MAY** not monitor AS*. **MAY** or **MAY** not monitor or drive D16-D31.
D16	**MUST** monitor and drive D00-D15. **MAY** or **MAY** not monitor AS*. **MAY** or **MAY** not monitor or drive D16-D31.
D32	**MUST** monitor and drive D00-D31. **MAY** or **MAY** not monitor AS*.
A16	**MUST** monitor A01-A15. **MAY** or **MAY** not monitor A16-A31.
A24	**MUST** monitor A01-A23. **MAY** or **MAY** not monitor A24-A31.

A32	**MUST** monitor A01-A31.
ALL	**MAY** or **MAY** not drive BERR*.

Notes :
1. The mnemonics D08(0), D08(EO), D16, and D32 are defined in Table 2-10.
2. The mnemonics A16, A24, and A32 are defined in Table 2-9.

2.3.3 BUS TIMER

The block diagram of the BUS TIMER is shown in Figure 2-4. BUS TIMERS can be designed to drive BERR* low after various periods of time. Table 2-7 shows how the BTO( ) mnemonic is used to describe the various types of BUS TIMERS.

OBSERVATION 2.13:
The dotted DTACK* and BERR* lines shown in Figure 2-4 allow to implement a BUS TIMER in one of two ways:

a. To drive BERR* low when the first data strobe stays low for longer than the bus time-out period, regardless of the levels on the DTACK* and BERR* lines.

b. To drive BERR* low when the first data strobe stays low for longer than the bus time-out period, but only if both DTACK* and BERR* are high at the point of time-out.

Table 2-7. Use Of The BTO( ) Mnemonic To Specify The Time-Out Period Of BUS TIMERS

The Following Mnemonic	When Applied to a	Means that it
BTO(x)	BUS TIMER	drives BERR* low if the first data strobe stays low for longer than x microseconds

2.3.4 LOCATION MONITOR

The block diagram of the LOCATION MONITOR is shown in Figure 2-5. The dotted lines in the diagram show signals whose use varies among the various types of LOCATION MONITORS. Table 2-8 shows how the various types of LOCATION MONITORS use these lines.

Table 2-8. RULES and PERMISSIONS That Specify The Use Of The Dotted Lines By The Various Types Of LOCATION MONITORS

Type of LOCATION MONITOR	Use of dotted lines
A16	**MUST** monitor A01-A15. **MAY** or **MAY** not monitor A16-A31.
A24	**MUST** monitor A01-A23.

	MAY or **MAY** not monitor A24-A31.
A32	**MUST** monitor A01-A31.
ALL	**MAY** or **MAY** not monitor AS*.

Note:
The mnemonics A16, A24, and A32 are defined in Table 2-9.

2.3.5 Addressing Modes

MASTERS broadcast an address over the DTB at the beginning of each cycle. This broadcasted address might be a 16, 24 or 32-bit address, depending upon the capabilities of the MASTER broadcasting it. The 16-bit addresses are “short addresses” the 24-bit addresses are “standard addresses” and the 32-bit addresses are “extended addresses”.

Table 2-9 shows the various mnemonics used to describe the addressing capabilities and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-9. Mnemonics That Specify Addressing Capabilities

The Following Mnemonic	When Applied to a	Means that it
A16	MASTER	can generate cycles with short (16 bit) addresses.
	SLAVE	can accept cycles with short (16 bit) addresses
	LOCATION MONITOR	can monitor cycles with short (16 bit) addresses.
A24	MASTER	can generate cycles with standard (24 bit) addresses.
	SLAVE	can accept cycles with standard (24 bit) addresses.
	LOCATION MONITOR	can monitor cycles with standard (24 bit) addresses.
A32	MASTER	can generate cycles with extended (32 bit) addresses
	SLAVE	can accept cycles with extended (32 bit) addresses.
	LOCATION MONITOR	can monitor cycles with extended (32 bit) addresses.

The MASTER broadcasts an Address Modifier (AM) code along with each address to tell SLAVES whether the address is short, standard, or extended.

Short addresses are generated by A16 MASTERS, and accepted by A16 SLAVES. Standard addresses are generated by A24 MASTERS, and accepted by A24 SLAVES. Extended addresses are generated by A32 MASTERS, and accepted by A32 SLAVES.

Short addressing is intended primarily for addressing I/O devices. It allows A16 SLAVES to be designed with less logic, since they do not have to decode as many address lines. While I/O boards can be designed to decode standard addresses and extended addresses, short addressing usually makes this unnecessary.

Standard and extended addressing modes are intended primarily for addressing memory, although there is no rule against designing I/O boards that also respond to these addressing modes. Standard and Extended addressing modes allow much larger addressing ranges.

RULE 2.9:
SLAVE boards **MUST** decode all of the address modifier lines.

OBSERVATION 2.14:
RULE 2.9 allows a SLAVE to differentiate short addresses, standard addresses, and extended addresses.

OBSERVATION 2.15:
In addition to the three modes of addressing described here, there is a fourth mode which is used on interrupt acknowledge cycles (see chapter 4). These interrupt acknowledge cycles can be distinguished from data transfer cycles by the fact the the IACK* signal line is low instead of high.

RULE 2.10:
Whenever a MASTER broadcasts an address over the address bus, it **MUST** ensure that IACK* is high.

PERMISSION 2.7:
A MASTER **MAY** either drive IACK* high during the address broadcast or it **MAY** leave IACK* undriven. (The bus terminators will then hold it high.)

RULE 2.11:
SLAVES **MUST NOT** respond to DTB cycles when IACK* is low.

RECOMMENDATION 2.2:
Since many systems will include a mixture of A16, A24, and A32 SLAVES, include A16 and A24 capability on 32-bit CPU cards in addition to the A32 capability, and include A16 capability on CPU cards with A24 capability.

2.3.6 Basic Data Transfer Capabilities

There are four basic data transfer capabilities associated with the DTB: D08(EO) (Even and Odd byte), D08(0) (Odd byte only), D16, and D32. These capabilities allow flexibility when interfacing different types of processors and peripherals to the bus.

Eight-bit processors can be interfaced to the bus as D08(EO) MASTERS. Sixteen-bit processors can be interfaced to the bus as D16 MASTERS. The D16 SLAVE module is useful for interfacing 16 bit memory devices or 16 bit I/O SLAVES to the DTB.

Many existing peripheral chips have registers that are only 8 bits wide. While these chips often have several of these registers, they cannot provide the contents of two registers simultaneously when a D16 MASTER attempts to access two adjacent locations with a double byte read cycle. These 8-bit peripheral ICs can be interfaced to the DTB as a D08(0) SLAVE. D08(0) SLAVES provide only BYTE(1) or BYTE(3) locations and respond only to single byte accesses. (Since single byte accesses to these odd byte locations always take place over D00-D07, this simplifies the D08(0) SLAVE'S interface logic.)

RECOMMENDATION 2.3:
Since most 16-bit microprocessors can also access memory 8 bits at a time, include D08(EO) MASTER capability on 16-bit CPU boards in addition to the D16 capability. In addition to allowing 8 bit data transfers to and from memory, this also allows them to access D08(0) SLAVES.

RECOMMENDATION 2.4:
Since most 32-bit microprocessors can also transfer data to and from memory 8 and 16 bits at a time, include D08(EO) and D16 capability on 32-bit CPU boards in addition to D32 capability. The D08(EO) capability not only allows 8-bit data transfers to and from memory, but it also allows the MASTER to access D08(0) SLAVES.

OBSERVATION 2.16:
It might seem logical to define "even byte only" SLAVES which respond to the even byte memory locations adjacent to the D08(0) SLAVES. But" this cannot be done because there is only one data transfer acknowledge line. If a MASTER were to select both an even byte and an odd byte location simultaneously" by doing a double byte transfer, both SLAVES would drive the same data acknowledge (DTACK*) line and the MASTER would not know whether both boards had acknowledged the access.

RECOMMENDATION 2.5:
Since many systems will include MASTERS with various data transfer capabilities" include D08(EO) and D16 capability on 32-bit SLAVE (memory) boards in addition to the D32 capability" and include D08(EO) capability on 16-bit SLAVE boards in addition to the D16 capability.

OBSERVATION 2.17:
Since D08(0) SLAVES respond only to odd byte addresses, they cannot provide contiguous memory. D08(0) SLAVES are useful only for I/O, status, or control registers, while D08(EO), D16 and D32 SLAVES are also useful for memory.

Table 2-10 shows the various mnemonics used to describe the basic data transfer capabilities, and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-10. Mnemonics That Specify Basic Data Transfer Capabilities

The Following Mnemonic	When Applied to a	Means that it
D08(EO)	MASTER	can generate the following cycles:
	SLAVE	can accept the following cycles:
	LOCATION MONITOR	can monitor the following cycles:
		Single byte read cycles:
		BYTE(0) READ
		BYTE(1) READ
		BYTE(2) READ
		BYTE(3) READ
		Single byte write cycles:
		BYTE(0) WRITE
		BYTE(1) WRITE
		BYTE(2) WRITE
		BYTE(3) WRITE
D08(0)	SLAVE	can accept the following cycles:
		Single byte read cycles:
		BYTE(1) READ
		BYTE(3) READ
		Single byte write cycles:
		BYTE(1) WRITE
		BYTE(3) WRITE
D16	MASTER	can generate the following cycles:
	SLAVE	can accept the following cycles:
	LOCATION MONITOR	can monitor the following cycles:
		Double byte read cycles:
		BYTE(0-1) READ
		BYTE(2-3) READ
		Double byte write cycles:
		BYTE(0-1) WRITE
		BYTE(2-3) WRITE
D32	MASTER	can generate the following cycles:
	SLAVE	can accept the following cycles:
	LOCATION MONITOR	can monitor the following cycles:
		Quad byte read cycle:

		BYTE(0-3) READ
		Quad byte write cycle:
		BYTE(0-3) WRITE

Note: (EO) is Even and Odd; (O) is Odd only.

2.3.7 Block Transfer Capabilities

MASTERS often access several memory Iocations in ascending order. When this is the case, block transfer cycles are very useful. They allow the MASTER to provide a single address, and then access data in that location and those at higher addresses, without providing additional addresses.

When a MASTER initiates a block transfer cycle, the responding SLAVE Iatches the address into an on-board address counter. The MASTER, upon completing the first data transfer, (i.e. driving data strobes high) does not allow the address strobe to go high. Instead, it repeatedly drives the data strobe(s) low in response to data transfer acknowledgments from the SLAVE, and transfers data to or from sequential memory locations in ascending order.

To access the next location(s), the SLAVE increments an on-board counter that generates the address for each transition of the data strobe(s).

OBSERVATION 2.18:
Block transfer cycles of indefinite Iength are not allowed, because this complicates the design of memory boards. Specifically, all block transfer SLAVES (the one that responds, as well as those that do not) would need to Iatch the initial address and then increment the address counter on each bus transfer. AlI SLAVES would then have to decode the incremented address to see if the block transfer has crossed a board boundary into their address range. While this is certainly possible, such address decoding typically limits access times of the SLAVE. To simplify the design of these SLAVES, and to permit faster access times, RULE 2.12 has been formulated.

RULE 2.12:
Block transfer cycles **MUST NOT** cross any 256 byte boundary.

OBSERVATION 2.1.9:
RULE 2.12 limits the maximum Iength of block transfers to 256 bytes. However, knowing that only A01 through A07 will change during the course of the block transfer simplifies the design of block transfer SLAVES. The upper address Iines only have to be decoded once, at the beginning of the block transfer cycle, allowing much faster access times on all subsequent data transfers.

OBSERVATION 2.20:
In some cases it might be necessary to transfer a large block of data which crosses one or more 256-byte boundaries. In such a case, if the hardware on the board which does the block transfer is designed to recognize the arrival at a 256-byte boundary, it can momentarily drive AS* high and then initiate another block transfer without the intervention of system software.

The block read cycle is very similar to a string of read cycles. Likewise, the block write cycle is very similar to a string of write cycles. The difference is that only the initial address is broadcast by the MASTER and the address strobe is held low during all of the data transfers.

OBSERVATION 2.21:
Control of the DTB cannot be transferred during block transfers because the address strobe is held low through all of the data transfers, and control of the DTB can only be transferred while the address strobe is high.

Table 2-11 lists the various mnemonics used to describe block transfer capabilities and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-11. Mnemonics That Specify Block Transfer Capabilities

The Following Mnemonic	When Applied to a	Means that it
BLT	D08(EO) MASTER	can generate the following cycles:
	D08(EO) SLAVE	can accept the following cycles:
	D08(EO) LOCATION MONITOR	can monitor the following cycles:
		Block read cycles:
		SINGLE BYTE BLOCK READ
		Block write cycles:
		SINGLE BYTE BLOCK WRITE
	D16 MASTER	can generate the following cycles:
	D16 SLAVE	can accept the following cycles:
	D16 LOCATION MONITOR	can monitor the following cycles:
		Block read cycles:
		DOUBLE BYTE BLOCK READ
		Block write cycles:
		DOUBLE BYTE BLOCK WRITE
	D32 MASTER	can generate the following cycles:
	D32 SLAVE	can accept the following cycles:
	D32 LOCATION MONITOR	can monitor the following cycles:
		Block read cycles:
		QUAD BYTE BLOCK READ
		Block write cycles:
		QUAD BYTE BLOCK WRITE

2.3.8 Read-Modify-Write Capability

In multiprocessor systems which share resources such as memory and I/O, a method is needed to allocate these resources. One very important goal of this allocation algorithm is to ensure that a resource being used by one task cannot be used by another at the same time. The problem is best described by an example:

Two processors in a distributed processing system share a common resource (e.g., a printer). Only one processor can use the resource at a time. The resource is allocated by a bit in memory -i.e., if the bit is set, the resource is busy; if it is cleared, the resource is available. To gain use of the resource, processor A reads the bit and tests it to determine whether it is cleared. If the bit is cleared, processor A sets the bit to lock out processor B. This operation takes two data transfers: a read to test the bit, and a write to set the bit. However, a difficulty might arise if the bus is given to processor B between these two transfers. Processor B might then also find the bit cleared and assume the resource is available. Both processors will then set the bit in the next available cycle and attempt to use the resource.

This conflict is avoided by defining a read-modify-write cycle which prevents transferring control of the DTB between the read portion and the write portion of the cycle. This cycle is very similar to a read cycle immediately followed by a write cycle. The difference is that the address strobe is held low during both transfers. This ensures that, unlike a read cycle followed by a write cycle, control of the DTB cannot be transferred during a read-modify-write cycle, as this is only possible while the address strobe is high.

Table 2-12 lists the various mnemonics used to describe read-modify-write capabilities and how each is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-12. Mnemonics That Specify Read-Modify-Write Capabilities

The Following Mnemonic	When Applied to a	Means that it
RMW	D08(EO) MASTER	can generate the following cycles:
	D08(EO) SLAVE	can accept the following cycles:
	D08(EO) LOCATION MONITOR	can monitor the following cycles:
		Single byte read-modify-write cycles:
		BYTE(0) READ-MODIFY-WRITE
		BYTE(1) READ-MODIFY-WRITE
		BYTE(2) READ-MODIFY-WRITE
		BYTE(3) READ-MODIFY-WRITE
	D08(0) SLAVE	can accept the following cycles:
		Single byte read-modify-write cycles:
		BYTE(1) READ-MODIFY-WRITE
		BYTE(3) READ-MODIFY-WRITE

	D16 MASTER	can generate the following cycles:
	D16 SLAVE	can accept the following cycles:
	D16 LOCATION MONITOR	can monitor the following cycles:
		Double byte read-modify-write cycles:
		BYTE(0-1) READ-MODIFY-WRITE
		BYTE(2-3) READ-MODIFY-WRITE
	D32 MASTER	can generate the following cycles:
	D32 SLAVE	can accept the following cycles:
	D32 LOCATION MONITOR	can monitor the following cycles:
		Quad byte read-modify-write cycles:
		BYTE(0-3) READ-MODIFY-WRITE

2.3.9 Unaligned Transfer Capability

Some 32-bit microprocessors store and retrieve data in an unaligned fashion. For example, a 32-bit value might be stored in four different ways, as shown in Figure 2-6.

		Example A	Example B	Example C	Example D
	BYTE(3)				
4-byte group	BYTE(2)				XXXX
number 2	BYTE(1)			XXXX	XXXX
	BYTE(0)		XXXX	XXXX	XXXX
	BYTE(3)	XXXX	XXXX	XXXX	XXXX
4-byte group	BYTE(2)	XXXX	XXXX	XXXX	
number 1	BYTE(1)	XXXX	XXXX		
	BYTE(0)	XXXX			

Figure 2-6. Four Ways That 32 Bits Of Data Might Be Stored In Memory

The MASTER can transfer the 32 bits of data using several different sequences of DTB cycles. For example, it can transfer the data one byte at a time, using four single byte data transfers. However, a MASTER can accomplish the transfer much quicker by using one of the cycle sequences shown in Table 2-13.

OBSERVATION 2.22:
The sequences shown in Table 2-13 would be typical of a MASTER that accesses the byte locations in ascending order. The VMEbus protocol does not require this.

As shown in Table 2-13, each of these 32-bit transfers can be accomplished with a combination of singe byte and double byte transfers. However, examples B and D require three bus cycles when done this way. Because of this, the DTB protocol also includes two

triple byte transfer cycles. When used in combination with a single byte cycle, these triple byte cycles allow data to be stored as shown in examples B and D using only two bus cycles.

Some 32-bit microprocessors also store and retrieve data 16 bits at a time, in an unaligned fashion, as shown in Figure 2-7.

Table 2-13. Transferring 32 Bits Of Data Using Multiple Byte Transfer Cycles

Example	Cycle sequences used to accomplish the transfer	Data bus lines used (See Figure 2-6)	Byte locations accessed
A	Quad byte transfer	D00-D31	Grp 1, BYTE(0-3)
B	Single byte transfer	D00-D07	Grp 1, BYTE(1)
	Double byte transfer	D00-D15	Grp 1 , BYTE(2-3)
	Single byte transfer	D08-D15	Grp 2, BYTE(0)
	Triple byte transfer	D00-D23	Grp 1 , BYTE(1-3)
	Single byte transfer	D08-D15	Grp 2, BYTE(0)
C	Double byte transfer	D00-D15	Grp 1, BYTE(2-3)
	Double byte transfer	D00-D15	Grp 2, BYTE(0-1)
D	Single byte transfer	D00-D07	Grp 1, BYTE(3)
	Double byte transfer	D00-D15	Grp 2, BYTE(0-1)
	Single byte transfer	D08-D15	Grp 2, BYTE(2)
	or		
	Single byte transfer	D00-D07	Grp 1, BYTE(3)
	Triple byte transfer	D08-D31	Grp 2, BYTE(0-2)

		Example E	Example F	Example G	Example H
	BYTE(3)				
4-byte group	BYTE(2)				
number 2	BYTE(1)				
	BYTE(0)				XXXX
	BYTE(3)			XXXX	XXXX
4-byte group	BYTE(2)		XXXX	XXXX	
number 1	BYTE(1)	XXXX	XXXX		
	BYTE(0)	XXXX			

Figure 2-7. Four Ways That 16 Bits Of Data Might Be Stored In Memory

The MASTER can transfer the 16 bits of data using several different sequences of DTB cycles as listed in Table 2-14.

OBSERVATION 2.23:
The sequences listed in Table 2-14 would be typical of a MASTER that accesses the byte locations in ascending order. The VMEbus protocol does not require this.

As shown in Figure 2-13, the 16-bit transfer in example F can be accomplished with two single byte transfers. However, this requires two bus cycles. Because of this, the DTB protocol also includes a double byte transfer cycle which allows data to be stored as shown in example F using only one bus cycle.

OBSERVATION 2.24:
Since unaligned transfers make use of all 32 data lines, only D32 MASTERS and SLAVES can do unaligned transfers.

Table 2-15 lists how the unaligned transfer (UAT) mnemonic is used to describe MASTERS, SLAVES, and LOCATION MONITORS.

Table 2-14. Transferring 16 Bits Of Data Using Multiple Byte Transfer Cycles

Example	Cycle sequences used to accomplish the transfer	Data bus lines used	Byte locations accessed (See Figure 2-7)
E	Double byte transfer	D00-D15	Grp 1 , BYTE(0-1)
F	Single byte transfer	D00-D07	Grp 1, BYTE(1)
	Single byte transfer	D08-D15	Grp 1, BYTE(2)
	or		
	Double byte transfer	D08-D23	Grp 1, BYTE(1-2)
G	Double byte transfer	D00-D15	Grp 1, BYTE(2-3)
H	Single byte transfer	D00-D07	Grp 1, BYTE(3)
	Single byte transfer	D08-D15	Grp 2, BYTE(0)

Table 2-15. Mnemonic That Specifies Unaligned Transfer Capability

The Following Mnemonic	When Applied to a	Means that it
UAT	D32 MASTER	can generate the following cycles:
	D32 SLAVE	can accept the following cycles:
	D32 LOCATION MONITOR	can monitor the following cycles:
		Triple byte read cycles:
		BYTE(0-2) READ
		BYTE(1-3) READ

		Triple byte write cycles:
		BYTE(0-2) WRITE
		BYTE(1-3) WRITE
		Double byte read cycle:
		BYTE(1-2) READ
		Double byte write cycle:
		BYTE(1-2) WRITE

2.3.10 ADDRESS-ONLY Capability

The ADDRESS-ONLY cycle is the only cycle on the DTB that is not used to transfer data. It begins as a typical DTB cycle, with the address, address modifier code, IACK* and LWORD* lines becoming valid and the address strobe falling after a set-up time. However, the data strobes are never driven low. After holding the various lines strobed by the address strobe stable for a prescribed minimum period, the MASTER finishes the cycle without waiting for DTACK* or BERR* to go low. (The ADDRESS ONLY cycle is also the only type of DTB cycle that does not require a response in order to complete.)

Table 2-16 lists how the ADDRESS-ONLY mnemonic (ADO) is used to describe MASTERS and SLAVES.

OBSERVATION 2.25:
ADDRESS-ONLY cycles can be used to enhance board performance by allowing a CPU board to broadcast an address before it has determined whether or not that address selects a SLAVE on the VMEbus. Broadcasting the address in this fashion allow VMEbus SLAVES to decode the address concurrently with the CPU board.

RECOMMENDATION 2.6:
Since MASTERS might generate ADDRESS-ONLY cycles, design SLAVES to include ADO capability.

Table 2-16. Mnemonic That Specifies ADDRESS-ONLY Capability

The Following Mnemonic	When Applied to a	Means that it
ADO	MASTER	can generate ADDRESS-ONLY cycles
	SLAVE	can tolerate ADDRESS-ONLY cycles without loss or alteration of stored data.

2.3.11 Interaction Between DTB Functional Modules

Data transfers take place between MASTERS and SLAVES. The MASTER is the module controlling the transfer. The SLAVE which recognizes the address as its own is the responding SLAVE, and all other SLAVES are non-responding SLAVES.

After initiating a data transfer cycle, the MASTER waits for a response from the responding SLAVE. When the MASTER detects the response from the SLAVE it drives the data strobes and address strobe high, terminating the cycle. The SLAVE responds by releasing its response line.

OBSERVATION 2.26:
Although the address and data timing are Iargely independent, there are two exceptions. First, the MASTER waits until it has driven AS* low before driving either of the data strobes low. Second, the SLAVE acknowledges both the address strobe and the data strobes with either DTACK* or BERR*.

RULE 2.13:
IF a SLAVE responds to a data transfer cycle, THEN it **MUST** either drive DTACK* low or it **MUST** drive BERR* low, but not both.

OBSERVATION 2.27:
Because of possible bus skew due to different loading of the address strobe and the data strobes, the falling edge of the data strobes might be detected by the SLAVE slightly before the address strobe falling edge.

OBSERVATION 2.28:
The WRITE* line is high/low to identify a read/write cycle before the first data strobe is driven low, and remains stable until both data strobes are high.

RULE 2.14:
Before driving the data bus, the MASTER **MUST** ensure that the previous responding SLAVE has stopped driving the data bus. It does so by verifying that DTACK* and BERR* are both high before it drives the data strobe(s) to low on any cycle, and before it drives any of the data lines during a write cycle.

RULE 2.15:
At the end of a read cycle the responding SLAVE **MUST** release the data bus before allowing DTACK* to go high.

RULE 2.16:
When the MASTER reads data from the SLAVE, the SLAVE **MUST** maintain valid data on the data bus until the MASTER returns the first data strobe to high.

SUGGESTION 2.3:
For optimum performance, design MASTERS so that they drive the data strobes high as soon as possible after DTACK* or BERR^ goes low. Also, design SLAVES so that they release the data bus and DTACK* as soon as possible after detecting the data strobes high. This allows the maximum data transfer rate on the bus.

OBSERVATION 2.29:
Addressing information on the bus might change soon after a module drives DTACK or BERR* low, and before the MASTER drives the data strobes high.

A third type of module, called the LOCATION MONITOR, monitors the data transfer and generates either or both of two on-board signals whenever an access is done to a byte location that it monitors. If the access is a write cycle, then the on-board WRITE signal is generated. If the access is a read cycle, then the on-board READ signal is generated. If a read-modify-write cycle is performed, then both on-board signals are generated.

If the cycle takes too long, a fourth module, called a BUS TIMER intervenes by driving BERR* low, this completes the data transfer handshake, and allows the bus to resume operation.

RULE 2.17:
There is a strict interlock between the rising and falling edges of the data strobes and DTACK*/BERR*. Once a MASTER has driven its data strobe(s) low it **MUST NOT** drive its data strobe(s) high and finish a transfer without first receiving a data transfer acknowledge or a bus error response.

OBSERVATION 2.30:
A board containing a processor, which needs to direct data transfers between itself and other VMEbus boards, would contain a MASTER module. If the same board also contained memory accessible from the VMEbus, it would also contain a SLAVE module. A floating point processor or intelligent peripheral controller might receive commands through a SLAVE interface from a general purpose processor board. It then might act as a MASTER to access global VMEbus memory to execute the command it has been given.

2.4 TYPICAL OPERATION

MASTERS initiate data transfers over the DTB. The addressed SLAVE then acknowledges the transfer. After receiving the data transfer acknowledge, the MASTER terminates the data transfer cycle. The asynchronous nature of the DTB allows the SLAVE to control the time taken for the transfer.

Before doing any data transfers, a MASTER has to be granted exclusive control of the DTB. This ensures that multiple MASTERS will not try to use the DTB at the same time. The MASTER gains control of the DTB using the modules and signal lines of the Arbitration Bus. (Section 2.5 explains this further) The following discussion presumes that the MASTER has already been granted and has assumed control of the DTB.

2.4.1 Typical Data Transfer Cycles

Figure 2-8 shows a typical single byte read cycle. To start the transfer, the MASTER drives the addressing lines with the desired address and address modifier code. Since this example is a BYTE(1) READ CYCLE, the MASTER drives LWORD* high and A01 low. Since it is not doing an interrupt acknowledge cycle, it does not drive IACK* low. The MASTER then waits for a specified set-up time before driving AS* low, to allow the address lines and the address modifier lines to stabilize before the SLAVES sample them.

Each SLAVE determines whether it should respond by examining the levels on the address lines, the address modifier lines, and IACK*. While this is happening, the MASTER drives WRITE* high to indicate a read operation. The MASTER then verifies that DTACK* and BERR* are high to ensure that the SLAVE from the previous cycle is no longer driving the data bus. If this is the case, the MASTER then drives DS0* low, while keeping DS1* high.

The responding SLAVE then determines which 4-byte group and which byte of that group is to be accessed, and starts the transfer. After it has retrieved the data from its internal storage and placed it on data bus lines D00-D07, the SLAVE signals the MASTER by driving DTACK* low. The SLAVE then holds DTACK* low and maintains the data valid for as long as the MASTER holds DS0* low.

When the MASTER receives DTACK* driven to low, it captures the data on D00-D07, releases the address lines and drives DS0* and AS* to high. The SLAVE responds by releasing D00-D07 and releasing DTACK* to high.

OBSERVATION 2.31:
The MASTER in Figure 2-8 releases all of the DTB lines at the end of the data transfer. This is not required unless the MASTER'S REQUESTER released BBSY* during the data transfer as described in Section 2.5.

The cycle flow for double byte and quad byte data transfer cycles are very similar to the single byte cycle. Flow diagrams for these cycles are shown in Figures 2-9 and 2-10.

MASTER	**SLAVE**
ADDRESS THE SLAVE Present address Present address modifier Drive LWORD* high Drive IACK* high Drive AS* to low	
SPECIFY DATA DIRECTION Drive WRITE* high **SPECIFY DATA WIDTH** Wait until DTACK* high and BERR* high (indicates that previous SLAVE is no longer driving data bus) Drive DS0* to low and DS1* to high	**PROCESS ADDRESS** Receive address Receive address modifier LWORD* high Receive IACK* high Receive AS* low If address is valid for this SLAVE then select on-board device
	FETCH DATA Receive WRITE* high

	Read data from selected device Receive DS1* high Receive DS0* low Present data on lines D00-D07 **RESPOND TO MASTER** Drive DTACK* to low

ACQUIRE DATA
 	
Receive data on lines D00-D07
Receive DTACK* low
 	
TERMINATE CYCLE
 	
If last cycle then
Release address lines
Release address modifier lines
Release LWORD*
Release IACK*
Endif
Drive DS0* to high
Drive AS. to high	

END TERMINATION
 	
If last cycle then
Release DS0* and DS1*
Release AS*
Else go to ADDRESS THE SLAVE
Endif
	END RESPONSE TO MASTER
	
	Receive AS* and DS0* high
	
	Release D00-D07

	ACKNOWLEDGE TERMINATION Release DTACK*

Figure 2-8. An Example Of A Single Byte Read Cycle

MASTER	**SLAVE**
ADDRESS THE SLAVE Present address Present address modifier Drive LWORD* high Drive IACK* high Drive AS* to low	
SPECIFY DATA DIRECTION	**PROCESS ADDRESS**

Drive WRITE* low **SPECIFY DATA WIDTH** Wait until DTACK* high and BERR* high (indicates that previous SLAVE is no longer driving data bus) Place data on D00-D15 Drive DS0* and DS1* to low	Receive address Receive address modifier LWORD* high Receive IACK* high Receive AS* low If address is valid for this SLAVE then select on-board device
	STORE DATA Receive WRITE* low Receive DS1* low Receive DS0* low Capture data from lines D00-D15 Write data into selected device **RESPOND TO MASTER** Drive DTACK* to low

TERMINATE CYCLE
 	
Receive DTACK* low
If last cycle then
Release address lines
Release address modifier lines
Release LWORD*
Release IACK*
Endif
Drive DS0* and DS1* to high
Drive AS* to high	

END TERMINATION
 	
If last cycle then
Release DS0* and DS1*
Release AS*
Else go to ADDRESS THE SLAVE
Endif
	ACKNOWLEDGE TERMINATION
	
	Receive AS*, DS0*, and DS1* high
	
	Release DTACK*

Figure 2-9. An Example Of A Double Byte Write Cycle

MASTER	SLAVE
ADDRESS THE SLAVE Present address	

Present address modifier Drive LWORD* low Drive IACK* high Drive AS* to low	
SPECIFY DATA DIRECTION Drive WRITE* low **SPECIFY DATA WIDTH** Wait until DTACK* high and BERR* high (indicates that previous SLAVE is no longer driving data bus) Place data on D00-D31 Drive DS0* and DS1* to low	**PROCESS ADDRESS** Receive address Receive address modifier LWORD* low Receive IACK* high Receive AS* low If address is valid for this SLAVE then select on-board device
	STORE DATA Receive WRITE* low Receive DS1* low Receive DS0* low Capture data from lines D00-D31 Write data into selected device **RESPOND TO MASTER** Drive DTACK* to low

TERMINATE CYCLE
 	
Receive DTACK* low
If last cycle then
Release address lines
Release address modifier lines
Release LWORD*
Release IACK*
Endif
Drive DS0* and DS1* to high
Drive AS* to high	

END TERMINATION
 	
If last cycle then
Release DS0* and DS1*
Release AS*
Else go to ADDRESS THE SLAVE
Endif
	ACKNOWLEDGE TERMINATION
	
	Receive AS*, DS0*, and DS1* high
	
	Release DTACK*

Figure 2-10. An Example Of A Quad Byte Write Cycle

2.4.2 Address Pipelining

The VMEbus strobes the address and data with separate strobe signals. This allows a MASTER to broadcast the address for the next cycle while the data transfer for the previous cycle is still in progress. This is called “address pipelining”.

PERMISSION 2.8:
As soon as a SLAVE drives DTACK* or BERR* low, the MASTER **MAY** change the address and, after driving AS* high for a minimum time, drive AS* low again.

For example, when a SLAVE drives DTACK* low on a read cycle, the MASTER can place a new address on the address bus while it is reading the data from the bus. This amounts to an overlapping of one cycle with the next one, and permits greater speeds on the VMEbus.

RULE 2.18:
Since address pipelining can occur, SLAVES **MUST NOT** be designed on the assumption that they will never encounter pipelined cycles.

OBSERVATION 2.32:
The responding SLAVE might recognize its address and respond very quickly on the DTACK* or BERR* lines. Since the MASTER is permitted to remove the address after the responding SLAVE drives DTACK* or BERR* low, non-responding SLAVES might not be able to decode the addressing information before the MASTER removes it from the bus.

SUGGESTION 2.4:
Design SLAVES to capture the addressing information on the falling edge of AS*.

OBSERVATION 2.33:
Because a MASTER might broadcast a new address while a previous cycle is finishing, the designer needs to ensure that the assertion of the second address strobe does not invalidate the first address if it is still needed by on-board logic to maintain the data on the bus.

PERMISSION 2.9:
MASTERS **MAY** be designed without the ability for address pipelining. (e.g. They **MAY** wait until the responding SLAVE releases DTACK* or BERR* before driving AS* low for the next cycle.)

OBSERVATION 2.34:
A MASTER might drive AS* low for a new cycle before it drives data strobes high from the previous cycle. Because of this, there might be a period when the address strobe for the new cycle, as well as at least one data strobe from the previous cycle coincide during the cycle overlap.

SUGGESTION 2.5:

To assure reliable operation, design SLAVES to initiate data transfers to and from the bus on the falling edge of the data strobes, instead of a simultaneous low level on the address strobe and data strobes.

2.5 DATA TRANSFER BUS ACQUISITION

RULE 2.19:
Before transferring any data on the DTB, a MASTER **MUST** get permission to use it.

Several MASTERS might want to use the DTB at the same time. The process which determines which MASTER can use the DTB is called arbitration and is discussed in Chapter 3. Because arbitration is closely tied to the operation of the DTB, it is briefly described here.

Figure 2-11 provides two examples that show possible sequences when a MASTER (called MASTER A) finishes using the DTB and allows arbitration to take place.

In Example 1, MASTER A, partway into its Iast transfer, indicates that it no Ionger needs the DTB. It does this by having its REQUESTER release the bus busy (BBSY*) signal line. Since MASTER A gives this early notice that the DTB will soon be available, the arbitration is done during its Iast data transfer. The arbitration is completed and MASTER B is granted permission to use the DTB before MASTER A has finished its cycle, but it waits until MASTER A releases AS*. (This assures that MASTER B will not start driving the DTB before MASTER A has finished with its Iast data transfer.)

In Example 2, MASTER A waits until after its last transfer (i.e., after AS* is released) before releasing BBSY*. In this case the DTB is idle while the arbitration is done. MASTER B is then granted the bus and, since AS* is already high, it begins using the DTB immediately.

RULE 2.20:
Once a MASTER'S REQUESTER releases BBSY. to high, the MASTER **MUST NOT** drive AS* from high to low (i.e., it **MUST NOT** begin a new cycle) until its REQUESTER receives a new bus grant.

2.6 DTB TIMING RULES AND OBSERVATIONS

This section describes the timing RULES and OBSERVATIONS that govern the behavior of MASTERS and SLAVES. This timing information is in the form of Figures and Tables:

Table 2-17 lists a timing Table and timing diagrams that specify MASTER, SLAVE, and LOCATION MONITOR operation.

Table 2-18 defines the various mnemonics that are used in this section.

Tables 2-19 through 2-21 specify the use of the DTB signals.

Tables 2-22 through 2-27 specify the timing parameters of the DTB. (The reference numbers used in Tables 2-24 through 2-27 correspond to the timing parameter numbers in Tables 2-22 and 2-23.)

Figures 2-12 through 2-15 specify the timing RULES and OBSERVATIONS during address broadcasting time.

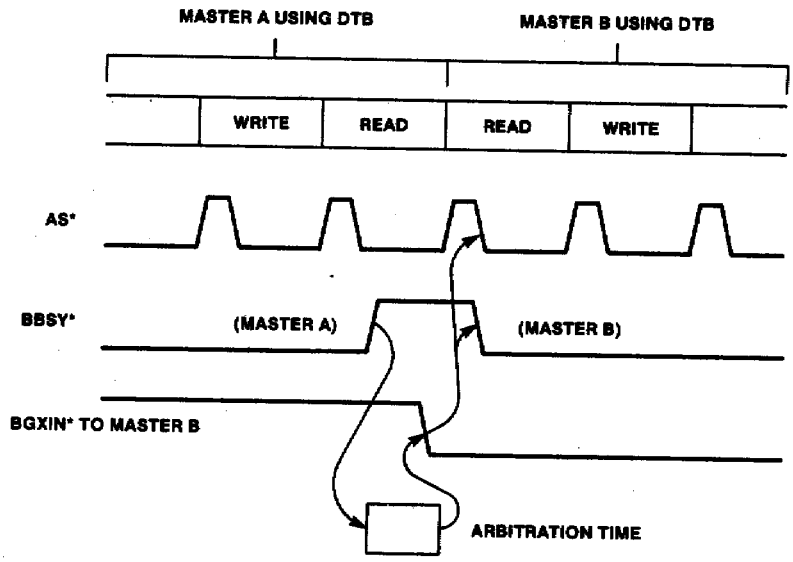
Figures 2-16 through 2-21 specify the timing RULES and OBSERVATIONS for MASTERS, SLAVES, and LOCATION MONITORS during data transfer time.

Figures 2-22 through 2.24 specify the timing RULES and OBSERVATIONS for MASTERS and SLAVES between DTB cycles.

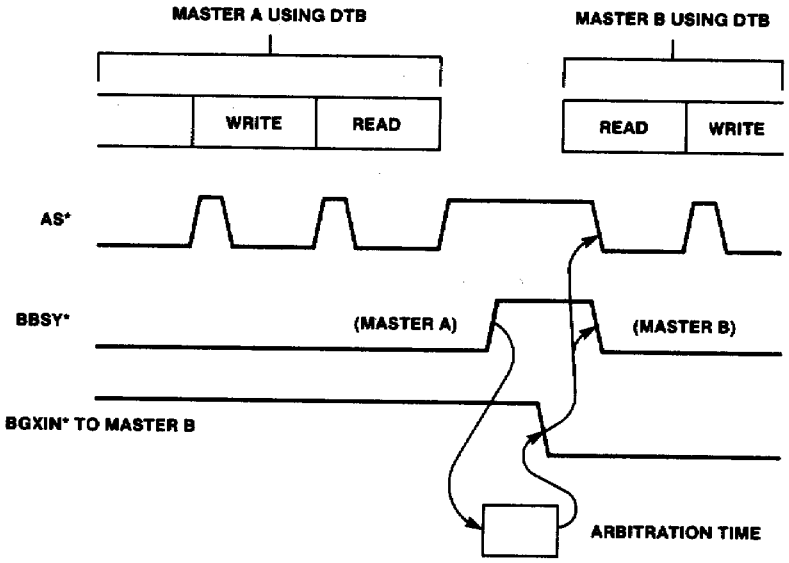
Figure 2-25 is the timing diagram for MASTER, SLAVE, and BUS TIMER during a timed-out cycle.

Figure 2-26 shows the timing during the mastership transfer of the DTB.

Example 2 -- Arbitration AFTER The Last Data Transfer



Example 1 -- Arbitration DURING The Last Data Transfer



Example 2 -- Arbitration AFTER The Last Data Transfer

Figure 2-11. Data Transfer Bus MASTER Exchange Sequence

In order to meet the specified timing RULES, board designers need to take into account the worst case propagation delays of the bus drivers and receivers used on their VMEbus boards. The propagation delay of the drivers depends on their output loads and manufacturers specifications do not always give enough information to calculate the propagation delays under various loads. To help the VMEbus board designer, some suggestions are offered in Chapter 6.

The OBSERVATIONS specify the timing of incoming lines signal transitions. These times can be relied upon as long as the backplane loading RULES in Chapter 6 are not violated. The

RULES for the bus terminators in Chapter 6 guarantee that the timing parameters for signal lines that are released after they have been driven, are met.

Typically, for each timing RULE there is a corresponding OBSERVATION. However, the time that is guaranteed in the OBSERVATION might differ from the time specified by the RULE. For example, a careful inspection of the timing diagrams shows that the MASTER is required to provide 35 nanoseconds of address and data set-up time, but the SLAVE is only guaranteed 10 nanoseconds. This is because the address and data bus drivers are not always able to drive the backplane's signal lines completely through the threshold region from low to high until the transition propagates to the end of the backplane and is reflected back. The falling edge of the address and data strobes, however, typically cross the 0.8-volt threshold without waiting for a reflection. The resulting set-up time at the SLAVE is the MASTER'S set-up time less two bus propagation times.

A special notation has been used to describe the data strobe timing. The two data strobes (DS0* and DS1*) will not always make their transitions simultaneously. For purposes of these timing diagrams, DSA* represents the first data strobe to make its transition (whether that is DS0* or DS1*). The broken line shown while the data strobes are stable is to indicate that the first data strobe to make a falling transition might not be the first to make its rising transition -- i.e., DSA* can represent DS0* on its falling edge and DS1* on its rising edge.

Table 2-17. Timing Diagrams That Define MASTER, SLAVE, And LOCATION MONITOR Operation
(See Table 2-22 for timing values)

Mnemonic	Type of cycles	Address Broadcast Timing Diag Figure(s)	Data Transfer Timing Diag Figure
ADO	ADDRESS-ONLY	2-12	N/A
D08(EO)	Single even byte transfers		
	BYTE(0) READ	2-12 & 2-13	2-16
	BYTE(2) READ	2-12 & 2-13	2-16
	BYTE(0) WRITE	2-12 & 2-13	2-18
	BYTE(2) WRITE	2-12 & 2-13	2-18
D08(EO)	Single odd byte transfers		
	BYTE(1) READ	2-12 & 2-13	2-16
or	BYTE(3) READ	2-12 & 2-13	2-16
D08(0)	BYTE(1) WRITE	2- 12 & 2- 13	2- 18
	BYTE(3) WRITE	2-12 & 2-13	2-18
D16	Double byte transfers		

	BYTE(0-1) READ	2-12 & 2-13	2-17
	BYTE(2-3) READ	2-12 & 2-13	2-17
	BYTE(0-1) WRITE	2-12 &-2-13	2-19
	BYTE(2-3) WRITE	2-12 & 2-13	2-19
D32	Quad byte transfers		
	BYTE(0-3) READ	2-12 & 2-13	2-17
	BYTE(0-3) WRITE	2-12 & 2-13	2-19
D08(E0):BLT	Single byte block transfers		
	SINGLE BYTE BLOCK READ	2-12 & 214	2-16
	SINGLE BYTE BLOCK WRITE	2-12 & 214	2-18
D16:BLT	Double byte block transfers		
	DOUBLE BYTE BLOCK READ	2-12 & 2-14	2-17
	DOUBLE BYTE BLOCK WRITE	2-12 & 2-14	2-19
D32:BLT	Quad byte block transfers		
	QUAD BYTE BLOCK READ	2-12 & 2-14	2-17
	QUAD BYTE BLOCK WRITE	2-12 & 2-14	2-19
D08(E0):RMW	Single byte RMW transfers		
	BYTE(0) READ-MODIFY-WRITE	2-12 & 2-15	2-20
	BYTE(1) READ-MODIFY-WRITE	2-12 & 2-15	2-20
	BYTE(2) READ-MODIFY-WRITE	2-12 & 2-15	2-20
	BYTE(3) READ-MODIFY-WRITE	2-12 & 2-15	2-20
D16:RMW	Double byte RMW transfers		
	BYTE(0-1) READ-MODIFY-WRITE	2-12 & 2-15	2-21
	BYTE(2-3) READ-MODIFY-WRITE	2-12 & 2-15	2-21
D32:RMW	Quad byte RMW transfers		
	BYTE(0-3) READ-MODIFY-WRITE	2- 12 & 2- 15	2-21
D32:UAT	Unaligned transfers		
	BYTE(0-2) READ	2-12 & 2-13	2-16
	BYTE(1-3) READ	2-12 & 2-13	2-16
	BYTE(1-2) READ	2-12 & 2-13	2-17
	BYTE(0-2) WRITE	2-1 2 & 2-13	2-18
	BYTE(1-3) WRITE	2-12 & 2-13	2-18
	BYTE(1-2) WRITE	2-12 & 2-13	2-19

Tables 2-19, 2-20, and 2-21 show how the various signal lines of the DTB are used to broadcast addresses and to transfer data. These Tables are referenced by the various timing

diagrams that follow. In order to keep these Tables compact, mnemonics are used to describe when and how the various lines are driven. These mnemonics are defined in Table 2-18.

Table 2-18. Definitions Of Mnemonics Used In Tables 2-19, 2-20, And 2-21

Mnemonic Description	Comments	
DVBM	DRIVEN VALID BY MASTER	RULE 2.21: The MASTER **MUST** drive DVBM lines to a valid level
DLBM	DRIVEN LOW BY MASTER	RULE 2.22: The MASTER **MUST** drive DLBM lines to a low level.
DHBM	DRIVEN HIGH BY MASTER	RULE 2.23: The MASTER **MUST** drive DHBM lines to a high level
dhbm?	DRIVEN HIGH BY MASTER?	PERMISSION 2.10: The MASTER **MAY** drive dhbm? lines high RULE 2.24: The MASTER **MUST NOT** drive dhbm? lines low..
dxbm?	DRIVEN BY MASTER?	PERMISSION 2.11: The MASTER **MAY** drive dxbm? lines, or it **MAY** leave these lines undriven. (When dxbm? lines are driven, they carry no valid information.)
DVBS	DRIVEN VALID BY SLAVE	RULE 2.25: The SLAVE **MUST** drive DVBS lines to a valid level.
dxbs?	DRIVEN BY SLAVE?	PERMISSION 2.12. The SLAVE **MAY** drive dxbs? lines leave these lines undriven., or it **MAY** (When dxbs? lines are driven, they carry no valid information.)
DVBB	DRIVEN VALID BY BOTH SLAVE AND MASTER	RULE 2.26: During the “read” portion of a read modify write cycle, the SLAVE **MUST** drive DVBB lines with valid data. During the "write" portion of a read-modify-write cycle, the MASTER **MUST** drive DVBB lines with valid data.
dxbb?	DRIVEN BY BOTH SLAVE AND MASTER?	PERMISSION 2.13: During the “read” portion of a read modify write cycle, the SLAVE **MAY** drive dxbb? lines, or it **MAY** leave them undriven. During the “write” portion of a read-modify-write cycle, the MASTER **MAY** drive dxbb? lines, or it **MAY** leave them undriven. (When dxbb? lines are driven, they carry no valid information.)

Table 2-19. Use Of Addressing Lines To Select A 4-Byte Group

Mnemonic	Addressing mode	A02-A15 See Note	A16-A23	A24-A31	IACK*
A16	SHORT	DVBM	dxbm?	dxbm?	dhbm?
A24	STANDARD	DVBM	DVBM	dxbm?	dhbm?
A32	EXTENDED	DVBM	DVBM	DVBM	dhbm?

Note: A01 is used in conjunction with LWORD. and the two data strobes to select which of the four bytes within the 4-byte group is accessed. (See Table 2-20).

Table 2-20. Use Of The DS1*, DS0*, A01, And LWORD* Lines To Select The Byte(s) Within A 4-Byte Group

Mnemonic	Type of cycles	DS1*	DS0*	A01	LWORD*
ADO	ADDRESS-ONLY	dhbm?	dhbm?	dxbm?	dxbm?
D08(EO)	Single even byte transfers				
	BYTE(O) READ or WRITE	DLBM	dhbm?	DLBM	dhbm?
	BYTE(2) READ or WRITE	DLBM	dhbm?	DHBM	dhbm?
D08(EO)	Single odd byte transfers				
or	BYTE(1) READ or WRITE	dhbm?	DLBM	DLBM	dhbm?
D08(0)	BYTE(3) READ or WRITE	dhbm?	DHBM	DLBM	dhbm?
D16	Double byte transfers				
	BYTE(O-1) READ or WRITE	DLBM	DLBM	DLBM	dhbm?
	BYTE(2-3) READ or WRITE	DLBM	DLBM	DHBM	dhbm?
D32	Quad byte transfers				
	BYTE(0-3) READ or WRITE	DLBM	DLBM	DLBM	DLBM
D08(EO) :BLT	Single byte block transfers				
	SINGLE BYTE BLOCK READ or WRITE	<----	NOTE1	---->	dhbm?
D16 :BLT	Double byte block transfers				
	DOUBLE BYTE BLOCK READ or WRITE	DLBM	DLBM	NOTE 2	dhbm?
D32:BLT	Quad byte block transfers				
	QUAD BYTE BLOCK READ or	DLBM	DLBM	DLBM	DLBM

	WRITE				
D08(EO) :RMW	Single byte RMW transfers				
	BYTE(0) READ-MODIFY-WRITE	DLBM	dhbm?	DLBM	dhbm?
	BYTE(1) READ-MODIFY-WRITE	dhbm?	DLBM	DLBM	dhbm?
	BYTE(2) READ-MODIFY-WRITE	DLBM	dhbm?	DHBM	dhbm?
	BYTE(3) READ-MODIFY-WRITE	dhbm?	DLBM	DHBM	dhbm?
D16:RMW	Double byte RMW transfers				
	BYTE(0-1) READ-MODIFY-WRITE	DLBM	DLBM	DLBM	dhbm?
	BYTE(2-3) READ-MODIFY-WRITE	DLBM	DLBM	DHBM	dhbm?
D32 :RMW	Quad byte RMW transfers				
	BYTE(0-3) READ-MODIFY-WRITE	DLBM	DLBM	DLBM	DLBM
D32:UAT	Unaligned transfers				
	BYTE(0-2) READ or WRITE	DLBM	dhbm?	DLBM	DLBM
	BYTE(1-3) READ or WRITE	dhbm?	DLBM	DLBM	DLBM
	BYTE(1 -2) READ or WRITE	DLBM	DLBM	DHBM	DLBM

Notes:

1. During Single byte block transfers, the two data strobes are alternately driven low . Either data strobe might be driven low on the first transfer. If the first accessed byte location is BYTE(0) or BYTE(2), then the MASTER drives DS1* to low first. If the first accessed byte location is BYTE(1) or BYTE(3), then it drives DS0* to low first. A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and might be either high or low depending upon which byte the single byte block transfer begins with. If the first byte location is BYTE(0) or BYTE(1), then the MASTER drives A01 to low. If the first byte location is BYTE(2) or BYTE(3), then the MASTER drives A01 to high.

An example of the use of DS0*, DS1*, A01, and LWORD* during a Single byte block transfer cycle which starts with BYTE(2) is given below:

		DS1*	**DS0***	**A01**	**LWORD***
First data transfer	BYTE(2)	DLBM	DHBM	DHBM	dhbm?
	BYTE(3)	DHBM	DLBM	dxbm?	dxbm?
	BYTE(0)	DLBM	DHBM	dxbm?	dxbm?
	BYTE(1)	DHBM	DLBM	dxbm?	dxbm?

Last data transfer	BYTE(2)	DLBM	DHBM	dxbm?	dxbm?

2. During a Double byte block transfer, A01 is valid only on the first data transfer (i.e. until the SLAVE drives DTACK* or BERR* low the first time) and is driven either high or low depending upon what double byte group the double byte block transfer begins with. If the first double byte group is BYTE(0-1), then the MASTER drives A01 to Iow. If the first double byte group is BYTE(2-3), then the MASTER drives A01 to high.

Table 2-21. Use Of The Data Bus Lines To Transfer Data

Mnemonic Type of cycles		**D24-D31**	**D16-D23**	**D08-D15**	**D00-D07**
ADO	ADDRESS-ONLY	dxbm?	dxbm?	dxbm?	dxbm?
D08(EO)	Single even byte transfers				
	BYTE(0) READ	dxbs?	dxbs?	DVBS	dxbs?
	BYTE(2) READ	dxbs?	dxbs?	DVBS	dxbs?
	BYTE(0) WRITE	dxbm?	dxbm?	DVBM	dxbm?
	BYTE(2) WRITE	dxbm?	dxbm?	DVBM	dxbm?
D08(EO)	Single odd byte transfers				
or	BYTE(1) READ	dxbs?	dxbs?	dxbs?	DVBS
	BYTE(3) READ	dxbs?	dxbs?	dxbs?	DVBS
D08(0)	BYTE(1) WRITE	dxbm?	dxbm?	dxbm?	DVBM
	BYTE(3) WRITE	dxbm?	dxbm?	dxbm?	DVBM
D16	Double byte transfers				
	BYTE(0-1) READ	dxbs?	dxbs?	DVBS	DVBS
	BYTE(2-3) READ	dxbs?	dxbs?	DVBS	DVBS
	BYTE(0-1) WRITE	dxbm?	dxbm?	DVBM	DVBM
	BYTE(2-3) WRITE	dxbm?	dxbm?	DVBM	DVBM
D32	Quad byte transfers				
	BYTE(0-3) READ	DVBS	DVBS	DVBS	DVBS
	BYTE(0-3) WRITE	DVBM	DVBM	DVBM	DVBM
D08(EO) :BLT	Single byte block transfers				
	SINGLE BYTE BLOCK READ	dxbs?	dxbs?	<----Note 1-- >	
	SINGLE BYTE BLOCK WRITE	dxbm?	dxbm?	<--- Note 1--->	
D16:BLT	Double byte block transfers				

	DOUBLE BYTE BLOCK READ	dxbs?	dxbs?	DVBS	DVBS
	DOUBLE BYTE BLOCK WRITE	dxbm?	dxbm?	DVBM	DVBM
D32:BLT	Quad byte block transfers				
	QUAD BYTE BLOCK READ	DVBS	DVBS	DVBS	DVBS
	QUAD BYTE BLOCK WRITE	DVBM	DVBM	DVBM	DVBM
D08(EO): RMW	Single byte RMW transfers				
	BYTE(0) READ-MODIFY-WRITE	dxbb?	dxbb?	DVBB	dxbb?
	BYTE(1) READ-MODIFY-WRITE	dxbb?	dxbb?	dxbb?	DVBB
	BYTE(2) READ-MODIFY-WRITE	dxbb?	dxbb?	DVBB	dxbb?
	BYTE(3) READ-MODIFY-WRITE	dxbb?	dxbb?	dxbb?	DVBB
D16:RMW	Double byte RMW transfers				
	BYTE(0-1) READ-MODIFY-WRITE	dxbb?	dxbb?	DVBB	DVBB
	BYTE(2-3) READ-MODIFY-WRITE	dxbb?	dxbb?	DVBB	DVBB
D32:RMW	Quad byte RMW transfers				
	BYTE(0-3) READ-MODIFY-WRITE	DVBB	DVBB	DVBB	DVBB
D32:UAT	Unaligned transfers				
	BYTE(0-2) READ	DVBS	DVBS	DVBS	dxbs?
	BYTE(1-3) READ	dxbs?	DVBS	DVBS	DVBS
	BYTE(1-2) READ	dxbs?	DVBS	DVBS	dxbs?
	BYTE(0-2) WRITE	DVBM	DVBM	DVBM	dxbm?
	BYTE(1-3) WRITE	dxbm?	DVBM	DVBM	DVBM
	BYTE(1-2) WRITE	dxbm?	DVBM	DVBM	dxbm?

Note:

1. During Single byte block transfers, data is transferred 8 bits at a time over D00-D07 or D08-D1 5. A single byte block read example is given below:

	D08-D15	**D00-D07**
First data transfer	DVBS	dxbs?
	dxbs?	DVBS

	DVBS	dxbs?
	dxbs?	DVBS
	DVBS	dxbs?
	dxbs?	DVBS
Last data transfer	DVBS	dxbs?

Table 2-22. MASTER, SLAVE, And LOCATION MONITOR Timing Parameters

PARAMETER NUMBER	MASTER See also Table 2-24		SLAVE See also Table 2-25		LOCATION MONITOR See also Table 2-26	
	MIN	MAX	MIN	MAX	MIN	MAX
1	O					
2	60					
4	35		10		10	
5	40		30		3	
6	0		0			
7	0		0			
8	35		10		10	
11	40		30		30	
12	35		10		10	
13		10		20		20
14	0		0			
15	0		0			
16	0		0			
17	40		0		30	
18	0		0		30	
19	40		0		30	
20	0		0			
21	0		0			
23	10		0		0	
24A	0					
24B	0					
25		25				
26	0		0			
27	-25		0			
28	30	2T	30			
29	0		0			
30	0		0			
31	0		0			
33			30		30	

NOTES:
1. AII times are in nanoseconds.
2. T = the timeout value in microseconds.

Table 2-23. BUS TIMER, Timing Parameters (see also Table 2-27)

PARAMETER NUMBER	MASTER	
	MIN	**MAX**
28	T	2T
30	0	

NOTE
T = the time-out value in microseconds.

Table 2-24. MASTER, Timing RULES And OBSERVATIONS

Note:
The numbers correspond to the timing parameters specified in Table 2-22.

1.	**RULE 2.27:** When taking control of the VMEbus, the MASTER **MUST NOT** drive any of IACK*, AM0-AM5, A01-A31, LWORD*, D00-D31, WRITE*, DS0*, DS1*, or AS* until after the previous MASTER allows AS* to rise above the low level. **OBSERVATION 2.35:** Chapter 3 describes how a MASTER'S REQUESTER is granted use of the VMEbus
2.	**RULE 2.28:** When taking control of the VMEbus, the MASTER **MUST NOT** drive any of IACK*, AM0-AM5, A01-A31 , LWORD*, D00-D31, WRITE*, DS0*, DS1*, or AS* until after its REQUESTER is granted the bus. **OBSERVATION 2.36:** Chapter 3 describes how a MASTER'S REQUESTER is granted use of the VMEbus
3.	**RULE 2.29:** When taking control of the VMEbus, the MASTER **MUST NOT** drive AS* low until this time after the previous MASTER allows AS* to rise above the low level. **OBSERVATION 2.37:** RULE 2.29 ensures that timing parameter 5 for SLAVES is guaranteed when there is an interchange of the DTB mastership.
4.	**RULE 2.30:** The MASTER **MUST NOT** drive AS* low until IACK* has been high, and the required lines among A01-A31, AM0-AM5, and LWORD* have been valid for this minimum time. **OBSERVATION 2.38:** Tables 2-19 and 2-20 specify the lines among A01-A31, AM0-AM5, and LWORD* that a MASTER is required to drive.

5.	**RULE 2.31:** When using the DTB for two consecutive cycles, the MASTER **MUST NOT** drive AS* low until it has been high for this minimum time.
6.	**RULE 2.32:** After a read cycle, the MASTER **MUST NOT** drive any of D00-D31 until both DTACK* and BERR* are high.
7.	**RULE 2.33:** During read cycles, the MASTER **MUST NOT** drive DSA* low until it has released all of D00-D31.
8.	RULE 2.34: During write cycles, the MASTER **MUST NOT** drive DSA* low until the required lines among D00-D31 have been valid for this minimum time. **OBSERVATION 2.39:** Table 2-21 specifies the lines among D00-D31 that a MASTER is required to drive.
9.	RULE 2.35: The MASTER **MUST NOT** drive DSA* low until both DTACK* and BERR* are high.
10.	**RULE 2.36:** The MASTER **MUST NOT** drive DSA* low until it has driven AS* low.
11 .	**RULE 2.37:** The MASTER **MUST NOT** drive DSA* low until DS0* and DS1* have both been simultaneously high for this minimum time.
12.	**RULE 2.38:** The MASTER **MUST NOT** drive DSA* low until WRITE* has been valid for this minimum time.
13.	**RULE 2.39:** During cycles where the MASTER drives both data strobes low, it **MUST** drive DSB* low within this maximum time after it drives DSA* low. **OBSERVATION 2.40:** Timing parameter 13 does not apply to transfers where only one data strobe is driven low.
14.	**RULE 2.40:** During all data transfer cycles, except read-modify-write cycles, the MASTER **MUST** hold the address valid and maintain the appropriate level on LWORD* until it detects the first falling edge on DTACK* or BERR*. **OBSERVATION 2.41**: During all data transfer cycles, except block transfer and read-modify-write cycles, there will be only one falling edge RULE 2.41:on DTACK* or BERR*.
15.	**RULE 2.41:** During read-modify-write cycles, the MASTER **MUST** hold the address valid and maintain the appropriate level on LWORD* until it detects the second falling edge on DTACK* or BERR*.
16.	**RULE 2.42:** During all data transfer cycles, the MASTER **MUST** maintain a valid address

	modifier code, and ensure that IACK* stays high until it detects the last falling edge on DTACK* or BERR*. **OBSERVATION 2.42:** During all data transfer cycles, except block transfer and read-modify-write cycles, there will be only one falling edge on DTACK* or BERR*.
17.	**RULE 2.43:** During ADDRESS-ONLY cycles, the MASTER **MUST NOT** change the levels on IACK*, A01-A31, AM0-AM5, and LWORD* for this minimum time after it drives AS* low.
18.	**RULE 2.44:** During all data transfer cycles, the MASTER **MUST** hold AS* low until it detects the last falling edge on DTACK* or BERR*
19.	**RULE 2.45:** The MASTER **MUST** hold AS* low for this minimum time.
20.	**RULE 2.46:** Once a MASTER has driven DSA* low, it **MUST** maintain that line low until it detects DTACK* or BERR* low.
21.	**RULE 2.47:** Once a MASTER has driven DSB* low, it **MUST** maintain that line low until it detects DTACK* or BERR* low.
22.	**RULE 2.48:** During write cycles, once the MASTER has driven DSA* low, it **MUST** not change any of D00-D31 until it detects DTACK* or BERR* low.
23.	**RULE 2.49:** Once a MASTER has driven DSA* low, it **MUST NOT** change the level on the WRITE* line, until this minimum time after both data strobes are high.
24A.	**RULE 2.50:** IF the MASTER drives or releases AS* to high after its REQUESTER releases BBSY*, THEN it **MUST** release IACK*, AM0-AM5, A01-A31, LWORD*, D00-D31, WRITE*, DS0* and DS1* before allowing AS* to rise above the low level. **OBSERVATION 2.43:** Chapter 3 describes how a MASTER'S REQUESTER releases the BBSY* line.
24B.	**RULE 2.51:** IF the MASTER drives or releases AS* to high before its REQUESTER releases BBSY*, THEN it **MUST** release AS*, IACK*, AM0-AM5, A01-A31, LWORD*, D00-D31, WRITE*, DS0* and DS1* before allowing its REQUESTER to release BBSY*. **OBSERVATION 2.44:** Chapter 3 describes how a MASTER'S REQUESTER releases the BBSY* line.
25.	**RULE 2.52:** IF the MASTER drives or releases AS* to high after its REQUESTER releases BBSY*, THEN it **MUST** release AS* within this time after allowing it to rise above the low

	level. **OBSERVATION 2.45:** Chapter 3 describes how a MASTER'S REQUESTER releases the BBSY* line.
26.	**OBSERVATION 2.46:** Timing parameter 26 guarantees that during read cycles, the data bus will not be driven until the MASTER drives DSA* low.
27.	**OBSERVATION 2.47:** During read cycles, the MASTER is guaranteed that the data bus will be valid within this time after DTACK* goes low. This time does not apply to cycles where the SLAVE drives BERR* low instead of DTACK*.
28.	**OBSERVATION 2.48:** The MASTER is guaranteed that neither DTACK* nor BERR* will go low until this minimum time after it drives DSA* low. The BUS TIMER guarantees the MASTER that if DTACK* has not gone low after its timeout period has elapsed and within twice its timeout period, then the BUS TIMER will drive BERR* low.
29.	**OBSERVATION 2.49:** During read cycles, the MASTER is guaranteed that the data bus will remain valid until it drives DSA* high**.**
30.	**OBSERVATION 2.50:** Timing parameter 30 guarantees that neither DTACK* nor BERR* will go high until the MASTER drives both DS0* and DS1* high.
31.	**OBSERVATION 2.51:** During read cycles, the MASTER is guaranteed that the data bus has been released by the time DTACK* and BERR* are high.

Table 2-25. SLAVE, Timing RULES And OBSERVATIONS

Note:
The numbers correspond to the timing parameters specified in Table 2-22.

4.	**OBSERVATION 2.52:** All SLAVES are guaranteed that IACK*, A01-A31, AM0-AM5, and LWORD* have been valid for this minimum time when they detect a falling edge on AS*.
5.	**OBSERVATION 2.53:** All SLAVES are guaranteed this minimum high time on AS* between DTB cycles.
6.	**OBSERVATION 2.54:** During read cycles, the responding SLAVE is guaranteed that none of D00-D31 will be driven by any other module until the responding SLAVE releases DTACK* and BERR* to high.
7.	**OBSERVATION 2.55:** During read cycles, the responding SLAVE is guaranteed that the data bus will be released by all other modules by the time DSA* goes low.
8.	**OBSERVATION 2.56:** During write cycles, the responding SLAVE is guaranteed that the data bus has been valid for this minimum time when it detects a falling edge on DSA*.
9.	**OBSERVATION 2.57:** The responding SLAVE is guaranteed that neither DS0* nor DS1* will go low until

	DTACK* and BERR* from the previous cycle have gone high.
10.	**OBSERVATION 2.58:** Due to bus skew, SLAVES on the DTB might detect a falling edge on DSA* before detecting the falling edge on AS*. However, SLAVES are guaranteed that a falling edge on DSA* will not precede the falling edge on AS* by more than this time.
11 .	**OBSERVATION 2.59:** SLAVES are guaranteed this minimum time during which both data strobes are simultaneously high between consecutive data transfers.
12.	**OBSERVATION 2.60:** SLAVES are guaranteed that WRITE* has been valid for this minimum time before a falling edge on DSA*.
13.	**OBSERVATION 2.61:** IF both data strobes are going to be driven low, THEN the responding SLAVE is guaranteed that DSB* will go low within this maximum time after DSA* has gone low.
14.	**OBSERVATION 2.62:** During all data transfer cycles except readmodifywrite cycles, the responding SLAVE is guaranteed that the address and LWORD* remain valid until it drives DTACK* or BERR* low for the first time, provided that it does so within the bus timeout period.
15.	**OBSERVATION 2.63:** During all readmodifywrite cycles, the responding SLAVE is guaranteed that the address and LWORD* remain valid until it drives DTACK* or BERR* low for the second time, provided that it does so within the bus timeout period.
16.	**OBSERVATION 2.64:** The responding SLAVE is guaranteed that IACK* and AM0-AM5 remain valid until it drives DTACK* or BERR* low for the last time, provided that it does so within the bus timeout period.
17.	**OBSERVATION 2.65:** SLAVES are guaranteed that IACK*, A01-A31, AM0-AM5, and LWORD* will remain valid for this minimum time after the falling edge of AS*. During ADDRESS-ONLY cycles this time is guaranteed by the MASTER. During all other cycle types, this time is derived from timing parameters 1 0, 1 4, 1 6, and 28.
18.	**OBSERVATION 2.66:** The responding SLAVE is guaranteed that AS* will remain low until it drives DTACK* or BERR* low, provided that it does so within the bus timeout period.
19.	**OBSERVATION 2.67:** SLAVES are guaranteed that the AS* will remain low for this minimum time.
20.	**OBSERVATION 2.68:** The responding SLAVE is guaranteed that once DSA* goes low, it will remain low until it drives DTACK* or BERR* low, provided that the SLAVE does so within the bus timeout period.
21.	**OBSERVATION 2.69:** The responding SLAVE is guaranteed that once DSB* goes low, it will remain low until it drives DTACK* or BERR* low, provided that the SLAVE does so within the bus timeout period.
22.	**OBSERVATION 2.70:** During write cycles, the responding SLAVE is guaranteed that the data bus will

	remain valid until it drives DTACK* or BERR* low, provided that it does so within the bus timeout period.
23.	**OBSERVATION 2.71:** The responding SLAVE is guaranteed that the WRITE* line remains valid until both data strobes are high.
26.	**RULE 2.53:** During read cycles, the responding SLAVE **MUST NOT** drive the data bus until DSA* goes low.
27.	**RULE 2.54:** During read cycles, the responding SLAVE **MUST NOT** drive DTACK* before it drives the data lines with valid data. **OBSERVATION 2.72:** RULE 2.54 does not apply to cycles where the responding SLAVE drives BERR* low instead of DTACK*.
28.	**RULE 2.55:** The responding SLAVE **MUST** wait this minimum time after DSA* goes low before driving DTACK* or BERR* low.
29.	**RULE 2.56:** During read cycles, once the responding SLAVE has driven DTACK* low, it **MUST NOT** change D00-D31 until DSA* goes high.
30.	**RULE 2.57:** Once the responding SLAVE has driven DTACK* or BERR* low, it **MUST NOT** release it until it detects both DS0* and DS1* high.
31.	**RULE 2.58:** During read cycles, the responding SLAVE **MUST** release all of D00-D31 before releasing DTACK* or BERR* to high.
32.	**Observation 2.73:** SLAVES are guarenteed that IACK*, LWORD*, A00-A31, and AM0-AM5 have been valid for this minimum time when they detect a falling edge on DSA*. This time is derived from timing parameters 4 and 10.
33.	**OBSERVATION 2.74:** During data transfer cycles, SLAVES are guaranteed that DS0* and/or DS1* will remain low for at least this minimum time. This time is derived from timing parameter 28, where the responding SLAVE is required to wait a minimum time before driving BERR* or DTACK* to low.

Table 2-26. LOCATION MONITOR, Timing Observations

Note:
The numbers correspond to the timing parameters specified in Table 2-22.

4.	**OBSERVATION 2.75:** The LOCATION MONITOR is guaranteed that IACK*, A01-A31 AM0-AM5, and LWORD* have been valid for this minimum time when it detects a falling edge on AS*.
5.	**OBSERVATION 2.76:**

	The LOCATION MONITOR is guaranteed this minimum high time on AS* between DTB cycles.
10.	**OBSERVATION 2.77:** Due to bus skew, LOCATION MONITORS on the DTB might detect a falling edge on DSA* before detecting the falling edge on AS*. However, The LOCATION MONITOR is guaranteed that the falling edge on DSA* will not precede the falling edge on AS* by more than this time.
11.	**OBSERVATION 2.78:** The LOCATION MONITOR is guaranteed this minimum time during which both data strobes are simultaneously high between consecutive data transfers.
12.	**OBSERVATION 2.79:** The LOCATION MONITOR is guaranteed that WRITE* has been valid for this minimum time when it detects a falling edge on DSA*.
13.	**OBSERVATION 2.80:** IF both data strobes are going to be driven low, THEN the LOCATION MONITOR is guaranteed that DSB* will go low within this maximum time after DSA* has gone low.
17.	**OBSERVATION 2.81:** The LOCATION MONITOR is guaranteed that IACK*, A01-A31, AM0-AM5, and LWORD* will remain valid for this minimum time after the falling edge of AS*. During ADDRESS ONLY cycles this time is guaranteed by the MASTER. During all other cycle types, this time is derived from timing parameters 10, 14, 16, and 28.
19.	**OBSERVATION 2.82:** The LOCATION MONITOR is guaranteed that the AS* will remain low for this minimum time.
23.	**OBSERVATION 2.83:** The LOCATION MONITOR is guaranteed that the WRITE* line remains valid until both data strobes go high.
32.	**OBSERVATION 2.84:** The LOCATION MONITOR is guaranteed that IACK*, LWORD*, A01-A31, and AM0-AM5 have been valid for this minimum time when it detects a falling edge on DSA*.
33.	**OBSERVATION 2.85:** During data transfer cycles, the LOCATION MONITOR is guaranteed that DS0* and/or DS1* will remain low for at least this minimum time. This time is derived from timing parameter 28, where the responding SLAVE is required to wait a minimum time before driving BERR* or DTACK* to low.

Table 2-27. BUS TIMER, Timing RULES

Note:
The numbers correspond to the timing parameters specified in Table 2-23.

28.	**RULE 2.59:** The BUS TIMER **MUST** wait at least its timeout time, but no longer than twice its timeout time, after the first data strobe goes low before driving BERR* low.
30.	**RULE 2.60:** Once it has driven BERR* low, the BUS TIMER **MUST NOT** release BERR* until it

	detects both DS0* and DS1* high.

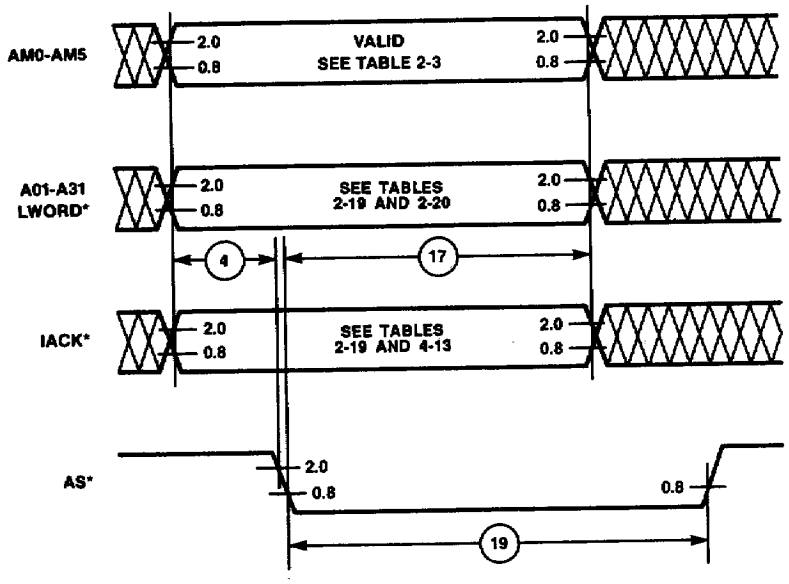


Figure 2-12. Address Broadcast Timing
ALL CYCLES

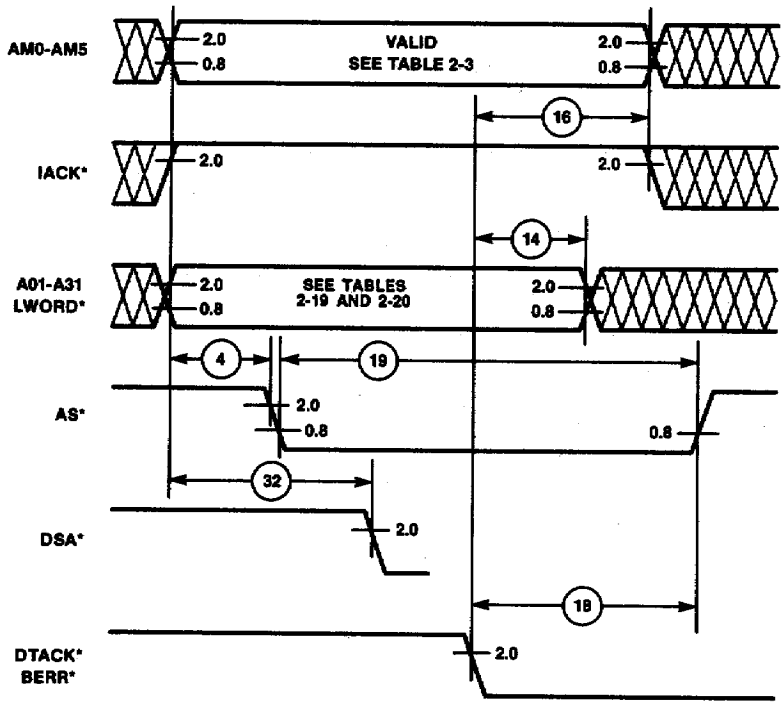


Figure 2-13. MASTER, Responding SLAVE, And LOCATI0N MONITOR
Address Broadcast Timing
Single Even Byte Transfers
Single Odd Byte Transfers
Double Byte Transfers
Quad Byte Transfers

Unaligned Transfers

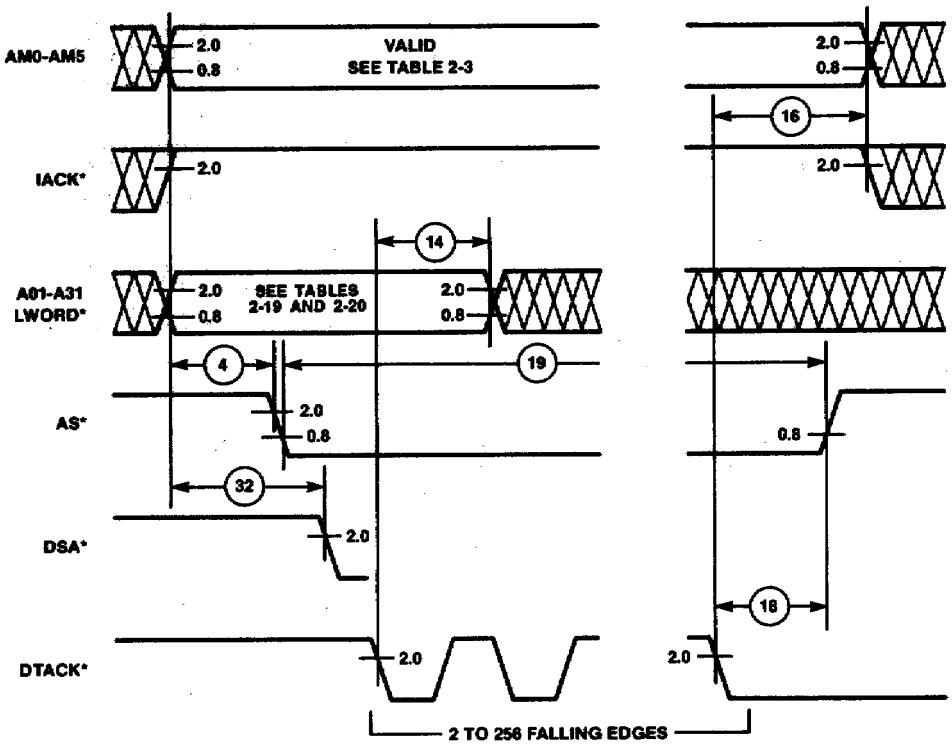


Figure 2-14. MASTER, SLAVE, And LOCATION MONITOR
Address Broadcast Timing
Single Byte Block Transfers
Double Byte Block Transfers
Quad Byte Block Transfers

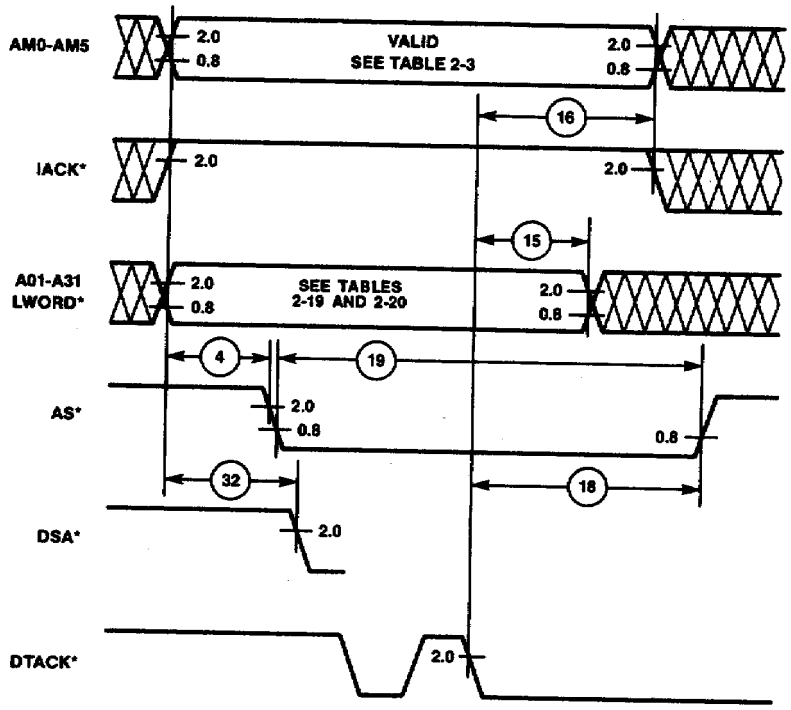


Figure 2-15. MASTER, SLAVE, And LOCATION MONITOR
Address Broadcast Timing
Single Byte RMW Cycles
Double Byte RMW Cycles
Quad Byte RMW Cycles

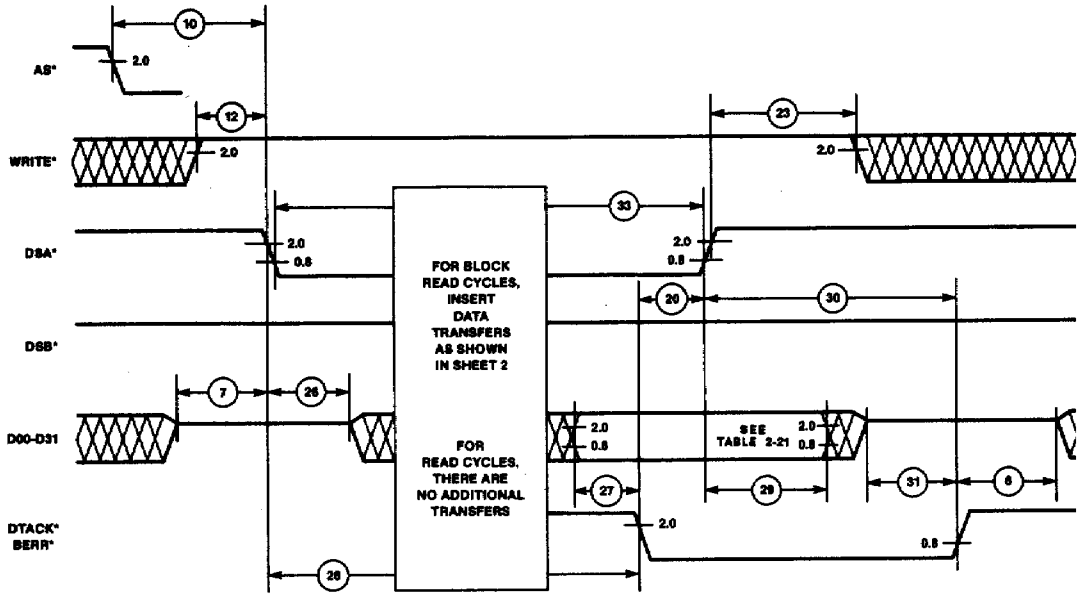


Figure 2-16. MASTER, SLAVE, And LOCATI0N MONITOR, Data Transfer Timing
Byte(0) Read
Byte(1) Read

Byte(2) Read
Byte(3) Read
Byte(0-2) Read
Byte(1-3) Read
Single Byte Block Read

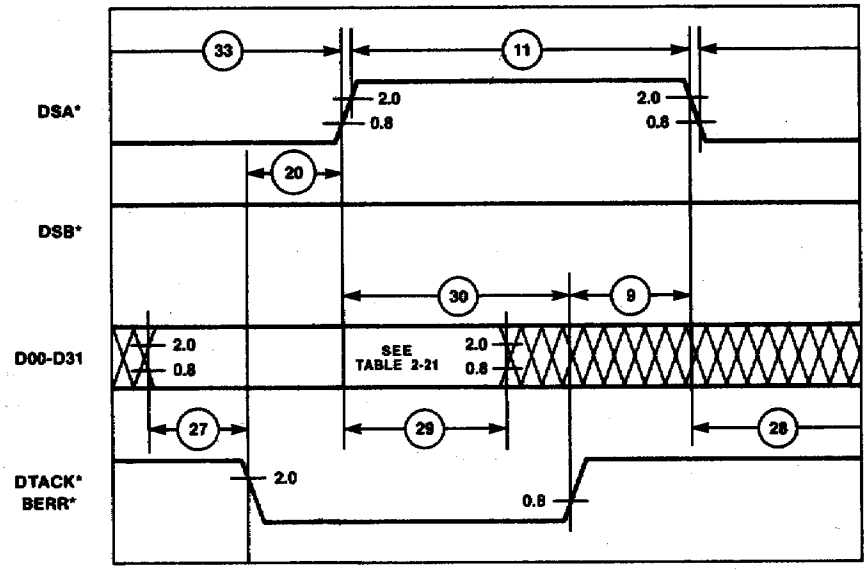


Figure 2-16b. MASTER, SLAVE, and LOCATION MONITOR Data Transfer Timing (cont'd)

Byte(0) Read
Byte(1) Read
Byte(2) Read
Byte(3) Read
Byte(0-2) Read
Byte(1-3) Read
Single Byte Block Read

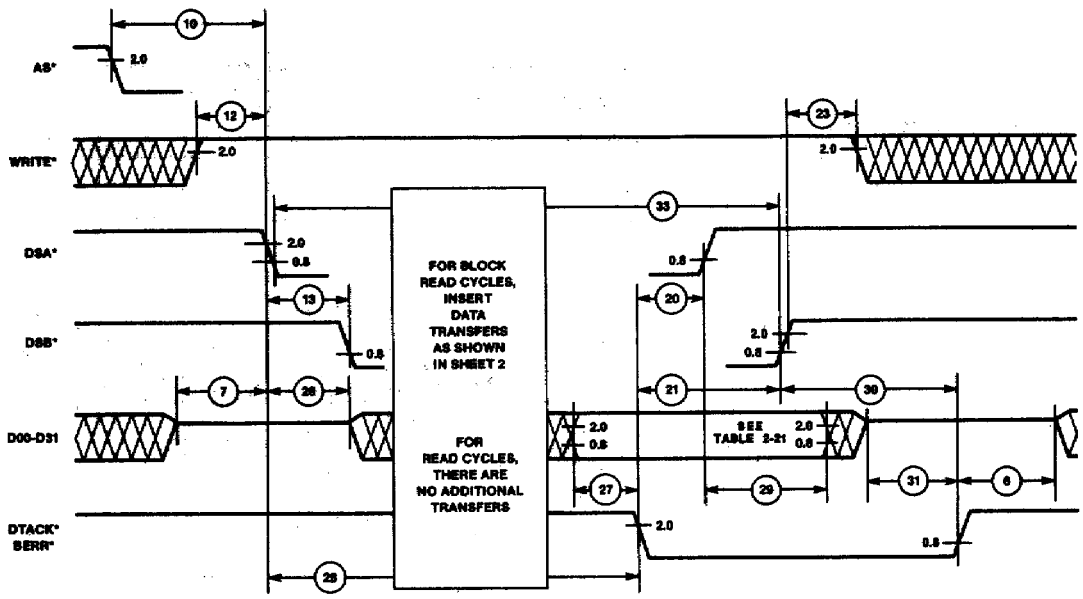


Figure 2-17. MASTER, SLAVE, and LOCATION MONITOR Data Transfer Timing
Byte(0-1) Read
Byte(2-3) Read
Byte(0-3) Read
Byte(1-2) Read
Double Byte Block Read
Quad Byte Block Read

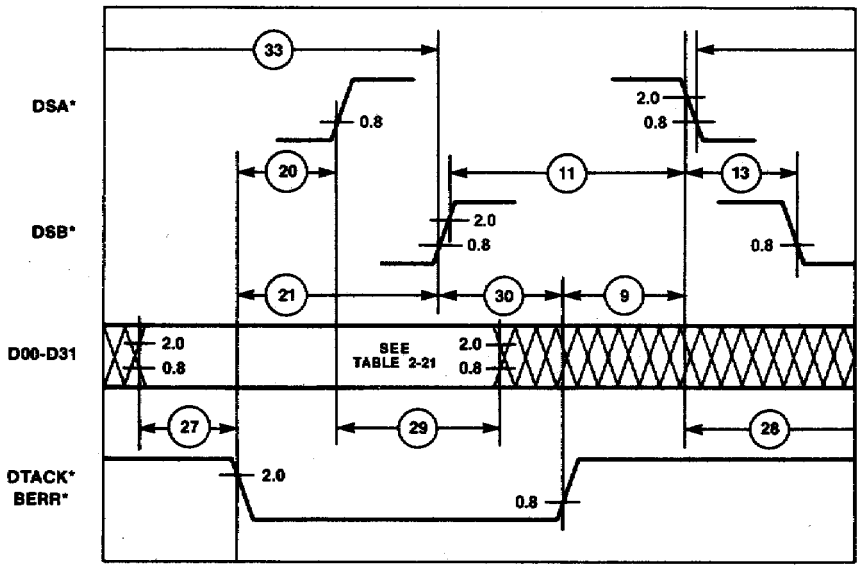


Figure 2-17b. MASTER, SLAVE, And LOCATION MONITOR Data Transfer Timing (cont'd)
Byte(0) Write
Byte(1) Write
Byte(2) Write

Byte(3) Write
Byte(0-2) Write
Byte(1-3) Write
Single Byte Block Write

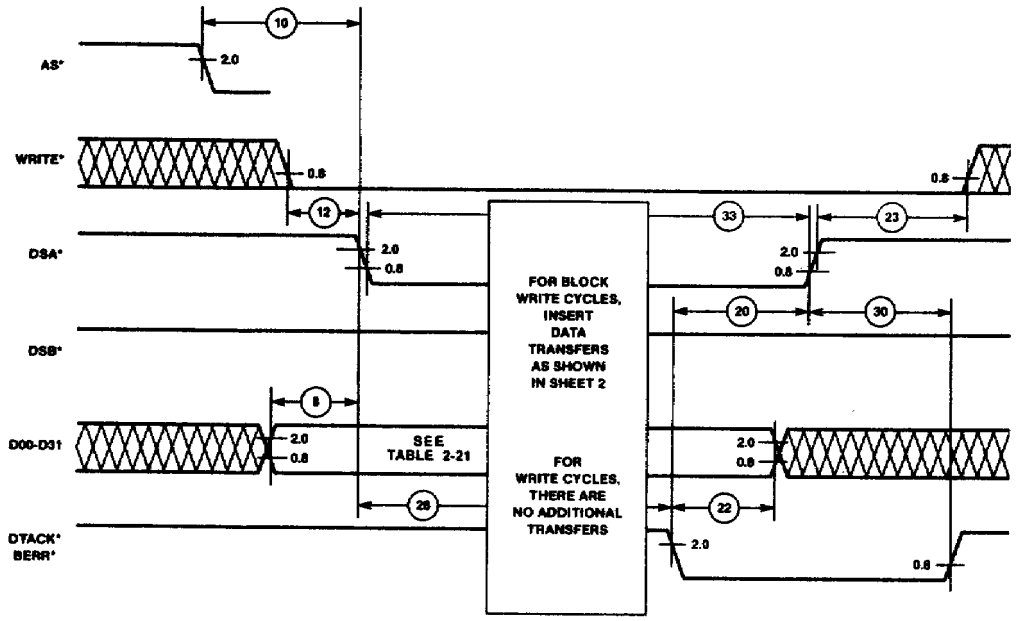


Figure 2-18. MASTER, SLAVE, And LOCATI0N MONITOR Data Transfer Timing

Byte(0) Write
Byte(1) Write
Byte(2) Write
Byte(3) Write
Byte(0-2) Write
Byte(1-3) Write
Single Byte Block Write

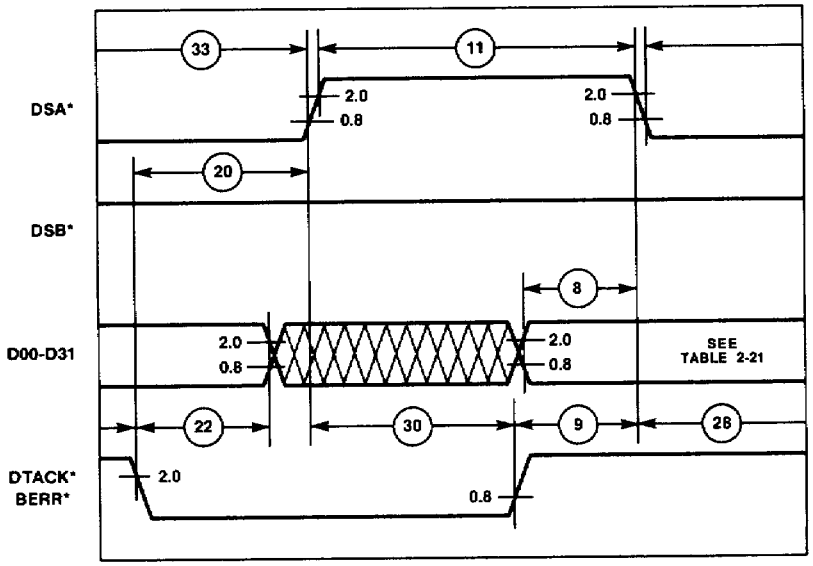


Figure 2-18b. MASTER, SLAVE, And LOCATI0N MONITOR Data Transfer Timing (cont'd)
Byte(0-1) Write
Byte(2-3) Write
Byte(0-3) Write
Byte(1-2) Write
Double Byte Block Write
Quad Byte Block Write

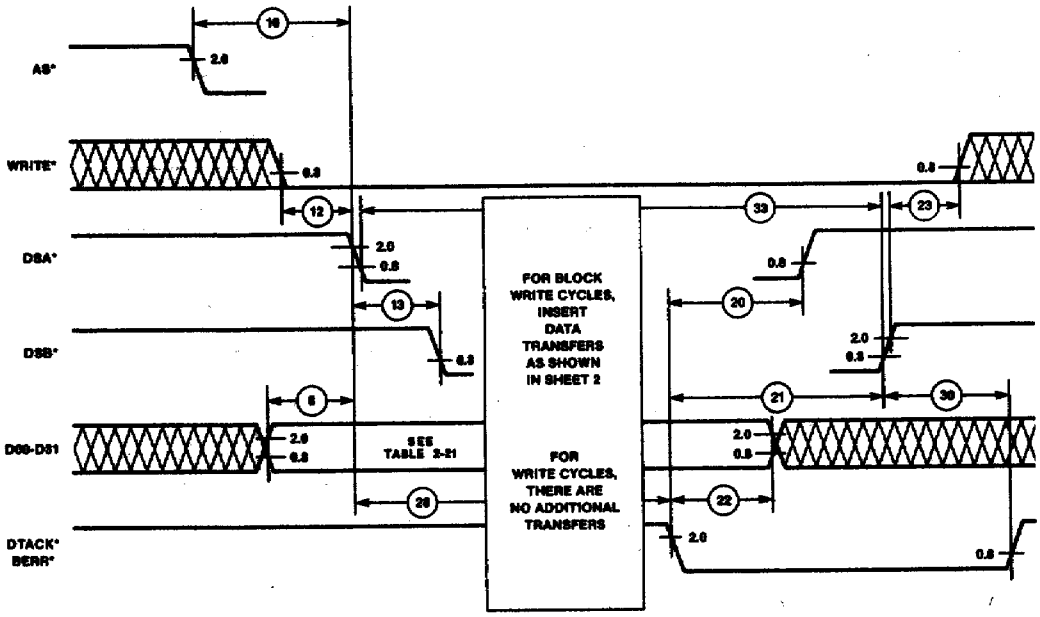


Figure 2-19. MASTER, SLAVE, And LOCATION MONITOR Data Transfer Timing
Byte(0-1) Write
Byte(2-3) Write
Byte(0-3) Write

Byte(1-2) Write
Double Byte Block Write
Quad Byte Block Write

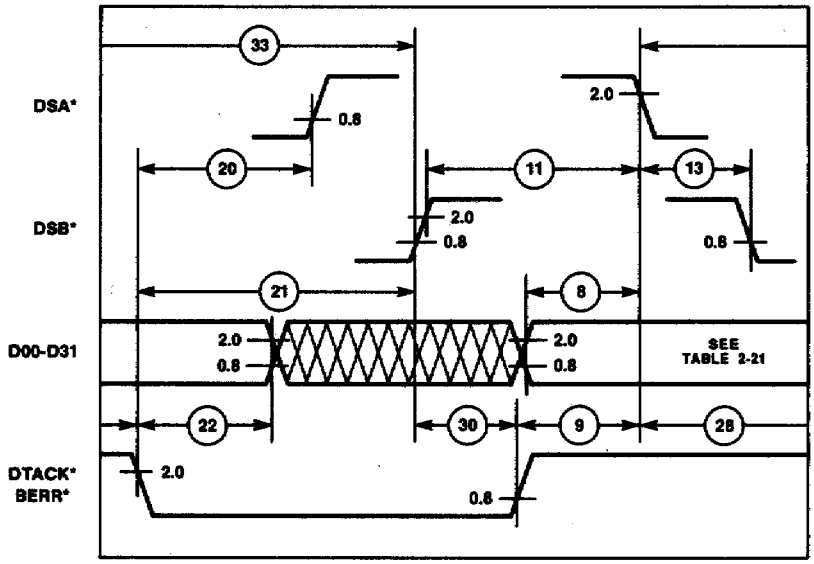


Figure 2-19b. MASTER, SLAVE, And LOCATION MONITOR Data Transfer Timing (cont'd)
Byte(0-1) Write
Byte(2-3) Write
Byte(0-3) Write
Byte(1-2) Write
Double Byte Block Write
Quad Byte Block Write

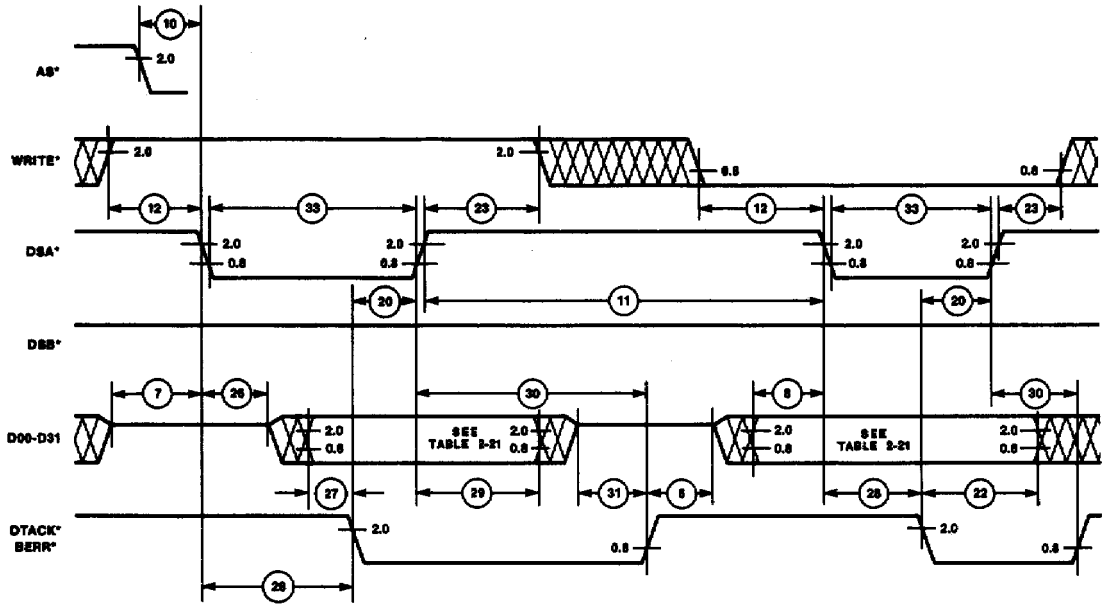


Figure 2-20. MASTER, SLAVE, And LOCATION MONITOR Data Transfer Timing Single Byte RMW Cycles

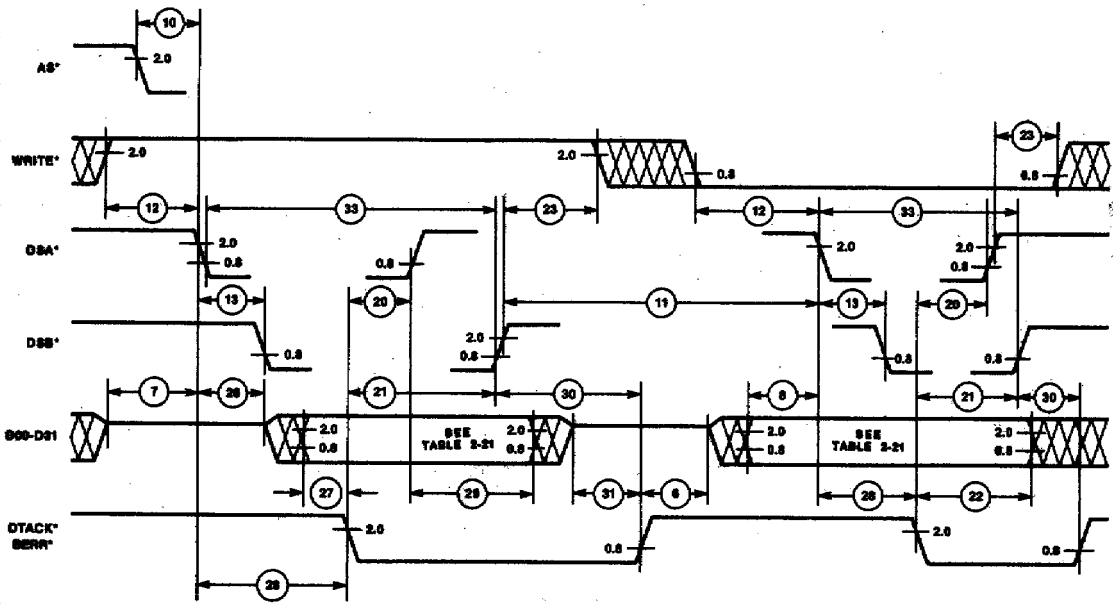


Figure 2-21 . MASTER, SLAVE, And LOCATION MONITOR Data Transfer Timing
Double Byte RMW Cycles
Quad Byte RMW Cycles

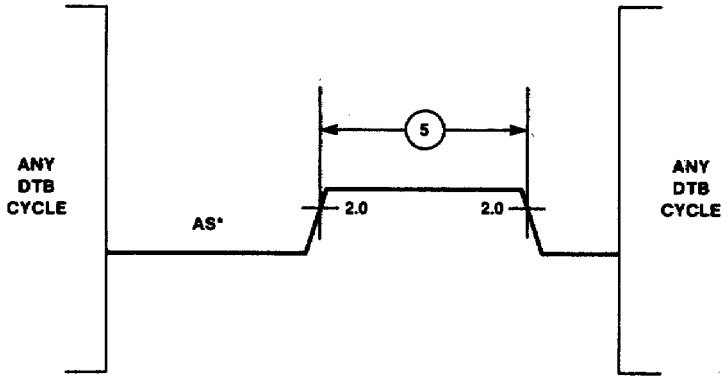


Figure 2-22. Address Strobe Intercycle Timing

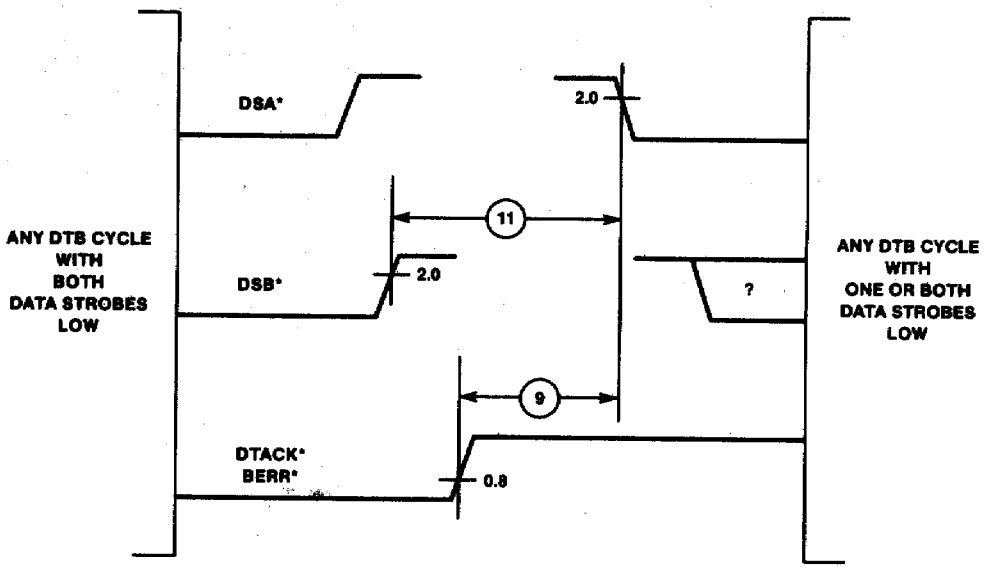


Figure 2-23. Data Strobe Intercycle Timing
A cycle where both data strobes go low followed by a cycle where one or both data strobes go low.

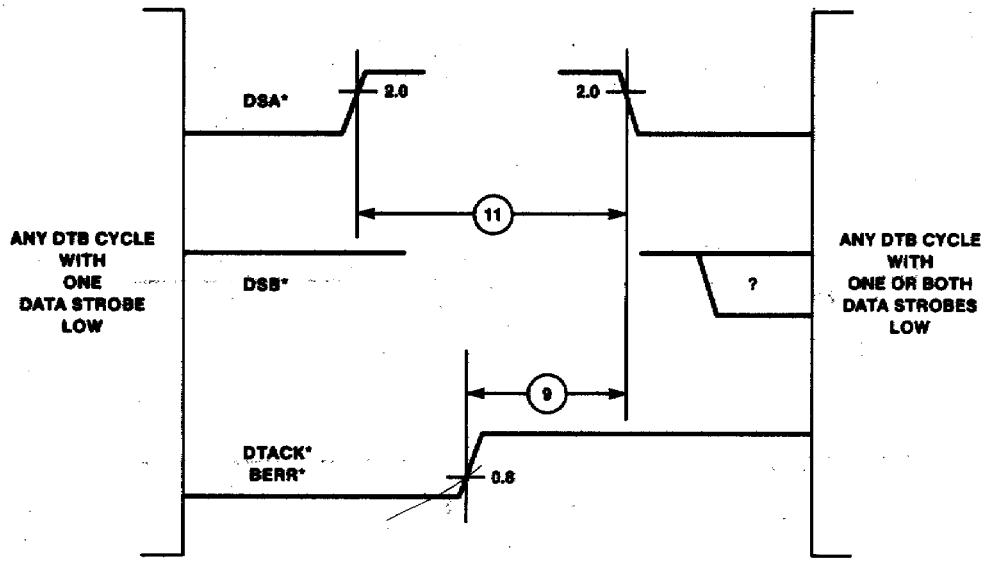


Figure 2-24. Data Strobe Intercycle Timing
A cycle where one data strobe goes low followed by a cycle where one or both data strobes go low

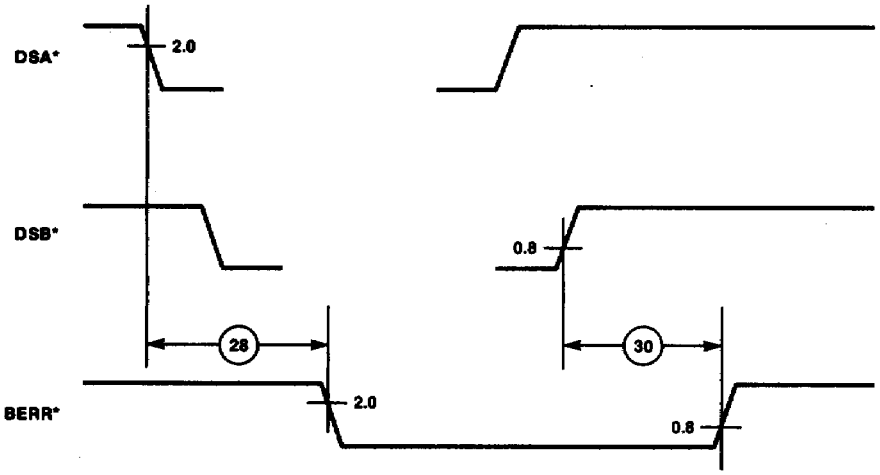


Figure 2-25. MASTER, SLAVE, And BUS TIMER Data Transfer Timing Timed-Out Cycle

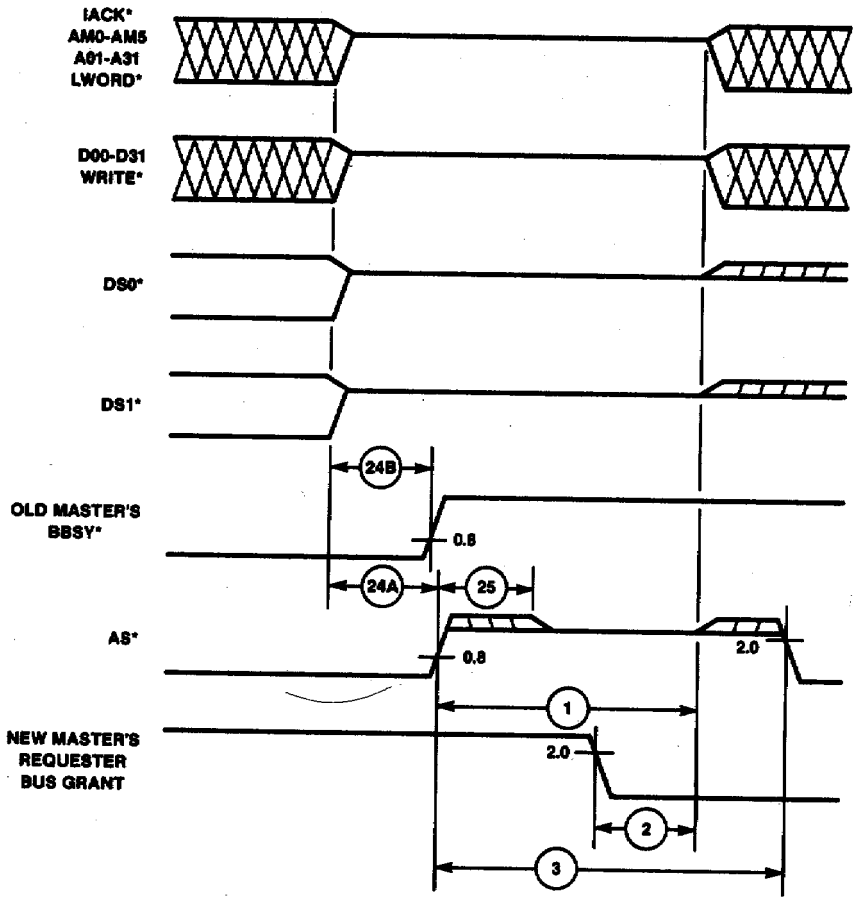


Figure 2-26. MASTER DTB Control Transfer Timing

CHAPTER 3

DATA TRANSFER BUS ARBITRATION

3.1 BUS ARBITRATION PHILOSOPHY

As microprocessor costs decrease, it is becoming more cost effective to design systems with multiple processors sharing global resources.

The most fundamental of these global resources is the Data Transfer Bus through which all other global resources are accessed. Therefore, any system that supports multiprocessing needs to provide an efficient allocation method for the Data Transfer Bus. Because speed of allocation is vital, a hardware allocation scheme is the only practical alternative. The VMEbus meets this need with its Arbitration subsystem. (See Figure 3-1).

The VMEbus arbitration subsystem:

a. Prevents simultaneous use of the bus by two MASTERS.
b. Schedules requests from multiple MASTERS for optimum bus use.

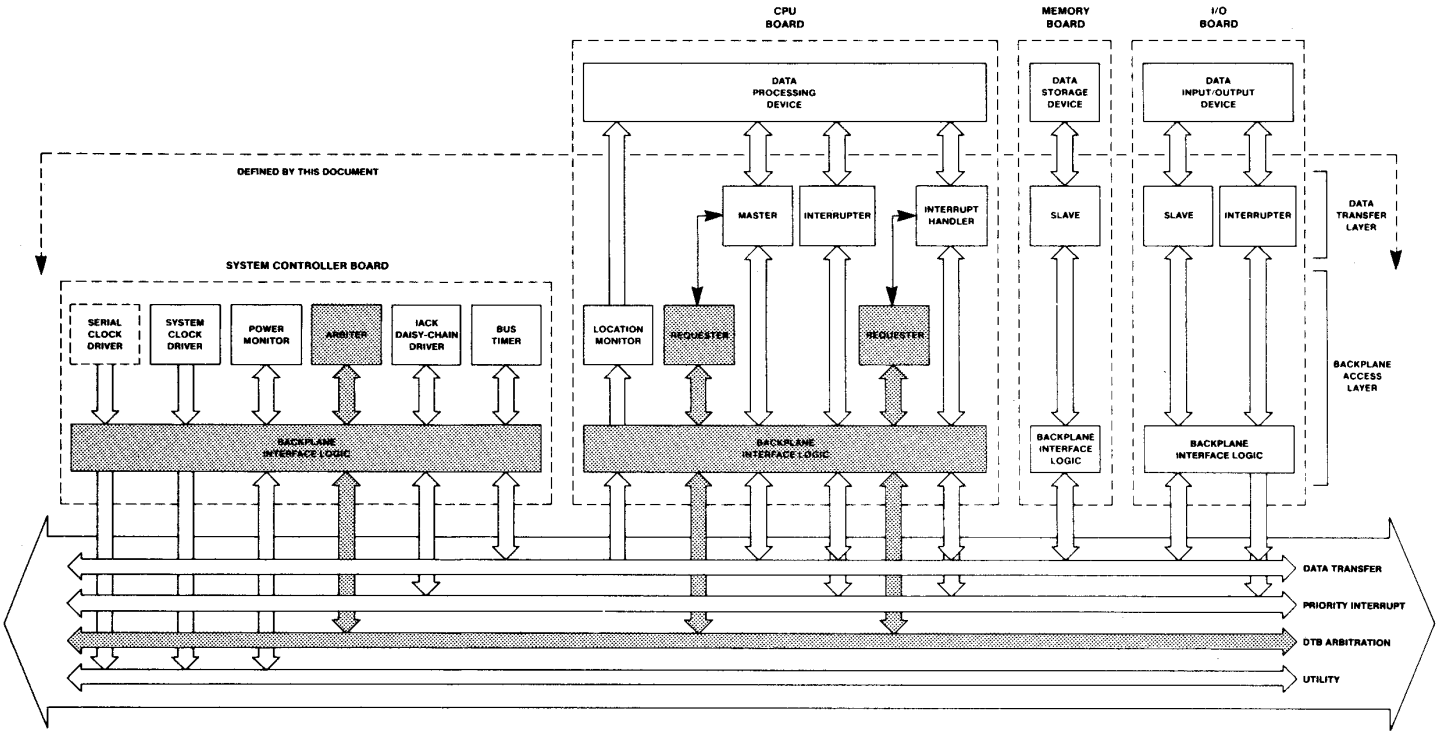


Figure 3-1. Arbitration Bus Functional Block Diagram

3.1.1 Types Of Arbitration

When several boards request use of the DTB simultaneously, the arbitration subsystem detects these requests and grants the bus to one board at a time. The decision of which board is granted the bus first depends upon what scheduling algorithm is used. Many algorithms are possible. The VMEbus describes three: prioritized, round-robin, and single level.

Prioritized arbitration assigns the bus according to a fixed priority scheme where each of four bus request lines has a priority from highest (BR3*) to lowest (BR0*).

Round-robin arbitration assigns the bus on a rotating priority basis. When the bus is granted to the REQUESTER on bus request line "BR(n)*" then the highest priority for the next arbitration is assigned to bus request line "BR(n-1)*".

Single level arbitration only accepts requests on BR3*, and relies on BR3*'s bus grant daisy-chain to arbitrate the requests.

PERMISSION 3.1:
Scheduling algorithms other than priority, round-robin, or single level **MAY** be used. For example, an ARBlTER'S algorithm might give highest priority to BR3*, but grant the bus to BR0* through BR2* on a round-robin basis.

3.2 ARBITRATION BUS LINES

The Arbitration Bus consists of six bused VMEbus lines and four daisy-chained lines. These daisy-chained lines require special signal names. The signals entering each board are called "Bus Grant IN" lines (BGxIN*), while the signals leaving each board are called "Bus Grant OUT" lines (BGxOUT*). The lines which leave slot n as BGxOUT* enter slot n+1 as BGxIN*. This is illustrated in Figure 3-2.

OBSERVATION 3.1:
In all descriptions in this chapter, the terms BRx*, BGxIN*, and BGxOUT* are used to describe the bus request and bus grant lines, where x takes on any value from zero to three.

In the VMEbus arbitration system, a REQUESTER module drives the following lines:

1 bus request line	(one of BR0* through BR3*)
1 bus grant out line	(one of BG0OUT* through BG3OUT*)
1 bus busy line	(BBSY*)

RULE 3.1:
IF a VMEbus board does not generate bus requests on some bus request levels,
THEN it **MUST** propagate the daisy-chain signals for those levels from its BGxIN* lines to its respective BGxOUT* lines.

PERMISSION 3.2:
The propagation for the unused lines of the bus grant daisy-chain can be done using jumpers or active logic. The latter approach allows selection of the request level under software control, while the former results in faster propagation through the daisy-chain.

Three types of ARBITERS are described in the VMEbus standard. They are:

PRIoritized	(PRI)

Round-Robin-Select (RRS)
SinGLe level (SGL)

The operation of these three types of ARBITERS is described in Section 3.3.

A PRI ARBlTER drives the following:

1 bus clear line	(BCLR*)
4 bus grant lines	(Slot 1 BG0IN* through BG3IN*) if the ARBITER'S board also has a REQUESTER. (Slot 1 BG0OUT* through BG3OUT*) if the ARBlTER'S board does not have a REQUESTER.

An RRS ARBITER drives the four Slot 1 BGxIN* or BGxOUT* lines and, optionally, the BCLR* line.

A SGL ARBlTER drives only BG3IN* or BG3OUT* at slot 1.

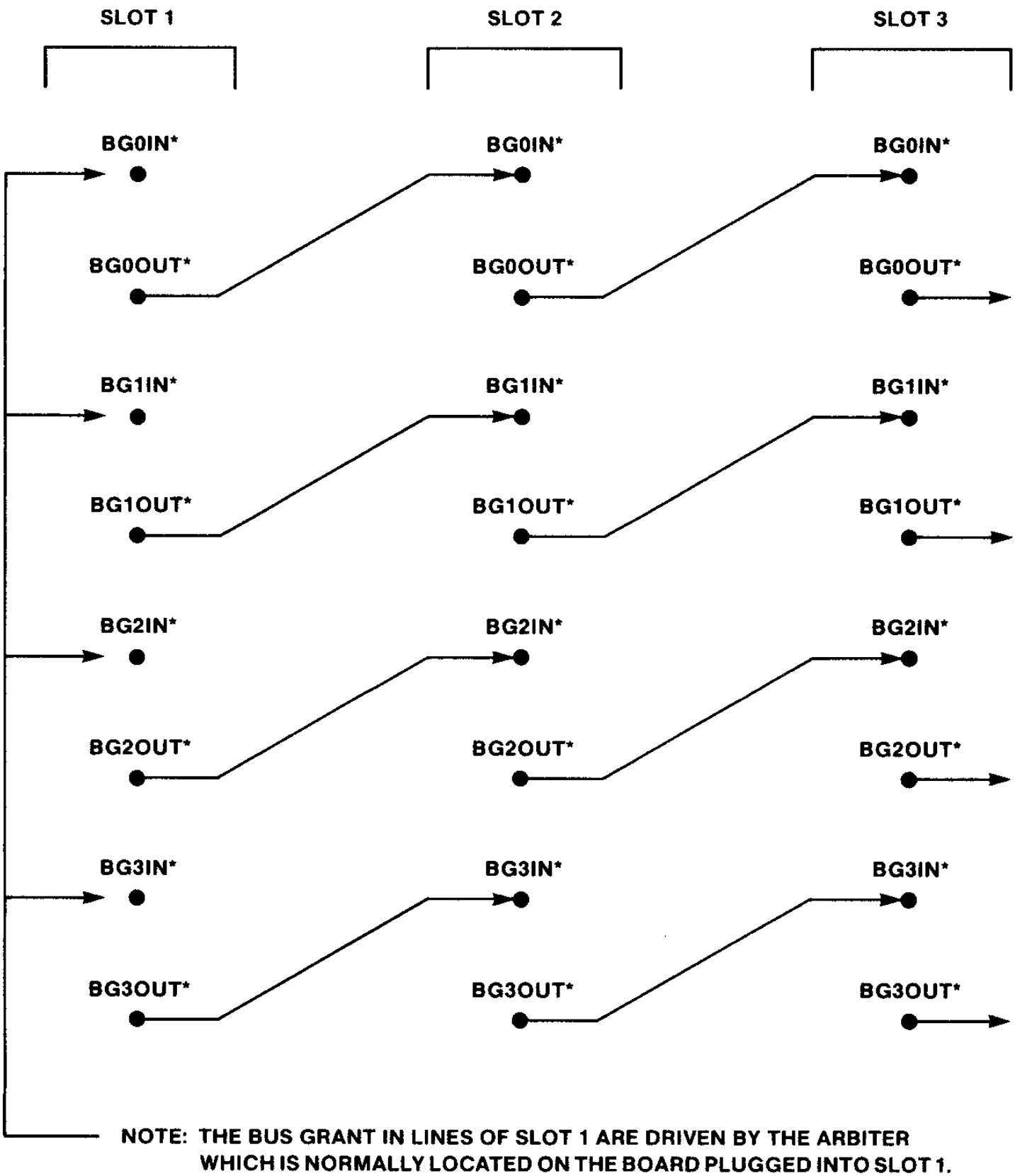


Figure 3-2. Illustration Of The Daisy Chained Bus Grant Lines

Two additional lines are connected with the arbitration system during power-up and power-down sequencing: SYSRESET*, and ACFAIL*. While their impact on the arbitration system is included in this chapter, these lines will be discussed further in Chapter 5.

3.2.1 Bus Request And Bus Grant Lines

The bus request lines are used by each REQUESTER to request use of the DTB. The bus grant lines allow the ARBITER to award use of the bus. It does this by driving a bus grant daisy-chain line low. This low level propagates down the daisy-chain, typically passing through several boards in the process. If a board never uses a particular request/grant level, the signal is passed through that board. Where the board uses a request/grant level x, the corresponding signal BGxIN* is gated on board. If its on-board REQUESTER is currently requesting the DTB on that level, it does not pass the low level on to its BGxOUT*. Otherwise, it passes on the low level.

RULE 3.2:
IF a VMEbus backplane slot is not occupied by a board, and if there are boards farther down the daisy-chain,
THEN jumpers **MUST** be installed at the empty slot to pass through the daisy-chain signal.

OBSERVATION 3.2:
The backplane mechanical specification in Chapter 7 describes a provision for the installation of jumpers at each slot.

RULE 3.3:
The ARBITER **MUST** always be located in slot 1.

3.2.2 **Bus Busy Line (BBSY*)**

Once a REQUESTER has been granted control of the Data Transfer Bus via the bus grant daisy-chain, it drives BBSY* low. It then has control of the DTB until it releases BBSY*, allowing the ARBITER to grant the DTB to some other REQUESTER.

3.2.3 Bus Clear Line (BCLR*)

The PRI ARBITER drives BCLR* low to inform the MASTER, currently in control of the DTB, when a higher priority request is pending. The current MASTER does not have to relinquish the bus within any prescribed time limit. It can continue transferring data until it reaches an appropriate stopping point, and then allow its on-board REQUESTER to release BBSY*.

PERMISSION 3.3:
Although RRS ARBITERS are not required to drive the BCLR* line they **MAY** do so.

SUGGESTION 3.1:
IF a RRS ARBITER drives the BCLR* line low,
THEN design it to do so whenever there is a request pending on any of the nongranted bus request lines.

3.3 FUNCTIONAL MODULES

The arbitration subsystem is composed of several modules:
a. One ARBITER
b. One or more REQUESTERS

Figures 3-3 and 3-4 provide block diagrams for the two types of Arbitration Bus modules.

RULE 3.4:
Output signal lines shown with solid lines in Figure 3-3 and 3-4 **MUST** be driven by the module, unless it would always drive them high.

RULE 3.5:
input signal lines shown with solid lines in Figures 3-3 and 3-4 **MUST** be monitored and responded to in the appropriate fashion.

OBSERVATION 3.3:
RULES and PERMISSIONS for driving and monitoring the signal lines shown with dotted lines in Figures 3-3 and 3-4 are given in Tables 3-1 and 3-2.

OBSERVATION 3.4:
IF an output signal line is not driven,
THEN terminators on the backplane ensure that it is high.

OBSERVATION 3.5:
Although SYSRESET* and ACFAIL* are not specified as part of the Arbitration Bus, they are important here because MASTERS, which are paired with REQUESTERS, respond to these signal lines. (SYSRESET* and ACFAIL* are driven by the POWER MONITOR module which is discussed in Chapter 5.)

3.3.1 ARBITER

The ARBITER is a functional module that decides which REQUESTER should be granted control of the DTB when several request it simultaneously. There are many possible algorithms that could be used to make this decision. Three types of ARBITERS are described in this specification: a prioritized (PRI) ARBITER, a round robin select (RRS) ARBITER, and a single level (SGL) ARBITER.

An ARBITER responds to incoming bus requests and grants the DTB to the appropriate REQUESTER with one of the bus grant lines.

When the ARBITER detects BBSY* high, and after it detects one or more bus requests, it issues a bus grant, corresponding to the highest priority bus request.

When the REQUESTER receives the bus grant, it drives BBSY* low and signals to its on-board MASTER or INTERRUPT HANDLER that it has been granted the DTB. After its on-

board MASTER or INTERRUPT HANDLER finishes using the DTB, the REQUESTER releases BBSY*. The resulting rising edge of BBSY* enables the ARBITER to issue another bus grant, based upon the levels of the bus request lines at that time.

In addition to the arbitration provided by the ARBITER, a secondary level of arbitration is provided by the bus grant daisy-chains. Because of these daisy-chains, REQUESTERS sharing a common request line are prioritized by slot position. The REQUESTER closest to slot 1 has the highest priority.

SGL ARBITERS respond only to bus requests on BR3* and depend on the BG3IN*/BG30UT* daisy-chain to do the arbitration.

The PRI ARBITER prioritizes the four bus request lines, from BR0* (the lowest) to BR3* (the highest), and responds with BG0IN* through BG3IN*, as appropriate. A PRI ARBlTER also informs any MASTER, currently in control of the bus, when a higher level request is pending by driving BCLR* low.

To visualize an RRS ARBITER, consider a mechanical switch being driven by a stepping motor. Each position on the switch connects a bus request line to its corresponding bus grant line. When the bus is busy, the switch is stopped on the current level. Upon release of the bus, the switch steps one position lower (i.e., from BR(n)* to BR(n-1)*) and tests for a request. It continues this scanning operation until a request is found, sending a bus grant over the appropriate line.

PERMISSION 3.4:
An ARBITER **MAY** be designed with built-in time-out feature that causes it to withdraw a bus grant if BBSY* is not driven low by a REQUESTER within a prescribed time.

OBSERVATION 3.6:
The time used by the ARBITER allowed by PERMISSION 3.4 needs to be longer than the longest possible bus grant daisy-chain propagation delay time, plus the time the slowest REQUESTER takes to generate BBSY*.

RULE 3.6:
Except for a time-out situation where no REQUESTER responds, once the ARBlTER grants the bus to a REQUESTER it **MUST NOT** generate a new bus grant before that REQUESTER generates a rising edge on the BBSY* line. (The REQUESTER generates a rising edge by driving BBSY* low and then releasing it.)

OBSERVATION 3.7:
IF an ARBITER uses a “snapshot” of the request lines taken prior to the rising edge of BBSY*,
THEN it might grant the bus to a REQUESTER that has since removed its request.

Note: The RULES and PERMISSIONS for monitoring and driving the dotted lines are given in Table 3-1.

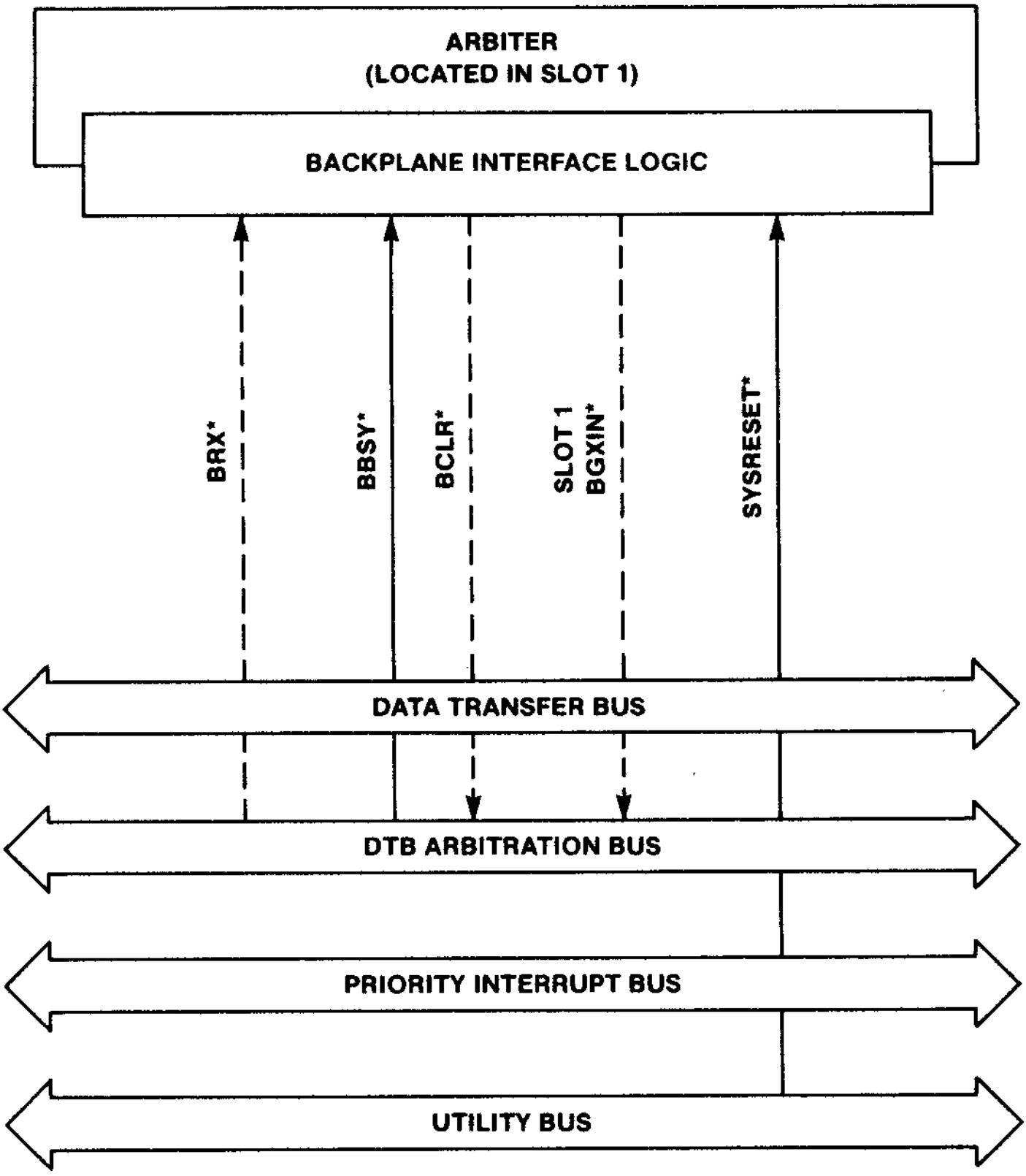


Figure 3-3. Block Diagram: ARBITER

Table 3-1. RULES And PERMISSIONS That Specify The Use Of The Dotted Lines By The Various Types Of ARBITERS Defined By This Document

Type of ARBITER	Use of dotted line
SGL	**MUST** drive slot 1 BG3IN*. **MUST** ensure slot 1 BG0IN*-BG2IN* are high. **MUST** monitor BR3*. **MAY** or **MAY** not drive BCLR*, or slot 1 BG0IN*-BG2IN **MAY** or **MAY** not monitor BR0*-BR2*.
RRS	**MUST** drive slot 1 BG0IN*- BG3IN*. **MUST** monitor BR0*-BR3*. **MAY** or **MAY** not drive BCLR*.
PRI	**MUST** drive slot 1 BG0IN*-BG3IN*, and BCLR*. **MUST** monitor BR0*-BR3*.

3.3.2 REQUESTER

Each REQUESTER in the system:

a. Monitors the DEVICE WANTS BUS from its on-board MASTER or INTERRUPT HANDLER and generates a bus request when the DTB is needed.

b. If it detects a low level on its BGxIN* line, and its on-board MASTER or INTERRUPT HANDLER does not need the DTB, then it passes on that low level to its BGxOUT*

c. If it detects a low level on its BGxIN* line, and its on-board MASTER or INTERRUPT HANDLER needs the DTB, it generates an on-board DEVICE GRANTED BUS signal to indicate the DTB is available, and drives the BBSY* signal low.

Two types of REQUESTERS are described in this specification: a Release When Done (RWD) REQUESTER and a Release On Request (ROR) REQUESTER.

The RWD REQUESTER releases BBSY* when the MASTER or INTERRUPT HANDLER drives the on-board DEVICE WANTS BUS signal false.

The ROR REQUESTER does not release BBSY* when its on-board DEVICE WANTS BUS signal goes false unless some other REQUESTER on the bus drives one of the bus request lines Iow. It monitors the four bus request lines and releases BBSY* only if another bus request is pending. ROR REQUESTERS reduce the number of arbitrations initiated by a MASTER which is generating a large percentage of the bus traffic. See Figure 3-4.

Assuming that the REQUESTER'S DEVICE WANTS BUS input is true, when it receives a bus grant it does 3 things:

a. It drives BBSY* to low.
b. It releases its low on BRx.

c. It drives the DEVICE GRANTED BUS on-board signal true, allowing the MASTER or INTERRUPT HANDLER to initiate bus transfers.

These events might occur in any order. It is even possible, although meaningless, that the MASTER or INTERRUPT HANDLER might not use the bus in response to this particular grant.

However, the following RULES apply:

RULE 3.7:
The REQUESTER **MUST** drive BBSY* to low for at least 90 nanoseconds.

RULE 3.8:
The REQUESTER **MUST** release bus request to high.

RULE 3.9:
The REQUESTER **MUST** maintain BBSY* low for at least 30 nanoseconds after it releases its bus request.

OBSERVATION 3.8:
The 30 nanosecond delay between the bus request's rising edge and the BBSY* rising edge ensures that the ARBITER does not mistakenly interpret the old bus request as a new one and issue another grant.

RULE 3.10:
The REQUESTER **MUST** hold BBSY* low until its bus grant goes high.

OBSERVATION 3.9:
RULE 3.10 ensures that the BBSY* transition to low has been seen by the ARBITER and that all segments of the bus grant daisy-chain have returned to high, in preparation for the next arbitration.

PERMISSION 3.5:
IF a REQUESTER has a bus request pending and, if it sees some other REQUESTER drive BBSY* low,
THEN it **MAY** withdraw its request by releasing the bus request line.

RULE 3.1 1:
IF a REQUESTER withdraws a bus request without having first been granted the bus,
THEN it **MUST** wait to do so until BBSY* goes low and it **MUST** do so within 50 nanoseconds after BBSY* goes low.

SUGGESTION 3.2:
Design REQUESTERS so that they pass on the bus grant daisy-chain as fast as possible after receipt of a bus grant. This will improve system performance.

Note: The RULES and PERMISSIONS for monitoring the dotted lines are given in Table 3-2.

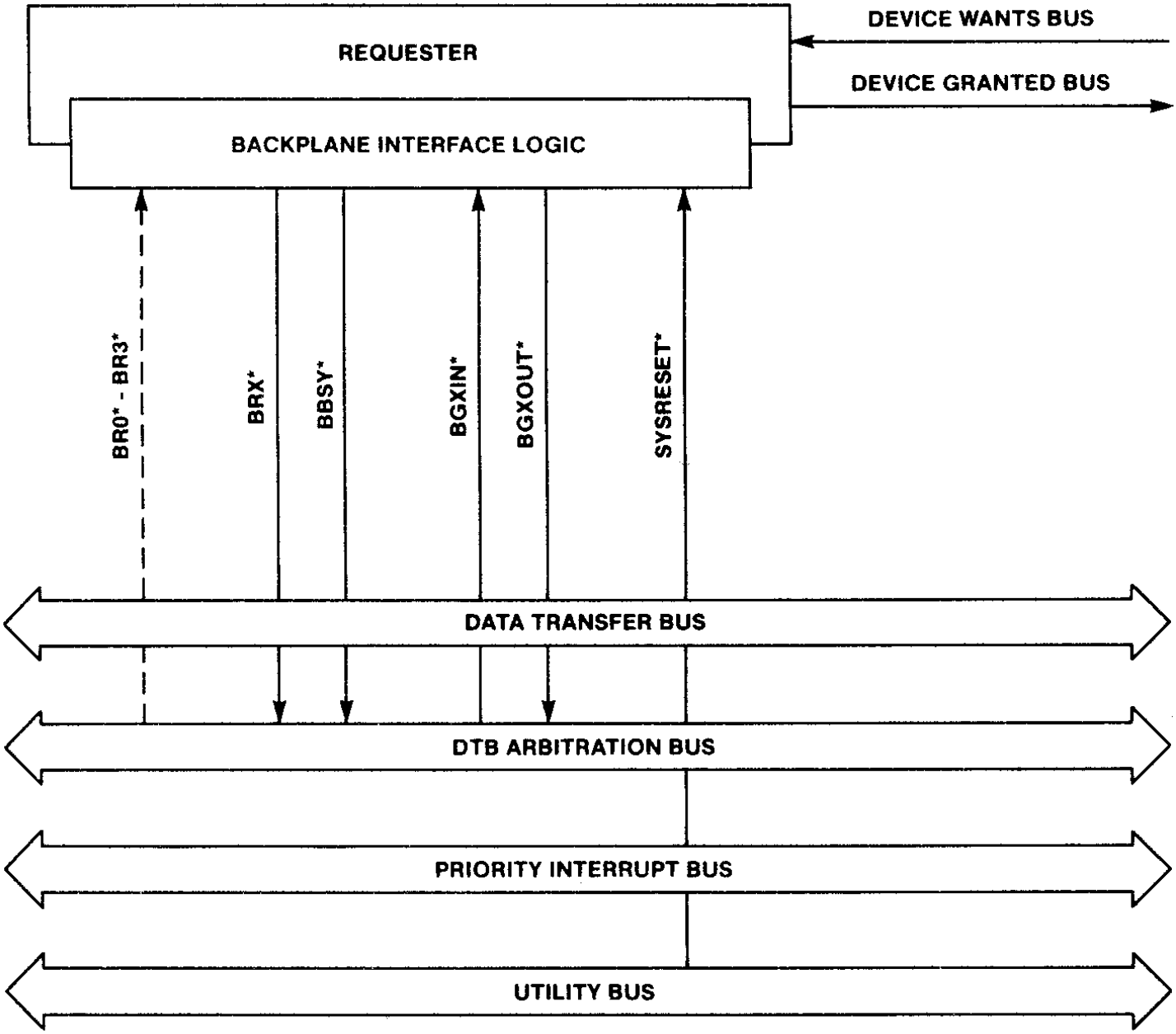


Figure 3-4. Block Diagram: REQUESTER

Table 3-2. RULES and PERMISSIONS That Specify The Use Of The Dotted Lines By The Various Types Of REQUESTERS Defined By This Document

Type of REQUESTER	Use of dotted line
RWD	**MAY** or **MAY** not monitor BR0*-BR3*.
ROR	**MUST** monitor BR0*-BR3*.

3.3.3 Data Transfer Bus MASTER

3.3.3.1 Release Of The DTB

The bus arbitration protocol determines how and when the DTB is granted to the various MASTERS and INTERRUPT HANDLERS in the system. It does not, however, control when MASTERS and INTERRUPT HANDLERS release the DTB.

MASTERS and INTERRUPT HANDLERS use several criteria in deciding when to release the DTB. INTERRUPT HANDLERS give up the bus after their interrupt acknowledge cycle. MASTERS give up the bus when they finish their data transfers.

Some MASTERS also monitor the ACFAIL* and BCLR* VMEbus signals. Both of these signals inform the MASTER that the DTB is needed for some higher priority activity. In the case of BCLR*, the MASTER'S design determines how long it takes to release the bus. For example, a MASTER on a disk controller board might not be able to relinquish the bus during a disk sector transfer without loss of data, so it might keep the bus until the sector transfer is finished. ACFAIL* informs the MASTER that an an AC power loss has been detected, and whatever problems the MASTER will face in surrendering the bus are insignificant compared to the needs of the total system.

RECOMMENDATION 3.1:
Design MASTERS to release the DTB within 200 microseconds after ACFAIL* goes low, except to participate in the ensuing power failure activities.

OBSERVATION 3.10:
The 200 microsecond specified in RECOMMENDATION 3.1 is intended to provide time for an orderly shut-down of the system.

Whatever criteria are used to decide when to release the DTB, arbitration is done before some other MASTER or INTERRUPT HANDLER begins using it. This arbitration takes place either during the last data transfer or after that transfer, depending on how much notice the MASTER or INTERRUPT HANDLER gives to its on-board REQUESTER.

PERMISSION 3.6:
MASTERS and INTERRUPT HANDLERS **MAY** release the DTB either during or after their last data transfer.

For example, if the MASTER notifies its on-board REQUESTER that it no longer wants the bus during its last data transfer, the REQUESTER releases BBSY* and arbitration takes place during the last transfer. But if the MASTER waits until the last transfer has completed before signaling its on-board REQUESTER, the DTB remains idle while the arbitration is done. (This was illustrated in Section 2.5.1)

Chapters 2 and 4 contain RULES that pertain to the release of the DTB.

SUGGESTION 3.3:
Design block transfer MASTER boards so that they signal their REQUESTER to release BBSY* during the last data strobe of the block transfer. If it is released at the beginning of the block transfer, high priority bus requests initiated during the block transfer might not be taken into account by the ARBITER until the next arbitration cycle.

3.3.3.2 Acquisition Of The DTB

To ensure that no DTB line is ever driven to opposite states by two MASTERS or INTERRUPT HANDLERS, when these modules take control of the DTB they are constrained by the following rule:

RULE 3.1 2:
When a MASTER or INTERRUPT HANDLER is given control of the DTB by its onboard REQUESTER, it **MUST** wait until it detects AS* high before turning on its DTB drivers.

OBSERVATION 3.11:
IF the previous MASTER or INTERRUPT HANDLER releases the bus DURING its last data transfer,
THEN RULE 3.12 ensures that the data transfer will be finished before the new MASTER or INTERRUPT HANDLER starts using the DTB. (If the previous MASTER or INTERRUPT HANDLER waited until the data transfer was finished before releasing the bus, AS* will already be high.)

3.3.3.3 Other Information

RECOMMENDATION 3.2:
To allow for prompt servicing of interrupt requests and for optimum use of the DTB, design MASTERS so that they release the DTB as soon as possible after they detect BCLR* Iow.

PERMISSION 3.7:
A MASTER or INTERRUPT HANDLER **MAY** have more than one REQUESTER, where each REQUESTER generates bus requests on a different bus request line.

OBSERVATION 3.12:
Where a MASTER or INTERRUPT HANDLER has two or more REQUESTERS, it can do high priority data transfers using one REQUESTER and low priority transfers using another REQUESTER.

3.4 TYPICAL OPERATION

3.4.1 Arbitration Of Two Different Levels Of Bus Request

Figures 3-5 and 3-6 illustrate the sequence of events that takes place when two REQUESTERS send simultaneous bus requests to a PRl ARBITER on different bus request lines. When the sequence begins, each of the REQUESTERS drives its respective bus request line low (REQUESTER A drives BR1* and REQUESTER B drives BR2*). The ARBITER detects BR1* and BR2* low simultaneously, and it drives BG2IN* low to its own slot (slot 1). That BG2IN* signal is monitored by REQUESTER B (also in slot 1). When REQUESTER B detects BG2IN* low, it responds by driving BBSY* low. REQUESTER B then releases the BR2* line and informs its own MASTER (MASTER B) that the DTB is available. When BBSY* goes low, the ARBlTER drives BG2IN* of slot 1 high.

When MASTER B completes its data transfer(s), and signals that fact by driving DEVICE WANTS BUS false, REQUESTER B releases BBSY*, provided that its BG2IN* has been received high and 30 nanoseconds have elapsed since it released BR2*.

The ARBITER interprets the release of BBSY* as a signal to arbitrate any current bus requests. Since BR1* is the only bus request being driven low, the ARBITER grants the DTB to REQUESTER A by driving BG1IN* low. REQUESTER A responds by driving BBSY* low. When MASTER A completes its data transfer(s) and signals that fact by driving DEVICE WANTS BUS false, REQUESTER A releases BBSY*, provided that its BG1IN* has been received high and 30 nanoseconds have elapsed since it released BR1*.

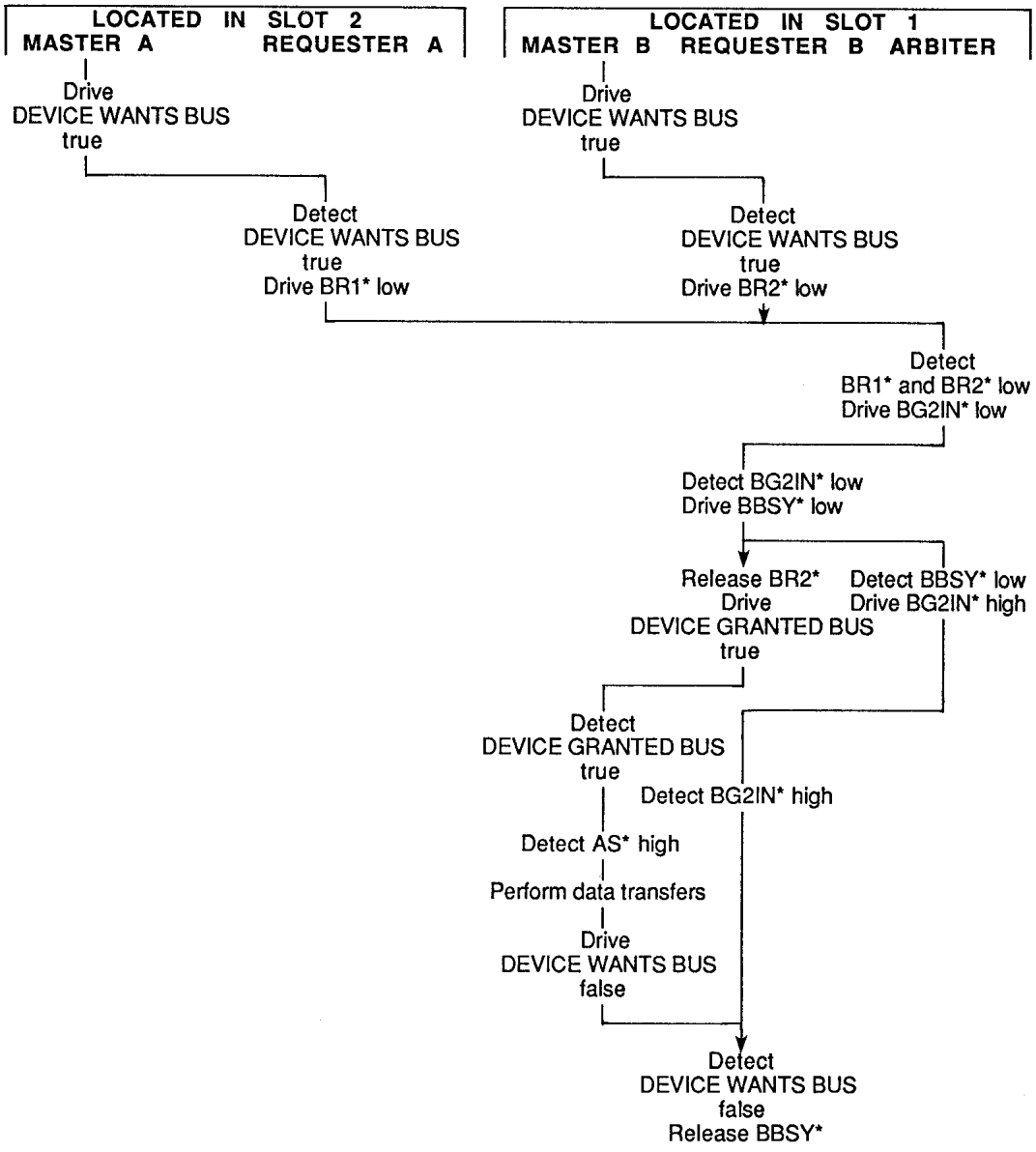
In this example, since no bus request lines are low when REQUESTER A releases BBSY*, the ARBlTER waits until it detects a bus request.

OBSERVATION 3.13:
The description illustrated in Figures 3-5 and 3-6 would hold for both PRl and RRS ARBITERS, unless we consider an RRS ARBITER where the last active request was level BR2*. In this case, the ARBITER would process the BR1* request first and then proceed to the BR2* request.

OBSERVATION 3.14:
BBSY* and the bus grants are fully interlocked as shown in Figure 3-6:

1) The ARBITER does not drive the bus grant high until it detects BBSY* low.
2) The REQUESTER does not release BBSY* to high until it detects the bus grant high.
3) The ARBITER does not drive the next bus grant low until it detects BBSY* high.
4) The next REQUESTER does not drive BBSY* low until it detects the bus grant low.

Note: DEVICE WANTS BUS and DEVICE GRANTED BUS are on-board signals between the MASTER and its REQUESTER. (See Figure 3-4)



(Continued on sheet 2)

Figure 3-5. Arbitration Flow Diagram
Two REQUESTERS, Two Request Levels (Sheet 1 Of 2)

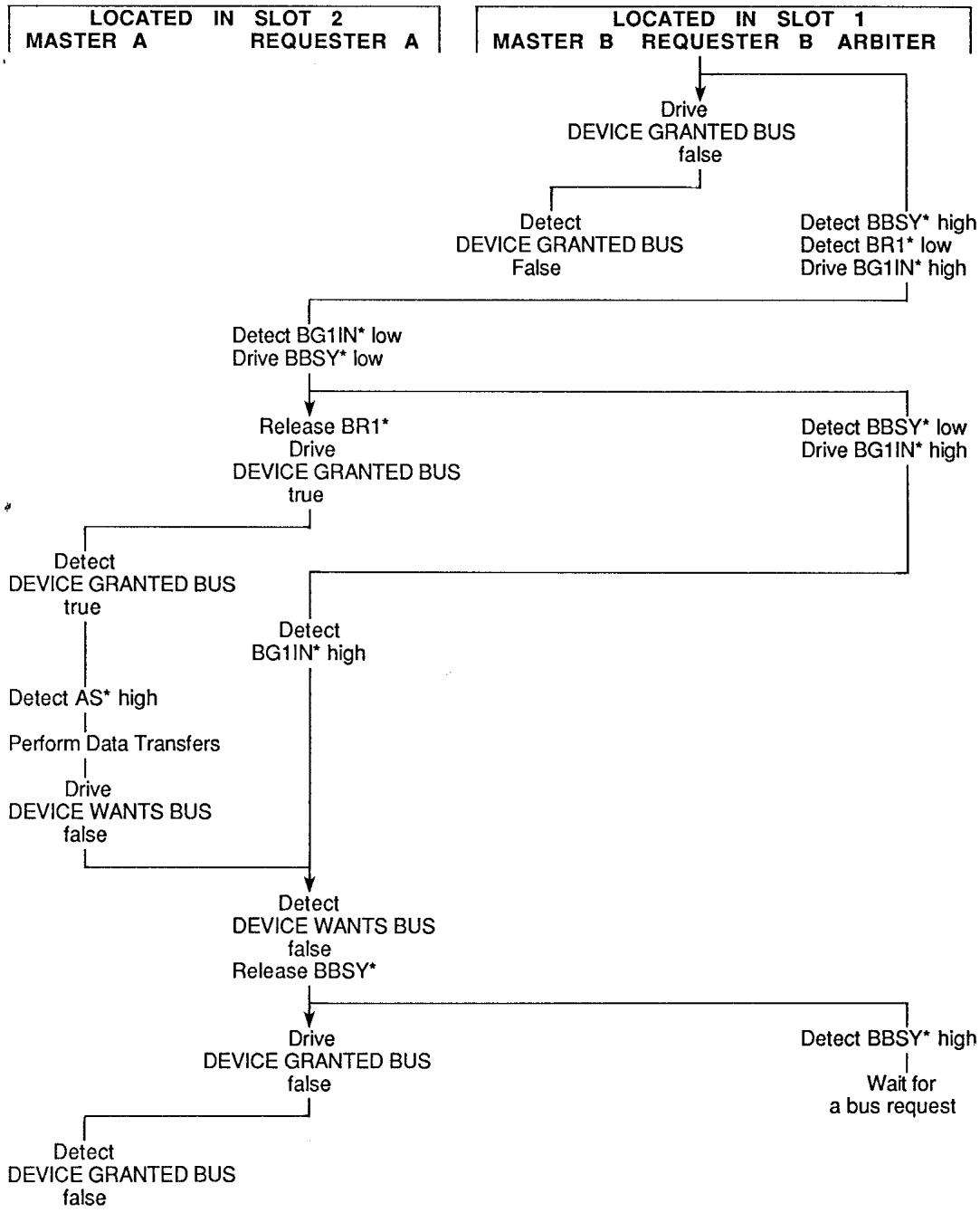


Figure 3-5b. Arbitration Flow Diagram
Two REQUESTERS, Two Request Levels (Sheet 2 Of 2)

Note: In this example each REQUESTER maintains its bus request line low until it is granted the DTB. In some cases a REQUESTER might release its bus request line without receiving a bus grant (see Section 3.3.2).

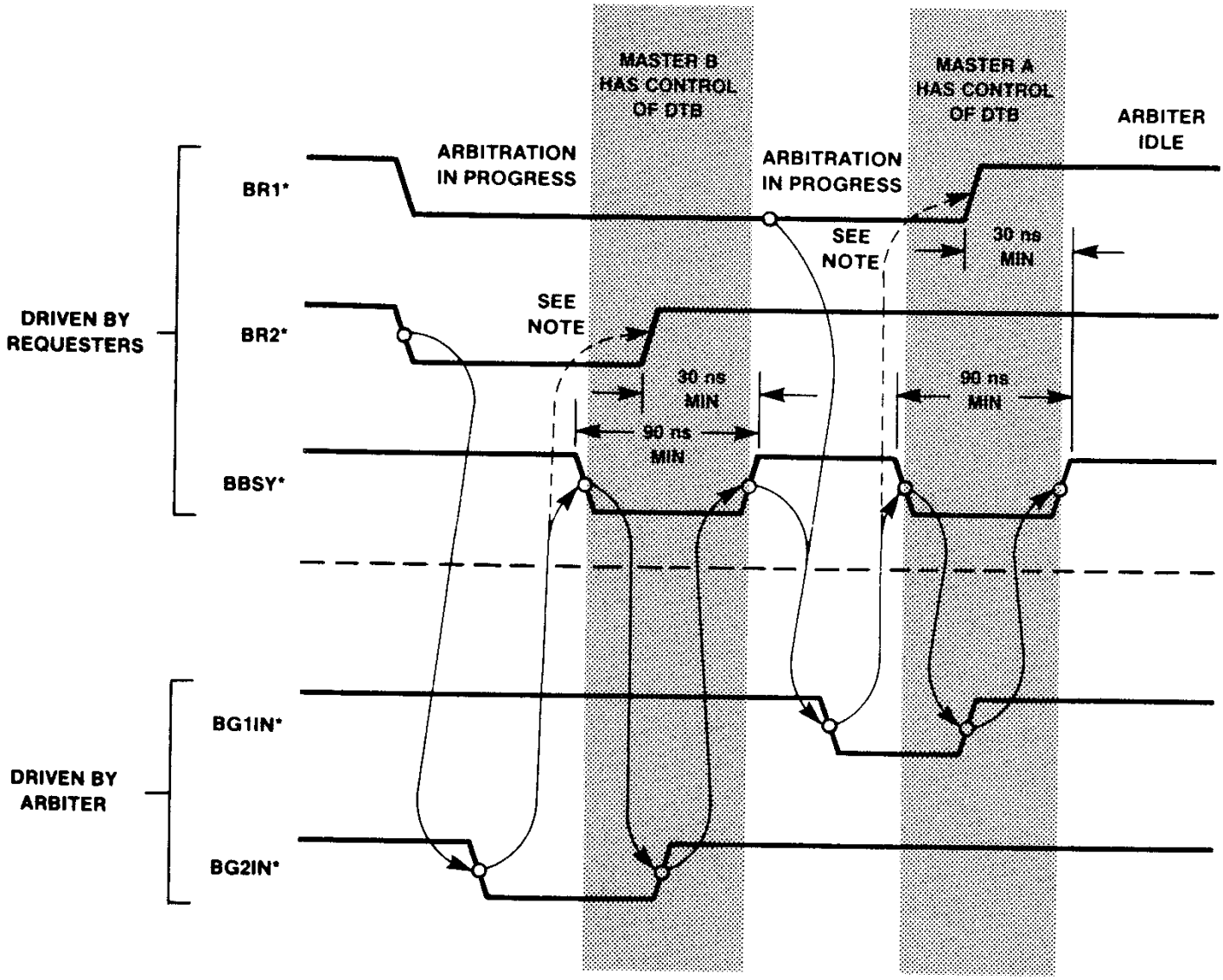


Figure 3-6. Arbitration Sequence Diagram
Two REQUESTERS, Two Request Levels

3.4.2 Arbitration Of Two Bus Requests On The Same Bus Request Line

Figures 3-7 and 3-8 illustrate the sequence of events which takes place when an ROR REQUESTER and an RWD REQUESTER send simultaneous requests to a PRI ARBITER on a common bus request line. In this example, the ARBITER and RWD REQUESTER are located on the system controller board in slot 1, with the ROR REQUESTER located in slot 2. When the sequence begins, both of the REQUESTERS drive BR1* low simultaneously. The ARBITER then drives BG1IN* low to its own slot (slot 1). That BG1IN* signal is monitored by REQUESTER A (also in slot 1). When REQUESTER A detects BG1IN* low, it responds by driving BBSY* low. REQUESTER A then releases BR1* and informs MASTER A that the DTB is available.

OBSERVATION 3.15:

Even though REQUESTER A releases BR1*, REQUESTER B continues to drive it low (see Figures 3-7 and 3-8).

After detecting BBSY* low, the ARBlTER drives BG1IN* high. When MASTER A has completed its data transfer(s), it drives DEVICE WANTS BUS false. When REQUESTER A detects this, and when the 30 nanoseconds delay since the release of BR1* has been satisfied, REQUESTER A releases BBSY*.

The ARBlTER interprets the release of BBSY* as a signal to arbitrate any current bus requests. Since the BR1* line is still low, the ARBITER drives BG1IN* low again. When REQUESTER A detects BG1IN* low, it drives its BG1OUT* low because it does not need the DTB. REQUESTER B then detects the low on its BG1IN* and responds by driving BBSY* low. When the ARBlTER detects the low on BBSY*, it drives BG1IN* high, which causes REQUESTER A to drive its BG1OUT* high.

Some time later, when MASTER B has finished its data transfers, it drives DEVICE WANTS BUS false, indicating that it has finished using the DTB.

Since REQUESTER B is an ROR REQUESTER, it does not release BBSY*, but keeps it driven low. In the event that MASTER B needs to use the DTB again, no arbitration will be required. In this example, however, REQUESTER A drives BR1* low, indicating a need to use the DTB, and REQUESTER B (which is monitoring the bus request lines) releases the BBSY* line. The ARBlTER then grants the DTB to REQUESTER A.

Note: DEVICE WANTS BUS and DEVICE GRANTED BUS are on-board signals between the MASTER and its REQUESTER. (See Figure 3-4)

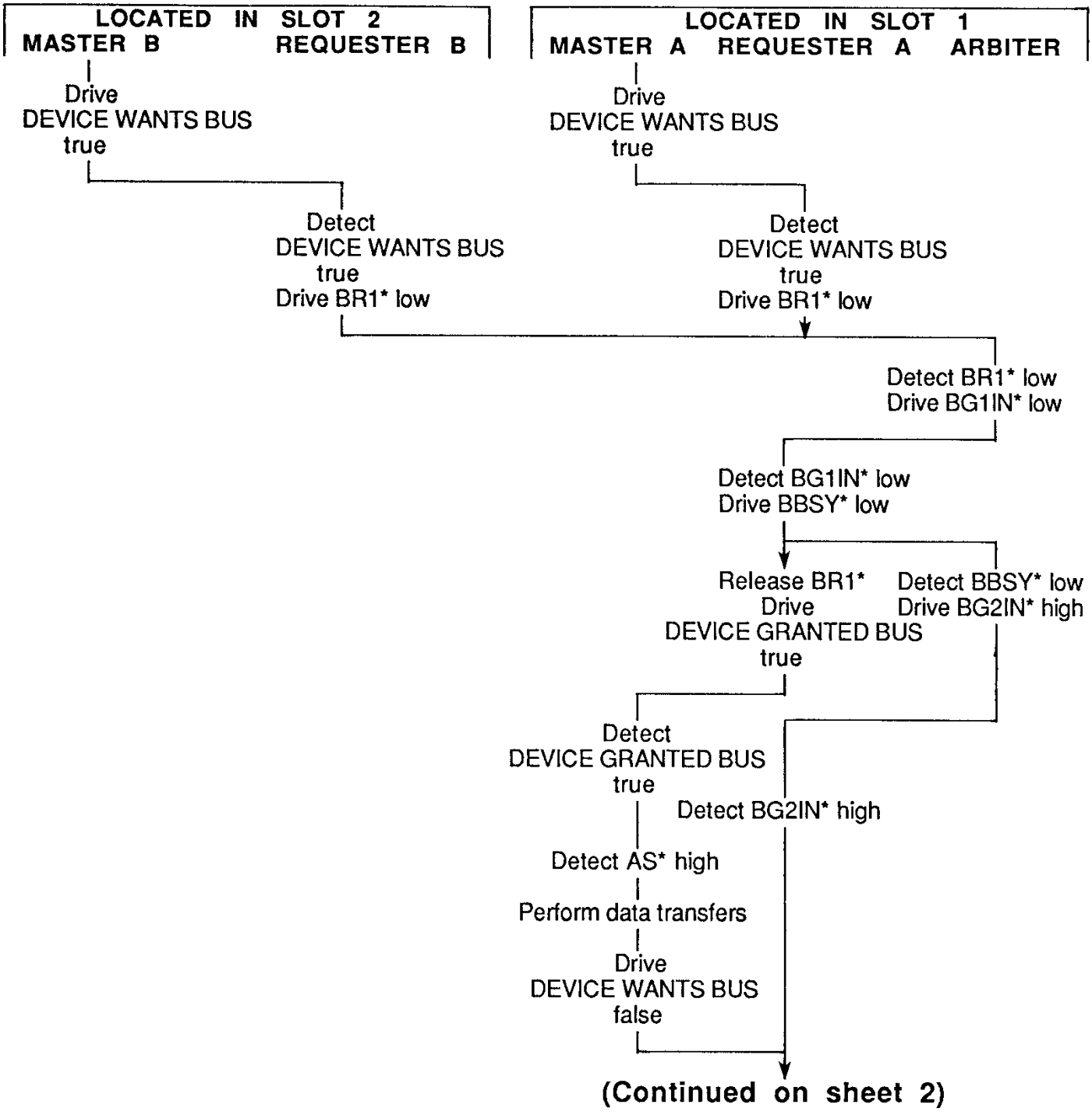


Figure 3-7. Arbitration Flow Diagram
Two REQUESTERS, Same Request Level (Sheet 1 Of 3)

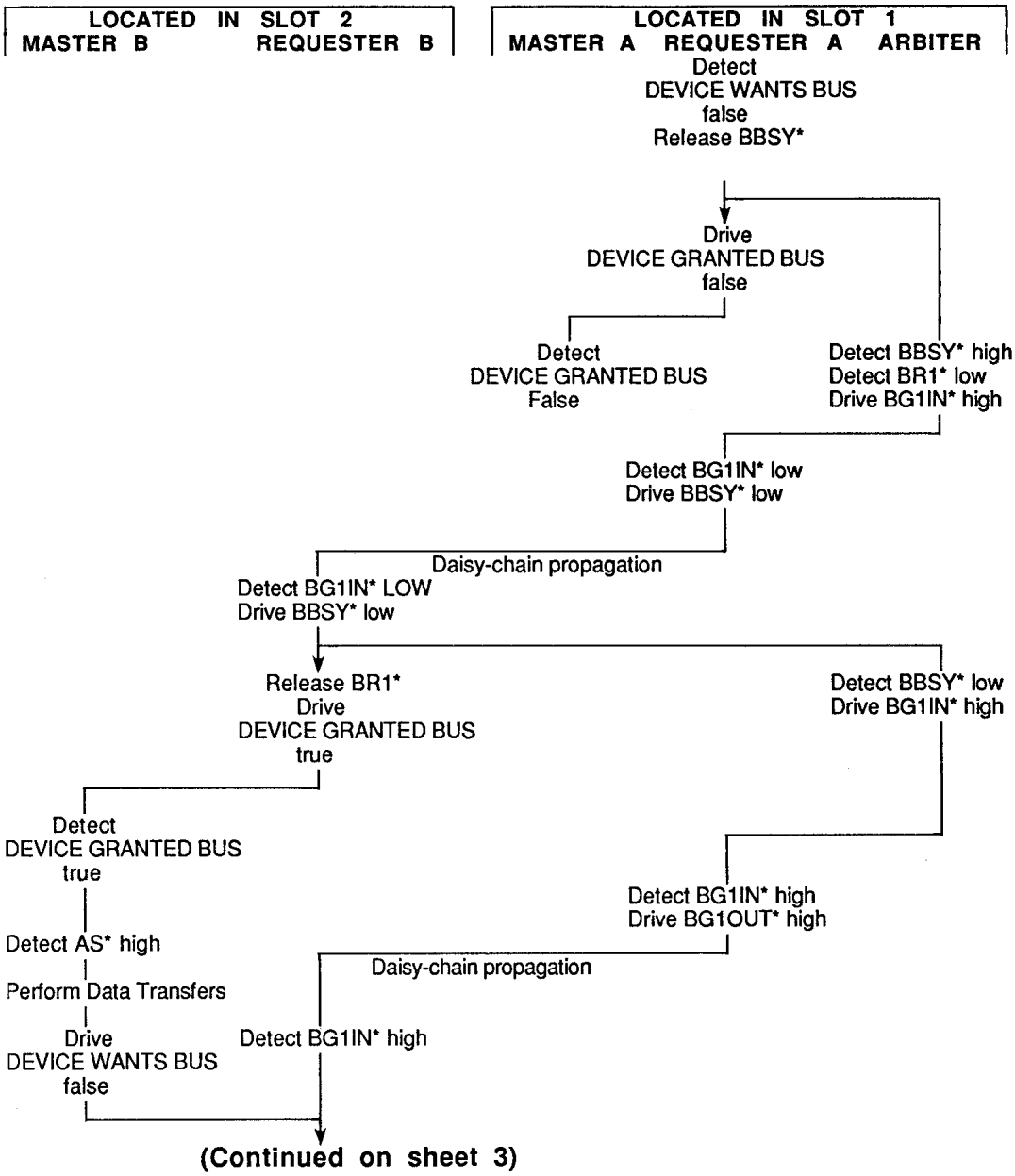


Figure 3-7b. Arbitration Flow Diagram

Two REQUESTERS, Same Request Level (Sheet 2 Of 3)

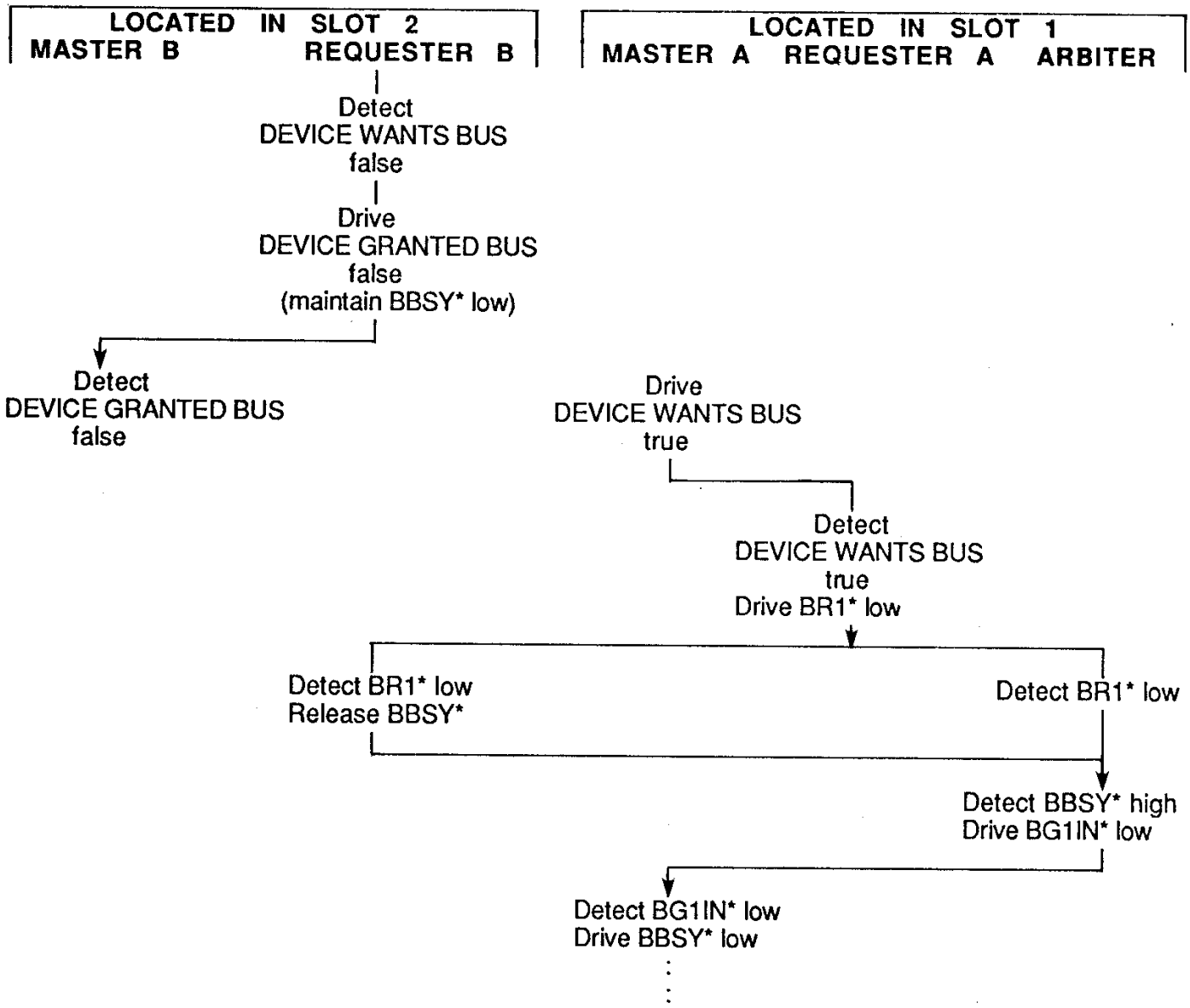


Figure 3-7c. Arbitration Flow Diagram
Two REQUESTERS, Same Request Level (Sheet 3 of 3)

3.5 RACE CONDITIONS BETWEEN MASTER REQUESTS AND ARBITER GRANTS

Suppose that there are two REQUESTERS: REQUESTER A and REQUESTER B, that share a common bus request line. REQUESTER B, which is farther down the daisy-chain, requests the bus and the ARBITER drives the corresponding bus grant line low. This bus grant arrives at REQUESTER A just as MASTER A signals that it wants the bus. If REQUESTER A is improperly designed, this situation might cause it to momentarily drive its BGxOUT* line low and then high again resulting in a low-going transient.

RULE 3.13:
REQUESTERS **MUST** be designed to ensure that no momentary low-going transients are generated on their BGxOUT* out line.

OBSERVATION 3.16:
If the REQUESTER is designed such that it latches the state of the on-board DEVICE WANTS BUS line upon the falling edge of its bus grant in line, and if that signal is in transition when the falling edge occurs, the outputs of the latch will sometimes oscillate, or remain in the threshold region between the high and low levels, for a short time. Because of this, the VMEbus specification does not set a time limit for the REQUESTER to pass along the bus grant. It only prohibits the REQUESTER from generating a low-going transient on its BGxOUT* line which might be interpreted as a bus grant by a REQUESTER further down the daisy-chain.

PERMISSION 3.8:
IF a REQUESTER detects that its on-board MASTER needs the bus between the time that it receives a bus grant intended for another REQUESTER and the time it would pass that bus grant on,
THEN it **MAY** treat the bus grant as its own. In this case the other REQUESTER will maintain its bus request until another bus grant is issued.

Note: In this example each REQUESTER maintains its bus request line low until it is granted the DTB. In some cases a REQUESTER might release its bus request line without receiving a bus grant (see Section 3.3.2).

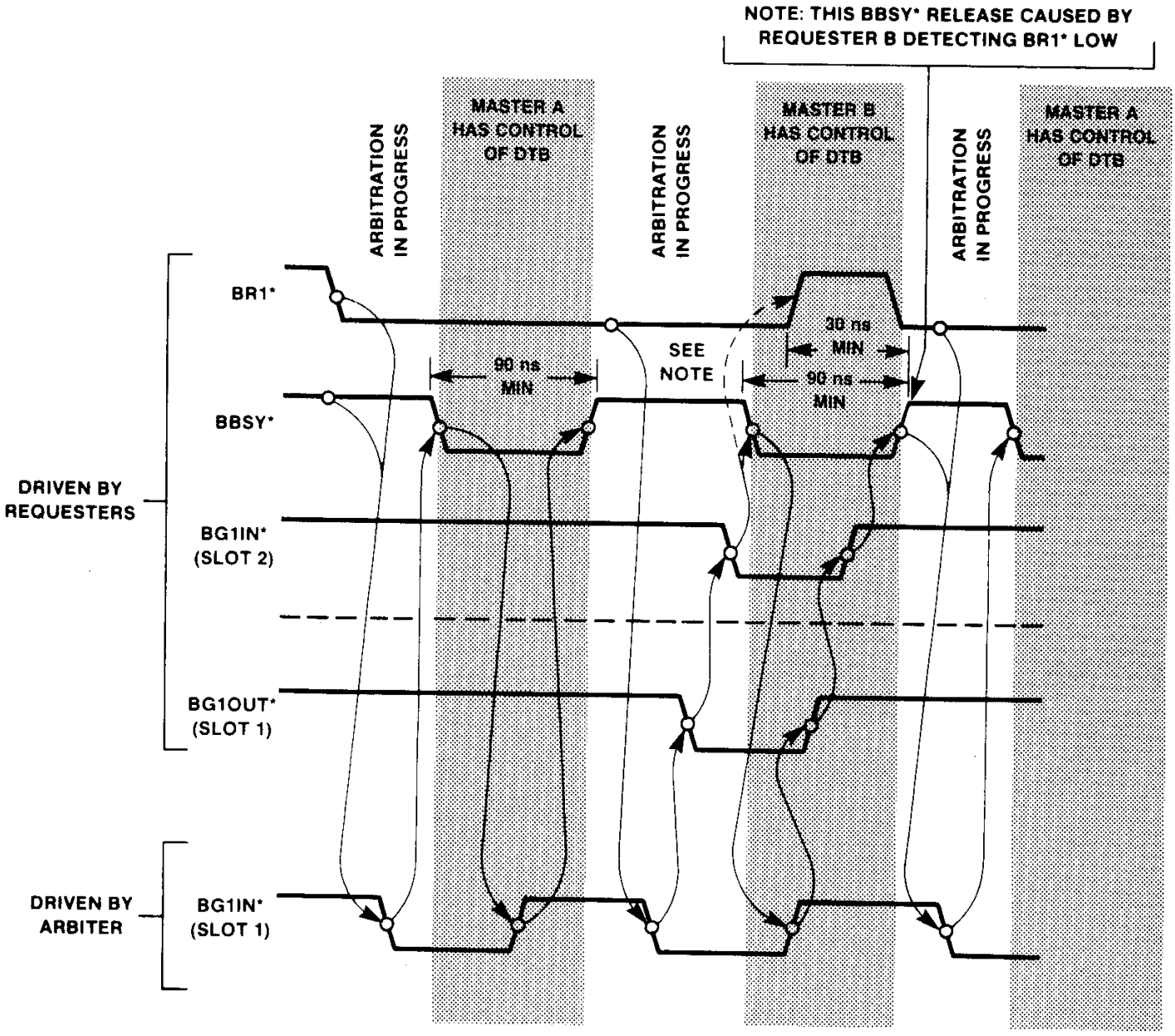


Figure 3-8. Arbitration Sequence Diagram
Two REQUESTERS, Same Request Level

CHAPTER 4

PRIORITY INTERRUPT BUS

4.1 INTRODUCTION

The VMEbus includes a Priority interrupt Bus which provides the signal lines needed to generate and service interrupts. Figure 4-1 shows a typical VMEbus system. INTERRUPTERS use the Priority Interrupt Bus to send interrupt requests to INTERRUPT HANDLERS which respond to these requests.

Any system which has interrupt capability includes software routines that are called interrupt service routines, and are invoked by the interrupts. interrupt subsystems can be classified into two groups:

a. Single handler systems, which have only one INTERRUPT HANDLER that receives and services all bus interrupts.
b. Distributed systems, which have two or more INTERRUPT HANDLERS that receive and service bus interrupts.

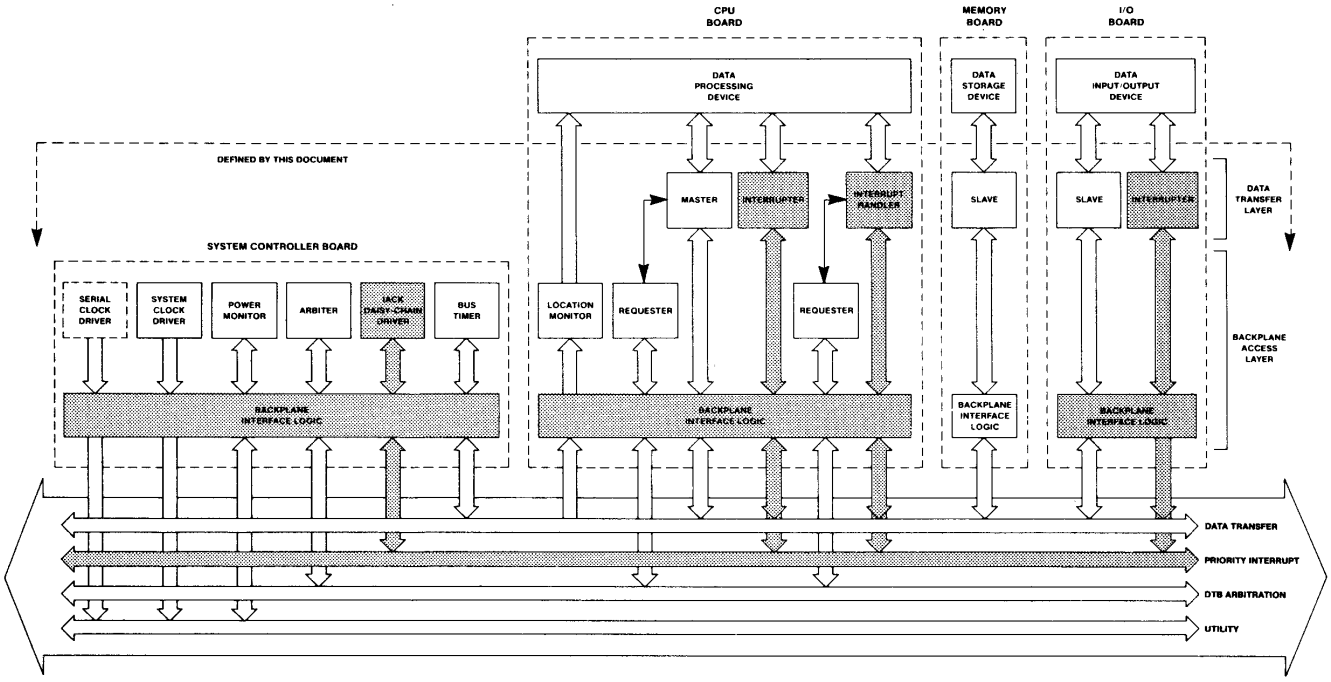


Figure 4-1. Priority Interrupt Bus Functional Block Diagram

4.1.1 Single Handler Systems

In a single handler system, all interrupts are received by one INTERRUPT HANDLER, and all interrupt service routines are executed by one processor. Figure 4-2 shows the interrupt structure of a single handler system. This type of architecture is well suited to machine or process control applications, where a supervisory processor co-ordinates the activities of

dedicated processors. The dedicated processors are typically interfaced to the machine or the process being controlled.

The supervisory processor is the destination for all bus interrupts, servicing them in a prioritized manner. The dedicated processors are not required to service interrupts from the bus, and can give primary attention to controlling a machine or process.

4.1.2 Distributed Systems

Figure 4-3 shows the interrupt structure of a distributed system. This system includes two or more INTERRUPT HANDLERS, each servicing only a subset of the bus interrupts. In a typical implementation, each of the INTERRUPT HANDLERS resides on a different processor board. This type of architecture is well suited to distributed computing applications, where multiple, co-equal processors execute the application software. As each of the co-equal processors executes part of the system software, it might need to communicate with the other processors. In the distributed system, each processor services only those interrupts directed to it, establishing dedicated communication paths among all processors.

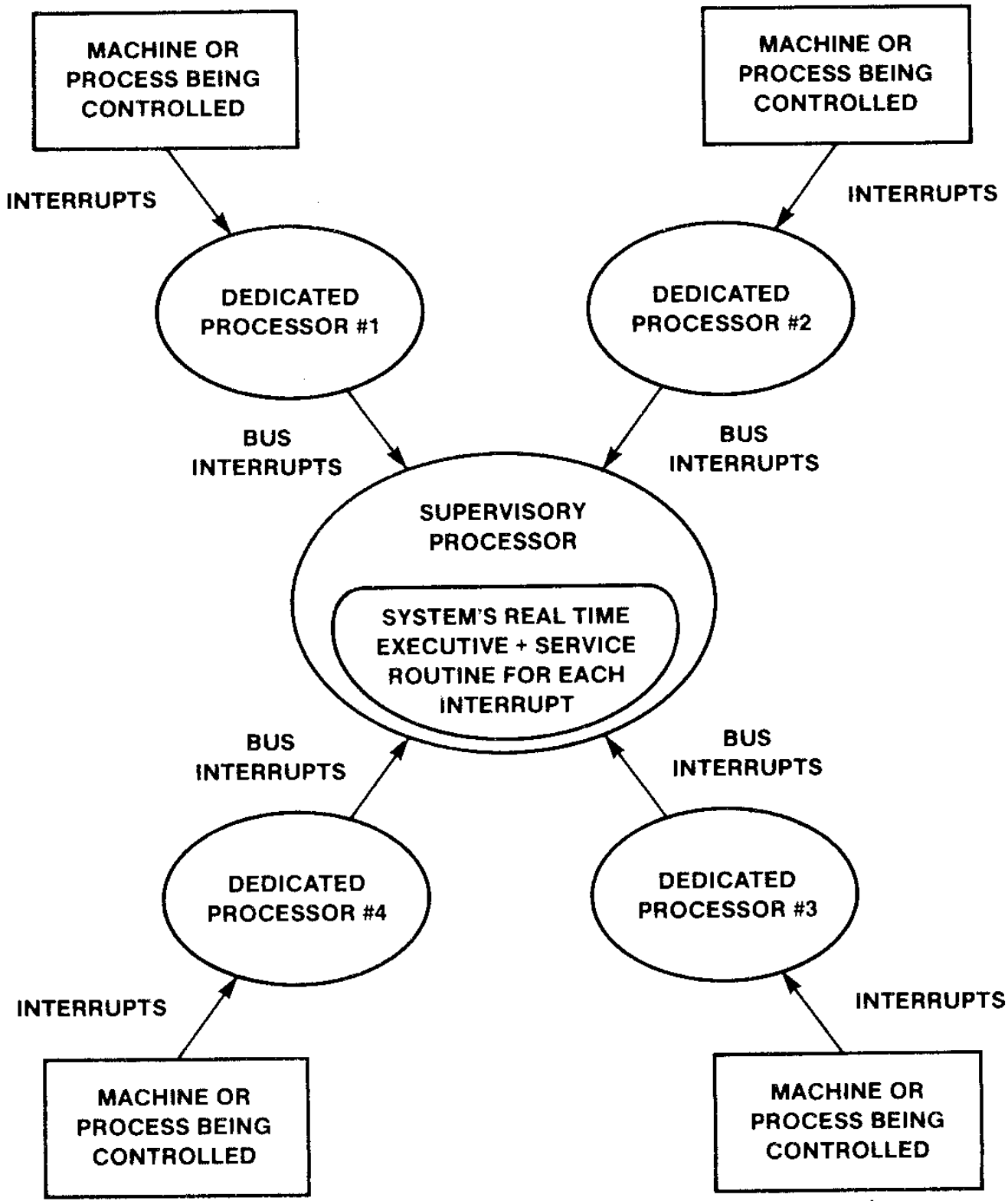


Figure 4-2. Interrupt Subsystem Structure: Single Handler System

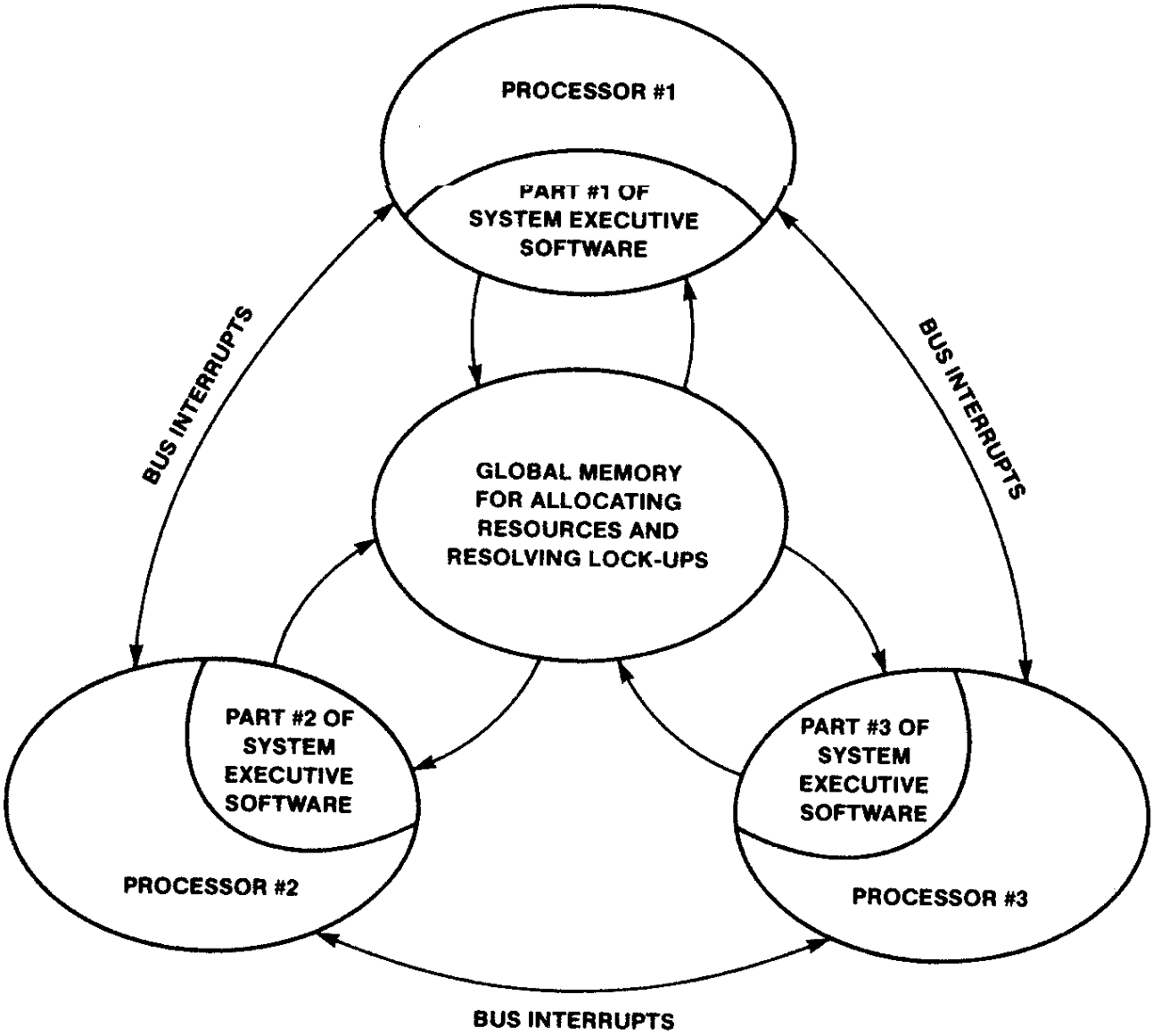


Figure 4-3. Interrupt Subsystem Structure: Distributed System

4.2 PRIORITY INTERRUPT BUS LINES

The Data Transfer Bus, the Arbitration Bus and the Priority Interrupt Bus are all used in the process of generating and handling bus interrupts.

The following discussion of the Priority Interrupt Bus assumes that the reader understands the operation of the Data Transfer Bus described in Chapter 2, and the Arbitration Bus described in Chapter 3.

The Priority Interrupt Bus consists of seven interrupt request signal lines, one interrupt acknowledge line, and one interrupt acknowledge daisy-chain:

IRQ1*	Interrupt Request 1
IRQ2*	Interrupt Request 2

IRQ3*	Interrupt Request 3
IRQ4*	Interrupt Request 4
IRQ5*	Interrupt Request 5
IRQ6*	Interrupt Request 6
IRQ7*	Interrupt Request 7
IACK*	Interrupt Acknowledge
IACKIN*/IACKOUT*	Interrupt Acknowledge Daisy-Chain

4.2.1 Interrupt Request Lines

INTERRUPTERS request interrupts by driving an interrupt request line low. In a single handler system, these interrupt request lines are prioritized, with IRQ7* having the highest priority.

4.2.2 Interrupt Acknowledge Line

The IACK* line runs the full length of the backplane and is connected to the IACKIN* pin of slot 1 (see Figure 4-4). When driven low, the IACKIN* pin causes the IACK DAISY-CHAIN DRIVER, located in slot 1, to propagate a falling edge down the interrupt acknowledge daisy-chain.

4.2.3 Interrupt Acknowledge Daisy-Chain - IACKIN*/IACKOUT.

Each of the seven interrupt request lines can be shared by two or more INTERRUPTER modules. The interrupt acknowledge daisy-chain assures that only one INTERRUPTER responds to the interrupt acknowledge cycle. This daisy-chain line passes through each board on the VMEbus. Each INTERRUPTER that is driving an interrupt request line low waits for a falling edge to arrive at its IACKIN* daisy-chain input. Only upon receiving this falling edge does an INTERRUPTER respond to an interrupt acknowledge cycle. It does not pass the falling edge on down the daisychain, preventing other INTERRUPTERS from responding to the interrupt acknowledge cycle.

RULE 4.1:
IF a VMEbus backplane slot is not occupied by a board, and if there are boards farther down the interrupt acknowledge daisy-chain,
THEN jumpers **MUST** be installed at the empty slot to pass through the daisy-chain signal.

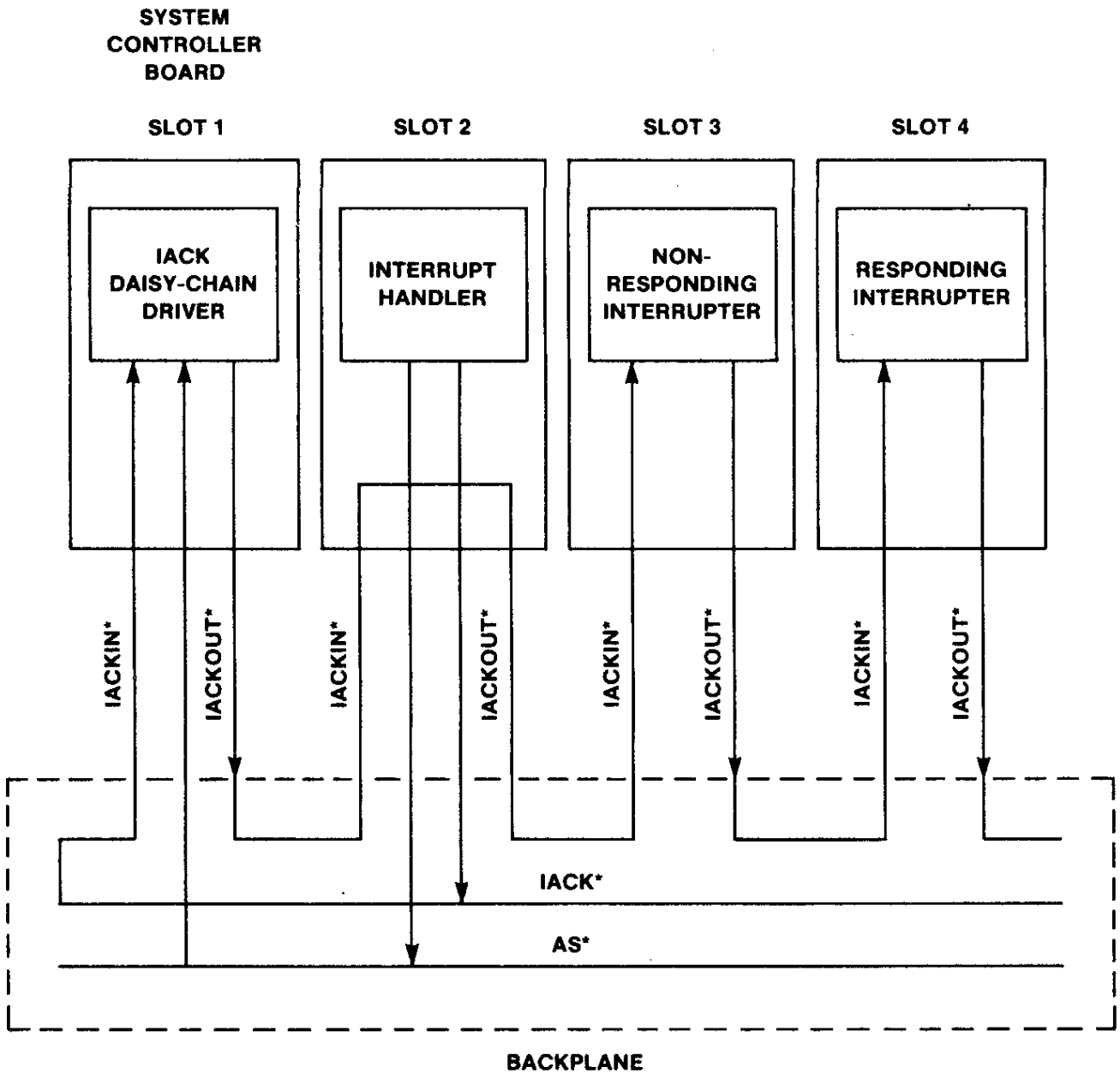


Figure 4-4. IACKIN*/IACKOUT* DAISY-CHAIN

4.3 PRIORITY INTERRUPT BUS MODULES - BASIC DESCRIPTION

There are three types of functional modules associated with the Priority Interrupt Bus. INTERRUPTERS, INTERRUPT HANDLERS, and IACK DAISY-CHAIN DRIVERS. The capabilities of INTERRUPT HANDLERS and INTERRUPTERS are described by a list of mnemonics that show what interrupt acknowledge cycle types they can generate and accept, respectively.

Sections 4.3.1 through 4.3.3 provide block diagrams for the three types of Priority Interrupt Bus modules: INTERRUPT HANDLER, INTERRUPTER, and IACK DAISY-CHAIN DRIVER.

RULE 4.2:

Output signal lines shown with solid lines in Figures 4-5 through 4-7 **MUST** be driven by the module, unless it would always drive them high.

OBSERVATION 4.1:
IF an output line is not driven,
THEN terminators on the backplane ensure that it is high.

RULE 4.3:
Input signal lines shown with solid lines in Figures 4-5 through 4-7 **MUST** be monitored and responded to in the appropriate fashion.

OBSERVATION 4.2:
RULES and PERMISSIONS for driving and monitoring signal lines shown with dotted lines in Figures 4-5 and 4-6, are given in Tables 4-1 and 4-2.

4.3.1 INTERRUPT HANDLER

The INTERRUPT HANDLER is used to accomplish several tasks:
a. It prioritizes the incoming interrupt requests within its assigned group of interrupt request lines (highest of IRQ1*-IRQ7*).
b. It uses its on-board REQUESTER to request the DTB and, when granted use of the DTB, initiates an interrupt acknowledge cycle, reading a STATUS/ID from the INTERRUPTER being acknowledged.
c. It initiates the appropriate interrupt servicing sequence, based on the information received in the STATUS/ID.

OBSERVATION 4.3:
The VMEbus specification does not dictate what will happen during the interrupt servicing sequence. Servicing of the interrupt might or might not involve use of the VMEbus.

The INTERRUPT HANDLER uses the DTB to read a STATUS/ID from the INTERRUPTER. In this respect, the INTERRUPT HANDLER acts like a MASTER and the INTERRUPTER acts like a SLAVE. However, there are four important differences. The INTERRUPT HANDLER:

a. always drives IACK* low.
b. is not required to drive the address modifier lines.
c. only uses the lowest three address lines (A01-A03).
d. never drives the data bus.

The INTERRUPT HANDLER always drives IACK* low when it accesses the bus. The MASTER either drives it high or does not drive it at all.

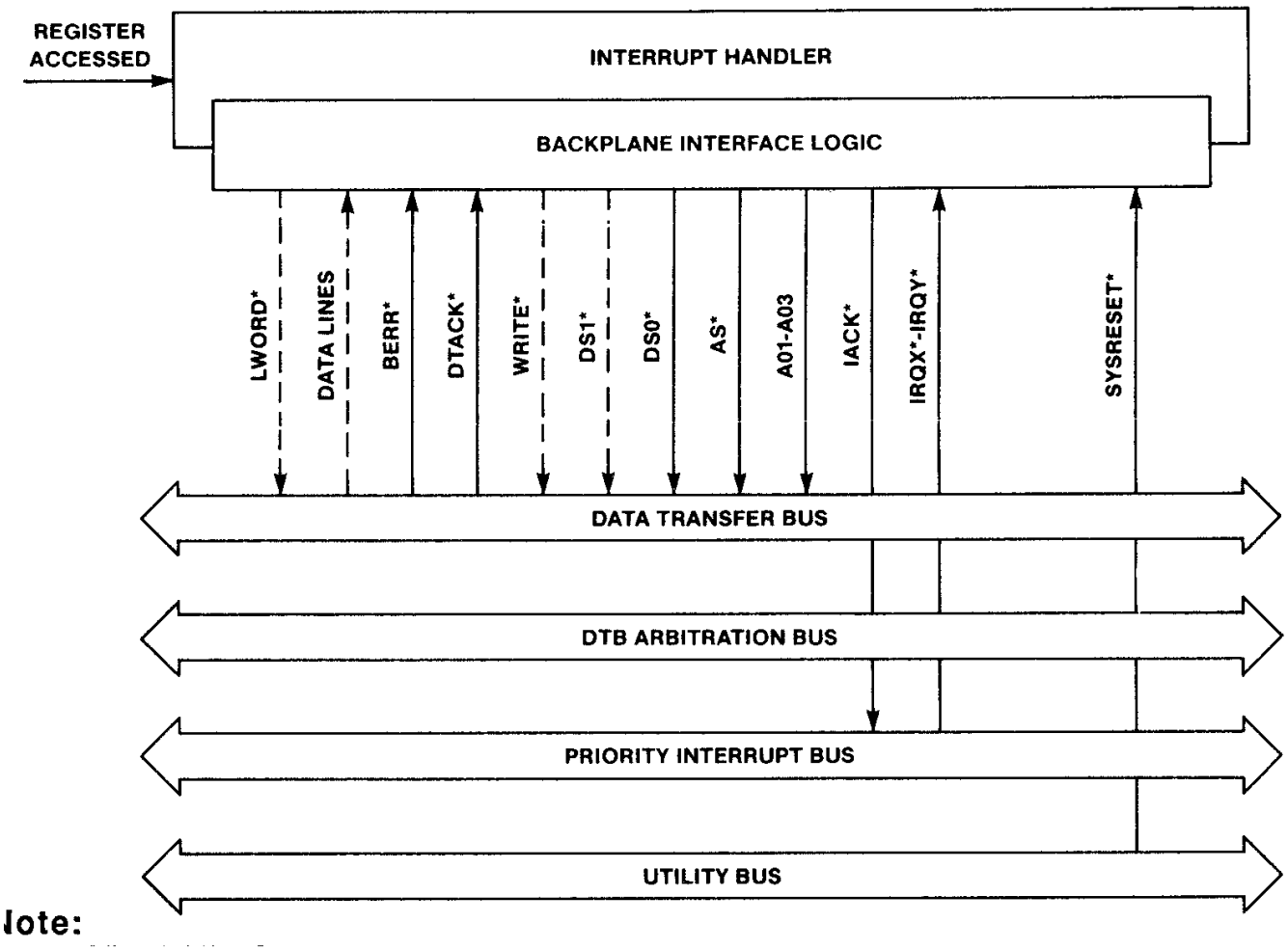
The INTERRUPT HANDLER does not have to drive the address modifier lines with a valid code, and it only drives the lowest three address lines (A01-A03) with valid information. The levels of these three address lines indicate which of the seven interrupt request lines is being acknowledged, as shown in Table 4-7. A MASTER drives 15, 23, or 31 address lines

(depending on the addressing mode) with the address of the SLAVE being accessed, and provides an address modifier code on the address modifier lines.

The INTERRUPT HANDLER does not drive the data lines (i.e., it does not ,,write,, to the INTERRUPTER) and does not have to drive the WRITE* line. A MASTER uses the data lines to a SLAVE bidirectionally and, during normal use, drives WRITE* low or high as required.

A block diagram of the INTERRUPT HANDLER is shown in Figure 4-5.

Note: The RULES and PERMISSIONS for monitoring and driving the dotted lines are given in Table 4-1.



Note:

Figure 4-5. Block Diagram: INTERRUPT HANDLER

Table 4-1. RULES And PERMISSIONS That Specify The Use Of The Dotted Lines In Figure 4-5 By The Various Types Of INTERRUPT HANDLERS

Type of INTERRUPT HANDLER	Use of dotted lines
D08(0)	**MUST** monitor D00-D07. **MAY** or **MAY** not drive LWORD* and DS1*.

	MAY or **MAY** not monitor D08-D31.
D16	**MUST** drive DS1*. **MUST** monitor D00-D15. **MAY** or **MAY** not drive LWORD*. **MAY** or **MAY** not monitor D16-D31.
D32	**MUST** drive DS1* and LWORD*. **MUST** monitor D00-D31.
ALL	**MUST** not drive WRITE* low.

Note: The mnemonics D08(0), D16, and D32 are defined in Table 4-5.

4.3.2 INTERRUPTER

The INTERRUPTER functions as follows:

a. It requests an interrupt from the INTERRUPT HANDLER which monitors its interrupt request line.
b. IF it receives a falling edge on the interrupt acknowledge daisy-chain input, THEN IF it is requesting an interrupt and the levels on the three valid address lines correspond to the interrupt request line it is using, and the width of the requested STATUS/ID is either equal to, or greater than the size it can supply,
THEN it supplies a STATUS/ID,
ELSE it passes the falling edge down the interrupt acknowledge daisychain.

Each INTERRUPTER module drives only one interrupt request line. The VMEbus specification describes a board that generates interrupt requests on several interrupt lines as having several INTERRUPTER modules.

PERMISSION 4.1:
Since the INTERRUPTER is just a conceptual model, logic on a VMEbus board **MAY** be shared between several INTERRUPTER modules.

The INTERRUPTER uses one of seven lines to request an interrupt. It then monitors the lowest three lines of the address bus (A01-A03), the IACKIN*/IACKOUT* daisychain, and optionally IACK*, to determine when its interrupt is being acknowledged. When acknowledged, it places its STATUS/ID on the data bus and signals the INTERRUPT HANDLER that the STATUS/ID is valid by driving DTACK* low.

There are five primary differences in the use of the DTB by the INTERRUPTER and the SLAVE. The INTERRUPTER:

a. only responds when its IACKIN* is low.
b. does not have to monitor the address modifier lines.
c. only monitors the lowest three address lines.
d. does not monitor the WRITE* line.
e. is permitted to respond with data of a different size than that requested.

The SLAVE monitors AS*, and interprets a falling edge on AS* as the signal that a valid bus cycle is in progress. It then proceeds to decode the appropriate number of address lines (15, 23, or 31), and the address modifier lines, and based on this information it determines whether it was addressed. However, the SLAVE responds only if IACK* is high.

The INTERRUPTER, on the other hand, interprets the falling edge on its IACKIN* line as a signal that it can respond to the interrupt acknowledge cycle in progress. It decodes only the lowest three address lines (A01-A03), ignoring the address modifier lines.

The INTERRUPTER does not need to monitor WRITE*, since it is never written to. SLAVES need to monitor WRITE* so that they can distinguish read cycles from write cycles.

The INTERRUPTER places a STATUS/ID on the bus, and responds with DTACK*, even if the LWORD*, DS1*, and DS0* lines call for a STATUS/ID whose width is greater than the INTERRUPTER is able to provide. For example, the INTERRUPT HANDLER might drive LWORD* and both data strobes low, indicating that it will read 32-bits of STATUS/ID from D00-D31, but a D08(0) INTERRUPTER would still respond with its 8-bit STATUS/ID on D00-D07. In contrast, when a SLAVE cannot provide the requested data width, it either responds with BERR* or does not respond at all, typically resulting in a bus time-out.

OBSERVATION 4.4:
When an INTERRUPTER places a STATUS/ID on the data bus, any undriven data lines are read by the INTERRUPT HANDLER as high because of the bus terminators.

For example, if a D16 INTERRUPT HANDLER initiates a double byte interrupt acknowledge cycle, a D08(0) INTERRUPTER would place an 8-bit STATUS/ID on D00-D07. The upper 8 bits, read by the INTERRUPT HANDLER from D08-D15, are read as ones (high), since they are not driven by the D08(0) INTERRUPTER.

RULE 4.4:
Before responding to an interrupt acknowledge cycle, the INTERRUPTER:

1. MUST have an interrupt request pending.
2. The level of that request **MUST** match the level indicated on A01-A03.
3. The width of the requested STATUS/ID **MUST** be equal to or greater than the size it can respond with.
4. It **MUST** have received an incoming falling edge on its IACKIN* daisy-chain input.
IF any of these four conditions are not met,
THEN the INTERRUPTER **MUST NOT** respond to the interrupt acknowledge cycle.
IF condition 4 is met, but either 1, 2, or 3 is not
THEN the INTERRUPTER **MUST** pass the falling edge of IACKIN* to the next INTERRUPTER module in the daisy-chain.

A block diagram of the INTERRUPTER is shown in Figure 4-6.

Notes:

1. The RULES and PERMISSIONS for driving and monitoring the dotted lines are given in Table 4-2.
2. This input signal is present on RORA INTERRUPTERS only.

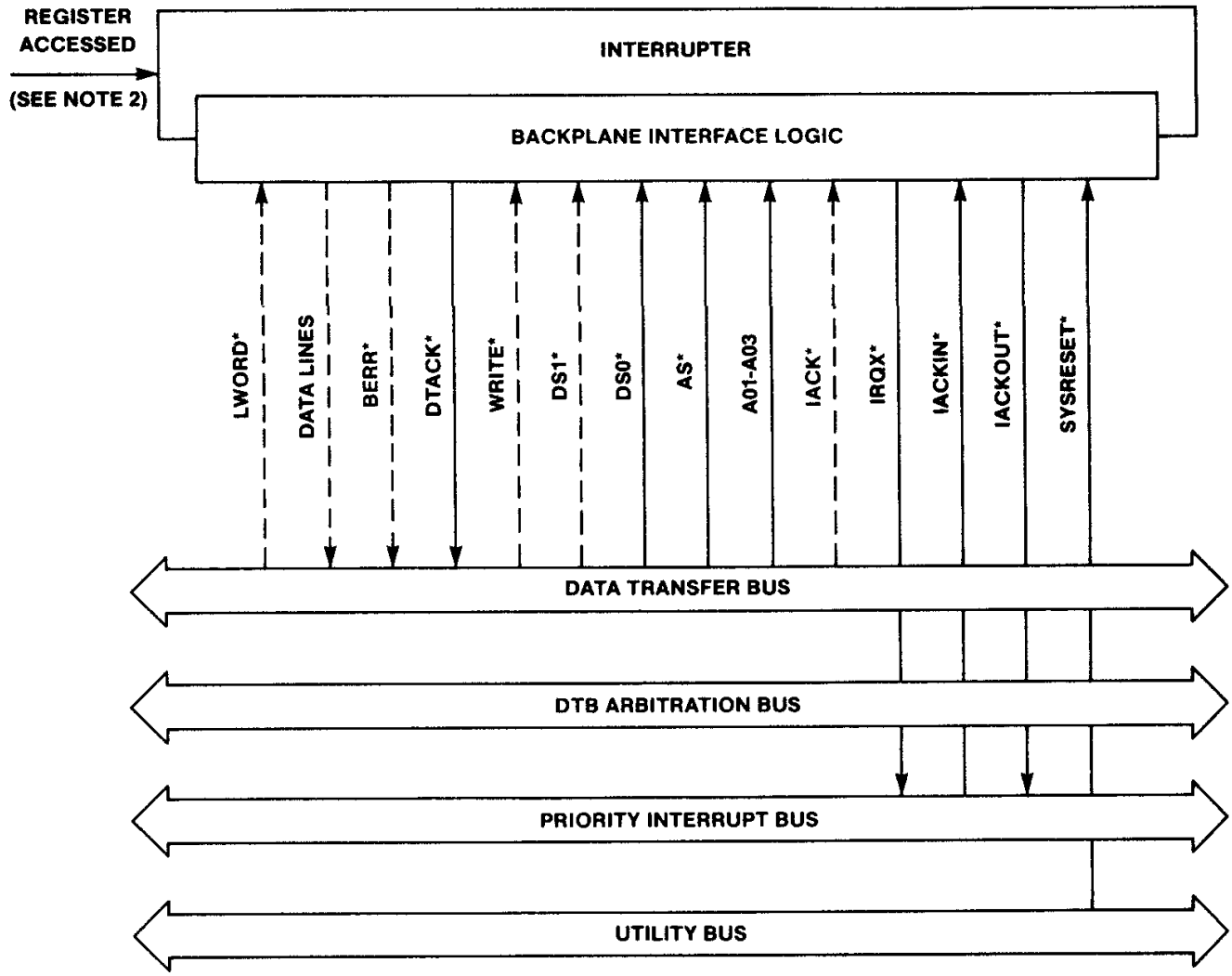


Figure 4-6. Block Diagram. INTERRUPTER

Table 4-2. RULES And PERMISSIONS That Specify The Use Of The Dotted Lines In Figure 4-6 By The Various Types Of INTERRUPTERS

Type of INTERRUPTER	Use of dotted lines
D08(O)	**MUST** drive D00-D07. **MUST NOT** drive D08-D31 low. has no reason to monitor LWORD* or DS1*.
D16	**MUST** monitor DS1*. **MUST** drive D00-D15. **MUST NOT** drive D1 6-D31 low. has no reason to monitor LWORD*.
D32	**MUST** monitor DS1* and LWORD*.

	MUST drive D00-D31.
ALL	**MAY** or **MAY** not monitor WRITE* and IACK*. **MAY** or **MAY** not drive BERR*.

Note:
The mnemonics D08(0), D16, and D32 are defined in Table 4-5.

4.3.3 IACK DAISY-CHAIN DRIVER

The IACK DAISY-CHAIN DRIVER is another module that interacts with INTERRUPT HANDLERS and INTERRUPTERS to coordinate the servicing of interrupts. It generates a falling edge on the interrupt acknowledge daisy-chain each time an INTERRUPT HANDLER initiates an interrupt acknowledge cycle.

A block diagram of the IACK DAISY-CHAIN DRIVER is given in Figure 4-7.

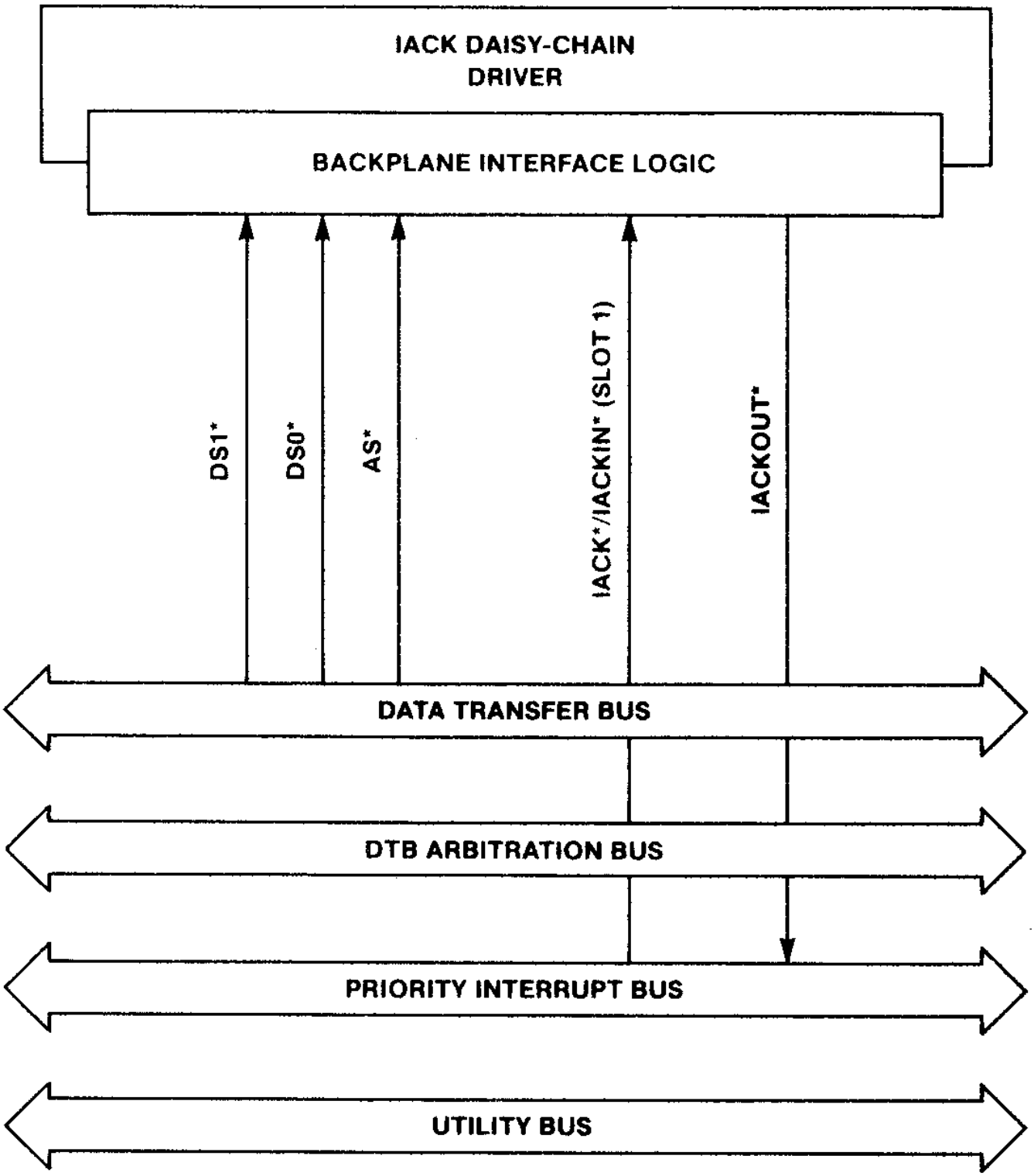


Figure 4-7. Block Diagram: IACK DAISY-CHAIN DRIVER

4.3.4 Interrupt Request Handling Capabilities

INTERRUPT HANDLERS can be designed to handle interrupt requests received on one to seven interrupt request lines. Table 4-3 shows how the IH( ) mnemonic is used to describe the interrupt handling capabilities of INTERRUPT HANDLERS.

Table 4-3. Use Of The IH( ) Mnemonic To Specify Interrupt Request Handling Capabilities

The Following Mnemonic	When Applied to an	Means that it
IH(x-y)	INTERRUPT HANDLER	can generate interrupt acknowledge cycles in response to interrupt requests on lines IRQx* through IRQy*.
IH(x)	INTERRUPT HANDLER	can generate interrupt acknowledge cycles in response to interrupt requests on line IRQx*.

4.3.5 Interrupt Request Generation Capabilities

INTERRUPTERS can be designed to generate an interrupt request on any of the seven interrupt request lines. Table 4-4 shows how the 1( ) mnemonic is used to describe the interrupt request generation capabilities of INTERRUPTERS.

Table 4-4. Use Of The 1( ) Mnemonic To Specify Interrupt Request

Generation Capabilities

The Following Mnemonic	When Applied to an	Means that it
I(x)	INTERRUPTER	can generate an interrupt request on line IRQx*.

4.3.6 STATUS/ID Transfer Capabilities

There are three STATUS/ID transfer capabilities D08(0), D16 and D32. Table 4-5 shows how these mnemonics are used to describe the interrupt handling capabilities of INTERRUPT HANDLERS and INTERRUPTERS.

Table 4-5. Mnemonics That Specify STATUS/ID Transfer Capabilities

The Following Mnemonic	When Applied to an	Means that it
D08(0)	INTERRUPTER	responds to 8-bit, 1 6-bit, and 32-bit interrupt acknowledge cycles by providing an 8-bit STATUS/ID on D00-D07.
	INTERRUPT HANDLER	generates 8-bit interrupt acknowledge cycles in response to the requests on the interrupt request line(s) and reads an8-bit STATUS/ID from D00-D07.
D16	INTERRUPTER	responds to 1 6-bit and 32-bit interrupt acknowledge cycles by providing a 1 6-bit STATUS/ID on D00-D15
	INTERRUPT	generates 16-bit interrupt acknowledge cycles in

	HANDLER	response to the requests on the interrupt request line(s) and reads a16-bit STATUS/ID from D00-D15.
D32	INTERRUPTER	responds to 32-bit interrupt acknowledge cycles by providing a 32 bit STATUS/ID on D00-D31 .
	INTERRUPT HANDLER	generates 32-bit interrupt acknowledge cycles in response to the requests on the interrupt request line(s) and reads a32-bit STATUS/ID from D00-D31 .

4.3.7 Interrupt Request Release Capabilities

Many widely used peripheral ICs generate interrupt requests. Unfortunately, there is no standard method for indicating to these ICs when it is time for them to remove their interrupt request from the bus. Three methods are used:

a. When the relevant processor senses an interrupt request from a peripheral device, it enters an interrupt service routine, and READS a status register in the device. The peripheral device interprets this read cycle on its status register as a signal to remove its interrupt request.
b. When the relevant processor senses an interrupt request from a peripheral device, it enters an interrupt service routine, and WRITES to a control register in the device. The peripheral device interprets this write cycle to its control register as a signal to remove its interrupt request.
c. When the relevant processor senses an interrupt request from a peripheral device, it reads a STATUS/ID from the device. The peripheral device interprets this read cycle as a signal to remove its interrupt request.

The VMEbus specification calls INTERRUPTERS that use methods a and b Release On Register Access (RORA) INTERRUPTERS, and those that use method c Release On AcKnowledge (ROAK) INTERRUPTERS. Figure 4-8 shows how an ROAK INTERRUPTER releases its interrupt request line when the INTERRUPT HANDLER reads its STATUS/ID and how an RORA INTERRUPTER releases its interrupt request upon an access to a control or status register.

OBSERVATION 4.5:
The SLAVE that provided the access to the lNTERRUPTER'S control or status register is typically on the same board as the INTERRUPTER, and it generates an on-board signal to the INTERRUPTER when it has completed the register access.

RULE 4.5:
An RORA INTERRUPTER **MUST NOT** release its interrupt request line before it detects a falling edge on DSA* during the register access cycle. It **MUST** release the interrupt request line within 2 microseconds after the last data strobe goes high at the end of the register access cycle.

RULE 4.6:

An ROAK INTERRUPTER **MUST NOT** release its interrupt request line before it detects a falling edge on DSA* during the interrupt acknowledge cycle which acknowledges its interrupt, and it **MUST** release its interrupt request line within 500 nanoseconds after the last data strobe goes high at the end of the STATUS/ID read cycle.

RULE 4.7:
Both RORA and ROAK INTERRUPTERS **MUST** provide a STATUS/ID during the interrupt acknowledge cycle that was initiated in response to their interrupt request.

RULE 4.8:
After an INTERRUPT HANDLER initiates an interrupt acknowledge cycle and reads the STATUS/ID from an RORA INTERRUPTER, it **MUST** ignore the low level on the interrupt request line for 2 microseconds after its on-board signal REGISTER ACCESSED goes true.

OBSERVATION 4.6:
RULE 4.8 prevents the INTERRUPT HANDLER from misinterpreting the low level on that line as a new interrupt request.

OBSERVATION 4.7:
The MASTER that accesses the INTERRUPTER'S control or status register is typically on the same board as the INTERRUPT HANDLER, and it generates an on-board signal to the INTERRUPT HANDLER when it has completed the register access.

PERMISSION 4.2:
IF a procedure is established to allow the MASTER to signal the INTERRUPT HANDLER that an access to the INTERRUPTER'S control or status registers has taken place
THEN the MASTER and INTERRUPT HANDLER **MAY** reside on different boards.

Table 4-6 shows how the RORA and ROAK mnemonics are used to describe INTERRUPTERS.

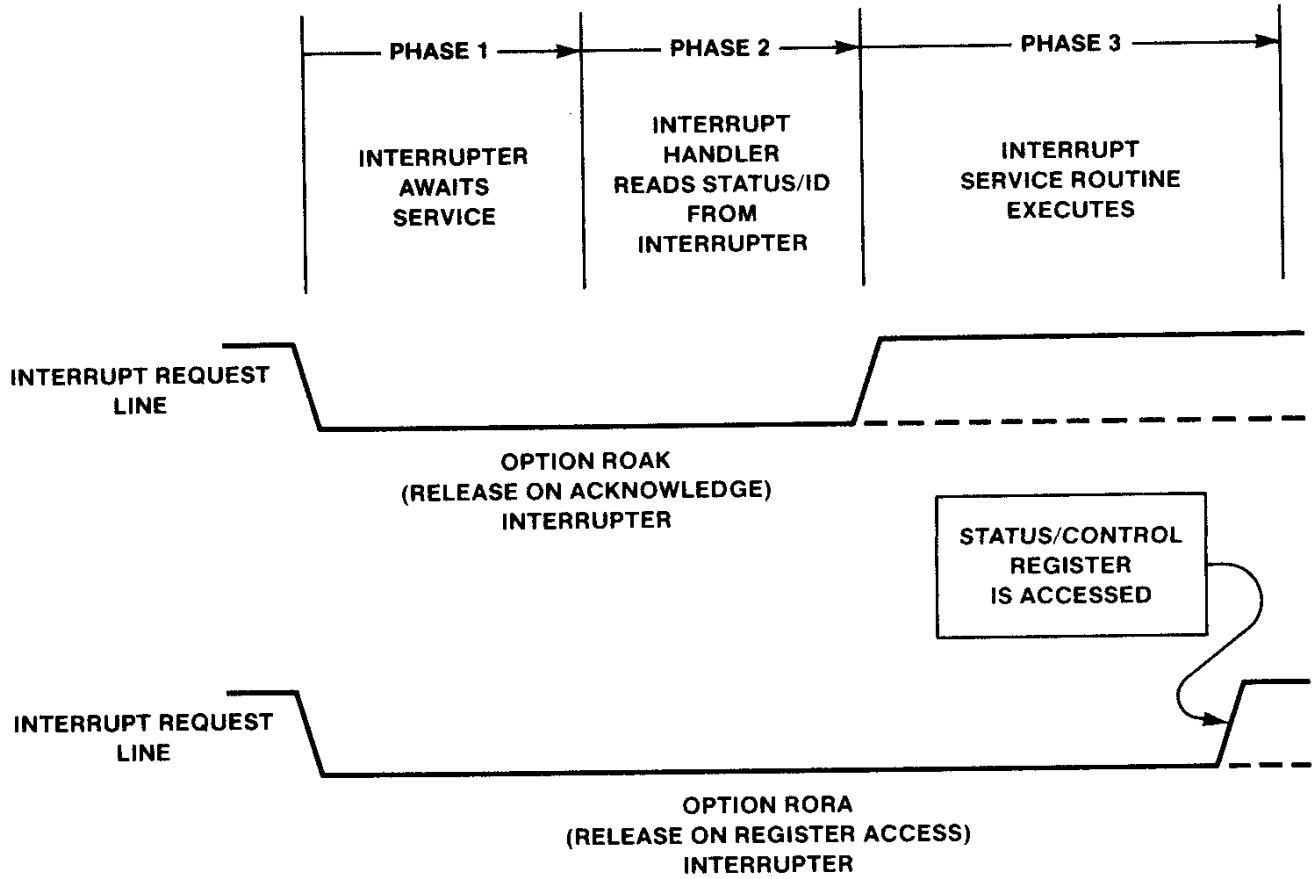


Figure 4-8. Release Of Interrupt Request Lines By ROAK And RORA INTERRUPTERS

Table 4-6. Mnemonics That Specify Interrupt Request Release Capabilities

The Following Mnemonic	When Applied to a	Means that it
RORA	INTERRUPTER	releases its interrupt request line when some MASTER accesses an on-board status or control register.
ROAK	INTERRUPTER	releases its interrupt request line when its STATUS/ID is read during an interrupt acknowledge cycle.

4.3.8 Interaction Between Priority Interrupt Bus Modules

In the following discussions, several on-board signals are defined to describe the interaction between the INTERRUPTER and INTERRUPT HANDLER modules and other on-board logic. These signals are only intended to illustrate the information which is passed to and from the modules, rather than to define their designs.

PERMISSION 4. 3:
VMEbus boards **MAY** be designed with on-board signals that differ from those used in the following discussions.

Figure 4-4 shows how the IACKIN*/IACKOUT* daisy-chain is routed through a typical configuration of boards on the VMEbus.

The IACK* line runs the full length of the backplane and can be driven by any INTERRUPT HANDLER that has control of the DTB. The backplane connects IACK* to the IACKIN* pin of slot 1. The IACK DAISY-CHAIN DRIVER resides in slot 1 and monitors the level of slot 1's IACKIN* line.

When an INTERRUPT HANDLER drives IACK* (and slot 1's IACKIN*) low, and then drives DSA* low, the IACK DAISY-CHAIN DRIVER generates a falling edge on its IACKOUT* pin. This pin is connected to the IACKIN* pin of slot 2. A jumper on the board in slot 2 routes the falling edge on the IACKIN* pin to the IACKOUT* pin, and through the backplane to the IACKIN* pin of the board in slot 3. The INTERRUPTER in slot 3 does not have a pending interrupt request, so it passes on the falling edge to its IACKOUT* pin. The INTERRUPTER in slot 4 then detects the falling edge on its IACKIN* line and responds by placing its STATUS/ID on the data bus, and then driving DTACK* low.

PERMISSION 4.4:
An INTERRUPTER **MAY** reside on the system controller board, installed in slot 1, along with the IACK DAISY-CHAIN DRIVER. Figure 4-9 shows how the two modules would be connected.

PERMISSION 4.5:
More than one INTERRUPTER **MAY** reside on a board. Figure 4-10 shows how this might be done.

OBSERVATION 4.8:
In some cases, board designers might not know whether or not the board they are designing will be installed in slot 1 , or in some other slot of a VMEbus system.

RECOMMENDATION 4.1:
IF a board includes both an IACK DAISY-CHAIN DRIVER and an INTERRUPTER, and might or might not be installed in slot 1,
THEN design it as shown in Figure 4-9.

PERMISSION 4.6:
Several boards containing IACK DAISY-CHAIN DRIVERS **MAY** be installed in a VMEbus system.

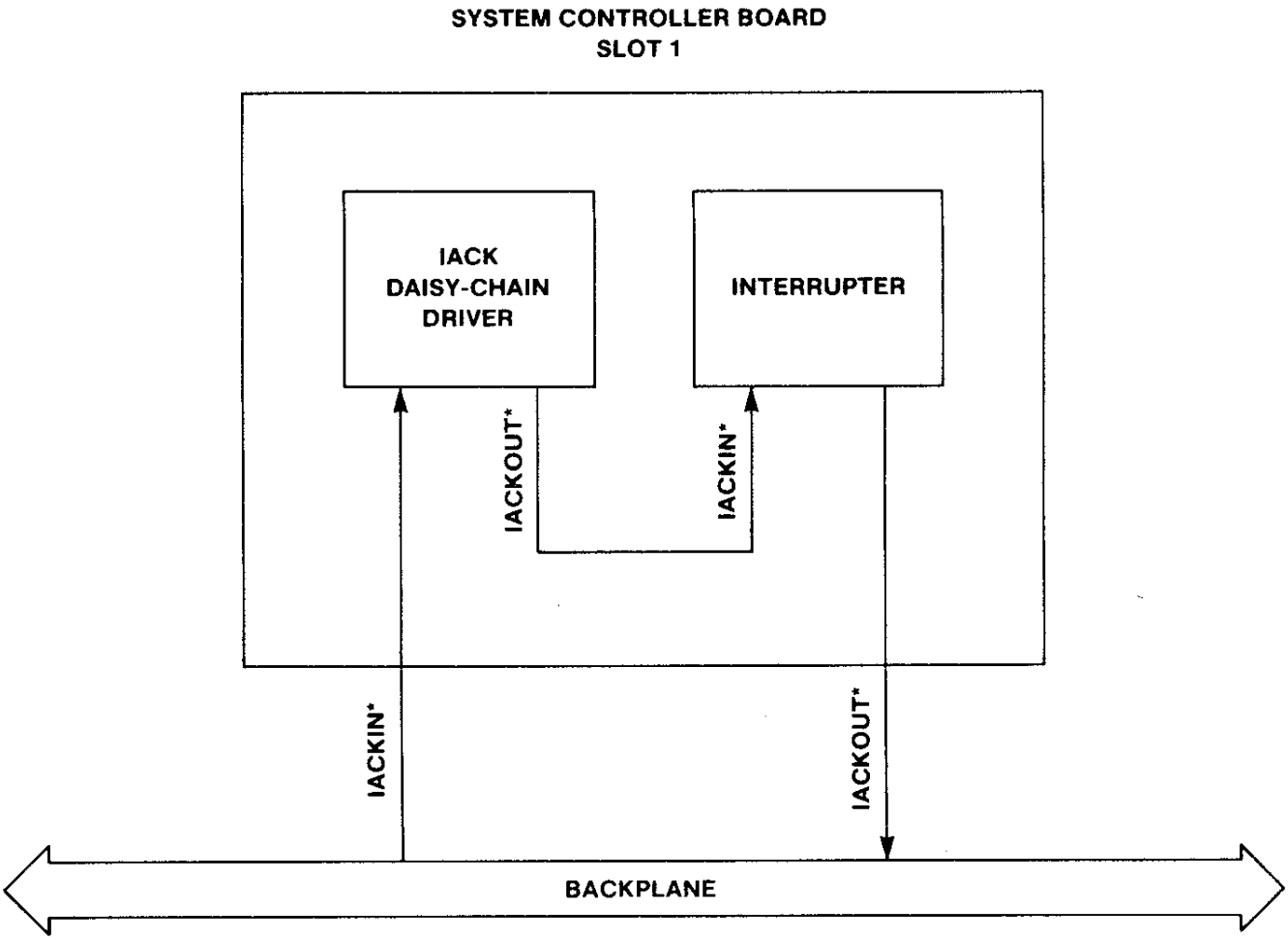


Figure 4-9. An IACK DAISY-CHAIN DRIVER And INTERRUPTER On The Same Board

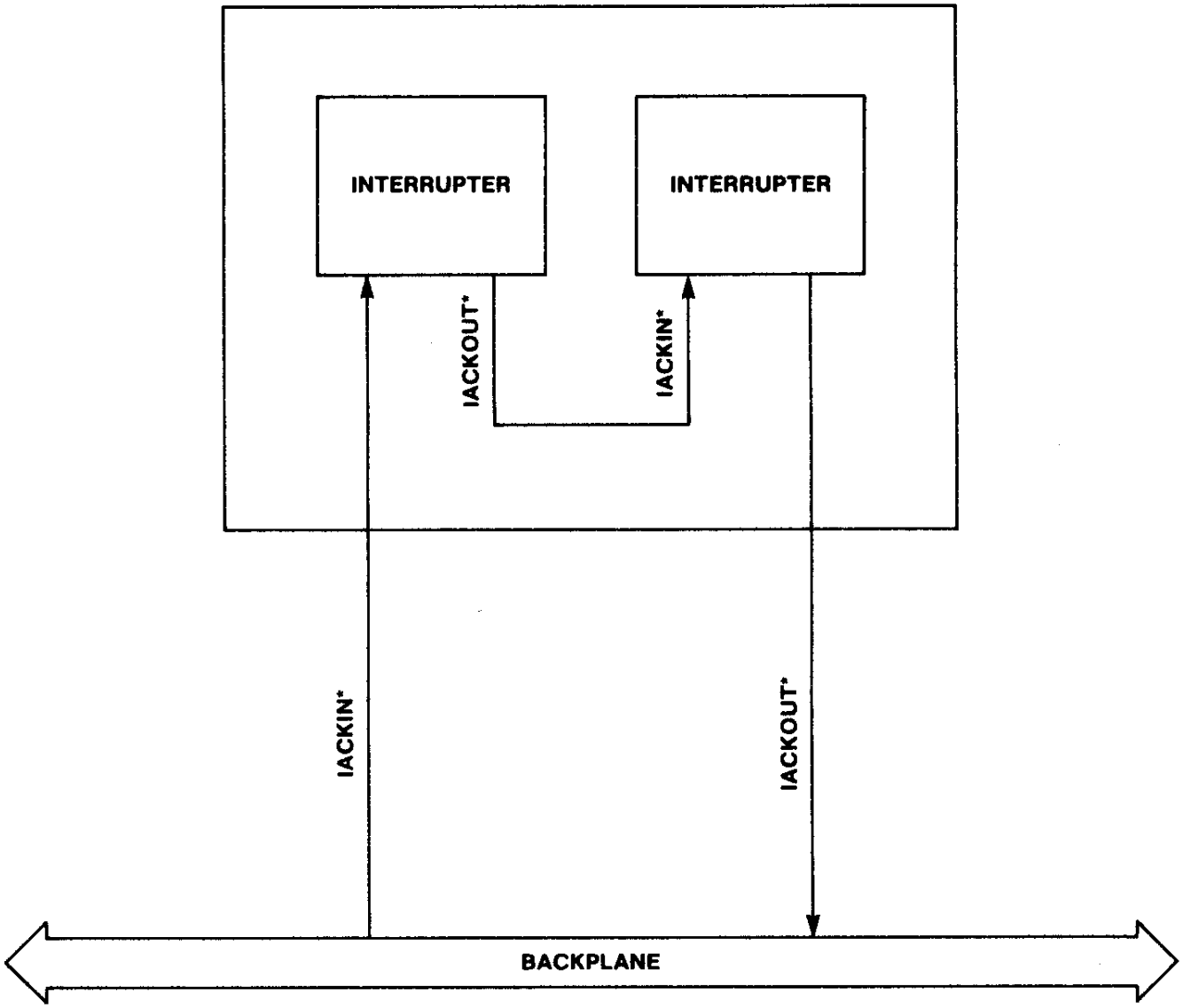


Figure 4-10. Two INTERRUPTERS On The Same Board

4.4 TYPICAL OPERATION

A typical interrupt sequence can be divided into three phases:
Phase 1: The interrupt request phase.
Phase 2: The interrupt acknowledge phase.
Phase 3: The interrupt servicing phase.

Figure 4-1 1 illustrates the timing relationships between the three phases.

Phase 1 starts when an INTERRUPTER drives an interrupt request line low and ends when the INTERRUPT HANDLER gains control of the DTB. During phase 2 the INTERRUPT HANDLER uses the DTB to read the lNTERRUPTER'S STATUS/ID. During phase 3 an interrupt servicing routine is executed. (This might or might not involve data transfers on the VMEbus.)

The protocol for the interrupt subsystem describes the module interaction required during phase 1 and phase 2. Any data transfers which take place during phase 3 will follow the Data Transfer Bus protocol described in Chapter 2.

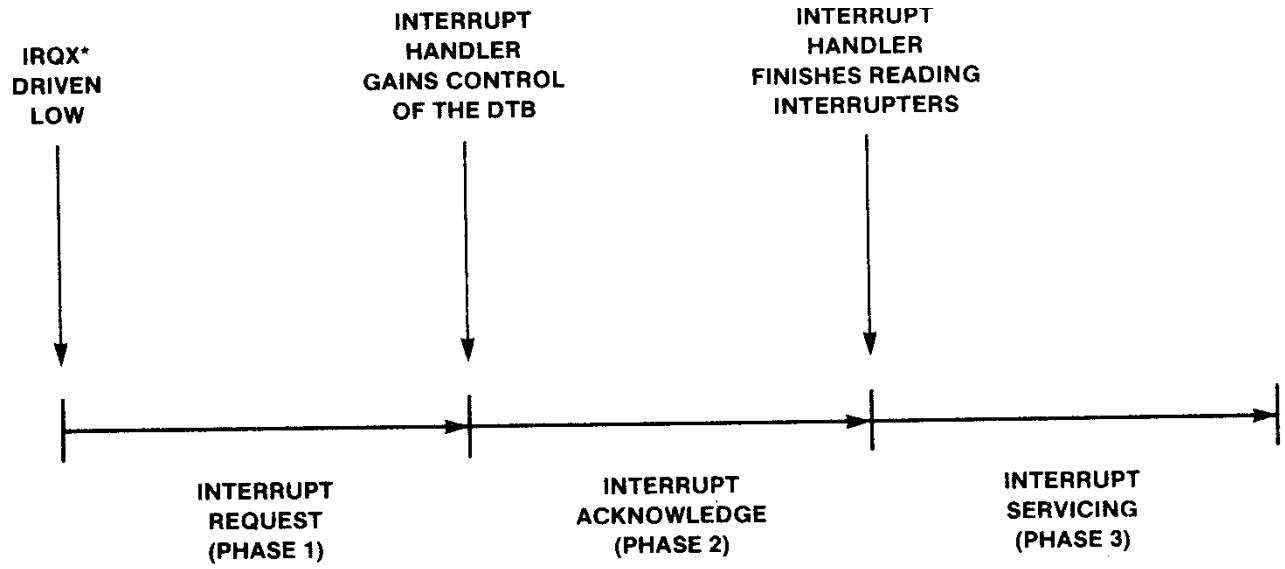


Figure 4-11 . The Three Phases Of An Interrupt Sequence

4.4.1 Single Handler Interrupt Operation

In single handler interrupt systems, the seven interrupt request lines are all monitored by a single INTERRUPT HANDLER. The interrupt request lines are prioritized such that IRQ7* has the highest priority, and IRQ1* has the lowest priority. When the INTERRUPT HANDLER detects simultaneous requests on two interrupt request lines, it acknowledges the highest priority request first.

4.4.2 Distributed Interrupt Operation

Distributed interrupt systems contain from two to seven INTERRUPT HANDLERS. For purposes of the following discussion, distributed interrupt systems will be considered in two groups:
a. distributed interrupt systems with seven INTERRUPT HANDLERS,
b. distributed interrupt systems with two to six INTERRUPT HANDLERS.

4.4.2.1 Distributed Interrupt Systems With Seven INTERRUPT HANDLERS

In distributed interrupt systems with seven INTERRUPT HANDLERS, each of the interrupt request lines is monitored by a separate INTERRUPT HANDLER. Each INTERRUPT HANDLER gains control of the DTB before it reads the STATUS/ID from INTERRUPTERS driving its interrupt request line.

OBSERVATION 4.9:

There is no specified relationship between the interrupt request line that an INTERRUPT HANDLER services and the bus request line used by its on-board REQUESTER. For example, an INTERRUPT HANDLER that services IRQ7* might have a REQUESTER that uses BR0*, and an INTERRUPT HANDLER that services IRQ1* might have a REQUESTER that uses BR3*. It is clear from this that there is no implied interrupt priority between lines serviced by different INTERRUPT HANDLERS.

Figure 4-12 illustrates a distributed interrupt system where INTERRUPT HANDLER A monitors IRQ2* and has an on-board REQUESTER which requests the DTB on BR2*. INTERRUPT HANDLER B monitors IRQ5* and has an on-board REQUESTER which requests the DTB on BR3*. Two INTERRUPTERS simultaneously drive IRQ2* and IRQ5* low, and the two INTERRUPT HANDLERS cause their on-board REQUESTERS to drive BR2* and BR3* low simultaneously. In this example, priority arbitration is used and, since both bus requests go low together, the ARBITER first grants control of the DTB to INTERRUPT HANDLER B's REQUESTER, and INTERRUPT HANDLER A waits until B has finished using the DTB.

OBSERVATION 4.10:
If round-robin arbitration is used, either of the INTERRUPT HANDLERS described in Figure 4-1 2 might be granted the bus first.

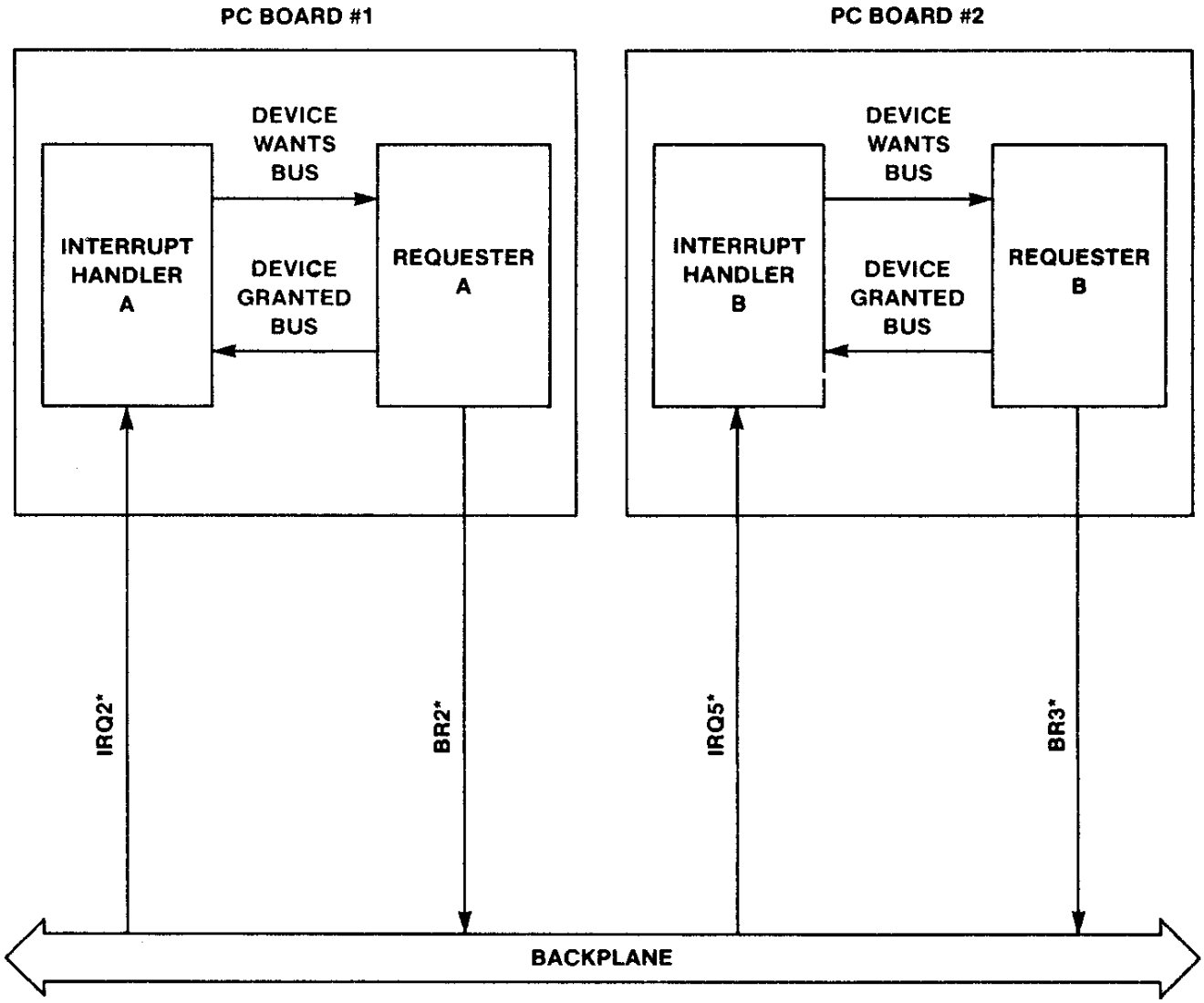


Figure 4-12. Two INTERRUPT HANDLERS, Each Monitoring One Interrupt Request Line

4.4.2.2 Distributed Interrupt Systems With Two To Six INTERRUPT HANDLERS

It is also possible to configure a distributed interrupt system in which two or more of the interrupt request lines are monitored by a single INTERRUPT HANDLER. Figure 4-13 illustrates a system configured with two INTERRUPT HANDLERS in which INTERRUPT HANDLER A monitors IRQ1*-IRQ4*, and INTERRUPT HANDLER B monitors IRQ5*-IRQ7*. In this case, the IRQ1*-IRQ4* lines are prioritized; IRQ4* = highest priority for INTERRUPT HANDLER A, and the IRQ5*-IRQ7* lines are prioritized; IRQ7* = highest priority for INTERRUPT HANDLER B. The DTB arbitration still determines which INTERRUPT HANDLER is granted the use of the DTB first.

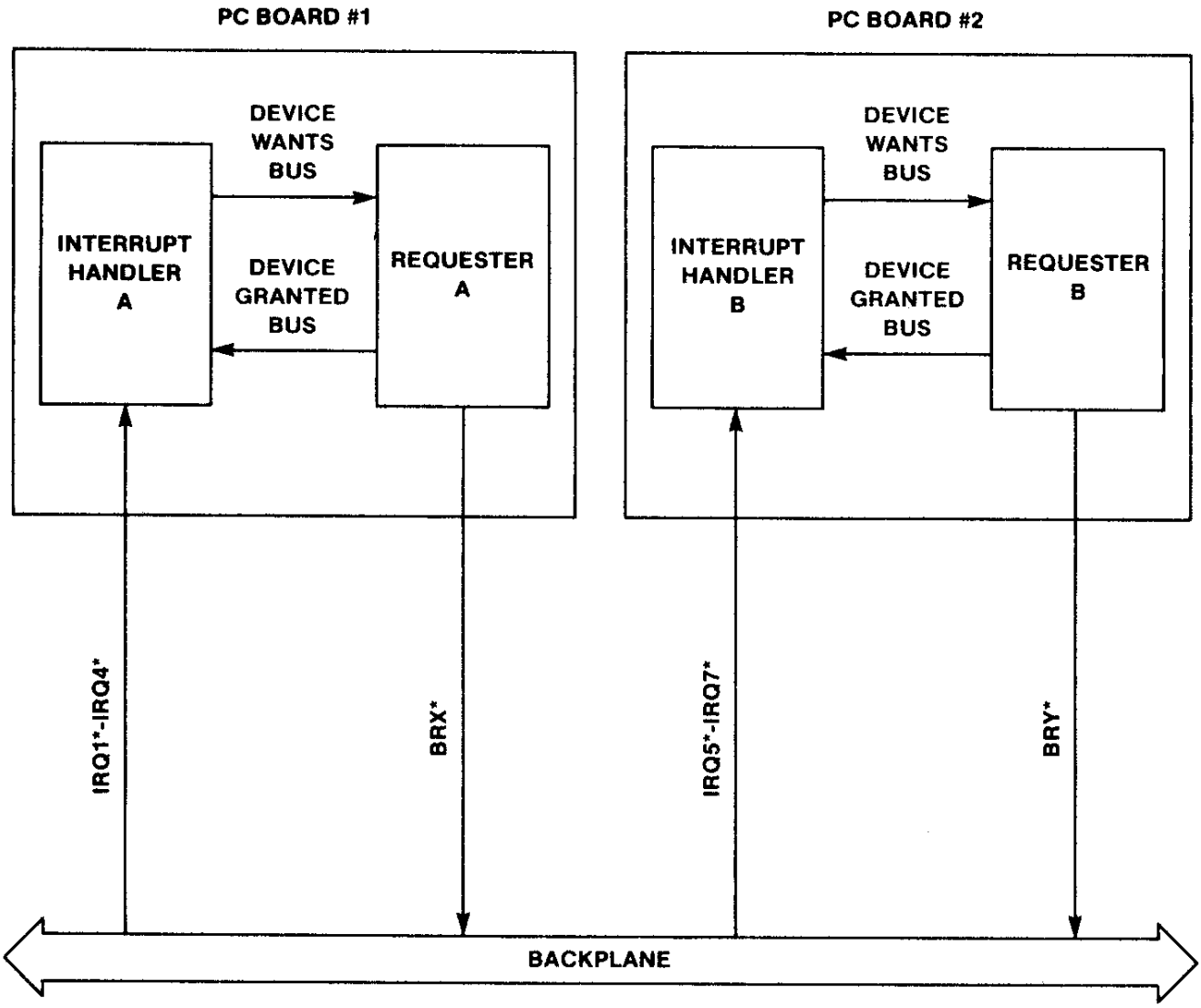


Figure 4-13. Two INTERRUPT HANDLERS, Each Monitoring Several Interrupt Request Lines

4.4.3 Example: Typical Single Handler Interrupt System Operation

Figure 4-14 illustrates the operation of a single handler interrupt system in which one INTERRUPT HANDLER monitors and prioritizes all seven interrupt lines. At the top of the diagram, a MASTER is using the DTB to move data within the system at a bus request level of 2. An INTERRUPTER in slot 3 requests an interrupt by driving IRQ4* low. When the INTERRUPT HANDLER detects the low level on IRQ4* it sends a signal to its on-board REQUESTER, indicating that it needs the bus. This REQUESTER then drives BR3* low. Upon detecting the bus request, the ARBITER drives BCLR* low, indicating that a higher priority REQUESTER is waiting for the DTB. (This example assumes a PRI ARBITER). When MASTER A detects the low level on BCLR*, it stops moving data and allows its requester to relinquish control of the DTB, and release BBSY*.

OBSERVATION 4.1 1:

The active MASTER is not required to relinquish the DTB within any specified time, but a prompt response to the BCLR* line allows the interrupt to be serviced quicker.

When the ARBITER detects BBSY* high, it grants the DTB to REQUESTER B, which informs its INTERRUPT HANDLER that the DTB is available (see Figures 2-26 and 227). The INTERRUPT HANDLER then puts out a 3-bit code on the lower three address lines, indicating that it is acknowledging the interrupt request on the IRQ4* line (see Table 4-7). At the same time, it drives IACK* low, indicating that it is acknowledging an interrupt, and drives AS* low. The low level on IACK* is connected, by a signal trace in the backplane, to the IACKIN* pin of slot 1 and causes the IACK DAISY-CHAIN DRIVER to generate a falling edge down the IACKIN*/IACKOUT* daisy-chain.

When the INTERRUPTER detects a falling edge on its incoming daisy-chain line (IACKIN*) it checks the lower three address bits to see if they match the interrupt request line which it is driving low. Since the 3-bit code matches the line on which it is making its interrupt request, when the INTERRUPTER detects the data strobe(s) low, it places its STATUS/ID on the data bus and drives the DTACK* line low. When the INTERRUPT HANDLER detects the low DTACK*, it reads the STATUS/ID and activates the appropriate interrupt service routine.

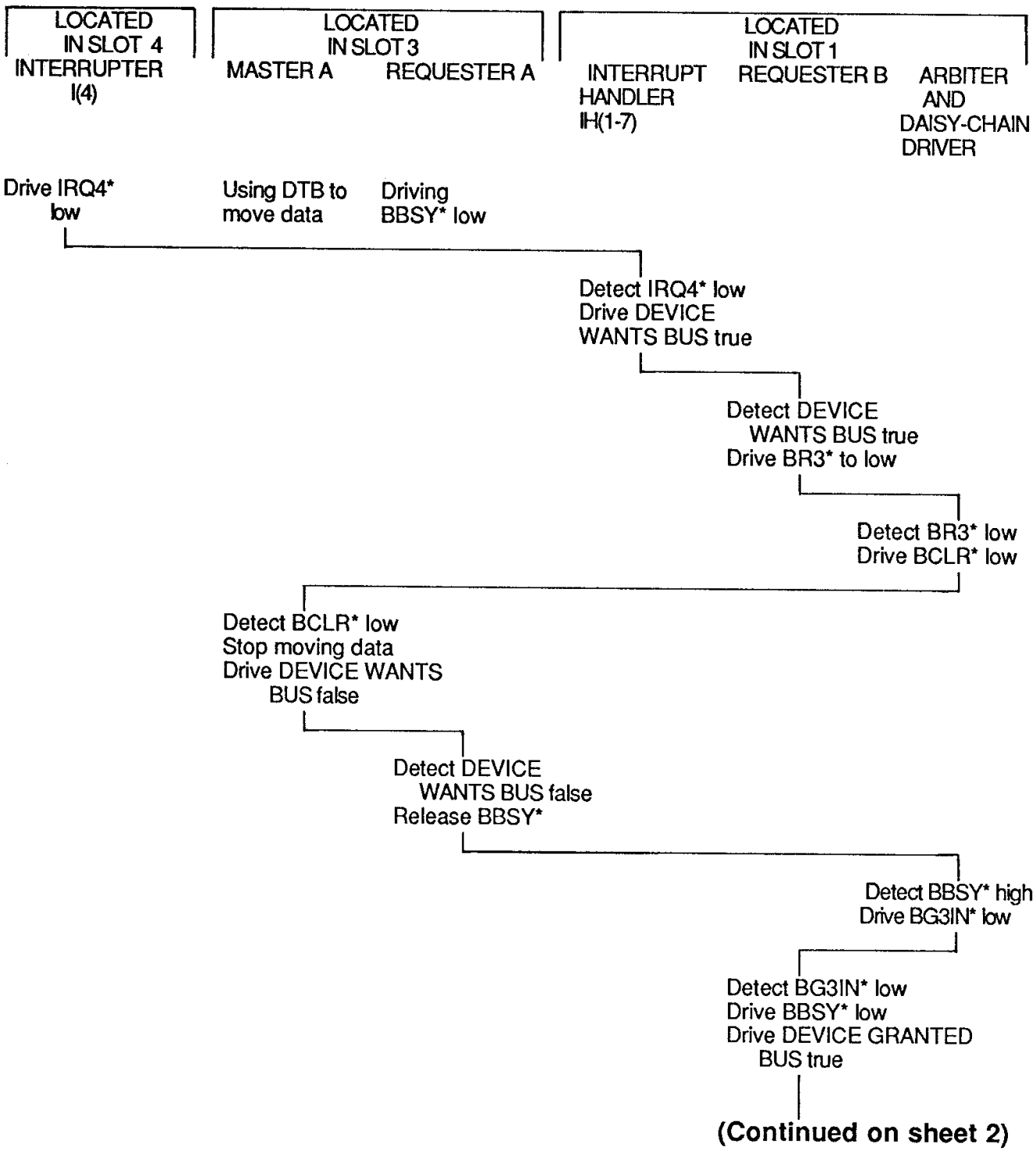


Figure 4-14. Typical Single Handler Interrupt System Operation Flow Diagram (Sheet 1 Of 2)

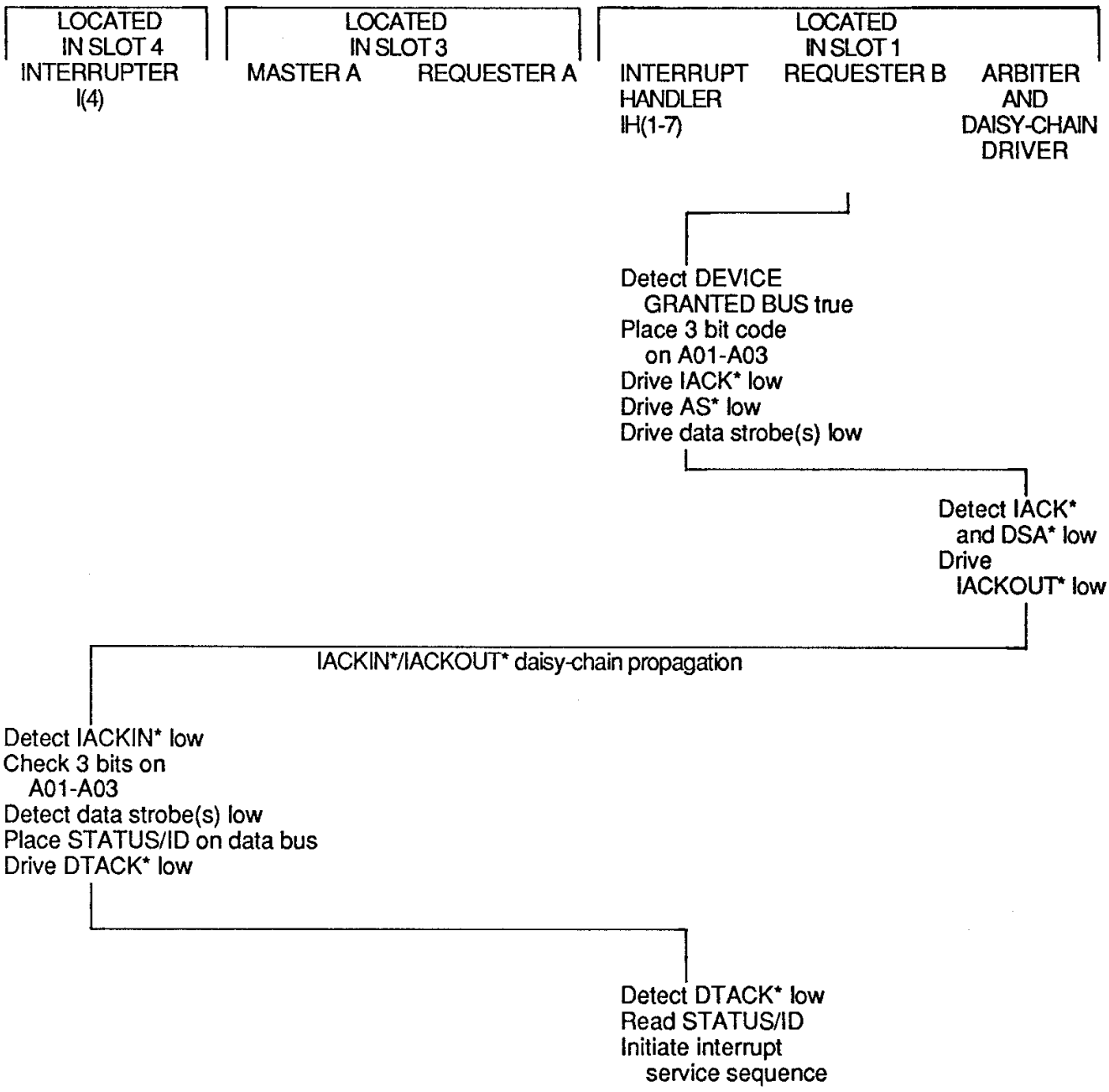


Figure 4-14b. Typical Single Handler Interrupt System Operation Flow Diagram (Sheet 2 Of 2)

Table 4-7. 3-Bit Interrupt Acknowledge Code

Interrupt line being acknowledged	Use of the address bus to broadcast the 3-bit interrupt acknowledge code		
	A03	A02	A01
IRQ1*	L	L	H
IRQ2*	L	H	L
IRQ3*	L	H	H

IRQ4*	H	L	L
IRQ5*	H	L	H
IRQ6*	H	H	L
IRQ7*	H	H	H

H = High level
L = Low level

4.4.4 Example: Prioritization Of Two Interrupts In A Distributed Interrupt System

Figure 4-1 5 illustrates the operation of a distributed interrupt system with two INTERRUPT HANDLERS. INTERRUPT HANDLER A monitors IRQ1*-IRQ4*, while INTERRUPT HANDLER B monitors IRQ5*-IRQ7*. INTERRUPT HANDLER A treats IRQ4* as its highest priority interrupt, while INTERRUPT HANDLER B treats IRQ7* as its highest priority interrupt. At the top of the diagram, INTERRUPTER C drives IRQ3* low, and INTERRUPTER D drives IRQ6* low. Both INTERRUPT HANDLERS detect their respective interrupt request lines low, and both simultaneously indicate to their on-board REQUESTER that they need the DTB. Both the REQUESTERS drive BR3* Iow. Upon detecting BR3* low, the DTB ARBITER drives BG3IN* low on slot 1. This low signal is passed down the BG3IN*/BG30UT* daisy-chain until it is detected by the REQUESTER B in slot 4. This REQUESTER then signals its on-board INTERRUPT HANDLER B that the DTB is available. INTERRUPT HANDLER B then reads the STATUS/ID from INTERRUPTER D.

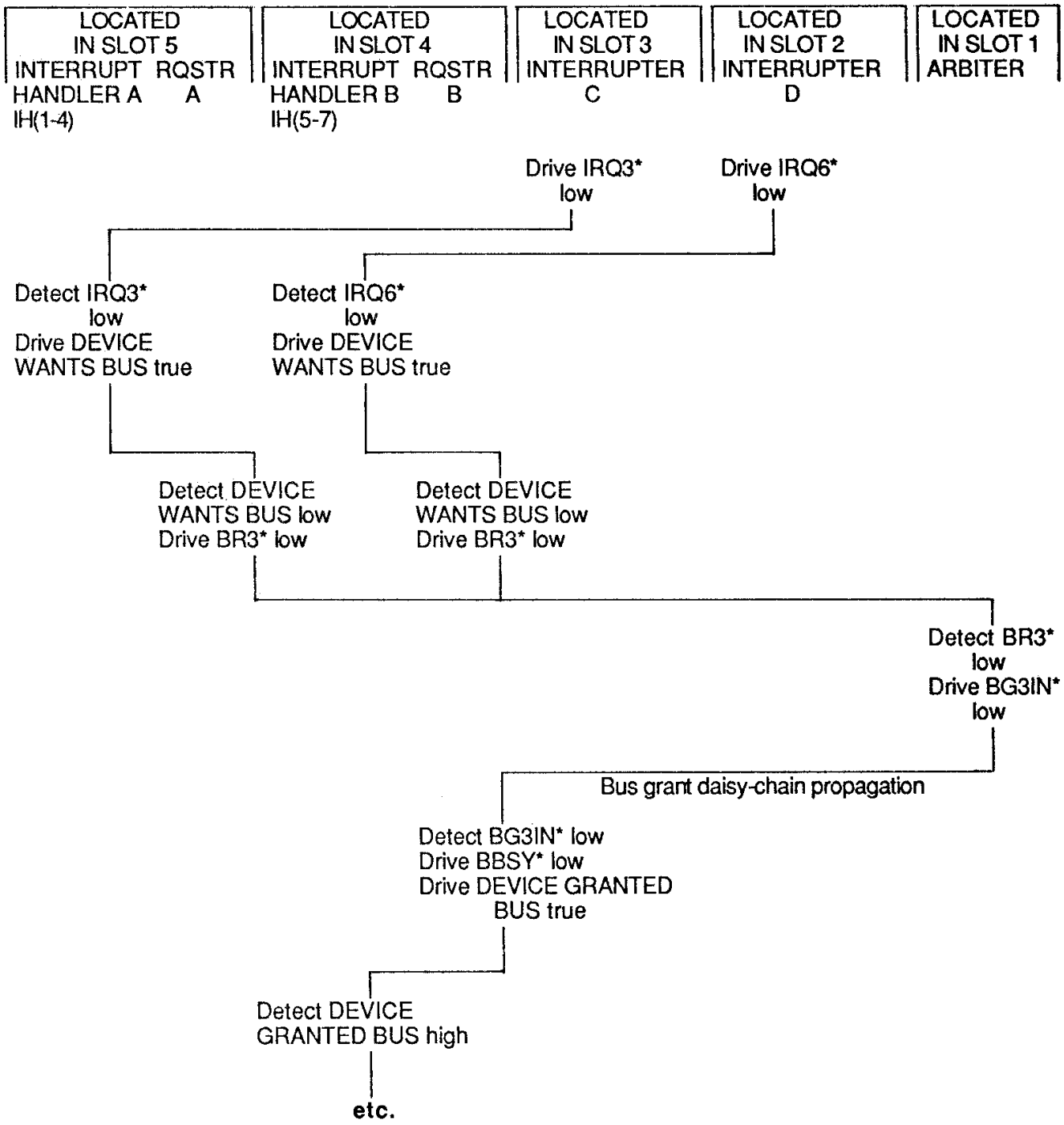


Figure 4-15. Typical Distributed interrupt System With Two INTERRUPT HANDLERS, Flow Diagram

4.5 PRIORITY INTERRUPT BUS TIMING RULES AND OBSERVATIONS

This section describes the timing RULES and OBSERVATI0NS that govern the behavior of INTERRUPT HANDLERS, INTERRUPTERS, and IACK DAISY-CHAIN DRIVERS during the selection of the responding INTERRUPTER (i.e. the INTERRUPTER that is to provide its STATUS/ID in response to the interrupt acknowledge cycle). This timing information is in the form of Figures and Tables.

The interrupt acknowledge cycle begins with the selection of the responding INTERRUPTER. This is called the INTERRUPTER selection portion of the cycle. Once an INTERRUPTER responds, the INTERRUPT HANDLER reads the STATUS/ID from it. This is called the STATUS/ID transfer portion of the cycle.

When the INTERRUPT HANDLER initiates an interrupt acknowledge cycle, there might be INTERRUPTERS between it and the INTERRUPTER being acknowledged that either:

a. do not have an interrupt pending,
b. have an interrupt pending, or on a different interrupt request line than the one being acknowledged.

Although these INTERRUPTERS do not respond with a STATUS/ID, they do participate in the interrupt acknowledge cycle by passing the falling edge from their IACKIN* line on to their IACKOUT* line. For this reason, these INTERRUPTERS are called participating INTERRUPTERS.

The first INTERRUPTER in the daisy-chain that has an interrupt pending on the interrupt request line being acknowledged responds with a STATUS/ID. For this reason it is called the responding INTERRUPTER. All other INTERRUPTERS are called non-participating INTERRUPTERS.

Table 4-8 lists timing Tables and timing diagrams that specify INTERRUPT HANDLER and INTERRUPTER operation.

Table 4-9 lists timing Tables and timing diagrams that specify IACK DAISY-CHAIN DRIVER operation.

Table 4-10 lists timing Tables and timing diagrams that specify participating INTERRUPTER operation.

Table 4-11 lists timing Tables and timing diagrams that specify responding INTERRUPTER operation.

Table 4.12 defines the mnemonics that are used **in Tables** 4-13 through 4-15.

Tables 4-1 through 4-15 specify the use of bus signal lines by the Priority Interrupt Bus functional modules.

Tables 4-16 through 4-19 specify the timing parameters for the Priority Interrupt Bus functional modules. (The reference numbers used in Tables 4-17 through 4-19 correspond to the timing parameter numbers in Table 4-1 6.)

Figures 4-16 through 4-23 are timing diagrams that specify the timing during interrupt acknowledge cycles.

Figure 4-24 specifies additional inter-cycle timing for the IACKIN*/IACKOUT daisychain .

All of the RULES and OBSERVATI0NS associated with the Figures listed below also apply to INTERRUPT HANDLERS, INTERRUPTERS, and IACK DAISY-CHAIN DRIVERS.

Figures 2-22 through 2.24 in Chapter 2 specify the timing for the address and data strobes between data transfer cycles.

Figure 2-25 specifies the timing of a timed-out cycle.

Figures 2-26 and 2-27 specify the timing during mastership transfer.

In order to meet; the specified timing, board designers have to take into account the worst case propagation delays of the bus drivers and receivers used on their VMEbus boards. The propagation delay of the drivers depends on their output loads. However, manufacturers specifications do not always give enough information to calculate the propagation delays under various loads. To help the VMEbus board designer, some suggestions are offered in Chapter 6.

The OBSERVATIONS specify the timing of incoming lines signal transitions. These times can be relied upon as long as the backplane loading RULES in Chapter 6 are not violated. The RULES for the bus terminators in Chapter 6 guarantee that the timing parameters for signal lines that are released after they have been driven, are met

Typically, for each timing RULE there is a corresponding timing OBSERVATI0N. However, the time that is guaranteed in the OBSERVATIONS might differ from the time specified by the RULE. For example, a careful inspection of the timing diagrams shows that the INTERRUPT HANDLER is required to provide 35 nanoseconds of address set-up time, but the INTERRUPTER is only guaranteed 1 0 nanoseconds. This is because the address drivers are not always able to drive the backplane’s signal lines completely through the low to high threshold region, until the transition propagates to the end of the backplane and is reflected back. The falling edge of the address strobe, however, typically crosses the 0.8-volt threshold without waiting for a reflection. Therefore, the resulting set-up time at the INTERRUPTER is the INTERRUPT HANDLER’S set-up time less two bus propagation times.

A special notation has been used to describe the data strobe timing. The two data strobes (DS0* and DS1*) do not always make their transitions simultaneously. For purposes of these timing diagrams, DSA* represents the first data strobe to make its

transition (whether that is DS0* or DS1*). DSB* represents the second data strobe to make its transition (whether that is DS1* or DS0*). The broken line shown while the data strobes are stable indicates that the first data strobe to make a falling transition might not be the first to make its rising transition -- i.e., DSA* might represent DS0* on its falling edge and DS1* on its rising edge.

Table 4-8. Timing Diagrams That Define INTERRUPT HANDLER And INTERRUPTER Operation

Mnemonic	Type of cycle	INTERRUPTER Selection Timing Diag Figures	STATUS/ID Transfer Timing Diag Figure
D08(0)	SINGLE BYTE STATUS/ID READ	2-12 & 4-16	4-20
D16	DOUBLE BYTE STATUS/ID READ	2-12 & 4-16	4-21
D32	QUAD BYTE STATUS/ID READ	2-12 & 4-16	4-21

Note:
See Tables 4-16 and 4-17 for timing parameters

Table 4-9. Timing Diagrams That Define IACK DAISY-CHAIN DRIVER Operation

Type of cycle	INTERRUPTER Selection Timing Diag
SINGLE BYTE STATUS/ID READ	Figure 4-17
DOUBLE BYTE STATUS/ID READ	Figure 4-17
QUAD BYTE STATUS/ID READ	Figure 4-17

Note:
See Tables 4-16 and 4-19 for timing values

Table 4-10. Timing Diagrams That Define Participating INTERRUPTER Operation

Type of cycle	INTERRUPTER Selection Timing Diag
SINGLE BYTE STATUS/ID READ	Figure 4-18
DOUBLE BYTE STATUS/ID READ	Figure 4-18
QUAD BYTE STATUS/ID READ	Figure 4-18

Note:
See Tables 4-16 and 4-18 for timing values

Table 4-11 . Timing Diagrams That Define Responding INTERRUPTER Operation

Mnemonic	Type of cycle	INTERRUPTER Selection Timing Diag Figures	STATUS/ID Transfer Timing Diag Figure
D08(0)	SINGLE BYTE STATUS/ID READ	4-19	4-22
D16	DOUBLE BYTE STATUS/ID READ	4-19	4-23
D32	QUAD BYTE STATUS/ID READ	4-19	4-23

Note:

See Tables 4-16 and 4-18 for timing parameters

Table 4-12. Definitions Of Mnemonics Used In Tables 4-13, 4-14, And 4-15

Mnemonic	Description	Comments
DVBIH	DRIVEN VALID BY INTERRUPT HANDLER	**RULE 4.9:** INTERRUPT HANDLER **MUST** drive DVBIH lines to a valid level.
DLBIH	DRIVEN LOW BY INTERRUPT HANDLER	**RULE 4.10:** INTERRUPT HANDLER **MUST** drive DLBIH lines to a low level
DHBIH	DRIVEN HIGH BY INTERRUPT HANDLER	**RULE 4.11:** INTERRUPT HANDLER **MUST** drive DHBIH lines to a high level
dhbih?	DRIVEN HIGH BY INTERRUPT HANDLER?	**PERMISSION 4 7:** INTERRUPT HANDLER **MAY** drive dhbih? lines high. **RULE 4.12:** INTERRUPT HANDLER **MUST NOT** drive dhbih? lines low during the cycle.
dxbih?	DRIVEN BY INTERRUPT HANDLER?	**PERMISSION 4.8:** INTERRUPT HANDLER **MAY** drive dxbih? lines during the cycle, or it **MAY** leave dxbih? lines undriven (When the line is driven it carries no valid information.)
DVBI	DRIVEN VALID BY INTERRUPTER	**RULE 4.13:** INTERRUPTER **MUST** drive DVBI lines to a valid level.
dhbi?	DRIVEN BY INTERRUPTER?	**PERMISSION 4.9:** INTERRUPTER **MAY** drive dhbi? lines high. **RULE 4.14:** INTERRUPTER **MUST NOT** drive dhbi? lines low.
dxbi?	DRIVEN BY INTERRUPTER?	**PERMISSION 4.10:** INTERRUPTER **MAY** drive dxbi? lines during the cycle, or it **MAY** leave the line undriven. (When the line is driven, it carries no valid information.)

Table 4-13. Use Of Addressing Lines During Interrupt Acknowledge Cycles

Interrupt Line Being Acknowledged	A03	A02	A01	IACK*
IRQ1*	DLBIH	DLBIH	DHBIH	DLBIH
IRQ2*	DLBIH	DHBIH	DLBIH	DLBIH

IRQ3*	DLBIH	DHBIH	DHBIH	DLBIH
IRQ4*	DHBIH	DLBIH	DLBIH	DLBIH
IRQ5*	DHBIH	DLBIH	DHBIH	DLBIH
IRQ6*	DHBIH	DHBIH	DLBIH	DLBIH
IRQ7*	DHBIH	DHBIH	DHBIH	DLBIH

Table 4-14. Use Of The DS1 *, DS0*, LWORD*, And WRITE* Lines During Interrupt Acknowledge Cycles

Mnemonic	Type of cycles	DS1*	DS0*	LWORD*	WRITE*
D08(0)	Single byte interrupt acknowledge	dhbih?	DLBIH	dhbih?	dhbih?
D16	Double byte interrupt acknowledge	DLBIH	DLBIH	dhbih?	dhbih?
D32	Quad byte interrupt acknowledge	DLBIH	DLBIH	DLBIH	dhbih?

Table 4-15. Use Of The Data Bus Lines To Transfer The STATUS/ID

Mnemonic	Type of cycles	D24-D31	D16-D23	D08-D15	D00-D07
D08(0)	Single, double, and quad byte interrupt acknowledge	dhbi?	dhbi?	dhbi?	DVBI
D16	Double and quad byte interrupt acknowledge	dhbi?	dhbi?	DVBI	DVBI
D32	Quad byte interrupt acknowledge	DVBI	DVBI	DVBI	DVBI

Table 4-16: INTERRUPT HANDLER, INTERRUPTER, And IACK DAISY-CHAIN DRIVER Timing Parameters

PARAMETER	INTERRUPT HANDLER See Table 4-17		INTERRUPTER See Table 4-18		IACK DAISY CHAIN DRIVER See Table 4-19	
	MIN	MAX	MIN	MAX	MIN	MAX
1	0					
2	0					
3	60					
4	35		10			
5	40		30		30	
6			0			
7			0			
9	0		0			
10	0					
11	40		30			
12	35		10			

13		10		20		
14	0		0			
16	0		0			
18	0		0			
19	40		30		30	
20	0		0			
21	0		0			
23	10		0			
24A	0					
24B	0					
25		25				
26	0		0			
27	-25		0			
28	30	2T	30			
29	0		0			
30	0		0			
31	0		0			
32			10		10	
34			40		40	
35				30		30
36			0			
37			0			
38A						
38B			0			
39				40		
40			30		30	
41			0			
42					30	
43			30			

NOTES:
1. All times are in nanoseconds.
2. T = the time-out value.

Table 4-17: INTERRUPT HANDLER, Timing RULES And OBSERVATIONS

Note: The numbers correspond to the timing parameters specified in Table 4-16.

1.	**RULE 4.15:** When taking control of the VMEbus, the INTERRUPT HANDLER **MUST NOT** drive any of IACK*, A01-A03, LWORD*, WRITE*, DS0*, DS1* or AS* until after the previous MASTER or INTERRUPT HANDLER allows AS* to rise above the low level. **OBSERVATION 4.12:** Chapter 3 describes how an INTERRUPT HANDLER'S REQUESTER is granted

	use of the VMEbus.
2.	**RULE 4.16:** When taking control of the VMEbus, the INTERRUPT HANDLER **MUST NOT** drive any of IACK*, A01-A03, LWORD*, WRITE*, DS0*, DS1*, or AS* until after it receives DEVICE GRANTED BUS true. **OBSERVATION 4.13:** Chapter 3 describes how an INTERRUPT HANDLER'S REQUESTER is granted use of the VMEbus.
3.	**RULE 4.1 7:** When taking control of the VMEbus, the INTERRUPT HANDLER **MUST NOT** drive AS* low until this time after the previous MASTER or INTERRUPT HANDLER allows AS* to rise above the low level. **OBSERVATION 4.1 4:** RULE 4.17 ensures that timing parameter 5 for INTERRUPTERS and SLAVES is guaranteed when there is an interchange of the DTB mastership.
4.	**RULE 4.18:** The INTERRUPT HANDLER **MUST NOT** drive AS* low until IACK* has been low, and LWORD* and A01 -A03 have been valid for this minimum time.
5.	**RULE 4.19:** When using the DTB for two consecutive cycles, the INTERRUPT HANDLER **MUST** drive AS* low until it has been high for this minimum time.
9.	**RULE 4.20:** The INTERRUPT HANDLER **MUST NOT** drive DSA* low until both DTACK* and BERR* are high.
10.	**RULE 4.21:** The INTERRUPT HANDLER **MUST NOT** drive DSA* low before it has driven AS* low.
11.	**RULE 4.22:** The INTERRUPT HANDLER **MUST NOT** drive DSA* low until DS0* and DS1* have been simultaneously high for this minimum time.
12.	**RULE 4.23:** The INTERRUPT HANDLER **MUST NOT** drive DSA* low until WRITE* has been high for this minimum time.
13.	**RULE 4.24:** During double byte or quad byte STATUS/ID read cycles, the INTERRUPT HANDLER **MUST** drive DSB* low within this maximum time after it drives DSA* low. **OBSERVATION 4.15:** Timing parameter 1 3 does not apply to single byte STATUS/ID reads.
14.	**RULE 4.25:** During all interrupt acknowledge cycles, the INTERRUPT HANDLER **MUST** hold the 3 bit interrupt acknowledge code valid, and maintain the appropriate level on LWORD* until it detects a falling edge on DTACK* or BERR*.
16.	**RULE 4.26:** During all interrupt acknowledge cycles, the INTERRUPT HANDLER **MUST** maintain IACK* low until it detects a falling edge on DTACK* or BERR*.
18.	**RULE 4.27:**

	The INTERRUPT HANDLER **MUST** hold AS* low until it detects DTACK* or BERR* low.
19.	**RULE 4.28:** The INTERRUPT HANDLER **MUST** hold AS* low for this minimum time.
20.	**RULE 4.29:** Once an INTERRUPT HANDLER has driven DSA* low, it **MUST** maintain it low until it detects DTACK* or BERR* low.
21.	**RULE 4.30:** Once an INTERRUPT HANDLER has driven DSB* low, it **MUST** maintain it low until it detects DTACK* or BERR* low.
23.	**RULE 4.31:** Once an INTERRUPT HANDLER has driven DSA* low, it **MUST** maintain a high on the WRITE* line until this minimum time after both data strobes are high.
24A.	**RULE 4.32:** IF the INTERRUPT HANDLER drives or releases AS* to high after its REQUESTER releases BBSY*, THEN it **MUST** release IACK*, A01-A03, LWORD*, WRITE*, DS0*, and DS1* before allowing AS* to rise above the low level.
24B.	**RULE 4.33:** IF the INTERRUPT HANDLER drives or releases AS* to high before its REQUESTER releases BBSY*, THEN it **MUST** release IACK*, A01-A03, LWORD*, WRITE*, DS0*, and DS1* before changing its DEVICE WANTS BUS signal from true to false.
25.	**RULE 4.34:** IF the INTERRUPT HANDLER drives or releases AS* to high after its REQUESTER releases BBSY*, THEN it **MUST** release AS* within this time after allowing it to rise above the low level.
26.	**OBSERVATION 4.16:** Timing parameter 26 guarantees that the data bus will not be driven until the INTERRUPT HANDLER drives DSA* low.
27.	**OBSERVATION 4.17:** The INTERRUPT HANDLER is guaranteed that the data bus will be valid within this time after DTACK* goes low. This time does not apply to cycles where the INTERRUPTER drives BERR* low instead of DTACK*.
28.	**OBSERVATION 4.18:** The INTERRUPT HANDLER is guaranteed that neither DTACK* nor BERR* will go low until this minimum time after it drives DSA* low. The BUS TIMER guarantees the INTERRUPT HANDLER that if DTACK* has not gone low after its time-out period has elapsed, and within twice its time-out period, then the BUS TIMER will drive BERR* low.
29.	**OBSERVATION 4.19:** The INTERRUPT HANDLER is guaranteed that the data bus remains valid until it drives DSA* high.
30.	**OBSERVATION 4.20:** This guarantees that neither DTACK* nor BERR* goes high until the INTERRUPT HANDLER drives both DS0* and DS1* high.

31.	**OBSERVATION 4.21:** The INTERRUPT HANDLER is guaranteed that the data bus has been released by the time DTACK* and BERR* are high.

Table 4-18. INTERRUPTER, Timing RULES And OBSERVATIONS

Note: The numbers correspond to the timing parameters specified in Table 4-16.

4.	**OBSERVATION 4.22:** INTERRUPTERS are guaranteed that IACK*, LWORD*, and A01-A03 have been valid for this minimum time when they detect a falling edge on AS*.
5.	**OBSERVATION 4.23:** All INTERRUPTERS are guaranteed this minimum high time on AS* between DTB cycles.
6.	**OBSERVATION 4.24:** The responding INTERRUPTER is guaranteed that none of D00-D31 will be driven by any other module until the responding INTERRUPTER releases DTACK* and BERR* to high.
7.	**OBSERVATION 4.25:** The responding INTERRUPTER is guaranteed that the data bus will be released by all other modules by the time DSA* goes low.
9.	**OBSERVATION 4.26:** The responding INTERRUPTER is guaranteed that neither DS0* nor DS1* will go low until DTACK* and BERR* from the the previous cycle have gone high.
11.	**OBSERVATION 4.27:** INTERRUPTERS are guaranteed this minimum time during which both data strobes are simultaneously high between cycles.
12.	**OBSERVATION 4.28:** INTERRUPTERS are guaranteed that WRITE* has been high for this minimum time when they detect a falling edge on DSA*.
13.	**OBSERVATION 4.29:** IF both data strobes are going to be driven low, THEN the responding INTERRUPTER is guaranteed that DSB* will go low within this maximum time after DSA*. And therefore: IF the DSB* does not go low within this maximum time, THEN the responding INTERRUPTER assumes that it is to respond with a single byte STATUS/ID.
14.	**OBSERVATION 4.30:** The responding INTERRUPTER is guaranteed that LWORD* and A01-A03 will remain valid until it drives DTACK* or BERR* low, provided it does so within the bus time-out period.
16.	**OBSERVATION 4.31:** The responding INTERRUPTER is guaranteed that IACK* will remain low until it drives DTACK* or BERR* low, provided it does so within the bus time-out period.
18.	**OBSERVATION 4.32:** The responding INTERRUPTER is guaranteed that AS* will remain low until it

	drives DTACK* or BERR* low, provided that it does so within the bus time-out period.
19.	**OBSERVATION 4.33:** INTERRUPTERS are guaranteed that the AS* will remain low for this minimum time.
20.	**OBSERVATION 4.34:** The responding INTERRUPTER is guaranteed that once DSA* goes low, it will remain low until the INTERRUPTER has driven DTACK* or BERR* low, provided that the INTERRUPTER does so within the bus time-out period.
21.	**OBSERVATION 4.35:** The responding INTERRUPTER is guaranteed that once DSB* goes low, it will remain low until the INTERRUPTER has driven DTACK* or BERR* low, provided that the INTERRUPTER does so within the bus time-out period.
23.	**OBSERVATION 4.36:** INTERRUPTERS are guaranteed that the WRITE. line remains high until both data strobes are high.
26.	**RULE 4.35:** The INTERRUPTER **MUST NOT** drive the data bus until DSA* goes low
27.	**RULE 4.36:** The responding INTERRUPTER **MUST NOT** drive DTACK* low before it drives the data lines with a valid STATUS/ID. **OBSERVATION 4.37:** This time does not apply to cycles where the INTERRUPTER drives BERR* low
28.	**RULE 4.37:** The responding INTERRUPTER **MUST** wait this minimum time after DSA* goes low before driving DTACK* or BERR* low.
29.	**RULE 4.38:** Once the responding INTERRUPTER has driven DTACK* low, it **MUST NOT** change D00-D31 until DSA* goes high.
30.	**RULE 4.39:** Once the responding INTERRUPTER has driven DTACK* or BERR* to low, it **MUST NOT** release it until it detects both DS0* and DS1* high.
31.	**RULE 4.40:** The responding INTERRUPTER **MUST** release all of D00-D31 before releasing DTACK* and BERR* to high.
32.	**OBSERVATION 4.38:** The responding INTERRUPTER is guaranteed that IACK*, LWORD*, and A01-A03 have been valid for this minimum time when it detects a falling edge on DSA*. This time is derived from timing parameters 4, and 10.
34.	**OBSERVATION 4.39:** The INTERRUPTER is guaranteed that AS* has been low for this minimum time, when it detects a falling edge on IACKIN*.
35.	**RULE 4.41:** A participating INTERRUPTER **MUST** drive its IACKOUT* high within this maximum time after the rising edge on AS*.
36.	**RULE 4.42:** The INTERRUPTER **MUST NOT** drive the data bus until its IACKIN* line goes low.

37.	**RULE 4.43:** IF a participating INTERRUPTER drives the data bus, THEN it **MUST** release it before driving its IACKOUT* line low.
38A.	**RULE 4.44:** A participating INTERRUPTER **MUST NOT** drive its IACKOUT* line low, until it detects its IACKIN* line low.
38B.	**RULE 4.45:** The responding INTERRUPTER **MUST NOT** drive its DTACK* line low, until it detects its IACKIN* line low.
39.	**OBSERVATION 4.40:** This time guarantees that each lNTERRUPTER'S IACKIN* line will go high within this time after the rising edge on AS*. This time is derived from timing parameter 35, where the IACK DAISY-CHAIN DRIVER and participating INTERRUPTERS are required to drive their IACKOUT* line high within a maximum time.
40.	**OBSERVATION 4.41:** All INTERRUPTERS are guaranteed that their IACKIN* line will stay high for this minimum time between consecutive DTB cycles.
41.	**OBSERVATION 4.42:** This time guarantees that A01-A03 and LWORD* remain valid until this time after the participating INTERRUPTER drives its IACKOUT* low, provided it does so within the bus time-out period.
43.	**OBSERVATION 4. 43:** This time guarantees that AS* remains low for this minimum time after the participating INTERRUPTER drives its IACKOUT* low, provided it does so within the bus time-out period.

Table 4-19. IACK DAISY-CHAIN DRIVER, Timing RULES And OBSERVATI0NS

Note:
The numbers correspond to the timing parameters specified in Table 4-16.

OBSERVATION 4.44:
Since the backplane connects IACK* to the slot 1 IACKIN*, these two signals are equivalent. And therefore, all RULES and OBSERVATIONS that apply to one, are applicable to the other as well.

5.	**OBSERVATION 4.45:** The IACK DAISY-CHAIN DRIVER is guaranteed this minimum high time on AS* between DTB cycles.
19.	**OBSERVATION 4.46:** The IACK DAISY-CHAIN DRIVER is guaranteed that the AS* will remain low for this minimum time. This time is derived from timing parameters 8, 1 6, and 27 of the INTERRUPTER.
32.	**OBSERVATION 4.47:** The IACK DAISY-CHAIN DRIVER is guaranteed that IACK* (and the slot 1 IACKIN*) has been valid for this minimum time when it detects a falling edge on DSA* .

34.	**RULE 4.46:** IF the IACKIN* line is low when the IACK DAISY-CHAIN DRIVER detects a falling edge on DSA*, THEN it **MUST** drive its IACKOUT* line low, but it **MUST NOT** do so until this time after the falling edge on DSA*. **OBSERVATION 4.48:** The IACK DAISY-CHAIN DRIVER does not drive its IACKOUT* line low every time DSA* goes low. It only does so when the IACK* line is low as well, indicating that an interrupt acknowledge cycle is in progress.
35.	**RULE 4.47:** IF the IACK DAISY-CHAIN DRIVER drives its IACKOUT* line low THEN it **MUST** drive its IACKOUT* high within this time after the rising edge of AS*.
40.	**RULE 4.48:** The IACK DAISY-CHAIN DRIVER **MUST NOT** drive IACKOUT* low until it has been high for this minimum time.
42.	**OBSERVATION 4.49:** IF the IACK DAISY-CHAIN DRIVER drives its IACKOUT* line low within the bus time-out period, THEN this time guarantees that IACK* (and the slot 1 IACKIN*) remains valid for this time.

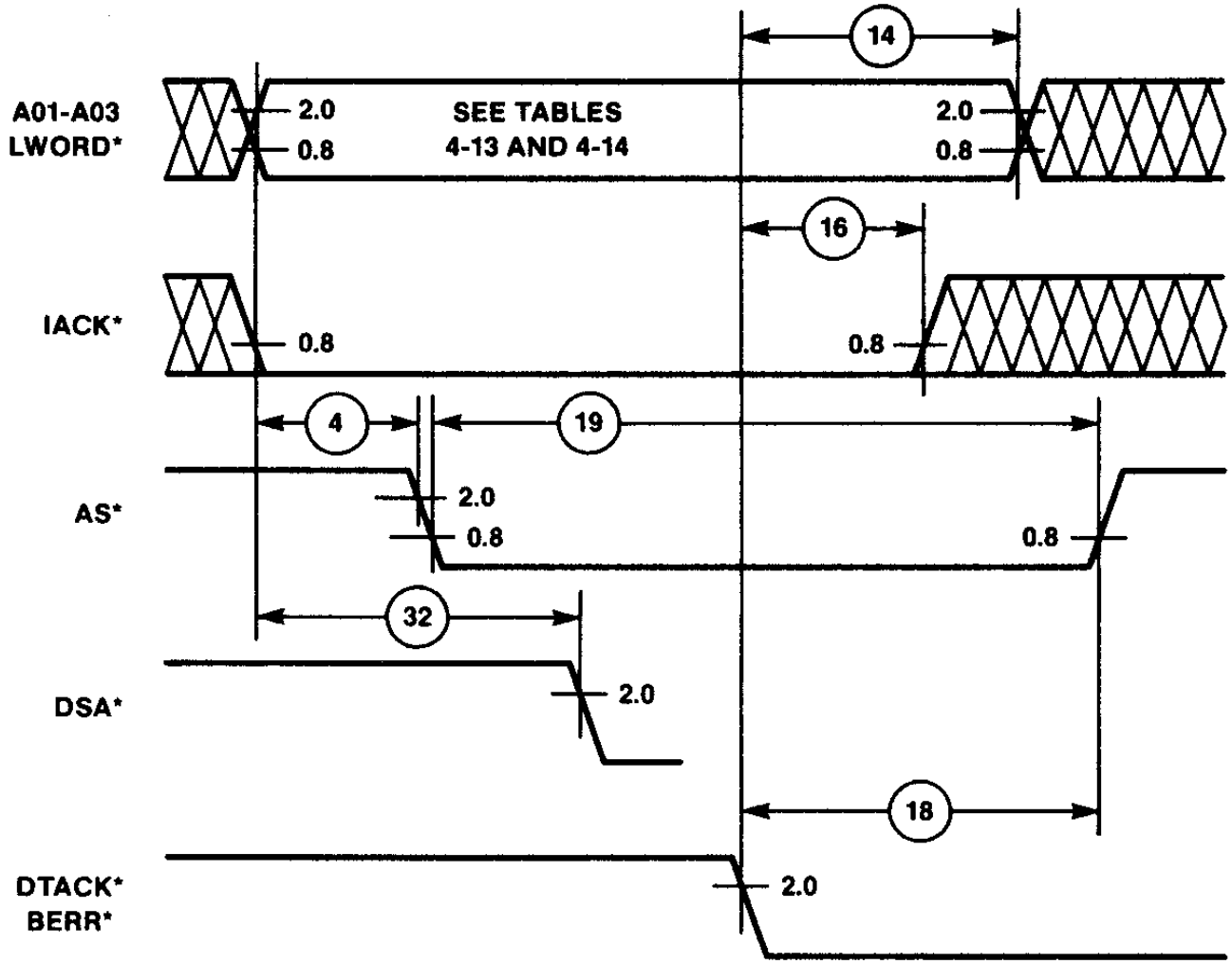


Figure 4-16. INTERRUPT HANDLER And INTERRUPTER
INTERRUPTER Selection Timing
SINGLE, DOUBLE And QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

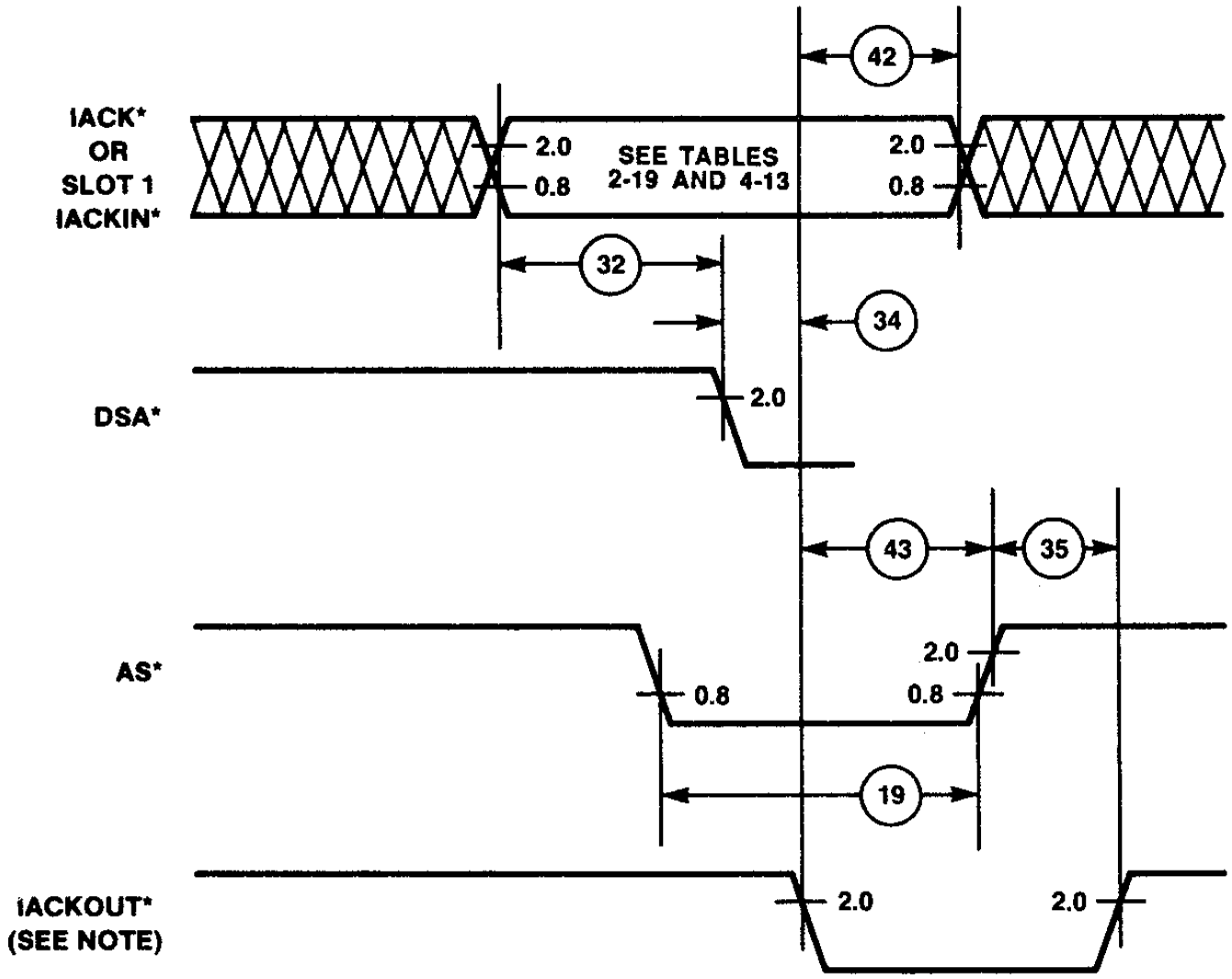


Figure 4-17. IACK DAISY-CHAIN DRIVER
INTERRUPTER Selection Timing
SINGLE, DOUBLE, And QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

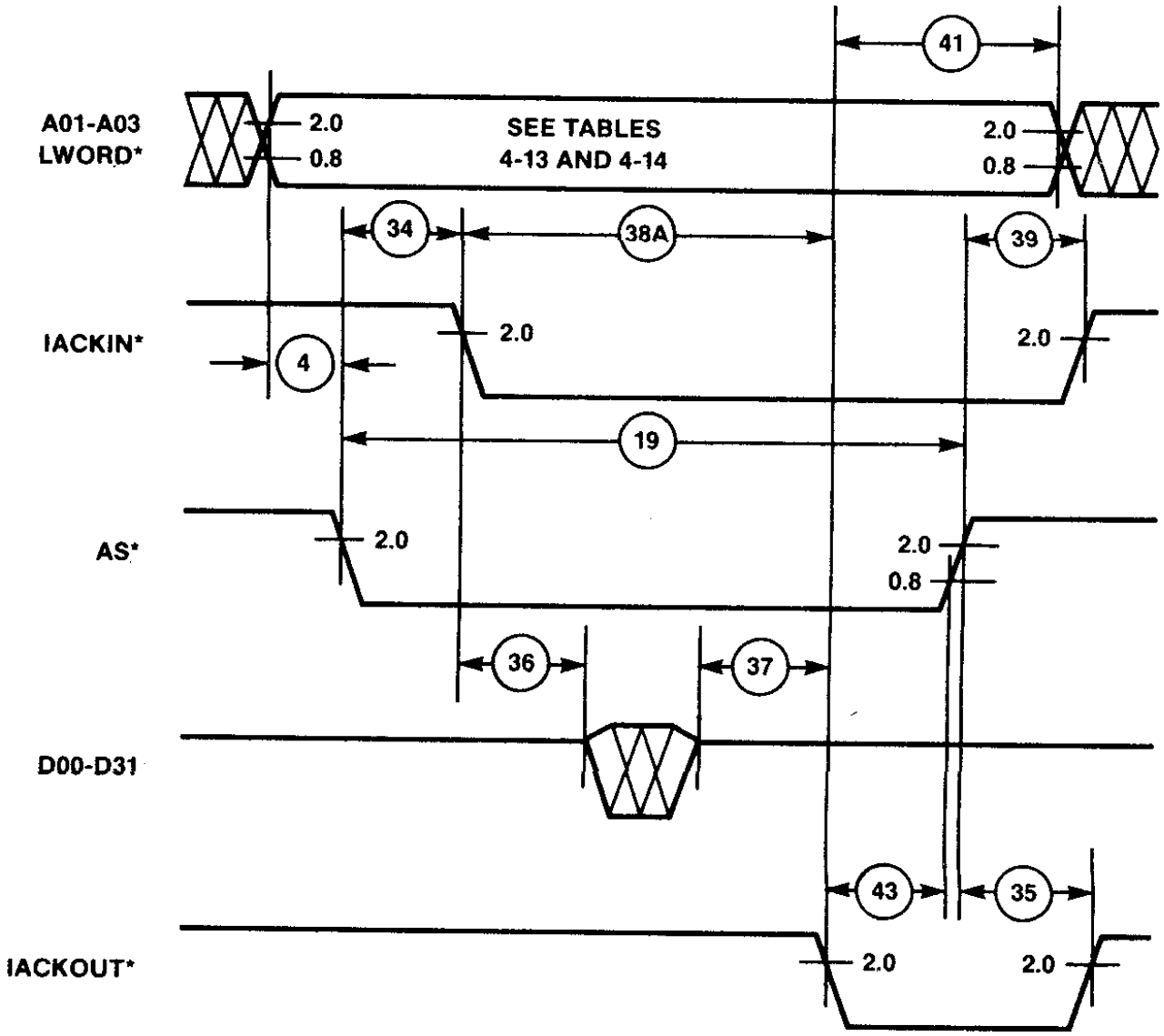


Figure 4-18. Participating INTERRUPTER
INTERRUPTER Selection Timing
SINGLE, DOUBLE, And QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

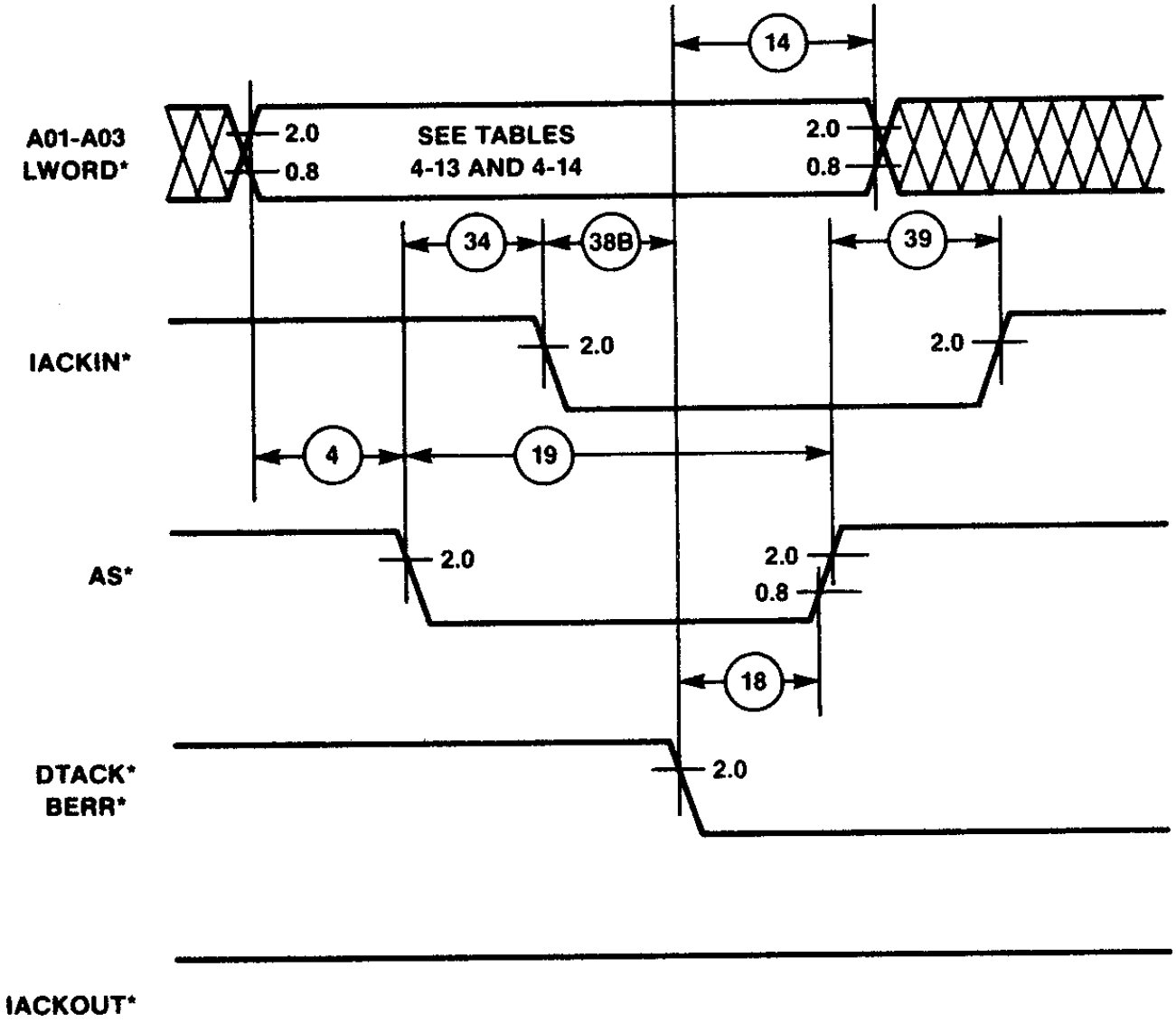


Figure 4-19: Responding INTERRUPTER
INTERRUPTER Selection Timing
SINGLE, DOUBLE, And QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

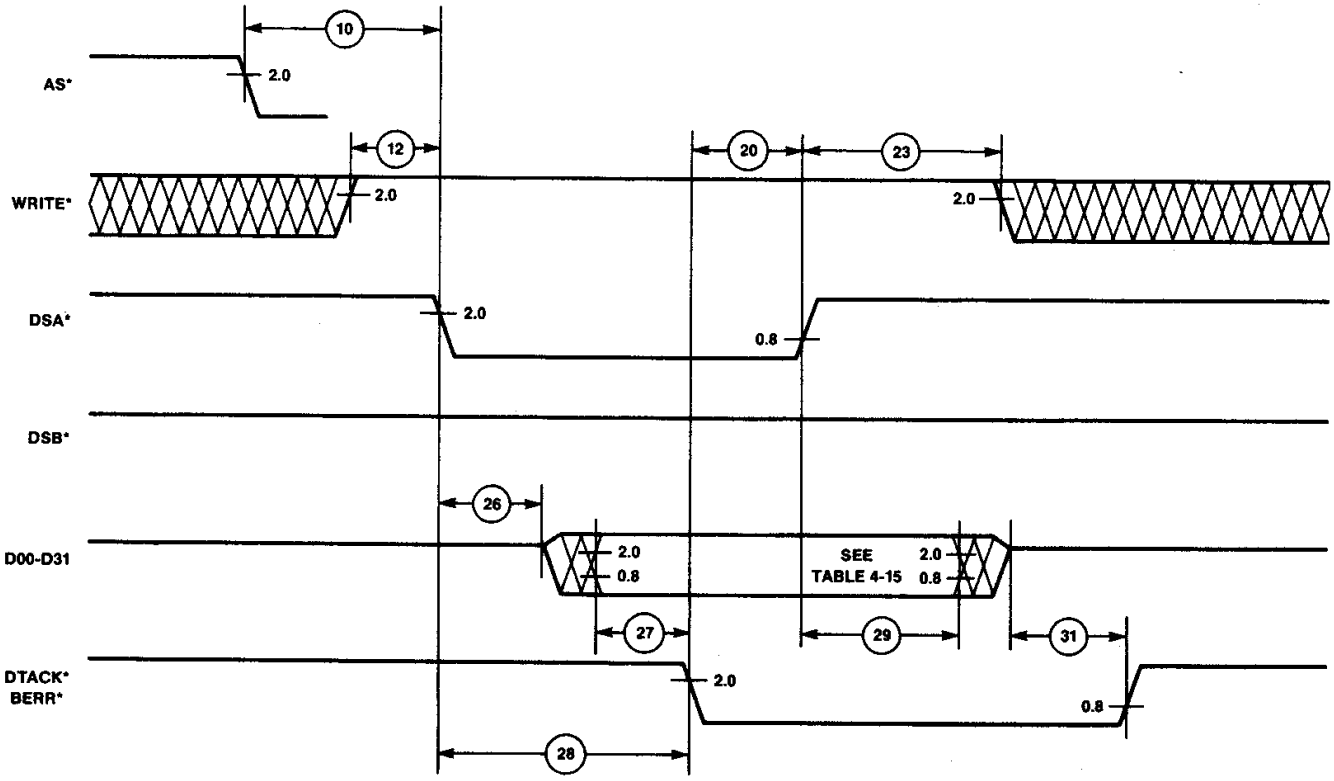


Figure 4-20: INTERRUPT HANDLER
STATUS/ID Transfer Timing
SINGLE BYTE INTERRUPT ACKNOWLEDGE CYCLE

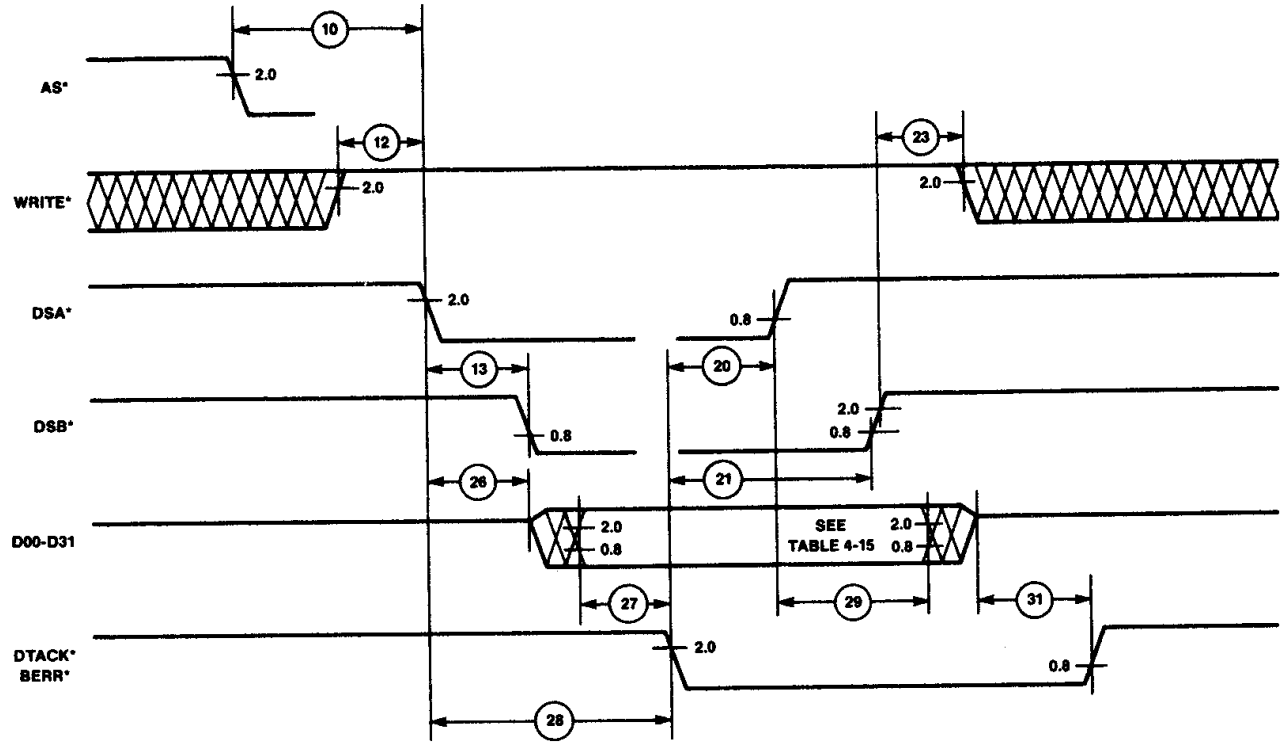


Figure 4-21: INTERRUPT HANDLER

STATUS/ID Transfer Timing
DOUBLE BYTE INTERRUPT ACKNOWLEDGE CYCLE
QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

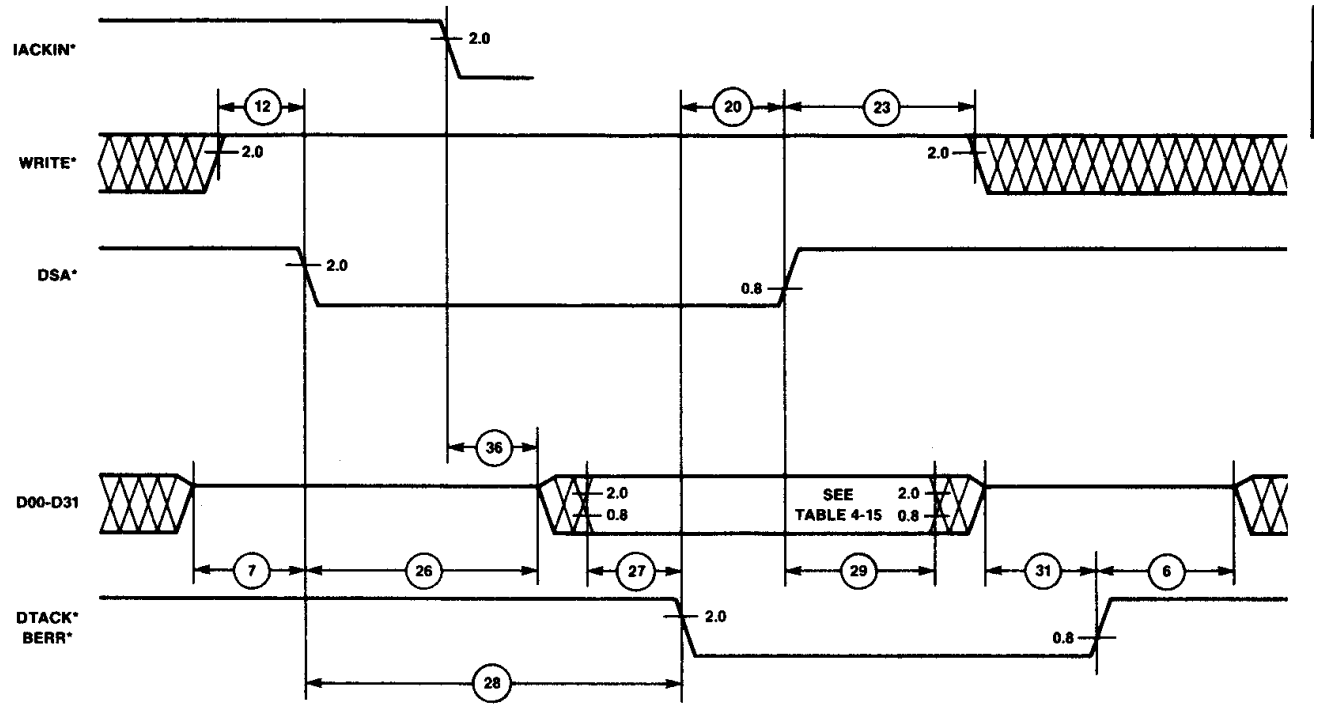


Figure 4-22: Responding INTERRUPTER
STATUS/ID Transfer Timing
SINGLE BYTE INTERRUPT ACKNOWLEDGE CYCLE

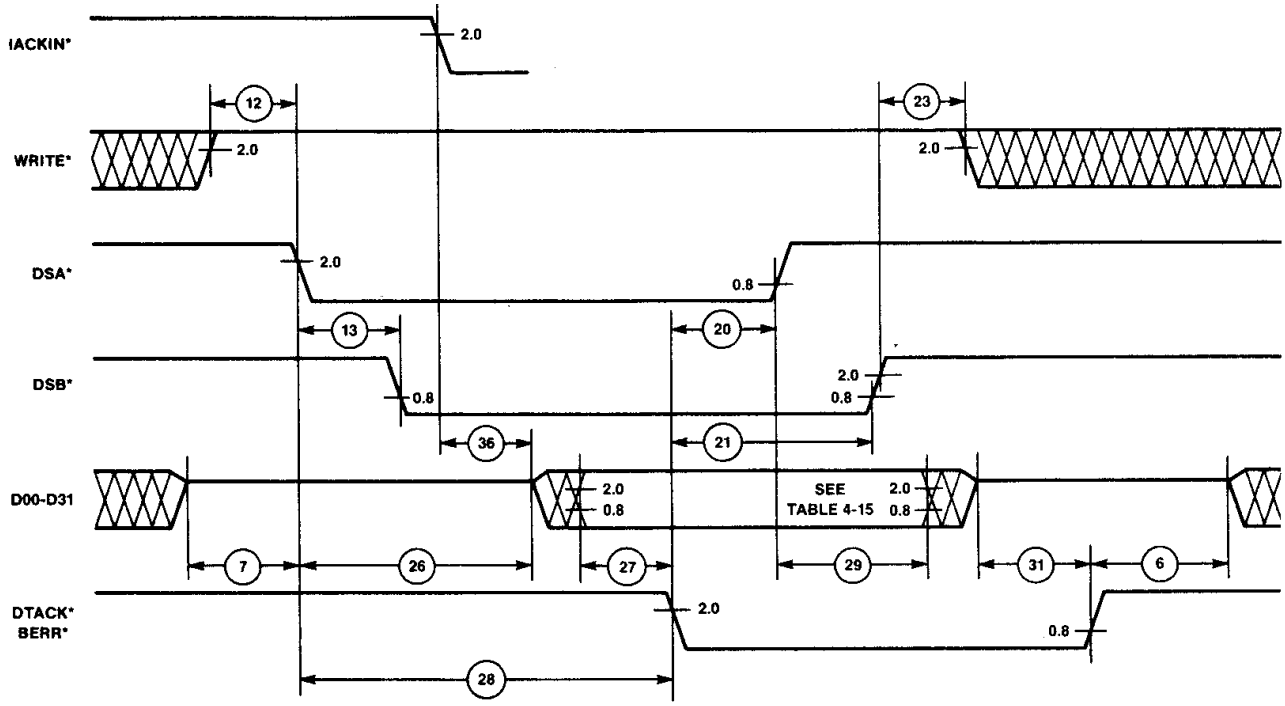


Figure 4-23. Responding INTERRUPTER
STATUS/ID Transfer Timing
DOUBLE BYTE INTERRUPT ACKNOWLEDGE CYCLE
QUAD BYTE INTERRUPT ACKNOWLEDGE CYCLE

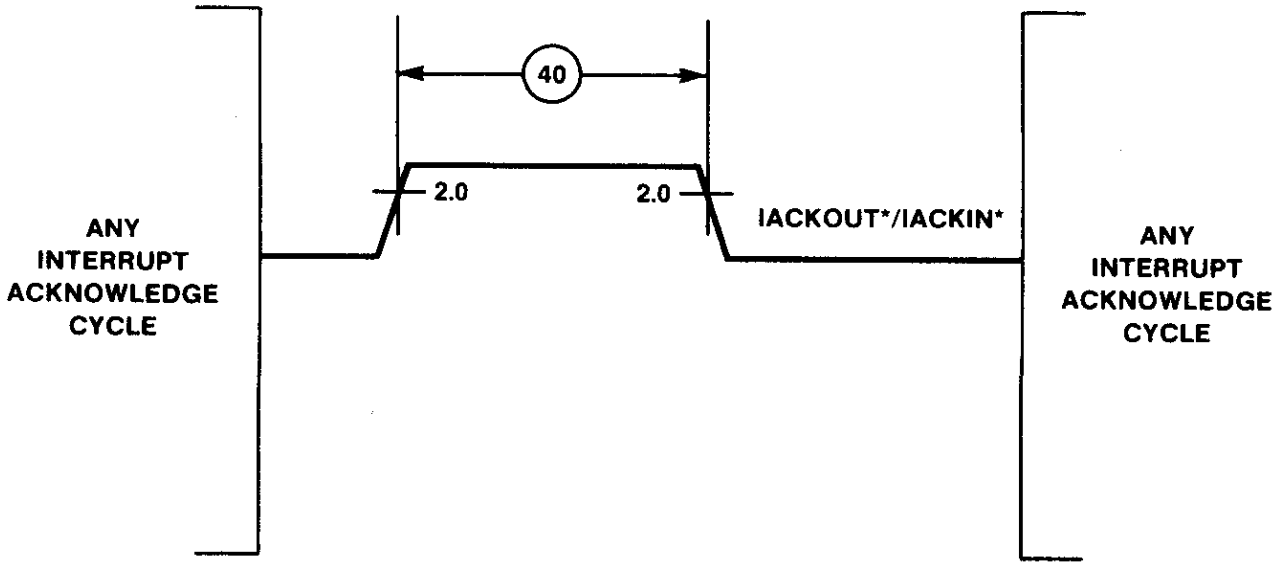


Figure 4-24. ACK DAISY-CHAIN DRIVER, Responding INTERRUPTER,
And Participating INTERRUPTER
IACK Daisy-Chain Inter-Cycle Timing

CHAPTER 5

UTILITY BUS

5.1 INTRODUCTION

This Chapter identifies and defines the signal lines and modules which provide utility functions for the VMEbus. The Utility Bus supplies periodic timing, initialization and diagnostic capability for the VMEbus (See Figure 5-1).

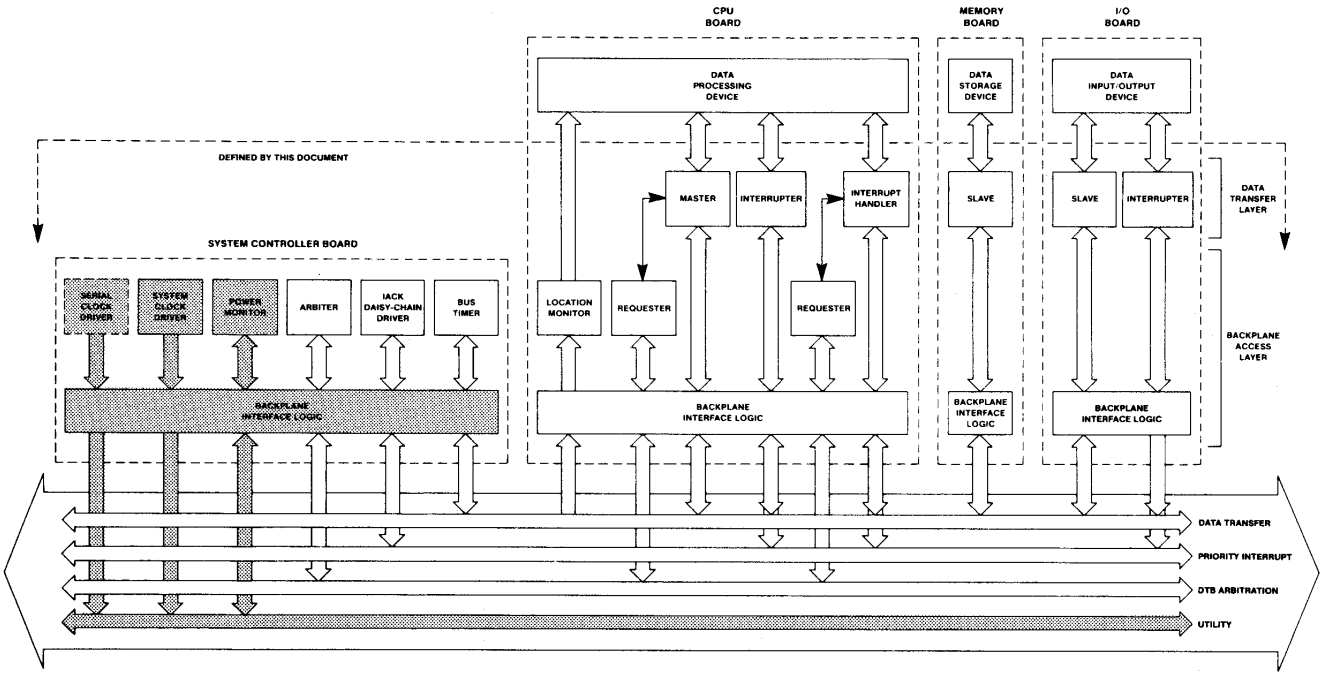


Figure 5-1. Utility Bus Block Diagram

5.2 UTILITY BUS SIGNAL LINES

The Utility Bus signal lines are listed below:

System Clock	(SYSCLK)
Serial Clock	(SERCLK)
Serial Data	(SERDAT*)
AC Fail	(ACFAIL*)
System Reset	(SYSRESET*)
System Failure	(SYSFAIL*)

5.3 UTILITY BUS MODULES

5.3.1 The SYSTEM CLOCK DRIVER

The system clock is an independent, non-gated, fixed frequency, 16 MHz, 50 percent (nominal) duty cycle signal. The SYSCLK driver is located on the system controller are located

in board slot one (see Chapter 1). It provides a known time base that is useful for counting off time delays. Figure 5-2 shows the SYSTEM CLOCK DRIVER timing diagram.

OBSERVATION 5.1:
SYSCLK has no fixed phase relationships with other VMEbus timing.

5.3.2 The SERIAL CLOCK DRIVER

The SERIAL CLOCK DRIVER provides a fixed frequency, special waveform signal used by VMSbus modules that reside on VMEbus boards. Its waveform is specified in the VMSbus specification. For the convenience of designers, the timing parameters in effect at the time that this document has been published are provided in Appendix C.

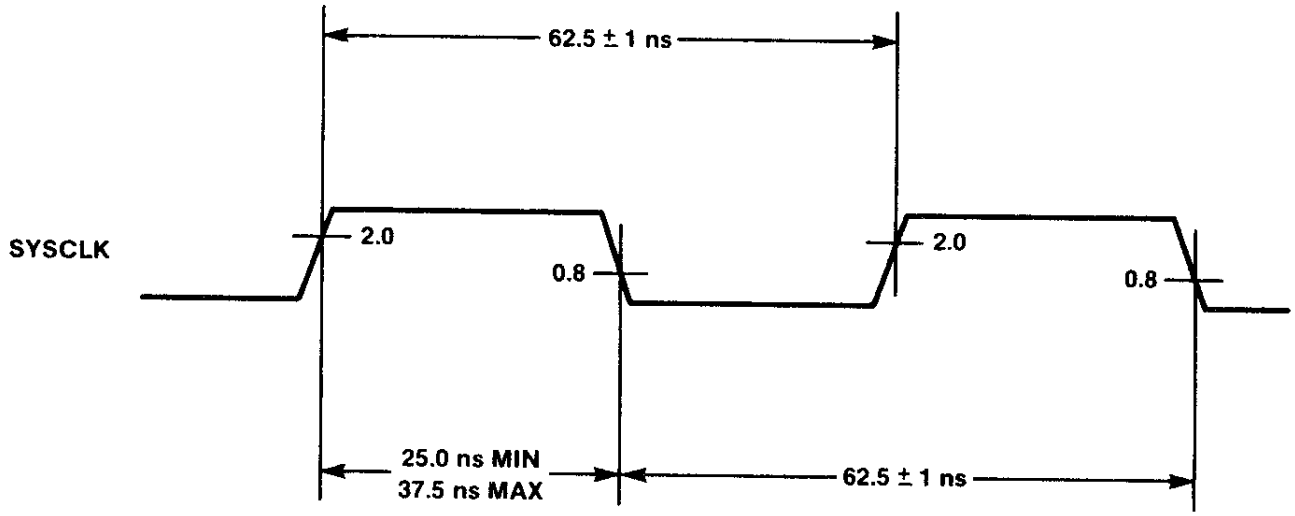


Figure 5-2. SYSTEM CLOCK DRIVER Timing Diagram

5.3.3 The POWER MONITOR

Figure 5-3 is the block diagram for the POWER MONITOR module. This module detects power failures and signals the VMEbus system in time to effect orderly shutdown. When power is then reapplied to the system the POWER MONITOR ensures that all other VMEbus modules are initialized.

The POWER MONITOR might also monitor a manually operated push button and initialize the VMEbus system whenever that button is depressed by the operator.

The ACFAIL* and SYSRESET* transitions, and the point at which the system DC voltages violate specification, have certain timing relationships. These relationships are shown in Figures 5-4 and 5-5.

PERMISSION 5.1:
VMEbus systems **MAY** be built with or without a POWER MONITOR module.

RULE 5.1:
POWER MONITORS **MUST** comply with the timing given in Figures 5-4 and 5-5.

PERMISSION 5.2:
The SYSRESET* line **MAY** be driven low by any VMEbus board to initialize the system from a manual push-button. Where a board drives SYSRESET*, but does not drive ACFAIL*, the timing in Figure 5-4 and Figure 5-5 does not apply.

RULE 5.2:
Whenever any board drives SYSRESET* low, it **MUST** hold SYSRESET* low for a minimum period of 200 milliseconds.

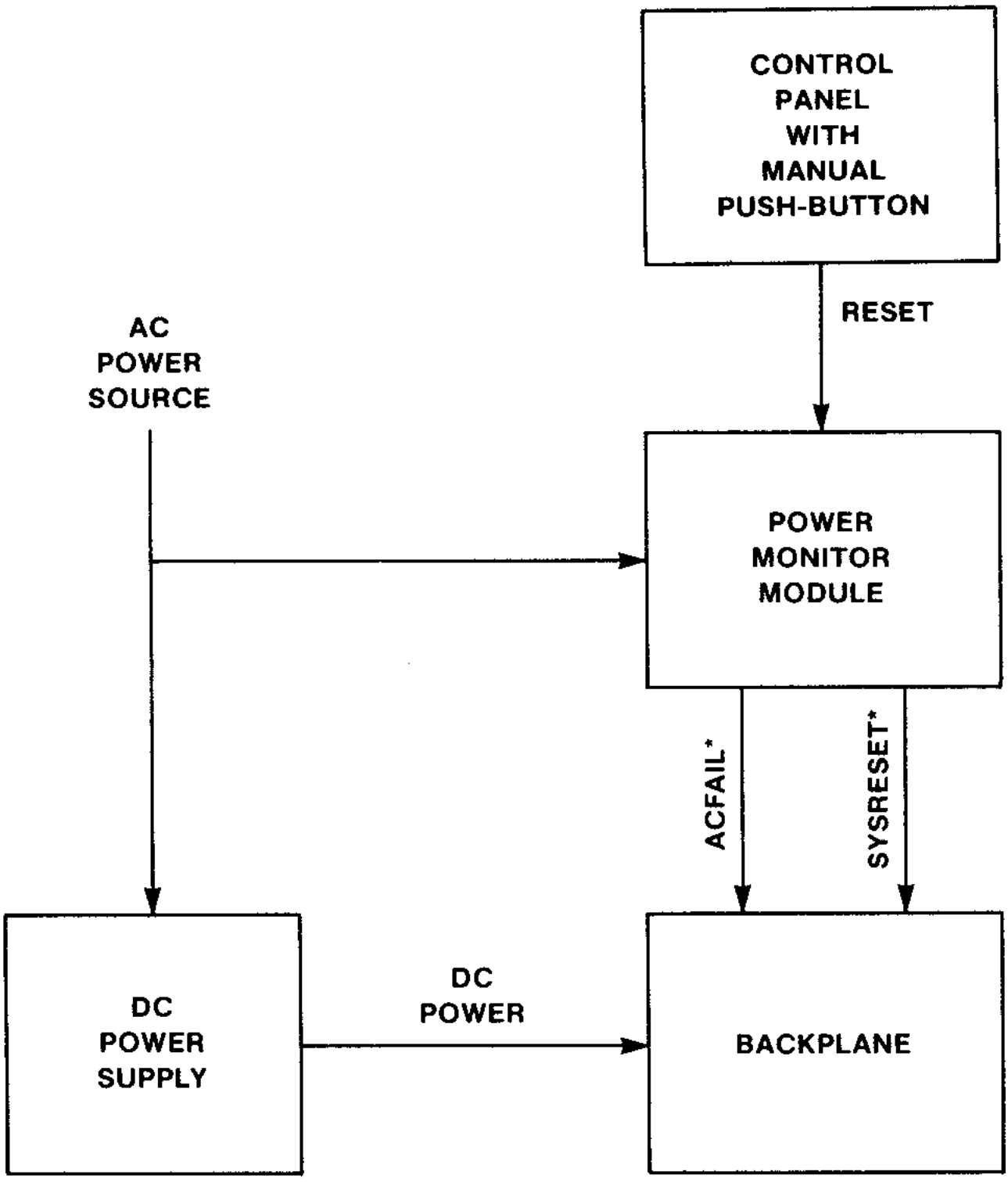


Figure 5-3. Block Diagram Of POWER MONITOR Module

5.4 SYSTEM INITIALIZATION AND DIAGNOSTICS

The VMEbus provides protocols which allow the system to be shut down and powered up in an orderly manner. Two signal lines are used in the power-down and power-up sequence: ACFAIL* and SYSRESET*. Another signal line is used in the power-up sequence: SYSFAIL*.

The following specifies the behavior of VMEbus functional modules during the powerdown sequence:

RECOMMENDATION 5.1:

Design MASTERS so that they do not request the bus for any purpose except powerfail activity, after ACFAIL* has been low for 200 microseconds.

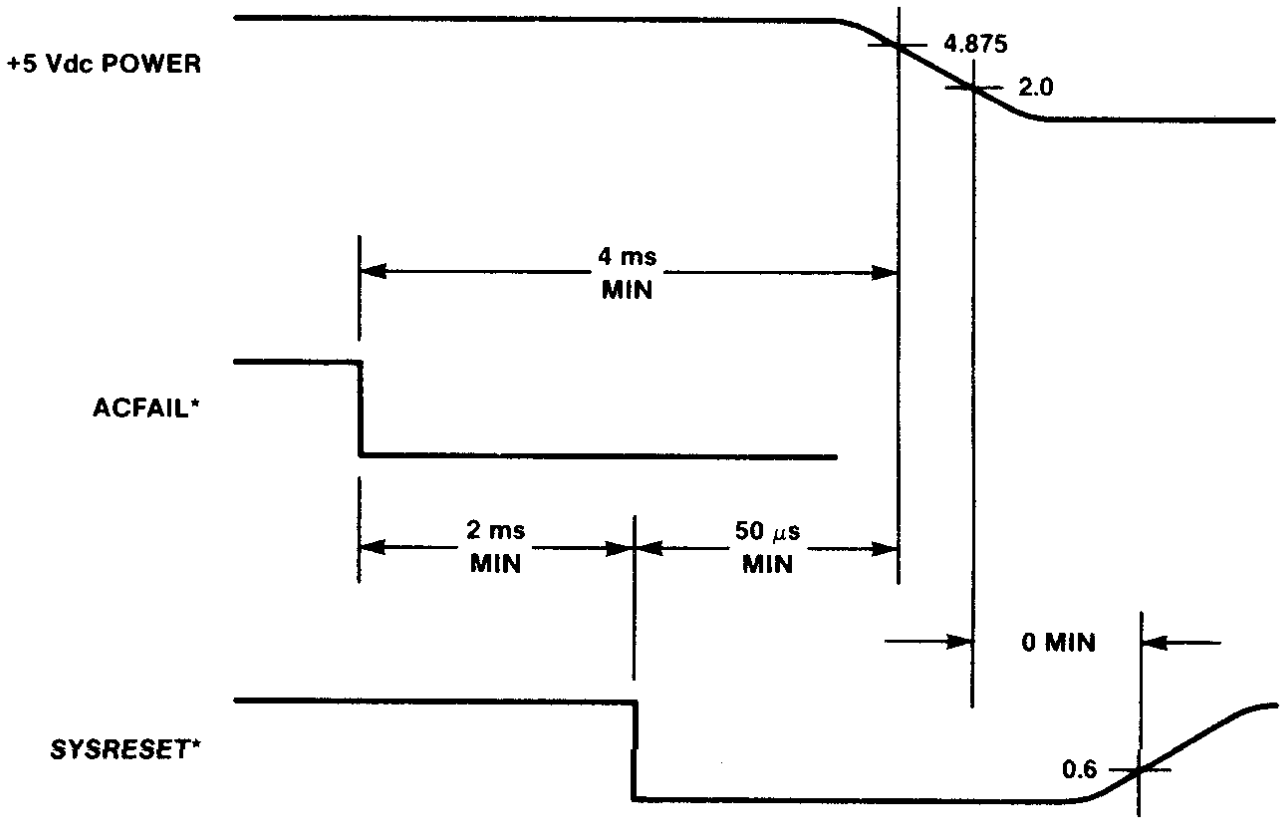


Figure 5-4. POWER MONITOR Power Failure Timing

Figure 5-5. POWER MONITOR System Restart Timing

RECOMMENDATION 5.2:
IF MASTERS and INTERRUPT HANDLERS have a bus request pending prior to detecting ACFAIL* low,
THEN they are to limit their subsequent non-power-fail activity to 200 microseconds.

OBSERVATION 5.2:
Bus accesses required to save and restore system data to VMEbus memory depend upon the application, and are not specified here. (The operating system has to assure that data saved during the shut-down process is restored prior to system operation.) In the case of a multiprocessing system, this might require some processor to processor communication .

System Reset (SYSRESET*) is an open-collector line driven by a POWER MONITOR module, or by any board in response to a push-button switch closure.

OBSERVATION 5.3:
Special circuitry is needed where push-button reset switches are used, to ensure that switch bounce does not cause the board to violate the 200 milliseconds minimum SYSRESET* low time.

RULE 5.3:
The SYSTEM CLOCK driver **MUST** continue to provide the specified SYSCLK waveform regardless of the state of the SYSRESET* line.

PERMISSION 5.4:
When SYSRESET* goes low, any board that requires more than 200 milliseconds to complete its initialization **MAY** turn on its SYSRESET* driver low to maintain SYSRESET* low for the required period.

RULE 5.4:
IF +5 Vdc power source is within its specified range when SYSRESET* goes low ,
THEN functional modules **MUST** satisfy the timing RULES given in Table 5-1 within the specified time after SYSRESET* goes low.

RULE 5.5:
IF SYSRESET* is low when the +5 Vdc enters its specified range,
THEN functional modules **MUST** satisfy the timing RULES given in Table 5-1 and then **MUST** refrain from driving specified lines until SYSRESET* goes high.

RULE 5.6:
After satisfying the RULES in Table 5-1, functional modules **MUST NOT** change the state of their drivers until SYSRESET* goes high, unless the +5 Vdc power source exits its specified range.

RULE 5.7:
IF +5 Vdc power source is within the specified range when SYSRESET* goes low and a MASTER or INTERRUPT HANDLER is driving AS*, DS0*, or DS1* low,
THEN it **MUST** maintain these strobes low long enough to satisfy the minimum low times given in Chapters 2 and 4.

Table 5-1. Module Drive During Power-Up And Power-Down Sequences

MODULE	MUST REFRAIN FROM DRIVING	AFTER SYSRESET* HAS BEEN LOW FOR
MASTERS and INTERRUPT HANDLERS	AS*, DS0*, or DS1* from high to low	5 usec
MASTERS and INTERRUPT	IACK*, LWORD*, AS*, DS0*, DS1*,	20 usec

HANDLERS	AM0-AM5, A01-A31, WRITE*, or D00-D31	
SLAVES and INTERRUPTERS	D00-D31, DTACK*, or BERR	30 usec
INTERRUPTERS	IRQ1*-IRQ7*	30 usec
BUS TIMER	BERR*	30 usec
ARBITER	BG0IN*-BG3IN* from high to low	5 usec
ARBITER	BG0IN*-BG3IN* low	30 usec
REQUESTERS	BBSY*	30 usec

SYSFAIL* is an open collector line that is held low when the system is powered-up and remains low until system self-tests are complete (see Figure 5-6). The following applies:

SUGGESTION 5-1:
On intelligent MASTER boards, include a locally accessible control register bit that is initialized to drive SYSFAIL* low when power is first applied to the board. This permits the board's local intelligence to do a local self-test and release SYSFAIL* only if the self-test passes.

SUGGESTION 5-2:
Design non-intelligent boards with a globally accessible control register bit that is initialized to drive SYSFAIL* low. This allows a MASTER on the VMEbus to run a test on the non-intelligent board and then write to the global control register bit releasing the board's SYSFAIL* driver.

SUGGESTION 5-3:
Where a SYSFAIL* control register bit is included on VMEbus boards, provide a status LED on the board's front panel to indicate the status of the control register bit. Then, if a system failure is indicated by the SYSFAIL* signal line, a visual inspection will help determine which board has failed.

RULE 5.3:
IF SYSFAIL* control register bits are included on VMEbus boards,
THEN they **MUST** drive SYSFAIL* low within 50 milliseconds after SYSRESET* goes low, as shown in Figure 5-6.

PERMISSION 5.4:
A VMEbus board **MAY** also drive SYSFAIL* low at any time during normal operation to indicate that it has detected some kind of failure.

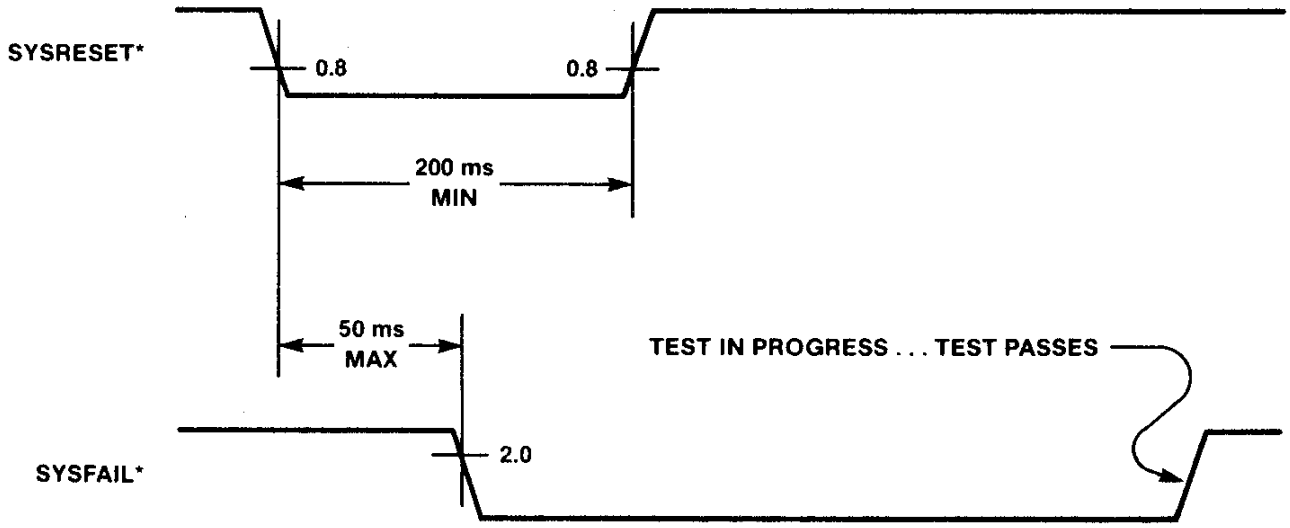


Figure 5-6. SYSRESET* And SYSFAIL* Timing Diagram

5.5 POWER PINS

Figure 5-7 gives the current rating for the VMEbus power pins at various temperatures.

OBSERVATION 5.4:
Some connector pins have a slightly higher contact resistance than others when plugged into the backplane. This produces unbalanced current flow in pins which are paralleled. Suppose that two pins are paralleled and are carrying a total of two amperes of current. If the contact resistance on one is 1 milliohm and the other is 2 milliohms, then one pin will be carrying only 0.67 amperes while the other carries 1.33 amperes!

RULE 5.9:
VMEbus connector pins **MUST** be capable of carrying the currents shown by the solid line in Figure 5-7.

OBSERVATION 5.5:
If one or more power pins fail completely, all of the load current flows through the remaining pins. For example, if half of the pins fail, the remaining pins carry twice the normal current. Depending upon the load current, this might cause damage to these remaining good pins.

SUGGESTION 5-4:
When designing a VMEbus board with a high current load, divide the board's area into zones which are each powered by a separate power grid. Don't connect these grids to each other on the VMEbus board. Instead, connect each to its own VMEbus power pin.

OBSERVATION 5.6:
IF a double height VMEbus board which draws more power than its P1 connector can provide is plugged into a subrack which contains only a J1 backplane,
THEN its P1 power pins will overheat, and might be damaged.

5.6 RESERVED LINE

RULE 5.10:
The VMEbus specification labels one signal line as RESERVED. The RESERVED line is terminated and bused. This line is set aside for future use and **MUST NOT** be used in any VMEbus board designs.

Notes:
1. The dotted line shows the upper limit for current that can be safely drawn per +5 Vdc power pin where two or more pins are connected to a common onboard.
2. The solid line shows the upper limit for current that can be safely drawn per +5 Vdc power pin where each pin is connected to a separate power grid.

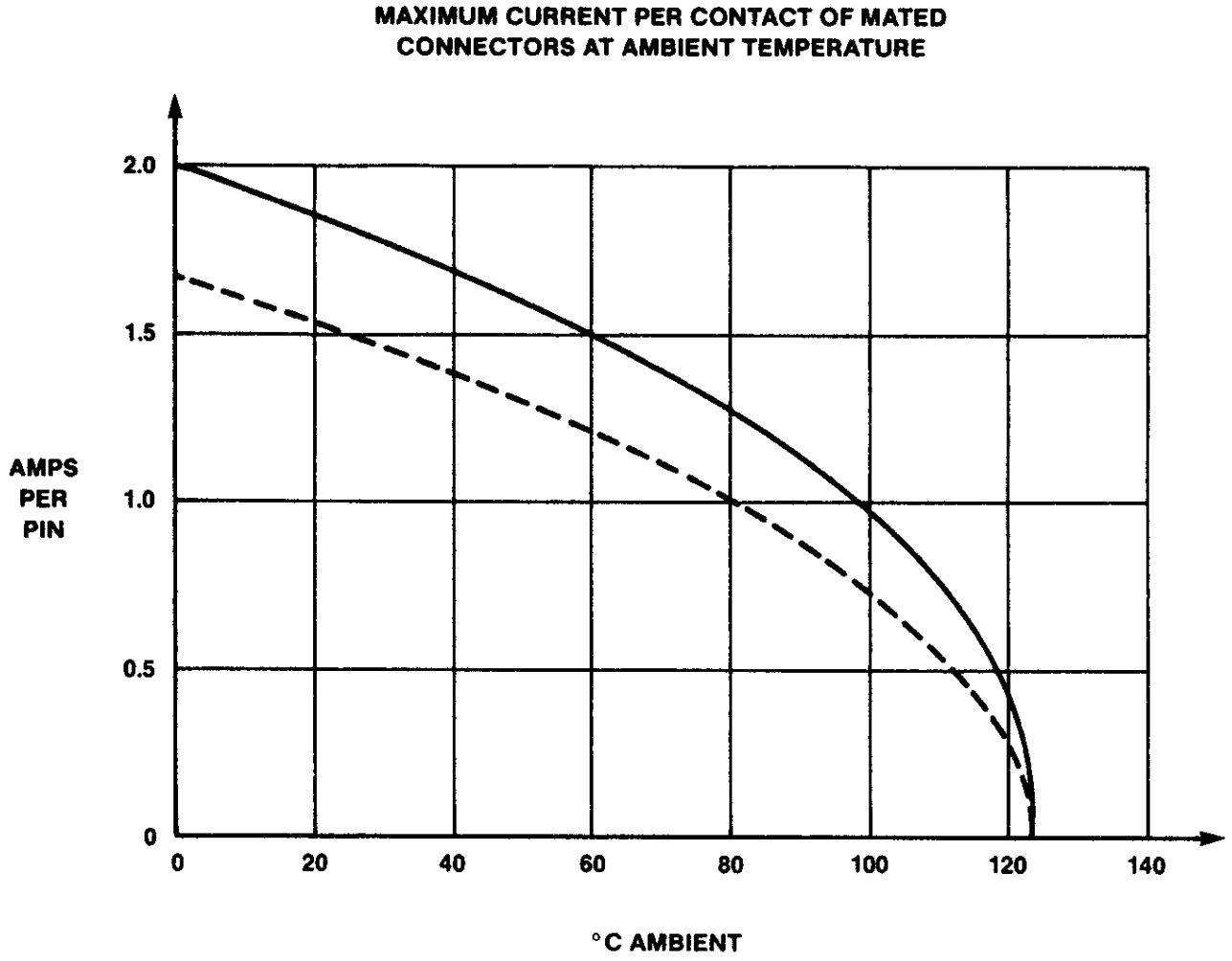


Figure 5-7. Current Rating For Power Pins

CHAPTER 6

ELECTRICAL SPECIFICATIONS

6.1 INTRODUCTION

The transmission of data between VMEbus boards such as processors, memories and I/0 devices takes place over one or two backplanes, depending on the design. The rules in this chapter ensure proper timing, minimal noise and minimal cross talk problems on the backplane signal lines. The design of VMEbus backplanes is governed by the following RULES:

RULE 6.1:
VMEbus backplanes **MUST NOT** have any signal conductors longer than 500 mm (19.68 inches).

RULE 6.2:
VMEbus backplanes **MUST NOT** have more than 21 slots.

RULE 6.3:
For those lines requiring termination (see Section 6.7) the backplane **MUST** provide some means for terminating them at both ends of the signal line.

RULE 6.4:
The backplane **MUST** provide power conductors for distribution of +5V, +5V STDBY, +12V, and -12V to all of the power pins specified in Section 7.6.

RULE 6.5:
The backplane **MUST** provide ground connections to all of the ground pins specified in Section 7.6.

PERMISSION 6.1:
VMEbus signal lines are normally driven by bipolar drivers, but any technology which complies with this specification **MAY** be used.

6.2 POWER DISTRIBUTION

Power in a VMEbus system is distributed on the backplane(s) as regulated direct current (DC) voltages. The available voltages are:

+5 Vdc	This is the main power source for most VMEbus systems. Most of the system circuitry, including TTL logic, MOS microprocessors, and memories, requires this voltage.
+/-12 Vdc	These are often used for powering RS232C drivers. They are also sometimes used for powering MOS and analog devices. In some cases -5 Vdc bias voltage or -5.2 Vdc ECL voltages are also derived from the -1 2 Vdc source using on-board regulators. These supplies

normally don't supply as much power to the VMEbus system as the +5 Vdc source does.

+5 Vdc STDBY — This is used to sustain memory, time-of-day clocks, etc, when the +5 Vdc power is lost.

6.2.1 DC Voltage Specifications

Table 6-1 summarizes the DC voltage specifications. The listed specifications are the maximum allowed variance as measured at the connector pins of any card plugged into the backplane.

RECOMMENDATION 6.1:
Design and connect backplanes so that the power supply sense point is located somewhere near the center of the backplane, and as close as possible to the point where power is introduced into the backplane.

OBSERVATION 6.1:
Placing the power supply sense point near the power input point prevents boards near the power input point from receiving too high a voltage.

Table 6-1. Bus Voltage Specification

MNEMONIC	DESCRIPTION	ALLOWED VARIATION (See OBSERVATION 6.2)	RIPPLE/NOISE BELOW 10 Mhz (Peak-to-Peak)
+5V	+5 Vdc	+0.25V/-0.125V	50 mV
+12V	+1 2 Vdc power	+0.60V / -0.36V	50 mV
-12V	-1 2 Vdc power	-0.60V / +0.36V	50 mV
+5V STDBY	+5 Vdc standby	+0.25V / -0.125V	50 mV
GND	Ground	REFERENCE	

OBSERVATION 6.2:
The non-symmetric variation given in Table 6-1 ensures that the DC power remains within the tolerance required by most ICs despite the typical voltage drops that occur in the power distribution network.

OBSERVATION 6.3:
The power consumed by some systems fluctuates over a wide range during normal system operation. For example, dynamic memory refreshing might cause significant fluctuations if large amounts of memory are refreshed at one time. In this case the response time of the voltage distribution system becomes important.

RECOMMENDATION 6.2:
Use bypass capacitors on VMEbus boards to minimize the effects of power transient.

6.2.2 Pin And Socket Connector Electrical Ratings

RULE 6.6:
The 96 pin connector used by the VMEbus **MUST** provide the following:

Voltage rating:	>= 100 volts DC, isolation pin to pin
Contact resistance:	<= 50 milliohms, at rated current
Insulation resistance:	>= 100 Megohms, pin to pin

6.3 ELECTRICAL SIGNAL CHARACTERISTICS

RULE 6.7:
VMEbus boards **MUST NOT** drive any backplane signal line to a higher steady-state voltage than the highest voltage on any of its +5V power pins, or to a lower steady state voltage than the lowest voltage on any of its GND pins.

RULE 6.8:
VMEbus boards **MUST** use drivers and receivers that meet the following characteristics:

Steady-state driver	low output level	=< 0.6 V
Steady-state receiver	low input level	=< 0.8 V
Steady-state driver	high output level	=> 2.4 V
Steady-state receiver	high input level	=< 2.0 V

Figure 6-1 gives a simple graphic representation of these levels.

VMEbus boards drive the backplane lines with three-state, open collector, and totem pole drivers. Section 6.4 specifies the drive and loading requirements for the various signal lines. Section 6.7 provides a summary, showing which types of drivers are used to drive each signal line.

Figure 6-1. VMEbus Signal Levels

2.94V	High Level	V (Terminator)
2.4 V	----------	Voh min
	//////////	STEADY STATE NOISE MARGIN
2.0 V	----------	Vih min
	Transition Region	
0.8V	----------	Vil max
	//////////	STEADY STATE NOISE MARGIN
0.6 V	----------	Vol max
	Low Level	

RULE 6.9:
When making voltage threshold measurements on a VMEbus board to verify compliance with timing specifications, the ground reference **MUST** be taken from the board's ground pm nearest the signal pin being measured, and the signal voltage **MUST** be measured on the board's connector pin.

6.4 BUS DRIVING AND RECEIVING REQUIREMENTS

This Section defines the driver and receiver specifications for all VMEbus signal lines. Table 6-2 lists all of the signals and shows which of the following subsections discusses it.

6.4.1 Bus Driver Definitions

Totem-pole, three-state, and open-collector drivers are defined as follows:

Totem-pole -- an active driver in both states which sinks current in the low state and sources current in the high state. Totem-pole drivers are used on signals having only a single driver per line (e.g., daisy-chain lines).

Three-state -- similar to a totem-pole driver except that it can go to a high impedance state (drivers turned off) in addition to low and high logic states. Three-state drivers are used for lines that can be driven by several devices at different points on the bus (e.g., address or data lines). Only one of these drivers can be active at any one time.

Open-collector -- sinks current in the low state but sources no significant current in the high state. Terminating resistors on the backplane ensure that the signal line voltage rises to a high level whenever it is not driven low. Open-collector drivers are used for signal lines which can be driven by several devices simultaneously (e.g. interrupt and bus request lines).

Table 6-2. Bus Driving And Receiving Requirements

SIGNAL NAME	SUB-SECTION 6.4.2.x
A01-A31	2
ACFAIL*	5
AM0-AM5	2
AS*	1
BBSY*	5
BCLR*	3
BERR*	5
BG0OUT*-BG3OUT*	4
BR0*-BR3*	5
D00-D31	2
DS0*	1
DS1 *	1
DTACK*	5
IACK*	2, 5
IACKOUT*	4
IRQ1 *-IRQ7*	5
LWORD*	2
SERCLK	3

SYSCLK	3
SYSFAIL*	5
SYSRESET*	5
WRITE*	2

6.4.2 Driving And Loading RULES For All VMEbus Lines

RULE 6.10:
All VMEbus boards **MUST** provide clamping on each VMEbus signal line that they monitor to prevent negative excursions below -1 .5 V.

OBSERVATION 6.4:
Standard 74LSxxx and 74Fxxx devices have internal clamping diodes on their inputs that will satisfy the clamping requirement specified in RULE 6.10.

RULE 6.11
VMEbus receivers **MUST** guarantee detection of a high logic level above a threshold of 2.0 volts, as shown in Figure 6-1.

RULE 6.12:
VMEbus receivers **MUST** guarantee detection of a low logic level below a threshold of 0.8 volts, as shown in Figure 6-1.

PERMISSION 6.2:
A three-state driver **MAY** be used as a totem-pole driver if its output is permanently

6.4.2.1 Driving And Loading RULES For High Current Three-State Lines
(AS*, DS0*, DS1*)

RULE 6.13:
IF a VMEbus board drives AS*, DS0*, or DS1*,
THEN its drivers for these lines **MUST** meet the following specifications:

Low state sink current:	IOL >= 64 mA
Low state voltage:	VOL <= 0.6V at I0L= 64 mA
High state source current:	IOH >=3 mA
High state voltage:	VOH >= 2.4V at I0H = 3 mA
Minimum source current with board pin grounded:	IOS >= 50 mA at 0V
Maximum source current with board pin grounded:	IOS < 225 mA at 0V

RULE 6.1 4:
When drivers are turned off, VMEbus boards **MUST** limit their loading of AS*, DS0* and DS1* to the following values:

Current sourced by board at 0.6 V, including leakage current:	IOZL+IIL <= 450 uA

Current sunk by board at 2.4 V including leakage current:	IOZH+IIH <= 100 uA
Total capacitive load on signal, including signal trace:	CT <= 20 pF

OBSERVATION 6.5:
The source and sink currents listed in RULES 6.13 and 6.14 include both driver and receiver currents sourced and sunk on the board.

SUGGESTION 6.1:
Use 74S241 or 74F241/244 devices to drive the lines AS*, DS0*, and DS1*.
Use 74LS240, 74LS241, or 74LS244 devices to receive the lines AS*, DSO*, and

6.4.2.2 Driving And Loading RULES For Standard Three-State Lines.
(A01-A31 , D00-D31 , AM0-AM5, IACK*, LWORD*, WRITE*)

RULE 6.1 5:
IF a VMEbus board drives the lines A01-A31, D00-D31, AM0-AM5, IACK*, LWORD*, or WRITE*,
THEN its drivers for these lines **MUST** meet the following specifications:

Low state Sink current:	IOL >= 48 mA
Low state voltage:	V0 <= 0.6V at IOL = 48 mA
High state source current:	IOH >= 3 mA
High state voltage:	VOH >= 2.4V at IOH = 3 mA
Minimum source current with board pin grounded:	IOS >= 50 mA at 0V
Maximum source current with board pin grounded:	IOS <= 225 mA at 0V

RULE 6.1.6:
When drivers are turned off, VMEbus boards **MUST** limit their loading of the lines A01-A31 , D00-D31 , AM0-AM5, IACK*, LWORD*, and WRITE* to the following values:

Current sourced by board at 0.6 V, including leakage current:	IOZL+IIL <= 700 uA
Current sunk by board at 2.4 V, including leakage current:	IOZH+IIH <= 150 uA
Total capacitive load on signal, including signal trace:	CT <= 20 pF

OBSERVATION 6.6:
The source and sink currents specified in RULES 6.15 and 6.16 include both driver and receiver currents sourced and sunk on the board.

SUGGESTION 6.2:
Use 74ALS645-1, 74F244 74AS573, or 74AS580 devices to drive the lines A01-A31, D00-D31 , AM0-AM5, IACK*, LWORD*, and WRITE*.
Use 74LS240, 74LS241, or 74LS244 devices to receive the lines A01-A31, D00-D31, AM0-AM5, IACK*, LWORD*, and WRITE*.

Use 74ALS645-1, 74ALS245A-1, 74ALS646-1, or 74ALS648-1 devices to transceive the lines A01-A31 , D00-D31 , AM0-AM5, IACK*, LWORD*, and WRITE*.

6.4.2.3 Driving And Loading RULES For High Current Totem-Pole Lines

(SERCLK, SYSCLK, BCLR*)

RULE 6.1.7:
VMEbus systems **MUST** have no more than one board driving each of the lines SERCLK, SYSCLK, or BCLR*. Its drivers for these lines **MUST** meet the following specifications:

Low state sink current:	I0L >= 64 mA
Low state voltage:	V0L <= 0.6V at I0L = 64 mA
High state source current:	IOH >= 3 mA
High state volta9e:	VOH >= 2.4Vat 10H =3 mA
Minimum source current with board pin grounded:	I0S >= 50 mA at 0 V
Maximum source current with board pin grounded:	I0S <= 255 mA at 0V

RULE 6.1 8:
All VMEbus boards **MUST** limit their loading of the lines SERCLK, SYSCLK, and BCLR* to the following values:

Current sourced by board at 0.6 V, including leakage current:	IOZL+IIL <= 600 uA
Current sunk by board at 2.4 V, including leakage current:	IOZH+IIH <= 50 uA
Total capacitive load on signal including signal trace, for system controllers (which have drivers):	CT <= 20 pF
Total capacitive load on signal, including signal trace, for other boards (which have no drivers)	CT <= 12 pF

OBSERVATION 6.7:
The source and sink currents specified in RULES 6.17 and 6.18 include both driver and receiver currents sourced and sunk on the board.

SUGGESTION 6.3:
Use 74S241 or 74F241/244 devices to drive the lines SERCLK, SYSCLK, and BCLR*.
Use 74LS240, 74LS241 or 74LS244 devices to receive the lines SERCLK, SYSCLK, and BCLR*.

6.4.2.4 Driving And Loading RULES For Standard Totem-Pole Lines

(BG0OUT*-BG30UT*/BG0IN*-BG31N*, IACKOUT*/IACKIN*)

IF a VMEbus board drives the lines BG0OUT*-BG30UT*/BG0IN*-BG31N*, or IACKOUT*/IACKIN* ,
THEN its drivers for these lines **MUST** meet the following specifications:

Low state sink current:	IOL >= 8 mA
Low state voltage:	VOL <= 0.6V at 10L = 8 mA

High state source current:	IOH >= 400 uA
High state voltage:	VOH >= 2.7V at 10H = 400 uA

All VMEbus boards **MUST** limit their loading of each of the lines BG0OUT*-BG30UT*/ BG0IN*-BG31N*, and iACKOUT*/IACKIN* to the following values:

Current sourced by board at 0.6 V, including leakage current:	IOZL+IIL <= 600 uA
Current sunk by board at 2.4 V including leakage current:	IOZH+IIH <= 50 uA
Total capacitive load on signal, including signal trace:	CT <= 20 pF

OBSERVATION 6.8:
The source and sink currents specified in RULES 6.19 and 6.20 include both driver and receiver currents sourced and sunk on the board.
SUGGESTION 6.4:
Use any standard device that meets the specifications above to drive the lines BG0OUT*-BG30UT*/BG0IN*-BG31N*, and IACKOUT*/IACKIN*.
Use 74LS240 74LS241, or 74LS244 devices to receive the lines BG3OUT*BG30UT;/BG0IN*-BG31N*, and IACKOUT*/IACKIN*.

6.4.2.5 Driving And Loading RULES For Open-Collector Lines
(BR0*-BR3*, BBSY*, IRQ1*-IRQ7*, DTACK*, BERR*, SYSFAIL*, SYSRESET*, ACFAIL*, and IACK*)

RULE 6.21:
IF a VMEbus board drives the lines BR0*-BR3*, BBSY*, IRQ1*-IRQ7*, DTACK* BERR*, SYSFAIL*, SYSRESET*, ACFAIL*, or IACK*
THEN its drivers for these lines **MUST** meet the following specifications:

Low state sink current:	IOL >= 48 mA
Low state voltage:	VOL =< 0.6V at I0L = 48 mA

RULE 6.22:
All VMEbus boards **MUST** limit their loading of the lines BR0*-BR3*, BBSY*, IRQ1*- IRQ7*, DTACK*, BERR*, SYSFAIL*, SYSRESET*, ACFAIL*, and IACK* to the following values:

Current sourced by board at 0.6 V including leakage current:	IOZL+IIL <= 400 uA (DTACK* and BERR*) IOZL+IIL <= 600 uA (all others)
Current sunk by board at 2.4 V, including leakage current:	IOZH+IIH <= 50 uA
Total capacitive load on signal,including signal trace:	CT <= 20 pF

OBSERVATION 6.9:
The sink current specified in RULES 6.21 and 6.22 includes both driver and receiver currents sourced and sunk on the board.

SUGGESTION 6.5:

Use 74S38 devices to drive the lines BR0*-BR3*, BBSY*, lRQ1*-lRQ7*, DTACK* BERR*, SYSFAIL*, ACFAIL*, and IACK*.
Use 74LS240, 74LS241, or 74LS244 devices to receive the lines BR0*-BR3*, BBSY* IRQ1*-lRQ7*, DTACK*, BERR*, SYSFAIL*, SYSRESET*, ACFAIL*, and IACK*.

SUGGESTION 6.6:
Since most TTL drivers do not work reliably when the +5 Vdc power source is out of its specified range, drive SYSRESET* on a POWER MONITOR module with a driver built from a discrete high-gain small signal transistor.

6.5 BACKPLANE SIGNAL LINE INTERCONNECTIONS

The VMEbus is a high performance interface system. Its design takes into account transmission line effects on the backplane. The address and data set-up times specified in Chapters 2 and 4 take into account the fact that most drivers available today do not reliably drive backplane signal lines from the low to the high level until there is a reflection from the end of the bus. Although these reflections serve a useful purpose, they cannot be excessive or ringing will result. The following paragraphs specify the backplane characteristics that achieve the desired result.

6.5.1 Termination Networks

RULE 6.23:
Termination networks **MUST** be used on each end of all VMEbus signal lines except the daisy-chain lines.

OBSERVATION 6.10:
The terminations in the VMEbus serve four purposes:

1. They reduce reflections from the ends of the backplanes.
2. They provide a high state pull-up for open-collector drivers.
3. They restore the signal lines to the high level when three-state devices are disabled .
4. They provide a standing current for the driver sink transistor to switch off, causing the signal line to rise more swiftly on positive transitions.

The Thevenin equivalent of the termination is shown in Figure 6-2. The voltage divider also shown provides this termination value.

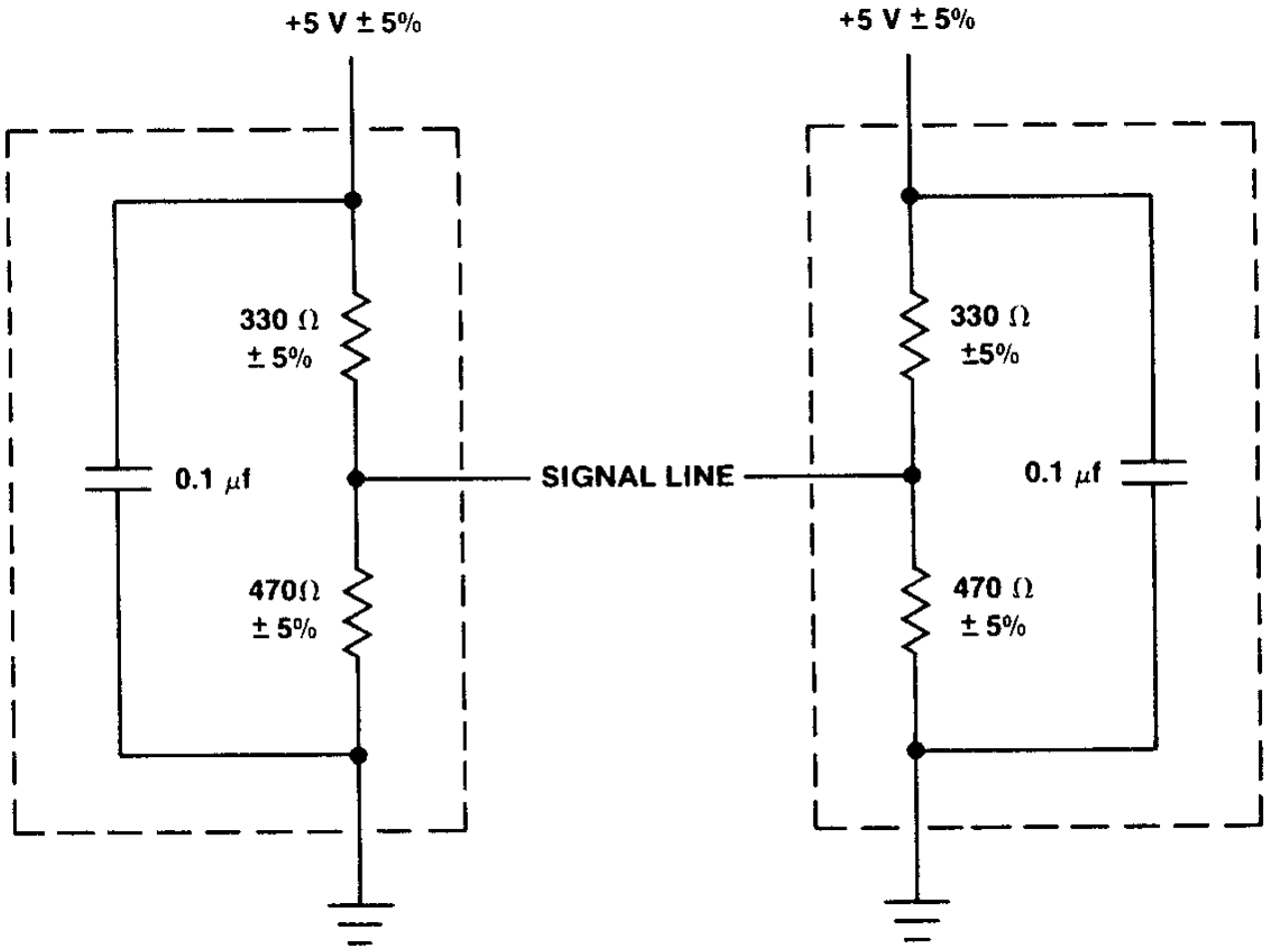
OBSERVATION 6.1 1:

IF a maximum tolerance of +/- 5% is maintained on the resistor values and source voltage used in the resistor network shown in Figure 6-2,
THEN the circuit shown will meet the tolerances shown for the Thevenin equivalent.

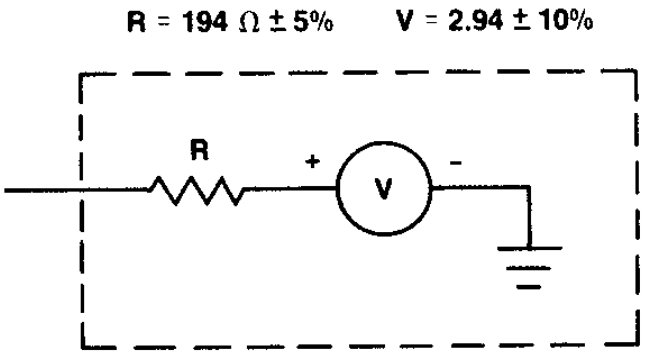
OBSERVATION 6.12:
The resistor network shown in Figure 6-2 presents its Thevenin equivalent impedance only when its +5V source is adequately decoupled to ground by a bypass capacitor.

RECOMMENDATION 6.3:
Provide a bypass capacitor with a value in the range of 0.01 to 0.1 uF as close as possible to the Vcc pin of each resistor termination package.

PERMISSION 6.3:
Any resistor network and voltage source **MAY** be used to provide the termination, as long as they provide the Thevenin equivalent shown in Figure 6-2.



RESISTOR NETWORKS THAT PROVIDE
THE REQUIRED TERMINATIONS



THEVENIN EQUIVALENT
FOR EACH TERMINATOR

Figure 6-2. Standard Bus Termination

6.5.2 Characteristic Impedance

Each signal line in the backplane has an associated characteristic impedance Zo. This characteristic impedance is important because discontinuities in Zo (due to capacitive effect and loads on the bus) and mismatches between Zo and the terminations can cause distortions of signal waveforms.

Figure 6-3 shows a microstrip signal line cross Section which is the normal configuration for a multilayer backplane signal line. The Zo is a function of the width and thickness of the line, the thickness of the dielectric, and its relative dielectric constant. Figure 6-4 shows characteristic impedance versus microstrip line width for common thickness of fiberglass-epoxy board. More information on Microstrip lines can be found in the MECL System Design Handbook, Motorola, 1983.

The terminations on the VMEbus signal lines reduce distortion of their signal waveforms. Although a perfect impedance match (which totally eliminates distortions due to reflections) is not maintained between the termination networks and the signal lines, it is important not to allow too great a mismatch, as might be the case if a signal line's Zo value is too Iow.

RECOMMENDATION 6.4:
When designing a VMEbus backplane, choose a signal line width and board thickness that gives a Zo (as calculated from Figure 6-4) as close as possible to 100 ohms.

The actual characteristic impedance of a backplane signal line is called the effective characteristic impedance (Zo,), and will be lower than Zo, due to the capacitance of plated-through holes and connector pins. This additional capacitance makes Zo, go below 100 ohms. Although plated-through holes are necessary to accommodate connectors, other holes should be kept to a minimum.

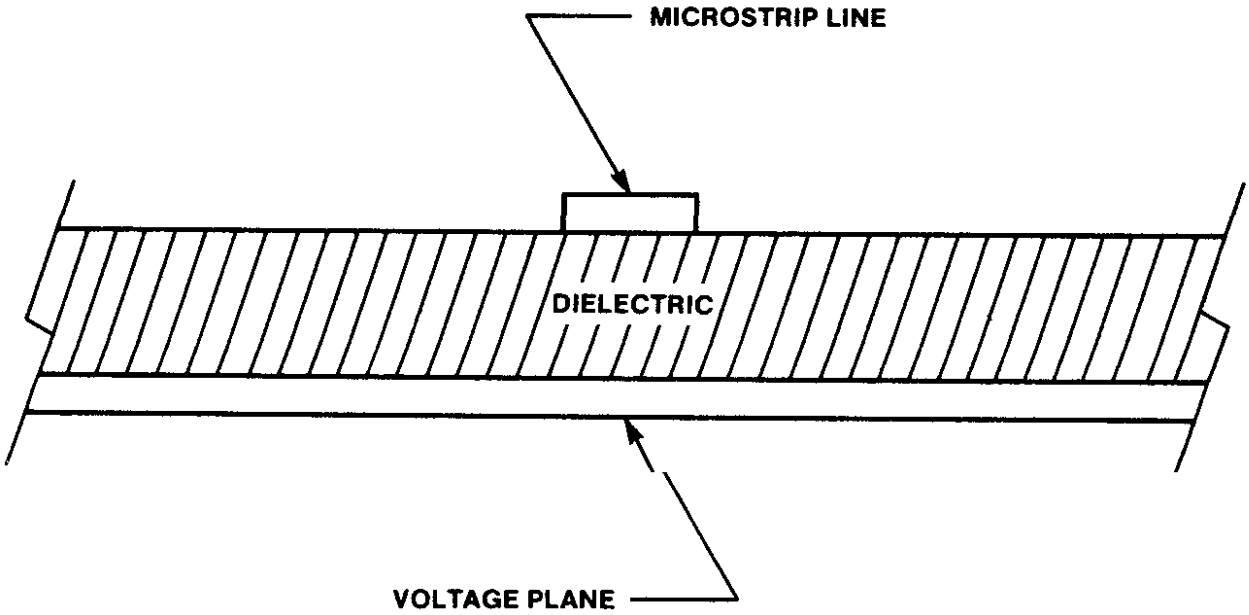


Figure 6-3. Backplane Microstrip Signal Line Cross Section

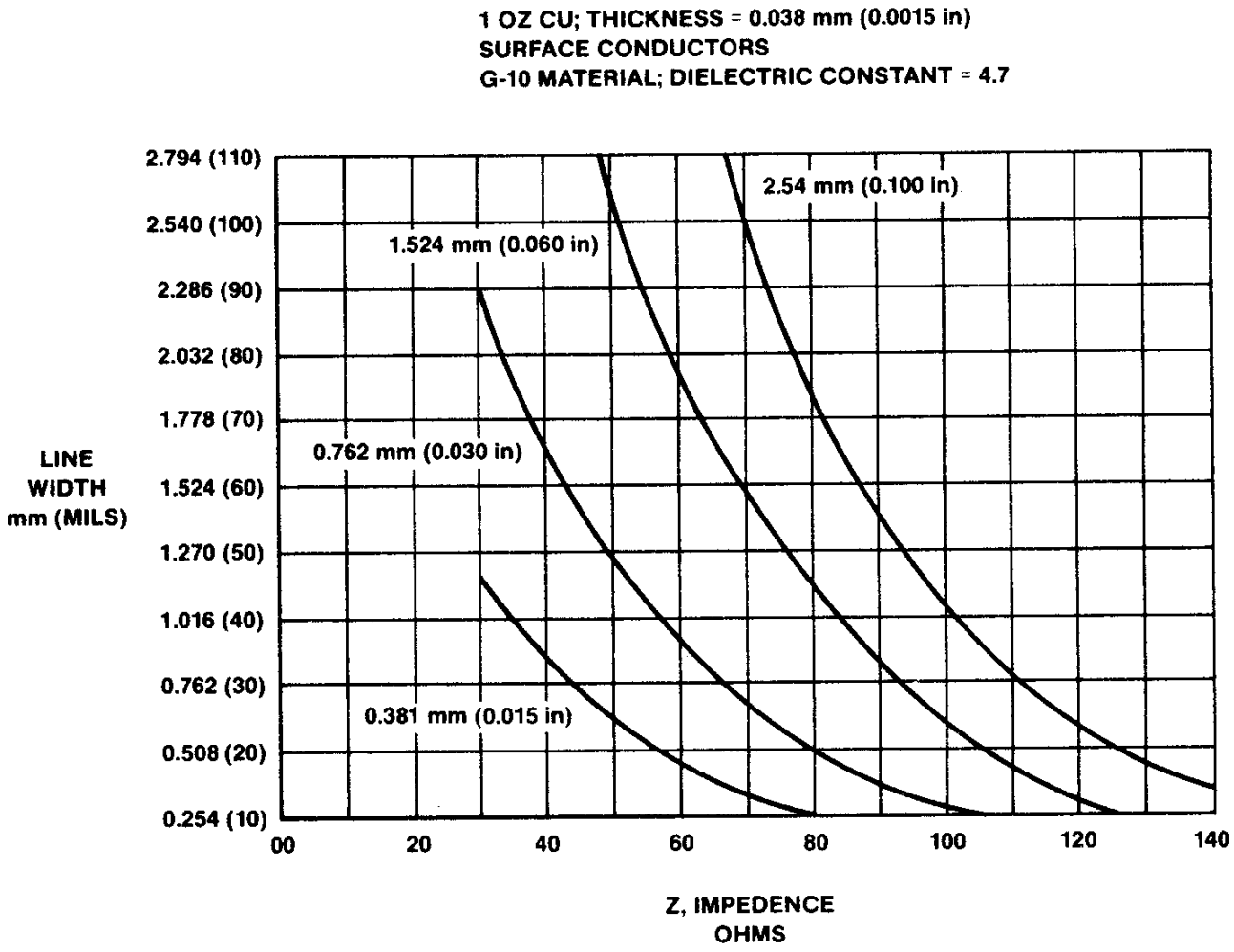


Figure 6-4. Zo Versus Line Width

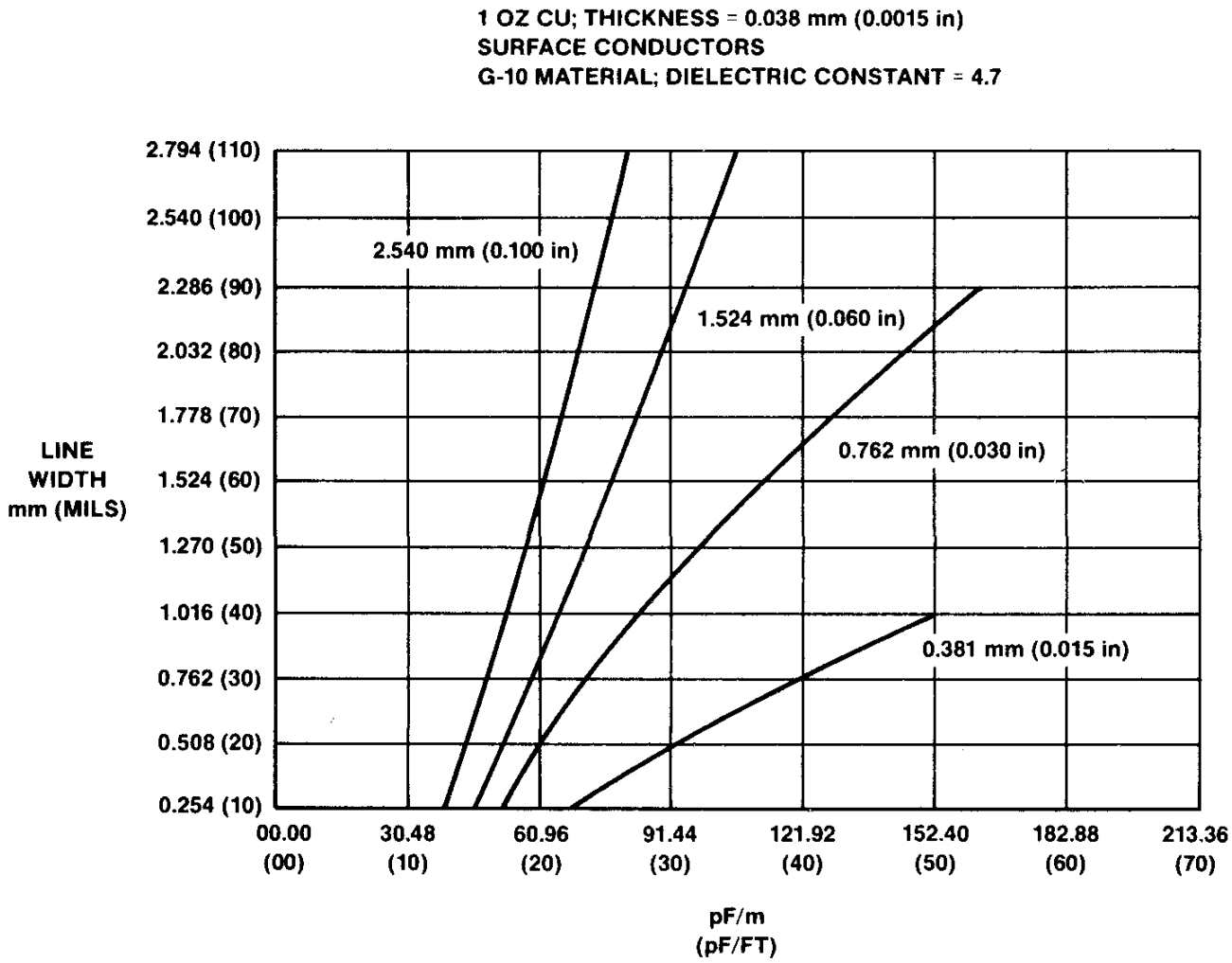


Figure 6-5. Co Versus Line Width

The backplane signal line impedance (without any boards plugged into the backplane) can be calculated with the following equation:

$$Zo = 1 / \text{sqrt}(1 + Cd/Co)$$

where

Zo =	The impedance of the microstrip line, ignoring the loading effects of plug-in pc boards, connectors, and plated-through holes. (See Figure 6-4)
Cd =	The distributed capacitance, per unit of distance, of the plated- through holes, and backplane connectors
Co =	The intrinsic line capacitance, per unit of distance, of the microstrip line, ignoring the loading effects of plug-in pc boards, connectors, and plated-through holes. (See Figure 6-5)
Zo' =	The backplane signal line impedance, including the loading effects of connectors, and plated-through holes but excluding the loading effects of plug-in pc boards.

OBSERVATION 6.13:
Typical Zo, values for a VMEbus backplane with no boards inserted, range from 50 to 60 ohms. If this impedance is 50 ohms, or higher it will provide satisfactory operation.

6.5.3 Additional Information

RULE 6.24:
Circuit traces from the 96 pin connectors to the on-board circuitry **MUST NOT** have a length of greater than 50.8 mm (2 inches).

OBSERVATION 6.14:
IF the trace from the 96 pin connector to on-board circuitry branches,
THEN the length of each branch is added to get the total length specified in RULE 6 .24.

RULE 6.25:
There **MUST NOT** be more than one driver driving the SYSCLK and SERCLK lines.

RULE 6.26:
IF the SYSTEM CLOCK DRIVER and the SERIAL CLOCK DRIVER modules are provided,
THEN they **MUST** be installed in slot 1 of the backplane.

OBSERVATION 6.15:
Locating the SYSTEM CLOCK DRIVER and the SERIAL CLOCK DRIVER on the board in slot 1 minimizes the distortion of their waveforms from the reflections off the ends of the backplane.

SUGGESTION 6.7:
If actual capacitance loading values cannot be obtained from manufacturer specifications sheets, the following values can be used to estimate the total capacitive loading of a VMEbus board:

The typical capacitance of a receiver:	3 - 5 pf
The typical capacitance of a driver:	10 - 12 pf
The typical capacitance of a transceiver:	15 - 18 pf
The typical capacitance of a 50.8 mm (2 inches) PC trace:	2 - 3 pf

OBSERVATION 6.1.6:
Circuit traces which run parallel to each other, such as in a backplane, sometimes induce signal transitions in each other. This phenomenon is commonly known as cross talk. When designing VMEbus backplanes, the spacings of lines and their position relative to ground and power planes have a large effect on the amount of cross talk observed.

SUGGESTION 6.8:
Propagation delays through bus drivers depend on how heavily they are loaded and VMEbus signal lines typically represent heavy loads. This has to be taken into account when calculating worst case timing. If the manufacturer’s data sheet for the driver gives a propagation delay for a 300 pf load, use that to do the worst case calculations. If the only propagation delay values are for a 30 pf load, add 10 ns to the propagation delay and 15 ns to the turn-on delay.

6.6 USER DEFINED SIGNALS

RECOMMENDATION 6.5:
If a board has a 96 pin connector in its P2 location, do not allow any of the P2 pins to be driven to a voltage greater than +15V. This reduces the likelihood of serious damage to the VMEbus system in the event that a signal trace from one of these pins is accidentally shorted to some other signal line.

6.7 SIGNAL LINE DRIVERS AND TERMINATIONS

This Section summarizes the types of drivers which have to be used for each of the signal lines on the VMEbus.
In order to simplify Table 6-3, an abbreviated notation is used to describe the various types of drivers. The notations used are shown below:

Totem-pole (high current)	-	TP HC
Totem-pole (standard)	-	TP STD
Three-state (high current)	-	3 HC
Three-state (standard)	-	3 STD
Open-collector	-	OC

For detailed specifications, see Section 6.4.

Table 6-3. Bus Driver Summary

SIGNAL MNEMONIC	SIGNAL NAME	DRIVER TYPE	BUSED AND TERMINATED?
A01-A31 (31 lines)	ADDRESS BUS	3 STD	YES
ACFAIL*	AC POWER FAILURE	OC	YES
AM0-AM5 (6 lines)	ADDRESS MODIFIER	3 STD	YES
AS*	ADDRESS STROBE	3 HC	YES
BBSY*	BUS BUSY	OC	YES
BCLR*	BUS CLEAR	TP HC	YES
BERR*	BUS ERROR	OC	YES
BG0IN*-BG3IN* BG0OUT*-BG30UT* (Daisy-chain)	BUS GRANT DAISY-CHAIN	TP STD	NO
BR0*-BR3* (4 lines)	BUS REQUEST	OC	YES
D00-D31 (32 lines)	DATA BUS	3 STD	YES
DS0*-DS1* (2 lines)	DATA STROBES	3 HC	YES

DTACK*	DATA TRANSFER ACKNOWLEDGE	OC	YES
IACK*	INTERRUPT ACKNOWLEDGE	3 STD or OC	YES
IACKIN*/IACKOUT* (Daisy-chain)	INTERRUPT ACKNOWLEDGE DAISY-CHAIN	TP STD	NO
IRQ1*-IRQ7* (7 lines)	INTERRUPT REQUEST	OC	YES
LWORD*	LONGWORD	3 STD	YES
RESERVED	RESERVED	-----	YES
SERCLK	SERIAL CLOCK	TP HC	YES
SERDAT*	SERIAL DATA	OC	YES
SYSCLK	SYSTEM CLOCK	TP HC	YES
SYSFAIL*	SYSTEM FAILURE	OC	YES
SYSRESET*	SYSTEM RESET	OC	YES
WRITE*	WRITE	3 STD	YES

CHAPTER 7

MECHANICAL SPECIFICATIONS

7.1 INTRODUCTION

Information is provided in this chapter to ensure that the VMEbus board assemblies, backplanes, subracks, and associated mechanical accessories are dimensionally compatible.

The mechanical dimensions given in this chapter conform to IEC publications 297-1, 297-3 297-3A and 603-2. The electrical characteristics for VMEbus connectors, as specified in Chapters 5 and 6 supersede publication 603-2 where they differ.

IEC publication 603-2 describes a family of connector types which are identified by labels of the form:

603-2-IEC-xxxxxx-xxx

All of the P1/J1 and P2/J2 connectors used on VMEbus boards and backplanes are members of this family. In this chapter, the label 603-2-IEC-xxxxxx-xxx is used when referring to all of these connector types as a group. The label 603-2-IEC-C096Mx-xxx is used when referring to the 96-pin male connector types within this family which are used on VMEbus boards. 603-2-IEC-C096Fx-xxx is used when referring to the 96-pin female connector types which are used on VMEbus backplanes.

Figure 7-1 is a front view of a 19-inch width subrack which shows how single height and double height VMEbus boards can be mixed in a single subrack. Boards are inserted into the subrack from the front, in a vertical plane with the component face of the board on the right.

PERMISSION 7.1:
A VMEbus system **MAY** be composed of single height boards, double height boards, or a mixture of both.

RULE 7.1:
Single height VMEbus subracks **MUST** have a single “J1” backplane.

RULE 7.2:
Double height VMEbus subracks **MUST** have
1) a “J1” backplane mounted in the upper portion of the subrack, OR
2) a "J1" and a "J2" backplane, with the J1 backplane mounted in the upper portion and the J2 backplane mounted in the lower portion.
OR
3) a double height backplane which provides both J1 and J2 connectors.

RULE 7.3:
VMEbus backplanes **MUST NOT** have more than 21 slots.

PERMISSION 7.2:
When using backplanes of fewer than 21 slots, the VMEbus subrack **MAY** be less than the standard 482.6 mm (1 9 inches) rack size.

RULE 7.4:
Except for the rack width, which varies depending on the number of slots it supports all subrack dimensions **MUST** agree with those given in this chapter to ensure mechanical compatibility between the boards and the subrack.

7.2 VMEbus BOARDS

RECOMMENDATION 7.1:
Make VMEbus boards 1.6 +0.2 mm (0.063 +0.008 inch) thick.

OBSERVATION 7.1:
The thickness of VMEbus boards is important because the subrack's guide rails are designed to accommodate boards of this thickness. Thicker boards might not fit into the guides and thinner boards might not be guided properly into the backplane's J1 and J2 connectors.

OBSERVATION 7.2:
The dimensioning of the 603-2-IEC-xxxxxx-xxx connectors provides a certain distance between the connector's mounting face and the center line of each of the connector's pins. This ensures that the P1 and P2 connector pin centers will alien properly with the J1 and J2 connectors of the backplane.

PERMISSION 7.3:
VMEbus boards **MAY** be designed with board thicknesses greater than 1 .6 mm (0.063 inch) if:
1. the thickness of the top and bottom edges of the board, which fit into the guiderails is reduced to 1.6 mm (0.63 inch) for distance of 2.5 mm (0.098 inch) from the top and bottom edges of the board (See Figure 7-2 and Figure 7-3) and
2. the mounting surface provided by the board for the IEC 602-3 connector(s) is 4.07 mm (0.160 inch) from the interboard separation plane. (See Figure 7-5)

Two board sizes are defined as standard VMEbus boards: single height and double height (see Figure 7-2 and Figure 7-3).

7.2.1 Single Height Boards

OBSERVATION 7.3:
A single height VMEbus board is 100mm (3.937 inch) high and 160 mm (6.299 inch) deep with an area of approximately 1 60.0 sq. cm (24.8 sq. inch).

RULE 7.5:
All single height boards **MUST** be designed according to the dimensions given in Figure 7-2.

RULE 7.6:

The hole pattern for the 96 pin 603-2-IEC-CO96Mx-xxx P1 connector **MUST** be as shown in Figure 7-2.

SUGGESTION 7.1:
Use the PC layout grid shown in Figure 7-2.

OBSERVATION 7.4:
Modular front panel hardware is available from several manufacturers which, when mounted on the grid shown in Figure 7-2, properly aligns with the front panel grid shown in Figure 7-7.

PERMISSION 7.4:
Components, other than the 603-2-IEC-CO96Mx-xxx connector, **MAY** be placed in such a way that they do not align with the grid pattern.

7.2.2 Double Height Boards

OBSERVATION 7.5:
A double height VMEbus board is 233.35 mm (9.187 inch) high and 160 mm (6.299 inches) deep with an area of approximately 373.4 sq. cm (57.9 sq. inch).

RULE 7.7:
All double height boards **MUST** be designed according to the dimensions given in Figure 7-3.

RULE 7.8:
The hole pattern for the 96 pin 603-2-IEC-CO96Mx-xxx P1 connector **MUST** be as shown in Figure 7-3.

RULE 7.9:
IF a 96 pin 603-2-IEC-CO96Mx-xxx connector is used for P2,
THEN its hole pattern **MUST** be as shown in Figure 7-3.

OBSERVATION 7.6:
As in the case of the single height boards above, the grid shown in Figure 7-3 properly aligns modular front panel components with the front panel grid shown in Figure 7-8.

PERMISSION 7.5:
Components, other than the 603-2-IEC-xxxxxx-xxx connectors, **MAY** be placed in such a way that they do not align with the grid pattern.

OBSERVATION 7.7:
There is a discontinuity of 1.27 mm (0.05 inch) between the 2.54 (0.10 inch) grid patterns for the upper and lower half of the board.

7.2.3 Board Connectors

The single height board has only one connector on its back edge. It is called the P1 connector. A double height board has either one or two connectors on its back edge. If it has one

connector, that connector is called the P1 connector and is located on the upper half of the back edge. If it has two connectors the upper one is called the P1 connector and the lower one is called the P2 connector.

RULE 7.10:
The P1 and P2 connectors of all VMEbus boards **MUST** meet or exceed the mechanical specifications of a 603-2-IEC-C096Mx-xxx class 2 connector, and **MUST** be mounted as shown in Figure 7-4.

OBSERVATION 7.8:
603-2-IEC class 2 connectors have a minimum mechanical endurance of 400 insertion/extraction cycles.

OBSERVATION 7.9:
The symmetry symbol in the box below each board outline in Figure 7-4 sets an upper limit on how much the connector's center line can be tilted with respect to the board's lower edge. This limit is described in the following rule.

RULE 7.11:
The perpendicular distance (d1) from the board's lower edge to the point A in Figure 7-4 **MUST NOT** differ its perpendicular distance (d2) to point B by more than 0.3 mm (0.012 inch).

RULE 7.12:
IF a VMEbus board is designed to use the center row of P2 for address or data bus expansion, or the board requires more power than the P1 connector can provide ,
THEN a 96 pin 603-2-IEC-C096Mx-xxx P2 connector **MUST** be provided and it **MUST** be mounted as shown in Figure 7-4.

PERMISSION 7.6:
Where neither address nor data bus expansion is required, and where the board does not require more power than the P1 connector can provide, any 603-2-IEC-xxxxxx-xxx connector **MAY** be used for P2 on a double height VMEbus board or the board **MAY** be designed without a P2 connector.

PERMISSION 7.7:
On double height boards, the two outside rows of pins on the P2 connector **MAY** be used to provide user defined connections (See Section 7.6.2).

PERMISSION 7.8:
I/0 cables **MAY** be connected to the front edge of a VMEbus board. No connectors are prescribed for these cable connections.

SUGGESTION 7.2:
Where possible, avoid the use of cable connections to the front edge of VMEbus boards. This makes it much easier to install and remove boards from the subrack during maintenance.

7.2.4 Board Assemblies

The board assembly typically consists of a PC board, as defined above with either one or two 603-2-IEC-xxxxxx-xxx connectors affixed to the board's back edge, electronic components, and an optional front panel with handles. For more detail on the front panels see Section 7.3.

RULE 7.13:
Solder filets, tracking, and components on VMEbus boards **MUST NOT** be closer than 2.5 mm (0.098 inch) from the top and bottom edges of the board to guarantee clearance between them and the board guides. Figure 7-2 and Figure 7-3 show these dimensions.

Figure 7-5 shows a cross sectional view of a PC board, its front panel, its connector and the backplane. The dimensions given are nominal values and are based upon the dimensions given in the other drawings in this chapter.

7.2.5 Board Widths

PERMISSION 7.9:
VMEbus boards **MAY** be designed to occupy more than one slot.

VMEbus boards designed to occupy a single slot of the subrack are called single width boards.

7.2.6 VMEbus Board Warpage, Lead Length and Component Height

During the manufacturing process boards sometimes become warped.

RULE 7.14:
The sum of warpage and component lead length **MUST** be less than or equal to 2.47 mm (0.097 inch) from where the solder side of an ideal (unwarped) board would be, and the sum of component height and warpage (in the other direction) **MUST** be less than or equal to 13.71 mm (0.54 inch) plus an integral multiple (N) of 20.32 mm (0.8 inch), from where the component side of an ideal (unwarped) board would be. (Where N = the number of slots that the board occupies minus 1 .)

OBSERVATION 7.10:
During insertion into the subrack, the component leads of a board might contact the right edge of the front panel to its left. For this reason, the board should be inserted carefully to avoid bending the component leads.

SUGGESTION 7.3:
Where possible, trim the component lead lengths to 1.52 mm (0.06 inch) This makes installation and removal of boards more convenient. Since the maximum allowable warpage is affected by the length of these component leads, trimming them allows boards with larger warpage to meet the specifications.

OBSERVATION 7.1 1:
The dimensions given in SUGGESTION 7.3 ensure that there will be a clearance of at east 2.54 mm (0.1 inch) between components of each board and the component leads of the board

to its right. This space allows adequate air flow and prevents additional warpage and vibration from causing interboard contact.

RULE 7.15:
All VMEbus boards **MUST** be measured, after they are assembled, to ensure that the combination of board warpage, component lead length, and component height do not exceed the specified limits when the board is inserted into a subrack In order to properly make these measurements, the board **MUST** be placed in a subrack (or a similar test fixture) while the measurements are taken.

Figure 7-6 shows a board (single or double height) in a subrack, and shows how the combination of board warpage, component lead length, and component height are to be measured. The interboard separation planes defined in that Figure provide the reference from which the measurements are made.

OBSERVATION 7.12:
A special test fixture, which simulates a subrack, is helpful in speeding up the measurements shown in Figure 7-6.

7.3 FRONT PANELS

This section provides the mechanical specifications for the single and double height board front panels and associated hardware.

PERMISSION 7.10:
VMEbus boards **MAY** be manufactured with or without front panels.

RECOMMENDATION 7.2:
Use front panels and associated hardware to prevent VMEbus boards from vibrating out of the subrack, and to guide airflow through the subrack.

RULE 7.16:
IF front panels are used,
THEN screws **MUST** be provided on those panels to secure the top and bottom of the panels to the subrack, and their screw threads **MUST** be M2.5 X 0.45 pitch (see Figure 7-7).

Figure 7-7 shows a single height, single width front panel. Figure 7-8 shows a double height, single width front panel The grid format on the rear face of these front panels are aligned with the board grids in Figure 7-2 and Figure 7-3 respectively.

Install front panel components such as Light Emitting Diodes (LEDs) and switches so that their centers are aligned with a front panel and point.

7.3.1 Handles

The VMEbus board designer **MAY** design VMEbus boards front pane’s with or without handles.

RECOMMENDATION 7.3
Provide handles to make VMEbus boards easier to remove from the subrack.

OBSERVATION 7.13
The handles available from various manufacturers vary somewhat in overall shape.

Choose handles whose depth and height conform to the dimensions shown in Figure 7-7 and Figure 7-8.

RECOMMENDATION 7.4
When mounting handles on VMEbus board front panels, choose one or more of the locations shown in Figure 7-7, Figure 7-8, Figure 7-11 and figure 7-12.

PERMISSION 7.12
On single height boards, handles **MAY** be installed in any of the following combinations:

1. Top only
2. Bottom only
3. Top and bottom

PERMISSION 7.13:
On double height boards handles **MAY** be installed in any of the following combinations:

1. Top only
2. Middle only
3. Bottom only
4. Top and middle
5. Bottom and middle
6. Top and bottom

OBSERVATION 7.14:
When double and single height boards share the same subrack, handles in the center of the double height front panels align with the handles of single height boards forming an unbroken line and giving a more consistent appearance.

OBSERVATION 7.15:
Removal of double height boards that have both a P1 and a 96-pin P2 connector requires up to 180 N (40.5 lbf) of extraction force. Placing handles at the top and bottom of double height boards makes removal of these boards easiest.

7.3.2 Front Panel Mounting

RECOMMENDATION 7.5:
IF front panels are used

THEN keep the reserved areas shown in Figure 7-9 and Figure 7-10 free of components to allow for installation of front panel mounting brackets. Locate the holes used to mount these brackets as shown in Figure 7-2 and Figure 7-3.

RECOMMENDATION 7.6.
IF front panels are used on double height boards
THEN provide at least one center mounting bracket at one of the two locations shown in Figure 7-10.

RULE 7.17:
IF front panels are used
THEN the test dimension shown in Figure 7-9 and Figure 7-10, from the rear face of the front panel to the rear face of the connector, **MUST** be maintained.

OBSERVATION 7.16:
The test dimensions from the rear face of the front panel to the front face of the backplane guarantee that the P1 and P2 connectors will be fully engaged and that the front panel fasteners will be able to secure the board into the subrack.

7.3.3 Front Panel Dimensions

All dimensioning of the front panel is done from a datum point 0.15 mm (0.006 inch) to the left of its top left corner as viewed from the front.

Single width front panels **MUST** be designed to the dimensions shown in Figure 7-7 and Figure 7-8.

RECOMMENDATION 7.7:
Make front panels 2.5 mm (0.098 inch) nominal thickness.

OBSERVATION 7.17:
The 20.02 mm (0.788 inch) width of the single slot front panels is 0.30 mm (0.012 inch) narrower than the 20.32 mm (0.8 inch) slot spacing. This prevents mechanical interference between adjacent front panels due to tolerances in the board assemblies and subracks.

RULE 7.19:
IF a board occupies more than one slot and has a front panel,
THEN the width of that front panel **MUST** be 20.02 mm (0.788 inch) plus an integral multiple (N) of 20.32 mm (0.8 inch), where N is the number of slots that the board occupies.

RULE 7.20:
Single slot front panels **MUST** be equipped with one fastener at the top and another at the bottom, located as shown in Figure 7-7 and Figure 7-8.

7.3.4 Filler Panels

Filler panels are sometimes used where the backplane positioning leaves a gap on I the left or right end of the subrack's front panel or where there are empty slots. These filler panels require no mounting brackets because they are not attached to printed circuit boards. They are secured to the subrack by screws or quarter-turn fasteners on their top and bottom ends like the front panels of VMEbus boards.

Filler panels **MUST** be designed to conform to the dimensions given in Figure 7-11 and Figure 7-12.

Single slot filler panels **MUST** be equipped with one fastener at the top and another at the bottom, located as shown in Figure 7-11 and Figure 7-12.

RECOMMENDATION 7.8:
Use filler panels on VMEbus based systems to maintain proper air flow within the subrack and to improve the appearance of the assembled VMEbus system.

SUGGESTION 7.6:
Equip filler panels with handles. This will give a more consistent appearance when they are installed in the same subrack with VMEbus boards that have front panels and handles.

SUGGESTION 7.7:
Provide additional mounting holes on filler panels wider than 101.60 mm (4.0 inch) to provide better attachment to the subrack.

7.3.5 Board Ejectors / Injectors

OBSERVATION 7.18:
Several vendors offer different types of board ejectors/injectors that make insertion and removal of VMEbus boards easier. Unfortunately, ejectors/injectors designed for use with one vendor's products do not always work with other vendor's products. Because no agreement has been reached between the various manufacturers, the VMEbus specification does not specify any as the preferred type.

OBSERVATION 7.19:
The insertion force for a single 603-2-IEC-CO96Mx-xxx connector can be as high as 90 N (20.23 lbf).

PERMISSION 7.14:
VMEbus boards **MAY** be equipped with any type of ejectors, injectors, or retainers as long as they do not make the products they are used on incompatible with boards or subracks designed to the VMEbus specifications.

7.4 BACKPLANES

The primary backplane is designated as the J1 backplane. In some cases this is the only backplane in the VMEbus system. When a double height subrack is used, this backplane is mounted in the upper portion of that subrack. When the expanded VMEbus is used, a second

backplane, designated the J2 backplane, is installed below the J1 backplane in the lower portion of the subrack. This J2 backplane buses only the center row (row b) of the P2 connector pins, allowing the two outer rows (rows a and c) to be used to implement the VMXbus or for any other user defined functions.

Board slots are designated 1, 2, 3, ... 21 , with the slot numbering starting at the left end of the subrack, as viewed from the front. The daisy-chain propagation starts with slot 1 and goes to 21.

RULE 7.23:
J1 VMEbus backplanes **MUST** bus all signals in all slots except for the daisy-chained signals. (See Section 7.6.1)

RULE 7.24:
When a J2 backplane is used to provide for 32 bit wide address and data transfers, it **MUST** bus all pins of the center row (row b) of those slots for which it has connectors. (See Section 7.6.2)

RULE 7.25:
The 96 pin 603-2-IEC-CO96Mx-xxx connectors **MUST** be used on all J1 VMEbus backplanes and on all J2 expansion backplanes.

RULE 7.26:
All VMEbus J1 backplanes **MUST** have some provision for jumpering the interrupt acknowledge and bus grant daisy-chains when boards are not plugged into a slot.

SUGGESTION 7.8:
To provide for the Jumpering of the J1 backplane daisy-chains, use 603-2-IECCO96Fx-xxx connectors that have wire-wrap pins.

SUGGESTION 7.9:
IF J1 connectors without wire-wrap pins are used,
THEN place all of the jumper pins for daisy-chains next to the J1 connector that they are jumpering.

PERMISSION 7.1 5:
As long as all other requirements are met, and proper spacing is maintained between the J1 and J2 connectors, the J1 and J2 backplanes **MAY** be designed as a single PC board.

SUGGESTION 7.10:
Use 603-2-IEC-CO96Fx-xxx connectors with wire-wrap pins for the connectors on J2 backplanes. This allows attachment of ribbon cables and secondary backplanes to those pins.

7.4.1 Backplane Dimensional Requirements

Figure 7-13 depicts a 21 slot backplane. Figure 7-14 shows where optional threaded studs can be provided to connect DC power cables to the backplane.

PERMISSION 7.16:
VMEbus backplanes **MAY** be designed with up to 21 slots.

RECOMMENDATION 7.9:
When designing a backplane with fewer than 21 slots, make the width

(N x 20.32mm) - 1.44mm, +0/-0.3 mm

where N is the number of slots. This allows two backplanes to be installed next to each other without wasting a slot position in the subrack.

RECOMMENDATION 7.10:

Do not design backplanes with widths greater than 425.28 mm (1 6.743 inch). Backplanes longer than this might not fit into widely available subracks.

PERMISSION 7.17:
VMEbus backplanes **MAY** be designed with or without threaded studs for power cable connections to the backplane. (See Figure 7-13 and Figure 7-14).

RULE 7.27:
Except for the width dimension which varies with the number of slots, and the optional threaded studs, VMEbus backplanes **MUST** be designed to the dimensions given in Figure 7-13 and Figure 7-14.

OBSERVATION 7.20:
The backplane dimensions shown in Figure 7-14 repeat every 20.32 mm (0.8 inch).

7.4.2 Signal LIne Termination Networks

RULE 7. 28:
VMEbus backplanes **MUST** either have built-in terminations or they **MUST** provide some way to connect plug-on terminator boards for terminating all of the signal lines indicated in Section 7.6.1 and Section 7.6.2.

PERMISSION 7.1 8:
The termination circuitry **MAY** either be built onto the ends of the backplane or it **MAY** be provided by separate modules that plug onto either the front or the back of the backplane at the two end slots.

RULE 7.29:
Backplane signal trace lengths, including any plug-on terminator boards, **MUST NOT** exceed 508 mm (20.0 inches).

OBSERVATION 7.21:

The entire board width of a 21 slot VMEbus backplane is required to accommodate the 603-2-IEC-xxxxxx-xxx connectors. Separate plug-on terminator modules are typically needed for such a backplane.

7.5 ASSEMBLY OF VMEbus SUBRACKS

This section shows how VMEbus subracks are assembled. All horizontal dimensions are from the left hand edge of the front opening of the subrack.

7.5.1 Subracks And Slot Widths

Figure 7-15 shows a typical double height 21 -slot subrack.

PERMISSION 7.19:
Double height subracks **MAY** be used to house double height boards or they **MAY** be sub-divided by the installation of a bracket, into two single height sections, one above the other.

OBSERVATION 7.22:
The bracket allowed in PERMISSI0N 7.19 provides two board guides: the lower guide for the board above it and the upper guide for the board below it.

SUGGESTION 7.11:
Where possible, install the left-most (slot 1) board guides so that their center line is 3.27 mm (0.129 inches) from the left end of the rack opening. This provides sufficient clearance for the component leads of the board installed in that slot, and doesn't waste any horizontal space. (If this board guide is installed farther to the right there will not be room for 21 slots.)

7.5.2 Subrack Dimensions

RULE 7. 30:
All double height VMEbus subracks **MUST** meet all of the dimensional requirements given in Figure 7-1 5, except for the width dimension, which varies according to how many slots are in the subrack.

RULE 7.31:
All single height VMEbus subracks **MUST** meet all of the dimensional requirements given in Figure 7-15, except for the width dimension, which varies according to how many slots are in the subrack, and the vertical distance between the lower and upper board guides, which is 1 00.2 mm, +0.4/-0.0 mm instead of 233.55 mm.

OBSERVATION 7.23:
The dimension from the front panel mounting surface to the front face of the backplane is particularly critical, since it guarantees correct connector engagement.

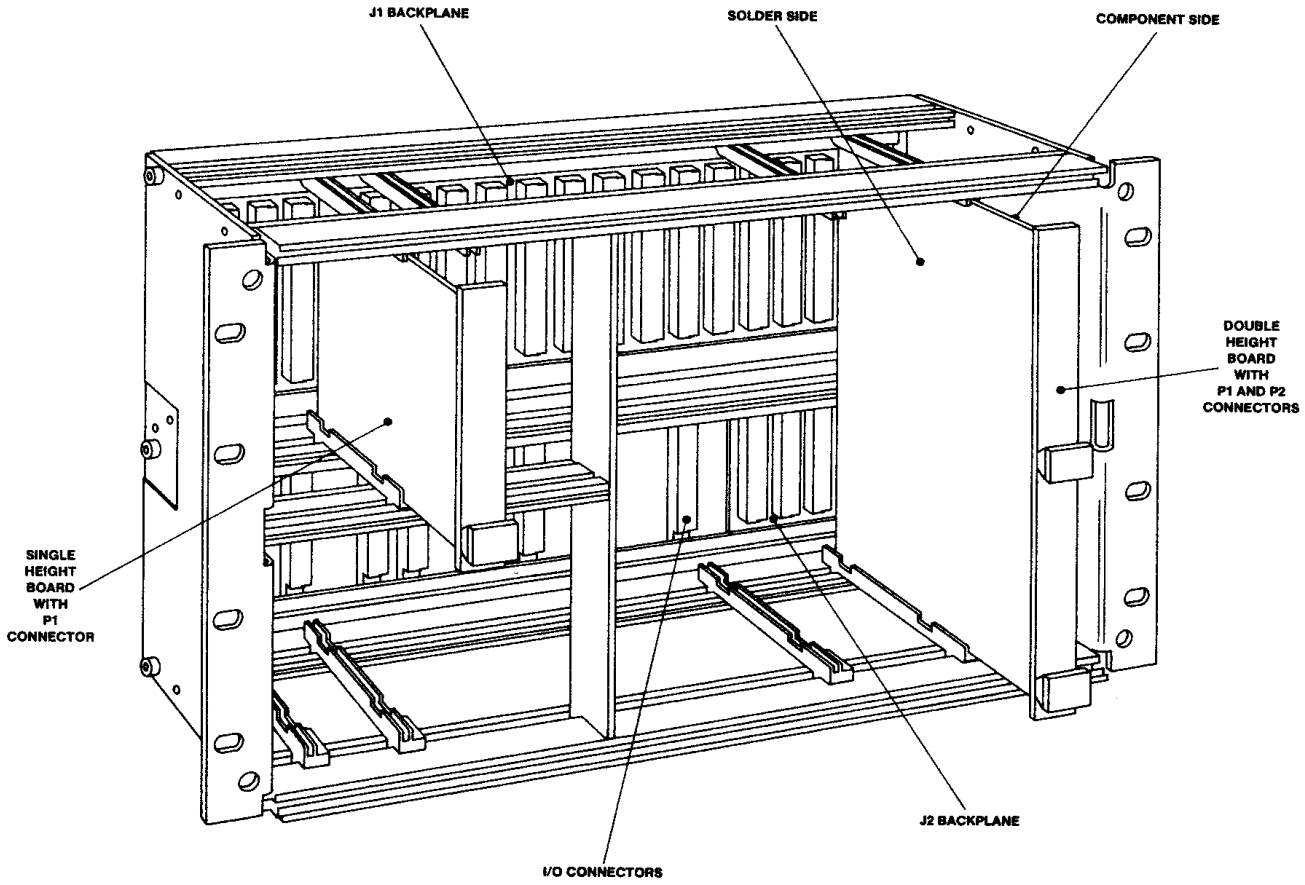
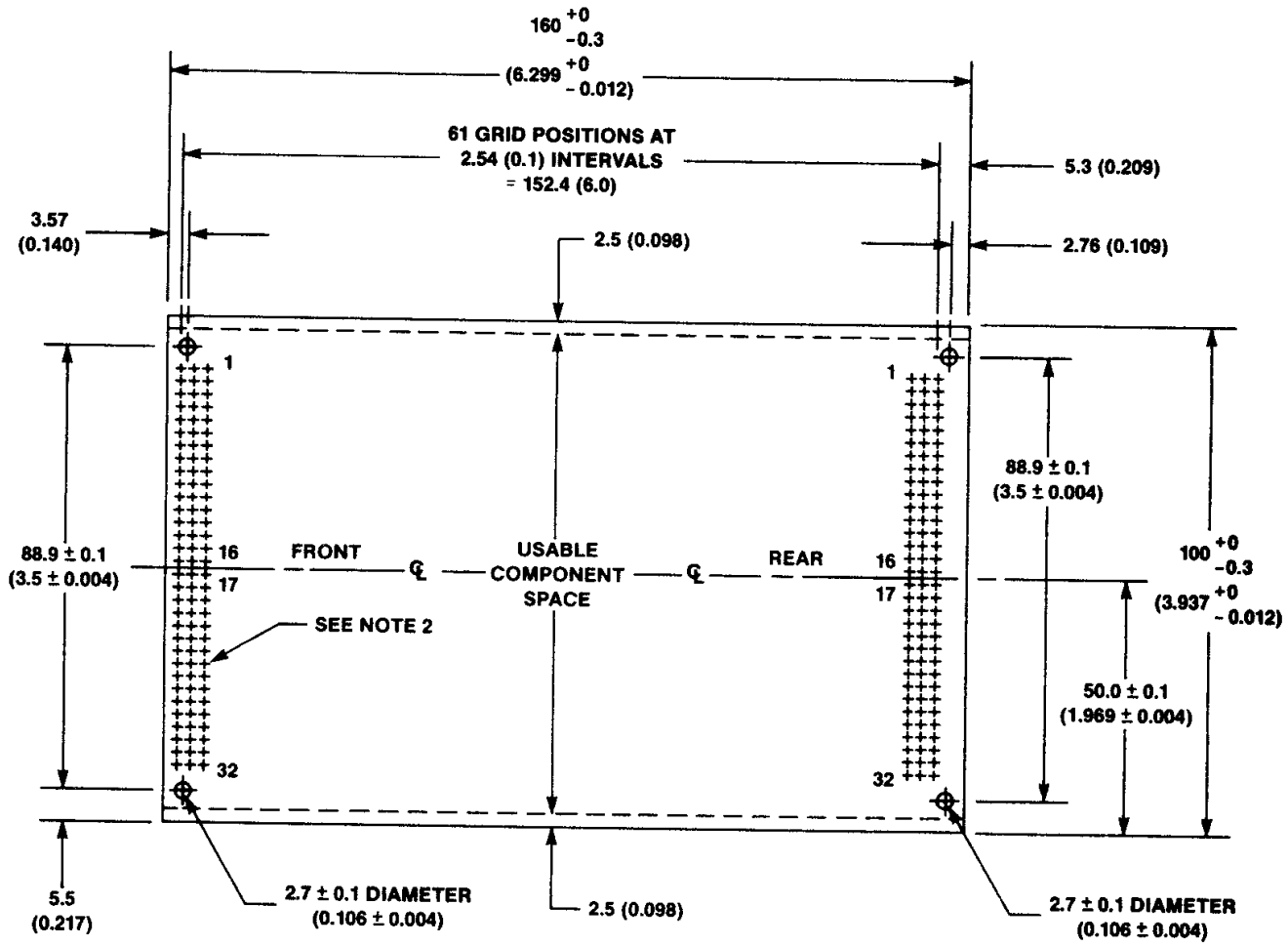


Figure 7-1. Subrack with Mixed Board Sizes



NOTES:

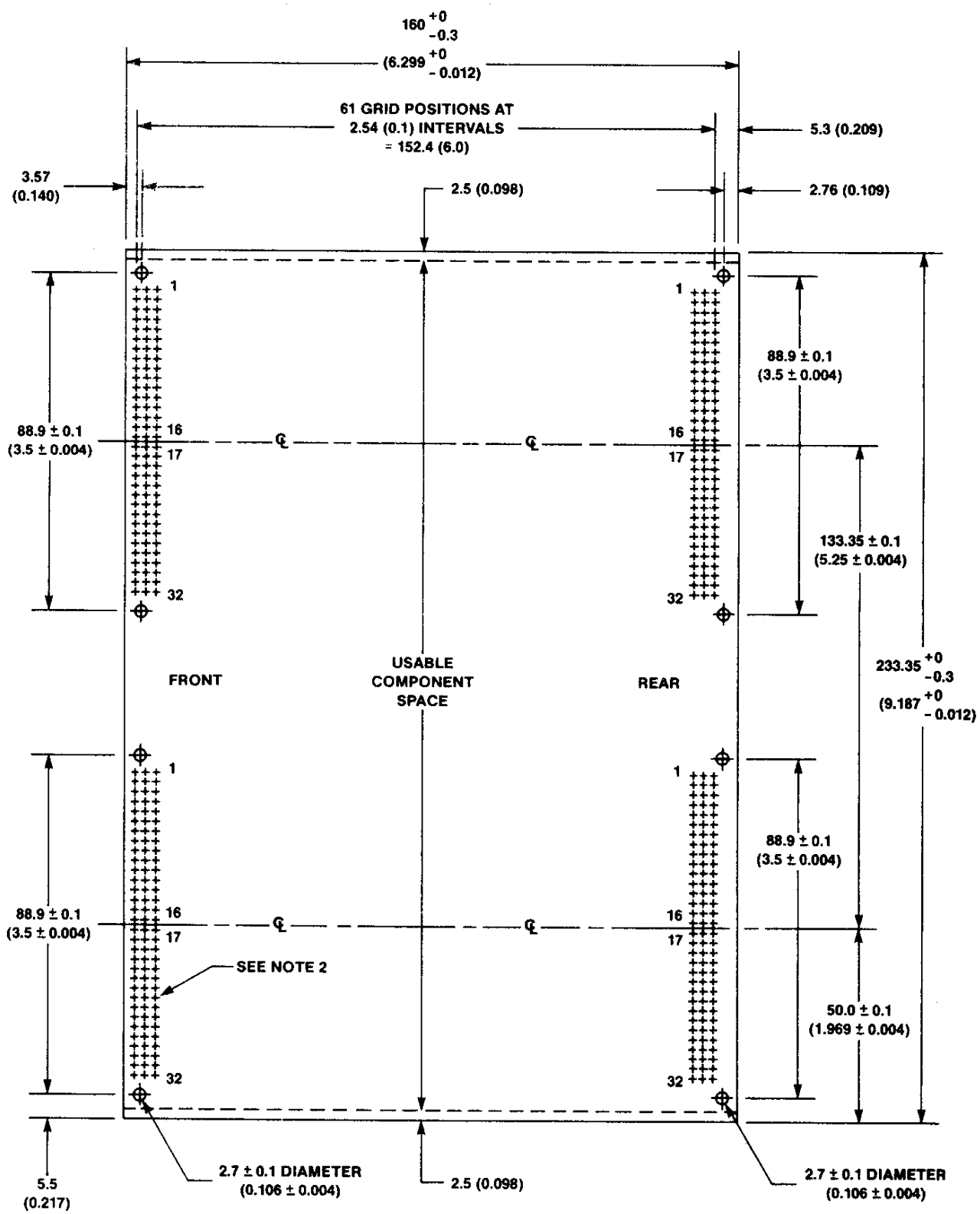
1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. These grids are provided to help the board designer to align components with the front panel grid.

3. RULE 7.32:

Boards **MUST** be 1 .6 +0.2 mm (0.063 +0.008 inch) thick in the guide area.

Figure 7-2. Single Height Board: Basic Dimensions



NOTES:

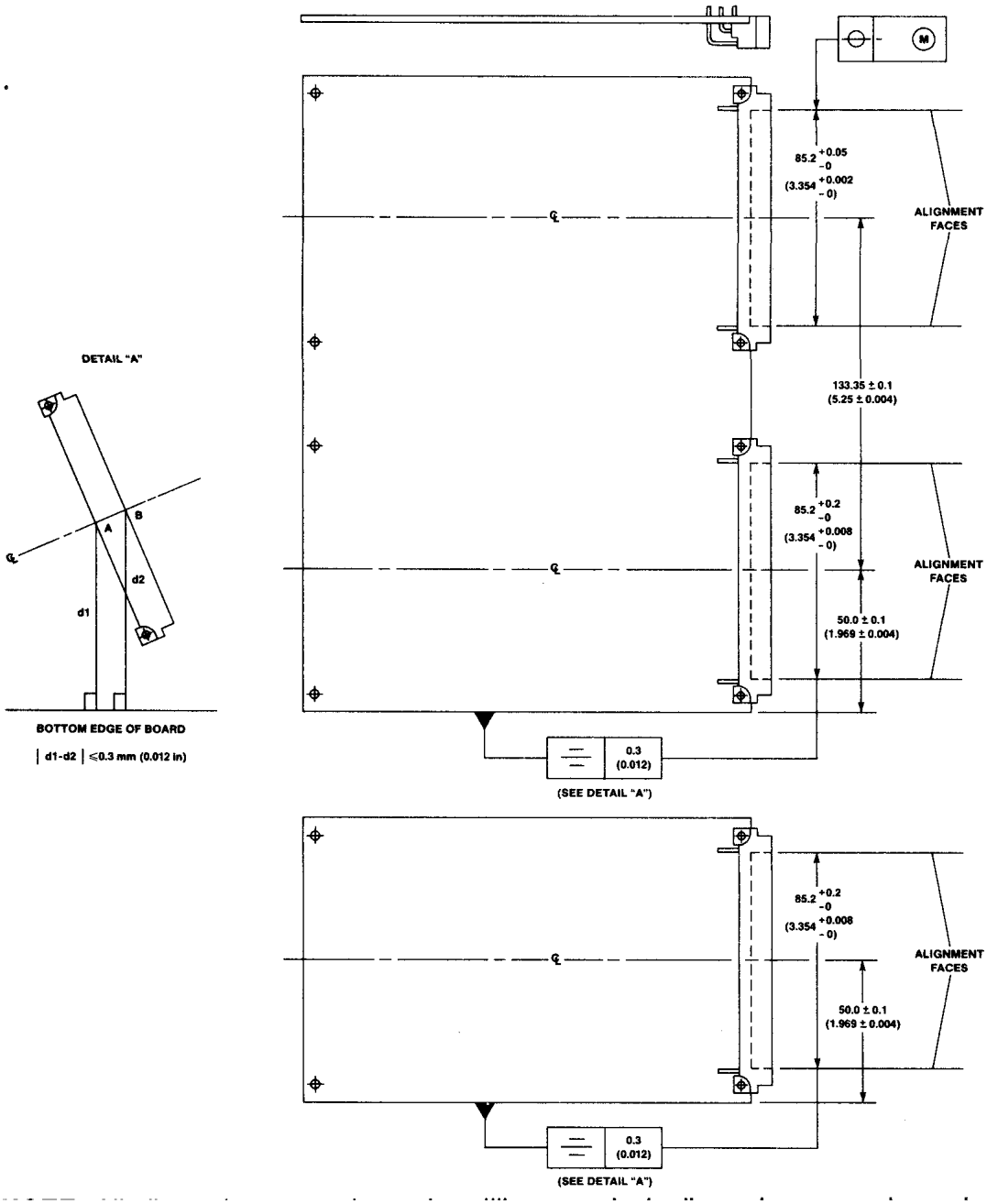
1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. These grids are provided to help the board designer to align components with the front panel grid.

3. RULE 7.33:

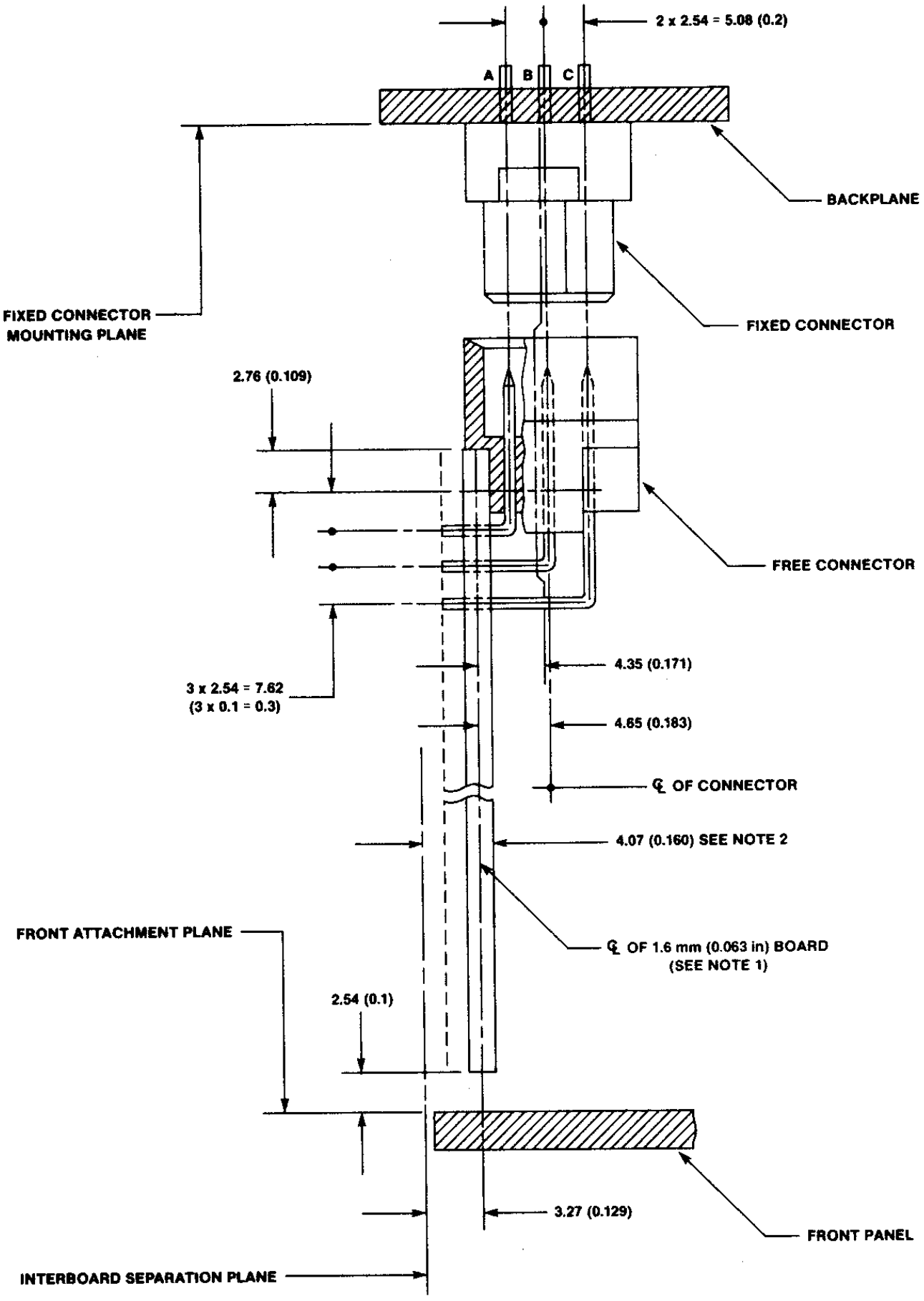
Boards **MUST** be 1.6 +0.2 mm (0.063 +0.008 inch) thick in the guide area.

Figure 7-3. Double Height Board: Basic Dimensions



NOTE:
All dimensions are shown in millimeters. inch dimensions are shown in parentheses.

Figure 7-4: Connector Position On Single And Double Height Boards

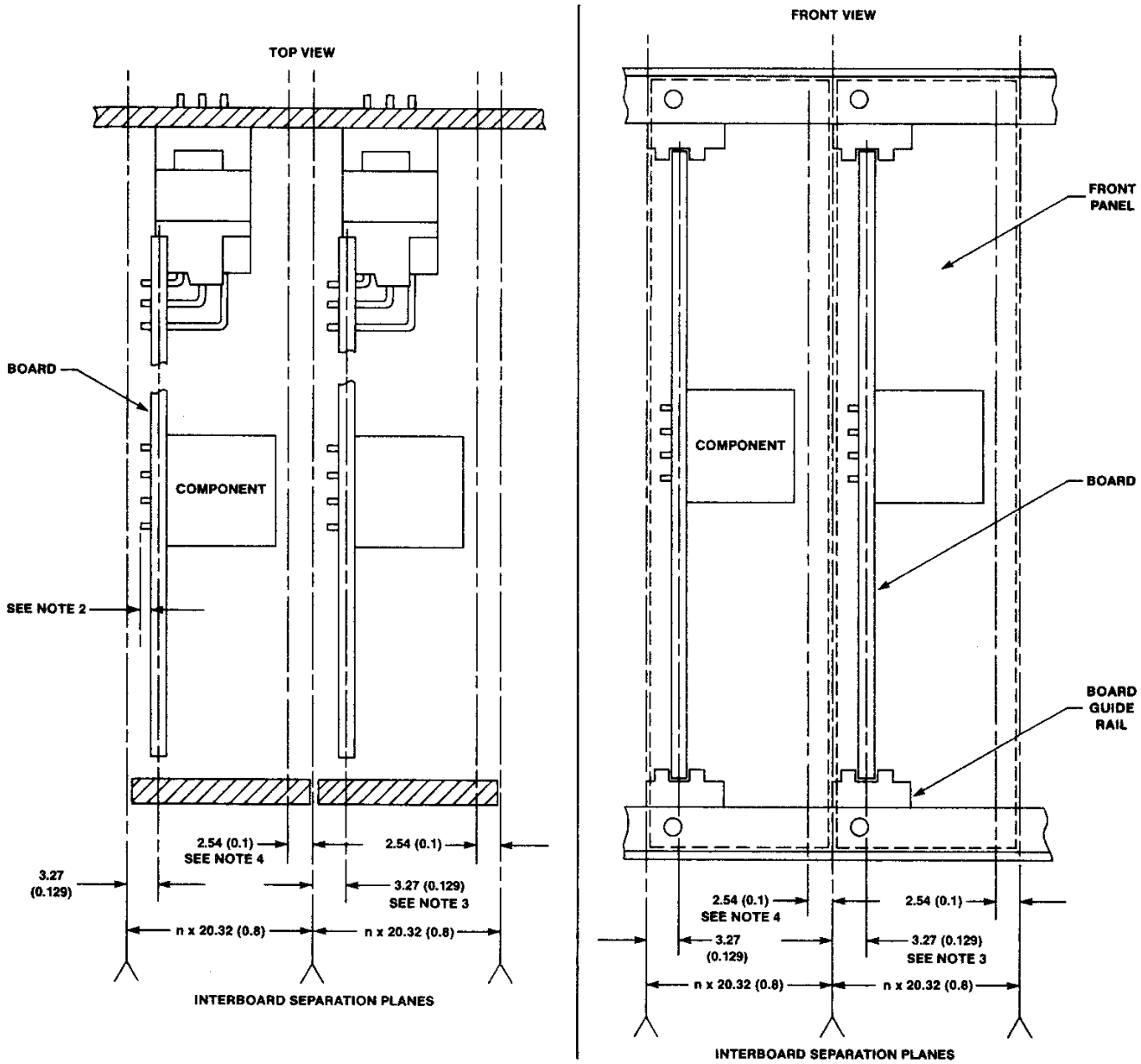

2 x 2.54 = 5.08 (0.2)
A
B
C
BACKPLANE
FIXED CONNECTOR
MOUNTING PLANE
FIXED CONNECTOR
2.76 (0.109)
FREE CONNECTOR
3 x 2.54 = 7.62
(3 x 0.1 = 0.3)
4.35 (0.171)
4.65 (0.183)
℄ OF CONNECTOR
4.07 (0.160) SEE NOTE 2
FRONT ATTACHMENT PLANE
℄ OF 1.6 mm (0.063 in) BOARD
(SEE NOTE 1)
2.54 (0.1)
3.27 (0.129)
FRONT PANEL
INTERBOARD SEPARATION PLANE

NOTES:

1. For additional inforamtion regarding the allowed thickness of boards, see Section 7.2.

2. This 4.077 mm (0.160 in) dimesions is the same regardless of the thickness of the board.

Figure 7-5. Cross Sectional View of Board, Connector, Backplane, and Front Panel



NOTES:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. SUGGESTION 7.12:

Trim component lead lengths to no more than 1.52 millimeters (0.06 inch).

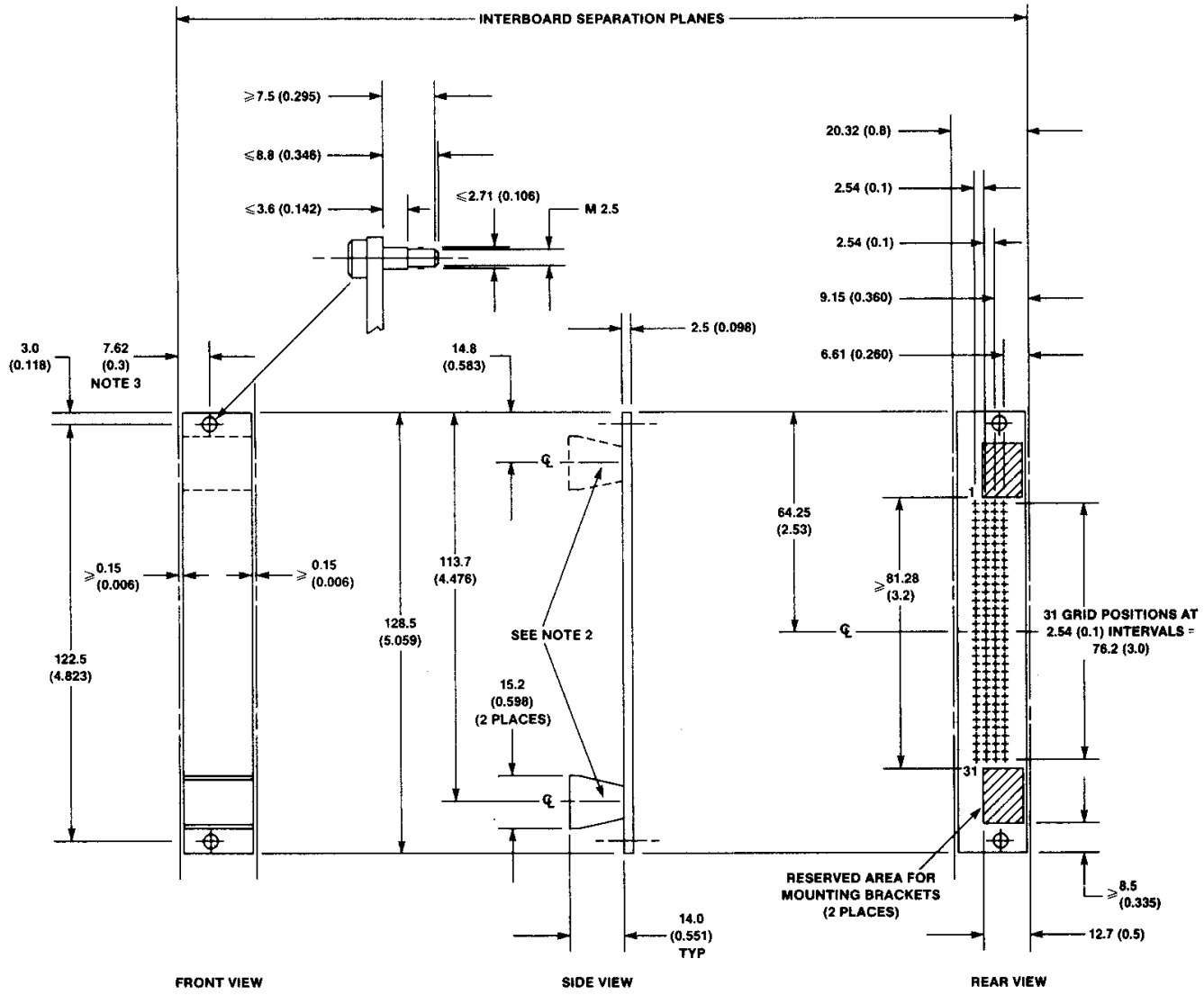
3. RULE 7.34:

Component leads and components mounted on the solder side of the board **MUST NOT** protrude through the interboard separation plane after it has been completely seated in the backplane.

4. RULE 7.35:

Components mounted on the component side of the board **MUST NOT** be any closer than 2.54 millimeters (0.1 inch) to the interboard separation plane after it has been completely seated in the backplane.

Figure 7-6. Component Height, Lead Length, and Board Warpage



NOTES:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. Dimensions given for height and depth of handles are suggestions only.

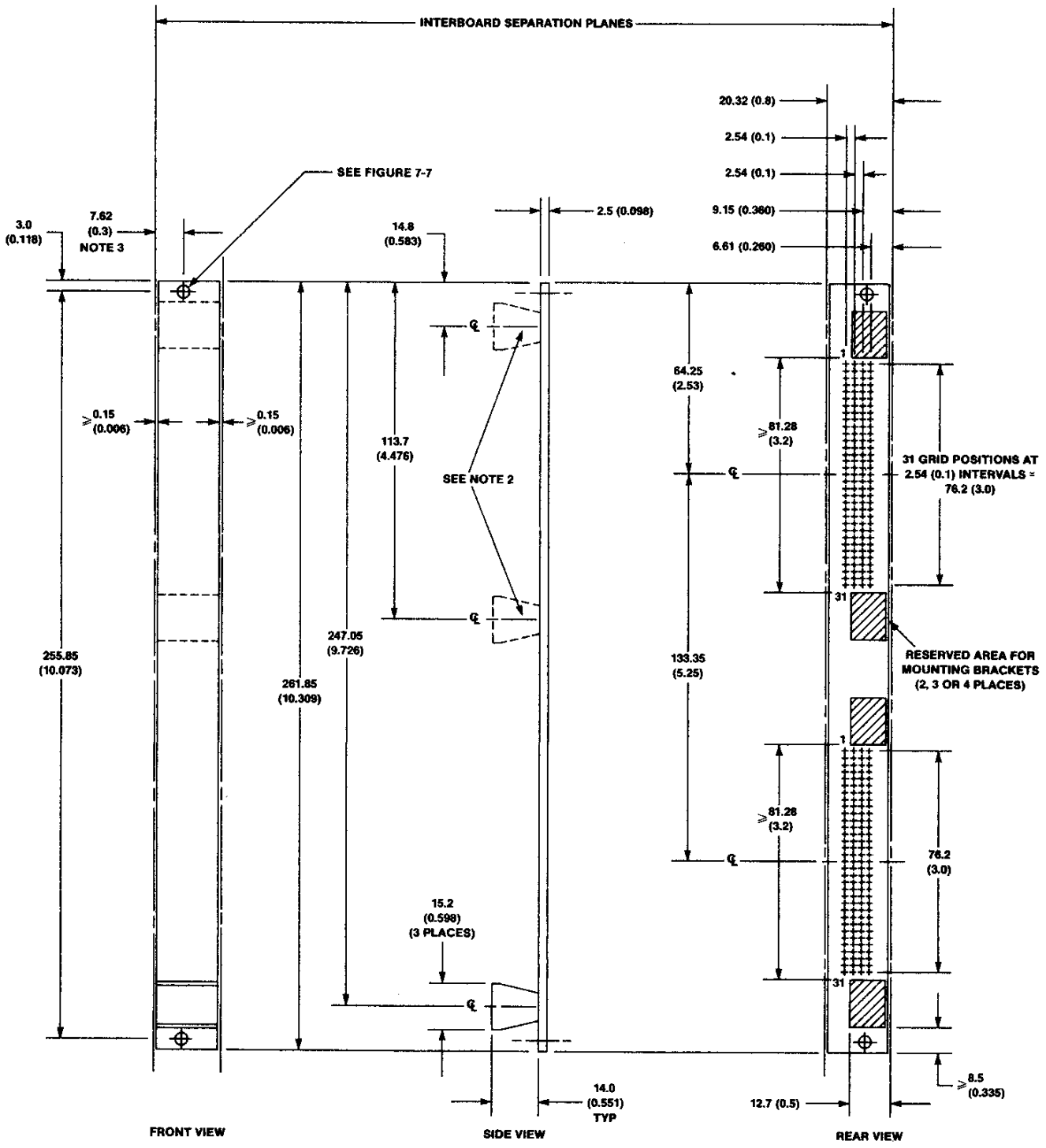
3. RECOMMENDATION 7.11:

Locate the mounting hole 7.62 mm (0.3 in) from the interboard separation plane.

PERMISSION 7.20:

The mounting hole **MAY** be located 12.7 mm (0.5 in) from the interboard separation plane.

Figure 7-7. Single Height, Single Width Front Panel



NOTES:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. Dimensions given for height and depth of handles are suggestions only.

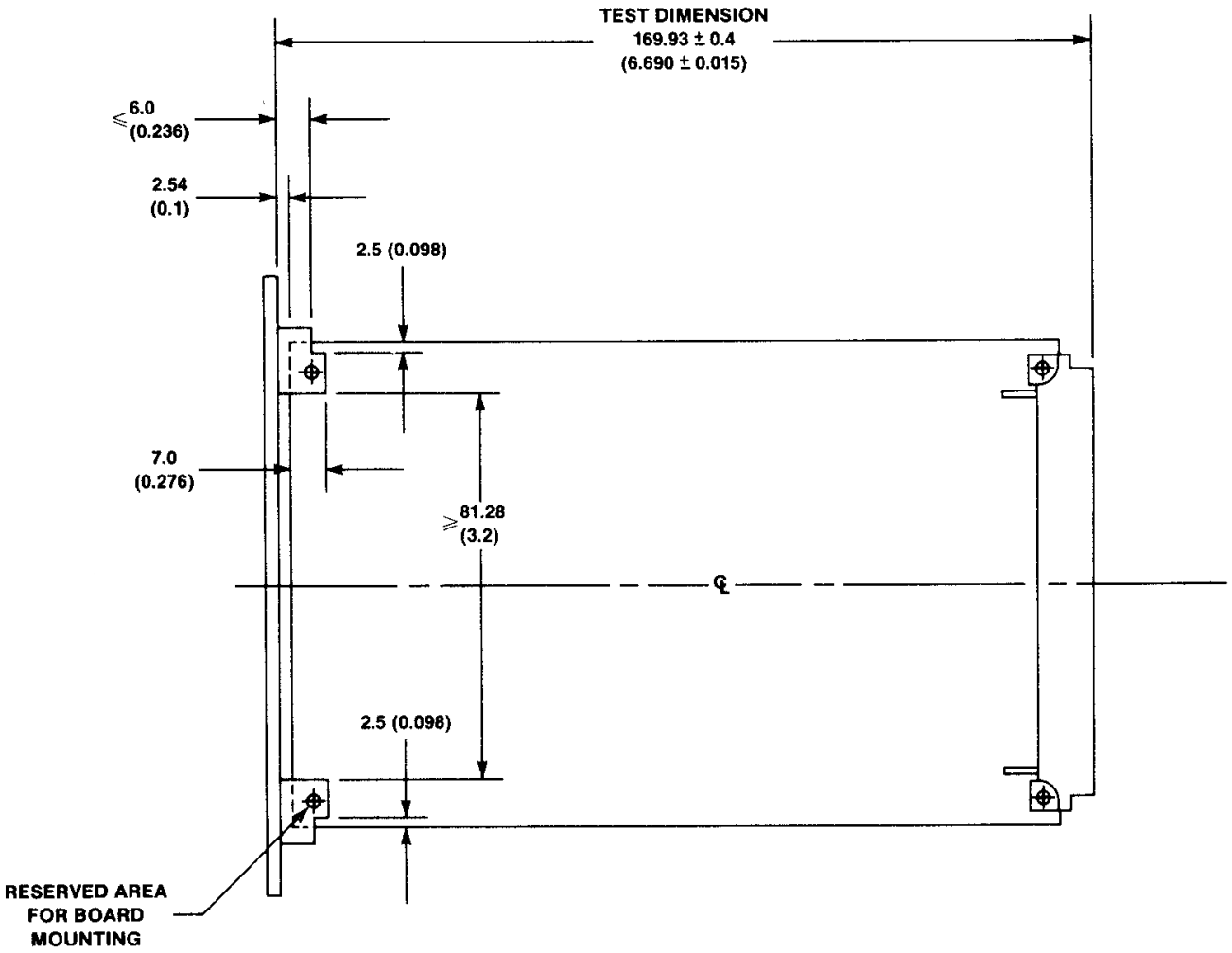
3. RECOMMENDATION 7.12:

Locate the mounting hole 7.62 mm (0.3 in) from the interboard separation plane.

PERMISSION 7.21:

The mounting hole **MAY** be located 12.7 mm (0.5 in) from the interboard separation plane.

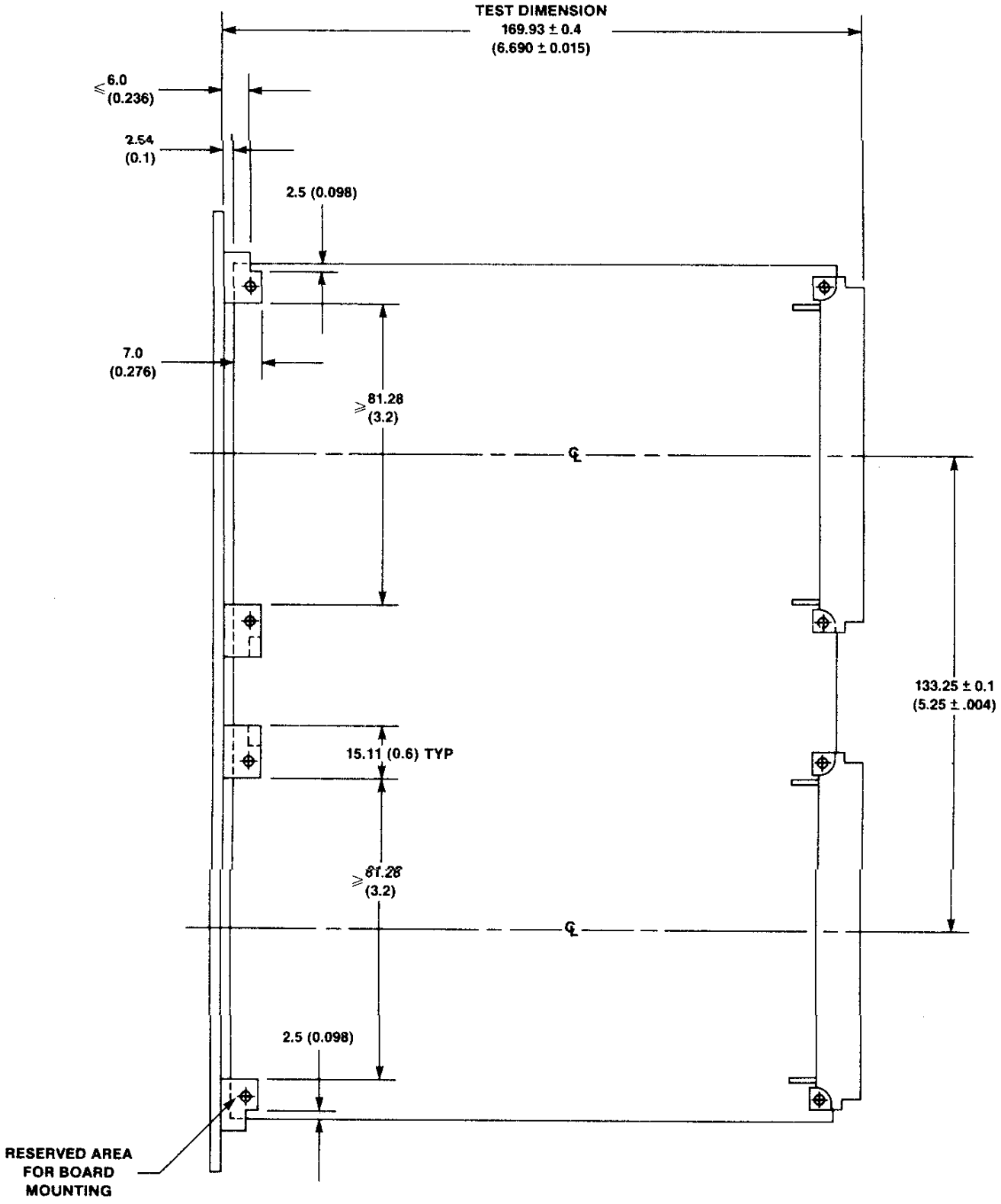
Figure 7-8. Double Height, Single Width Front Panel



NOTE:

1. All dimensions are shown in millimeters. inch dimensions are shown in parentheses.

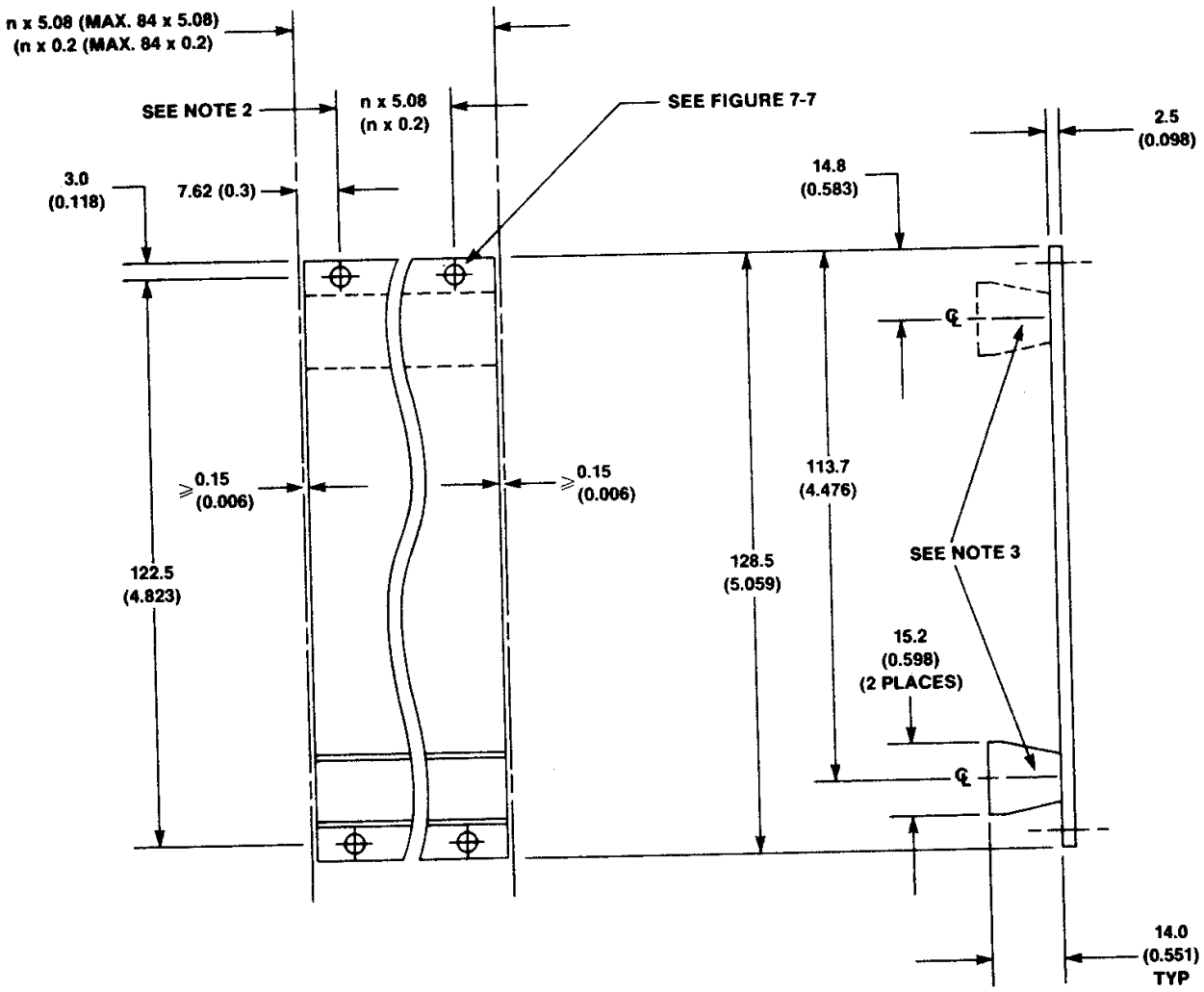
Figure 7-9. Front Panel Mounting Brackets And Dimension Of Single Height Boards.



NOTE:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

Figure 7-10. Front Panel Mounting Brackets And Dimension Of Double Height Boards.



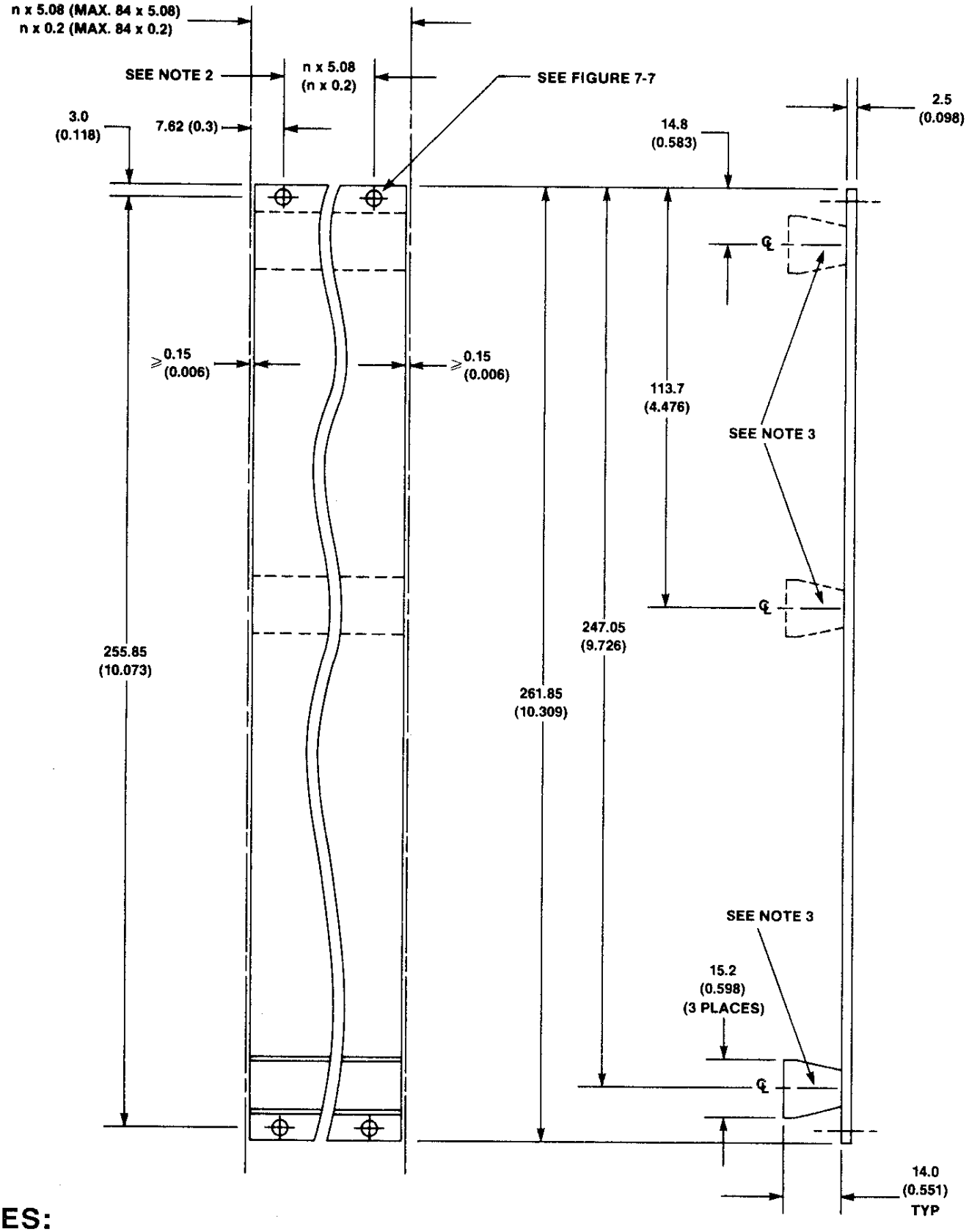
NOTES:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.

2. RECOMMENDATION 7.13:
Where a panel is more than 50.8 millimeters (2.0 inch) wide use at least 4 mounting holes; two at the top and two at the bottom.

3. Dimensions given for height and depth of handles are suggestions only.

Figure 7-11. Single Height Filler Panel



'ES:

NOTES:

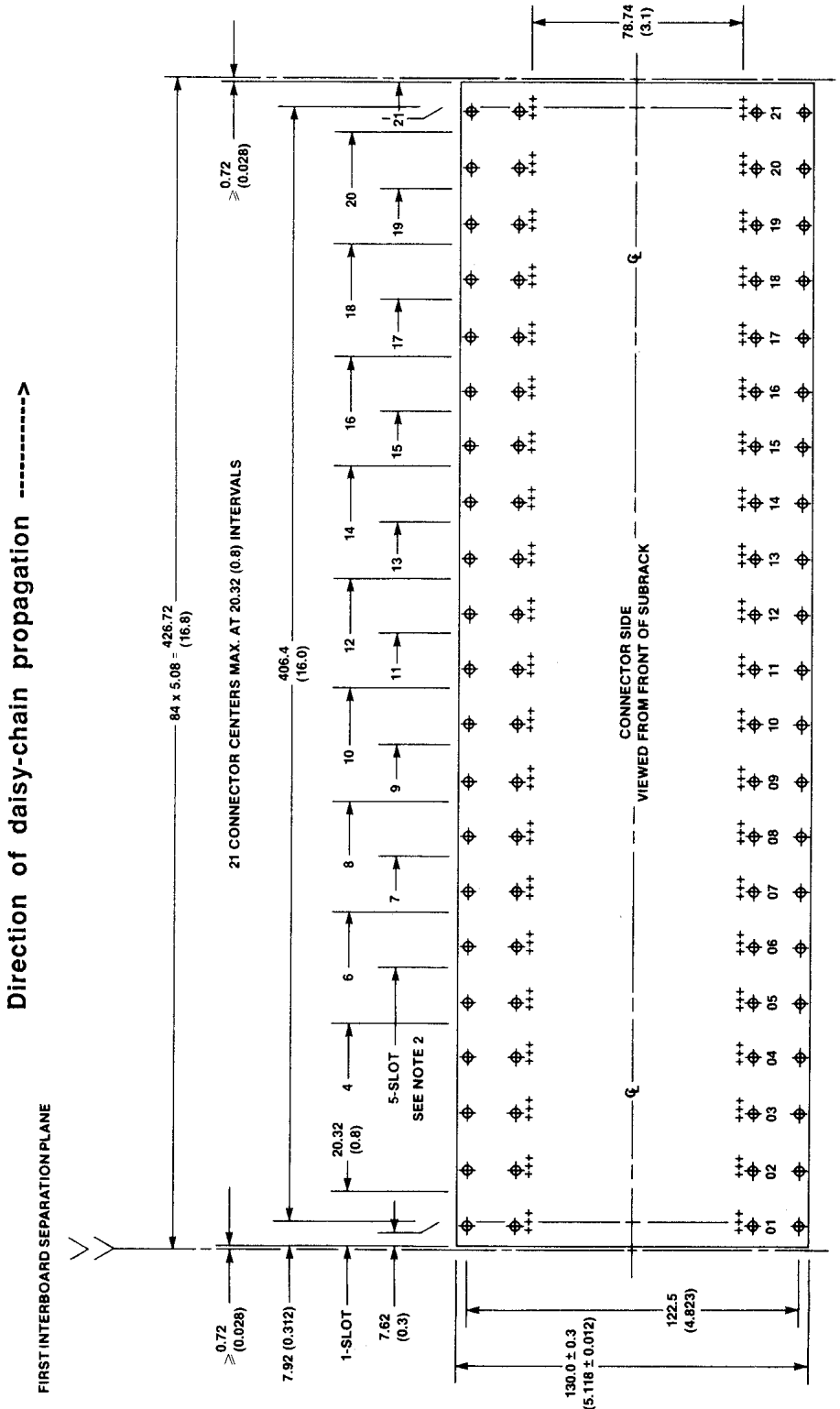
1. All dimensions are shown in millimeters. inch dimensions are shown in parentheses.

2. RECOMMENDATION 7.14:

Where a panel is more than 50.8 millimeters (2.0 inch) wide use at least 4 mounting holes; two at the top and two at the bottom.

3. Dimensions given for height and depth of handles are suggestions only.

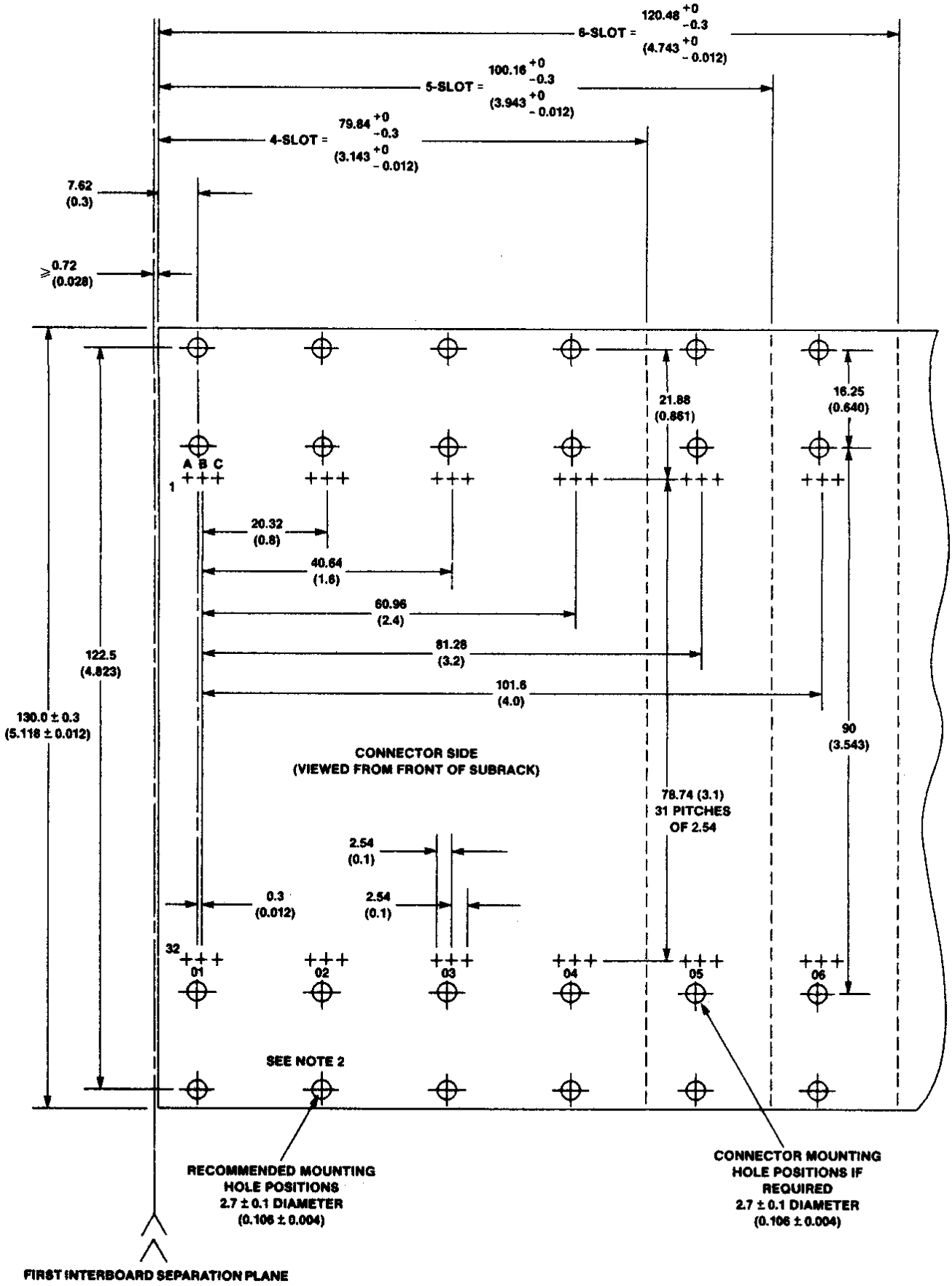
Figure 7-12. Double Height Filler Panel



NOTES:

1. All dimensions are show in millimeters. Inch dimensions are shown in parantheses.
2. Backplane width varies depending on the number of slots.

Figure 7-13. Backplane Overall Dimensions


6-SLOT = 120.48 +0 -0.3 (4.743 +0 -0.012)
5-SLOT = 100.16 +0 -0.3 (3.943 +0 -0.012)
4-SLOT = 79.84 +0 -0.3 (3.143 +0 -0.012)
7.62 (0.3)
≥ 0.72 (0.028)
21.88 (0.861)
16.25 (0.640)
A B C
1
20.32 (0.8)
40.64 (1.6)
60.96 (2.4)
81.28 (3.2)
101.6 (4.0)
122.5 (4.823)
130.0 ± 0.3 (5.118 ± 0.012)
90 (3.543)
CONNECTOR SIDE
(VIEWED FROM FRONT OF SUBRACK)
78.74 (3.1)
31 PITCHES
OF 2.54
2.54 (0.1)
0.3 (0.012)
2.54 (0.1)
32
01
02
03
04
05
06
SEE NOTE 2
RECOMMENDED MOUNTING
HOLE POSITIONS
2.7 ± 0.1 DIAMETER
(0.106 ± 0.004)
CONNECTOR MOUNTING
HOLE POSITIONS IF
REQUIRED
2.7 ± 0.1 DIAMETER
(0.106 ± 0.004)
FIRST INTERBOARD SEPARATION PLANE

NOTE:
1. All dimensions are shown in millimeters. Inch dimensions are shown in millimeters. Inch dimensions are show in parentheses.

Figure 7-14. Backplane Detailed Dimensions

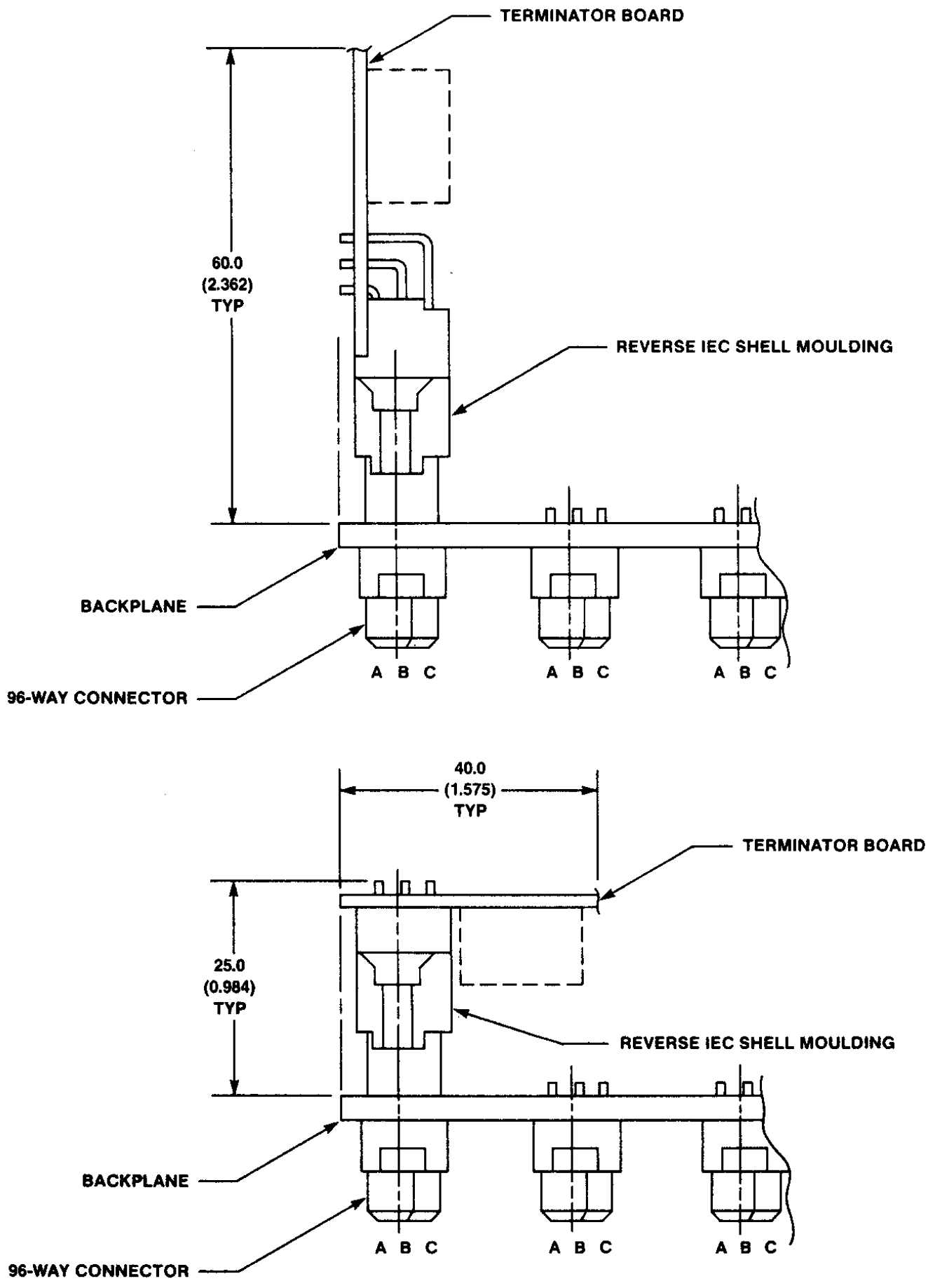

TERMINATOR BOARD
60.0
(2.362)
TYP
REVERSE IEC SHELL MOULDING
BACKPLANE
96-WAY CONNECTOR
A B C
A B C
A B C
40.0
(1.575)
TYP
TERMINATOR BOARD
25.0
(0.984)
TYP
REVERSE IEC SHELL MOULDING
BACKPLANE
96-WAY CONNECTOR
A B C
A B C
A B C

Figure 7-15. “Off-Board Type” Backplane Termination (Viewed From Top Of Backplane)

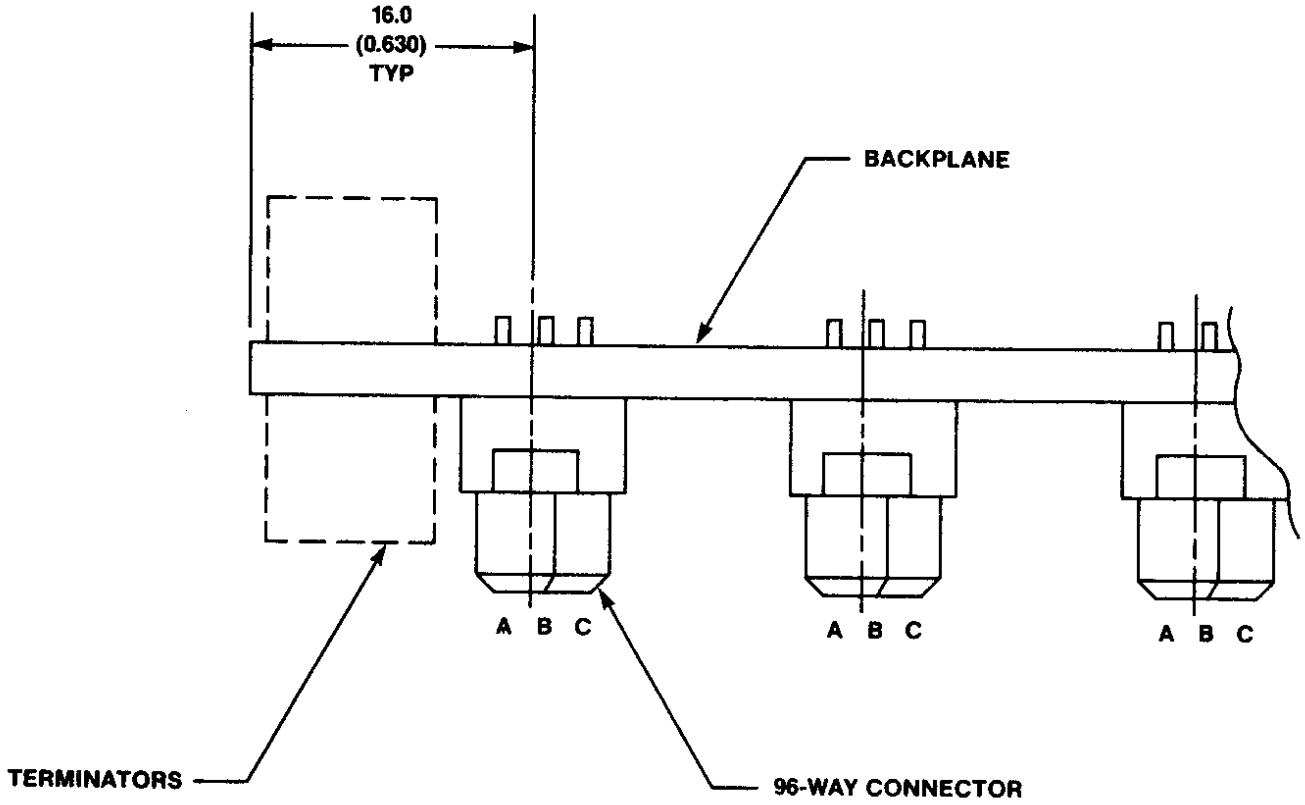
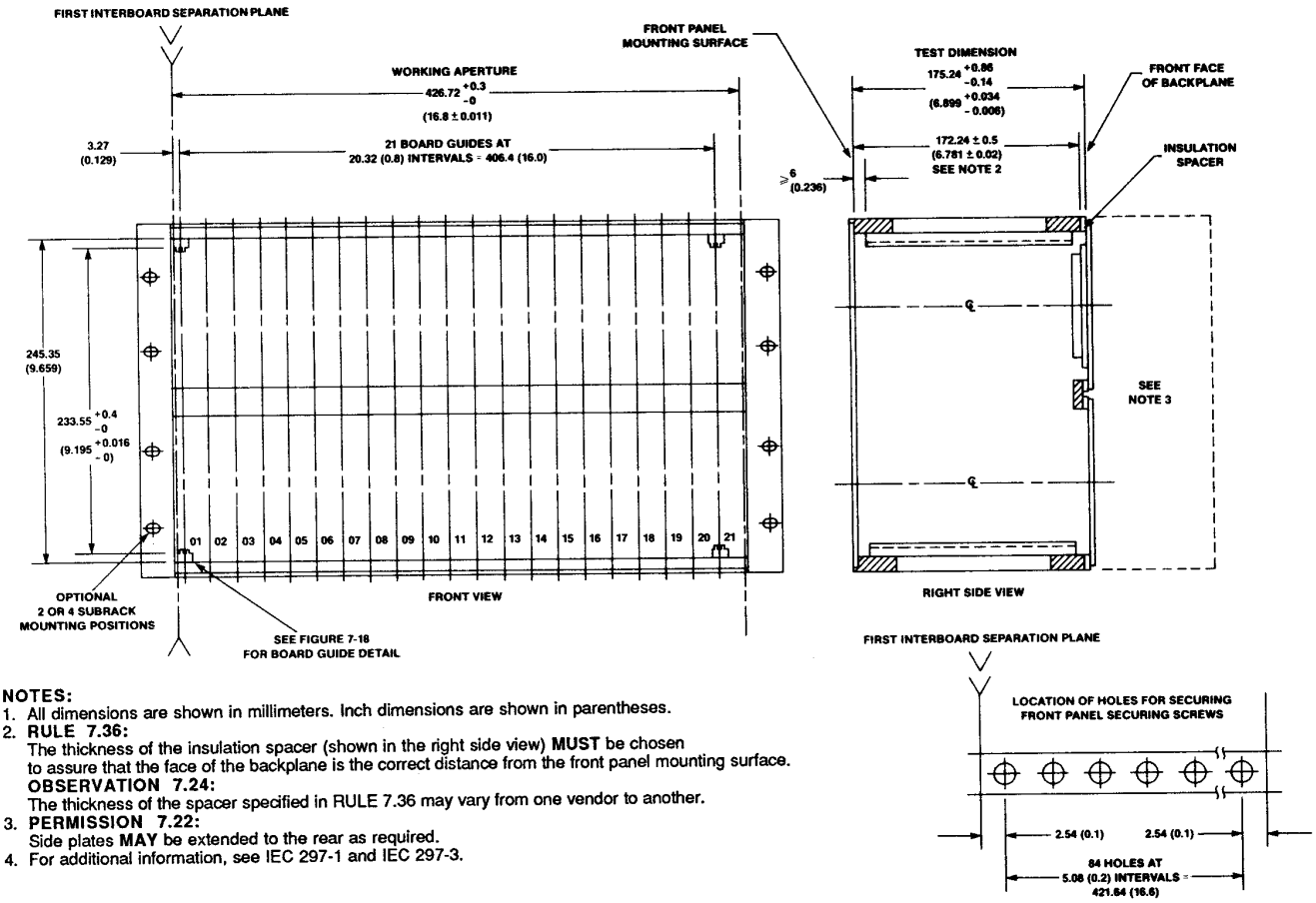


Figure 7-16. “On-Board Type” Backplane Termination (Viewed From Top Of Backplane)



NOTES:

1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.
2. **RULE 7.36:**
   The thickness of the insulation spacer (shown in the right side view) **MUST** be chosen to assure that the face of the backplane is the correct distance from the front panel mounting surface.
   OBSERVATION 7.24:
   The thickness of the spacer specified in RULE 7.36 may vary from one vendor to another.
3. **PERMISSION 7.22:**
   Side plates **MAY** be extended to the rear as required.
4. For additional information, see IEC 297-1 and IEC 297-3.

Figure 7-17: 21 Slot Subrack

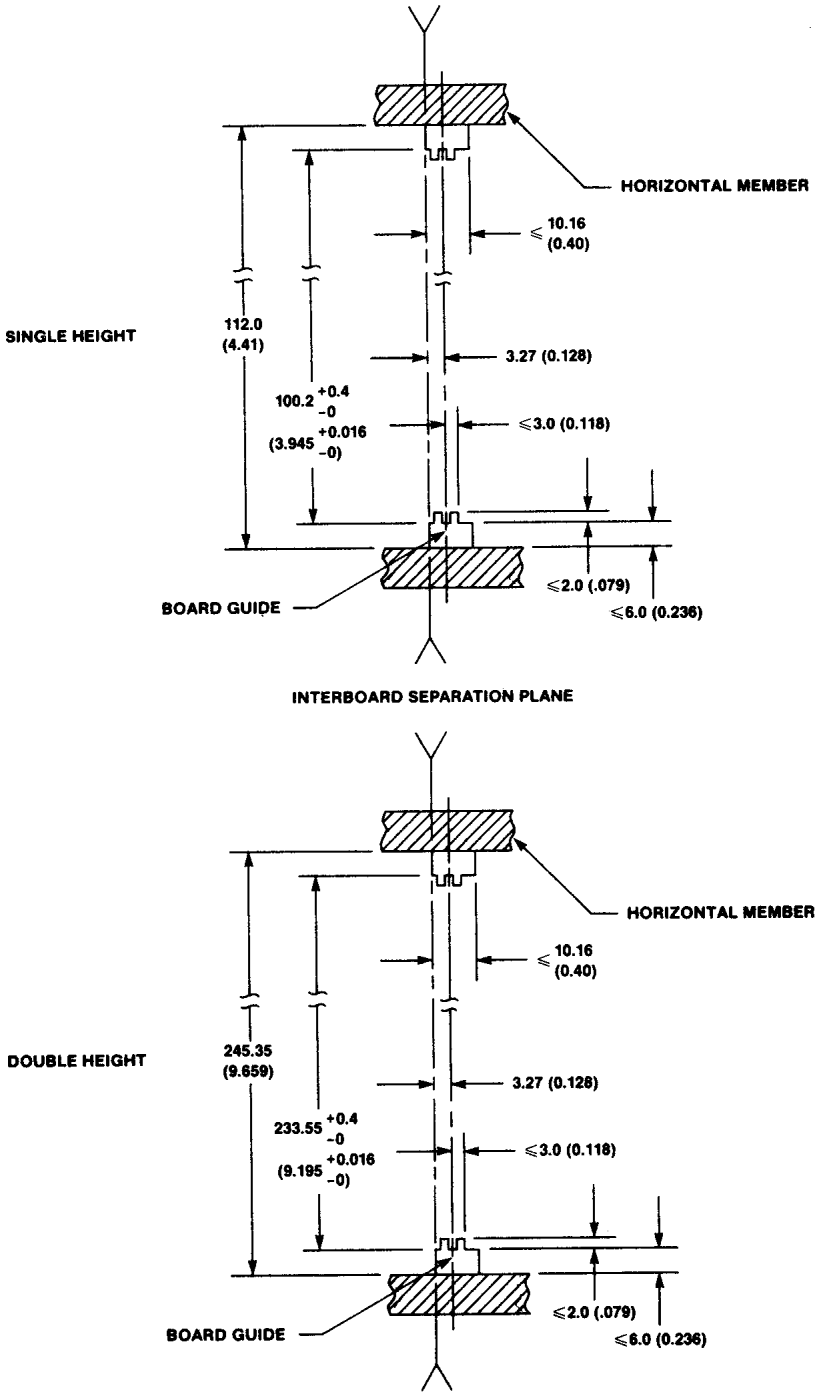


Figure 7-18. Board Guide Detail

7.6 VMEbus BACKPLANE CONNECTORS AND BOARD CONNECTORS

7.6.1 Pin Assignments For J1/P1 Connectors

Table 7-1 provides signal names for the J1/P1 connector pins. (The connector consists of three rows of pins labeled rows a, b, and c.)

Table 7-1 . J1 /P1 Pin Assignments

PIN NUMBER	ROW a SIGNAL MNEMONIC	ROW b SIGNAL MNEMONIC	ROW c SIGNAL MNEMONIC
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG20UT*	GND
10	SYSCLK	G3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK(1)	A17
22	IACKOUT*	SERDAT*(1)	A16
23	AM	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1 *	A08
31	-12V	+5VSTDBY	+12V
32	+5V	+5V	+5V

Note: (1): See Appendix C for further information on the use of these signals.

7.6.2 Pin Assignments For The J2/P2 Connector

Table 7-2 provides signal names for the J2/P2 connector pins. (The connector consists of three rows of pins labeled rows a, b, and c.)

Table 7-2. J2/P2 pin assignments

PIN NUMBER	ROW a SIGNAL MNEMONIC	ROW b SIGNAL MNEMONIC	ROW c SIGNAL MNEMONIC
1	User Defined	+5V	User Defined
2	User Defined	GND	User Defined
3	User Defined	RESERVED	User Defined
4	User Defined	A24	User Defined
5	User Defined	A25	User Defined
6	User Defined	A26	User Defined
7	User Defined	A27	User Defined
8	User Defined	A28	User Defined
9	User Defined	A29	User Defined
10	User Defined	A30	User Defined
11	User Defined	A31	User Defined
12	User Defined	GND	User Defined
13	User Defined	+5V	User Defined
14	User Defined	D16	User Defined
15	User Defined	D17	User Defined
16	User Defined	D18	User Defined
17	User Defined	D19	User Defined
18	User Defined	D20	User Defined
19	User Defined	D21	User Defined
20	User Defined	D22	User Defined
21	User Defined	D23	User Defined
22	User Defined	GND	User Defined
23	User Defined	D24	User Defined
24	User Defined	D25	User Defined
25	User Defined	D26	User Defined
26	User Defined	D27	User Defined
27	User Defined	D28	User Defined
28	User Defined	D29	User Defined
29	User Defined	D30	User Defined
30	User Defined	D31	User Defined
31	User Defined	GND	User Defined
32	User Defined	+5V	User Defined

APPENDIX A

GLOSSARY OF VMEbus TERMS

A16	A type of module that provides or decodes an address on address lines A01 through A15.
A24	A type of module that provides or decodes an address on address lines A01 through A23.
A32	A type of module that provides or decodes an address on address lines A01 through A31.
ARBITRATION	The process of assigning control of the DTB to a REQUESTER.
ADDRESS-ONLY CYCLE	A DTB cycle that consists of an address broadcast, but no data transfer. SLAVES do not acknowledge ADDRESS-ONLY cycles and MASTERS terminate the cycle without waiting for an acknowledgment.
ARBITER	A functional module that accepts bus requests from REQUESTER modules and grants control of the DTB to one REQUESTER at a time.
ARBITRATION BUS	One of the four buses provided by the VMEbus backplane. This bus allows an ARBITER module and several REQUESTER modules to coordinate use of the DTB.
ARBITRATION CYCLE	An ARBITRATION CYCLE begins when the ARBITER senses a bus request. The ARBITER grants the bus to a REQUESTER, which signals that the DTB is busy. The REQUESTER terminates the cycle by releasing the bus busy signal which causes the ARBITER to sample the bus requests again.
BACKPLANE (VMEbus)	A printed circuit (PC) board with 96-pin connectors and signal paths that bus the connector pins. Some VMEbus systems have a single PC board, called the J1 backplane. It provides the signal paths needed for basic operation. Other VMEbus systems also have an optional second PC board, called a J2 backplane. It provides the additional 96 pin connectors and signal paths needed for wider data and address transfers. Still others have a single PC board that provides the signal conductors and connectors of both the J1 and J2 backplanes.
BACKPLANE INTERFACE LOGIC	Special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc. The VMEbus specification prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.
BLOCK READ CYCLE	A DTB cycle used to transfer a block of 1 to 256 bytes from a SLAVE to a MASTER. This transfer is done using a string of 1, 2, or 4 byte data transfers. Once the block transfer is started, the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the MASTER broadcasts only one address and address modifier (at

	the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data for the next cycle is retrieved from the next higher location.
BLOCK WRITE CYCLE	A DTB cycle used to transfer a block of 1 to 256 bytes from a MASTER to a SLAVE. The block write cycle is very similar to the block read cycle. It uses a string of 1 , 2 or 4 byte data transfers and the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the next transfer is stored in the next higher location.
BOARD	A printed circuit (PC) board, its collection of electronic components, and either one or two 96 pin connectors that can be plugged into VMEbus backplane connectors.
BUS TIMER	A functional module that measures how long each data transfer takes on the DTB and terminates the DTB cycle if a transfer takes too long. Without this module, if the MASTER tries to transfer data to or from a non-existent SLAVE location it could wait forever for a SLAVE to respond. The BUS TIMER prevents this by terminating the cycle.
D08(O)	A SLAVE that sends and receives data 8 bits at a time over D00-D07, OR an INTERRUPT HANDLER that receives 8 bit STATUS/ID's over D00-D07 OR , an INTERRUPTER that sends 8 bit STATUS/ID's over D00-D07.
D08 (EO)	A MASTER that sends or receives data 8 bits at a time over either D00-D07 or D08-D15, OR a SLAVE that sends and receives data 8 bits at a time over either D00-D07 or D08-D15.
D16	A MASTER that sends and receives data 16 bits at a time over D00-D15, OR a SLAVE that sends and receives data 16 bits at a time over D00-D15, OR an INTERRUPT HANDLER that receives 16 bit STATUS/ID's over D00-D15, OR an INTERRUPTER that sends 16 bit STATUS/ID's over D00-D15.
D32	A MASTER that sends and receives data 32 bits at a time over D00-D31 , OR a SLAVE that sends and receives data 32 bits at a time over D00-D31 , OR an INTERRUPT HANDLER that receives 32 bit STATUS/ID's over D00-D31 , OR an INTERRUPTER that sends 32 bit STATUS/ID's over D00-D31.
DAISY-CHAIN	A special type of VMEbus signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot. There are four bus grant daisy-chains and one interrupt acknowledge daisy-chain on the VMEbus.
DATA TRANSFER BUS	One of the four buses provided by the VMEbus backplane. The DATA TRANSFER BUS allows MASTERS to direct the transfer of binary data between themselves and SLAVES. (DATA TRANSFER BUS is often abbreviated DTB.)

DATA TRANSFER BUS CYCLE	A sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a MASTER and a SLAVE. There are 34 types of data transfer bus cycles.
DTB	An acronym for DATA TRANSFER BUS.
FUNCTIONAL MODULE	A collection of electronic circuitry that resides on one VMEbus board and works together to accomplish a task.
IACK DAISY-CHAIN DRIVER	A functional module which activates the interrupt acknowledge daisy-chain whenever an INTERRUPT HANDLER acknowledges an interrupt request. This daisy-chain ensures that only one INTERRUPTER will respond with its STATUS/ID when more than one has generated an interrupt request on the same level.
INTERRUPT ACKNOWLEDGE CYCLE	A DTB cycle, initiated by an INTERRUPT HANDLER, that reads a "STATUS/ID" from an INTERRUPTER. An INTERRUPT HANDLER generates this cycle when it detects an interrupt request from an INTERRUPTER and it has control of the DTB.
PRIORITY INTERRUPT BUS	One of the four buses provided by the VMEbus backplane. The PRIORITY INTERRUPT BUS allows INTERRUPTER modules to send interrupt requests to INTERRUPT HANDLER modules, and INTERRUPT HANDLER modules to acknowledge these interrupt requests.
INTERRUPTER	A functional module that generates an interrupt request on the INTERRUPT BUS and then provides STATUS/ID information when the INTERRUPT HANDLER requests it.
INTERRUPT HANDLER	A functional module that detects interrupt requests generated by INTERRUPTERS and responds to those requests by asking for STATUS/ID information.
LOCATION MONITOR	A functional module that monitors data transfers over the DTB in order to detect accesses to the locations it has been assigned to watch. When an access occurs to one of these assigned locations, the LOCATION MONITOR generates an on-board signal.
MASTER	A functional module that initiates DTB cycles in order to transfer data between itself and a SLAVE module.
POWER MONITOR MODULE	A functional module that monitors the status of the primary power source to the VMEbus system and signals when the power has strayed outside the limits required for reliable system operation. Since most systems are powered by an AC source, the power monitor is typically designed to detect drop-out or brown-out conditions on AC lines.
READ CYCLE	A DTB cycle used to transfer 1, 2, 3, or 4 bytes from a SLAVE to a MASTER. The cycle begins when the MASTER broadcasts an address and an address modifier. Each SLAVE captures this address and address modifier, and checks to see if it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus, and acknowledges the transfer.

	Then the MASTER terminates the cycle.
READ - MODIFY - WRITE CYCLE	A DTB cycle that is used to both read from, and write to, a SLAVE location without I permitting any other MASTER to access that location during that cycle. This cycle I is most useful in multiprocessing systems where certain memory locations are used I to control access to certain systems resources. (For example, semaphore locations.)
REQUESTER	A functional module that resides on the same board as a MASTER or INTERRUPT HANDLER and requests use of the DTB whenever its MASTER or INTERRUPT HANDLER needs it.
SERIAL CLOCK DRIVER	A functional module that provides a periodic timing signal that synchronizes operation of the VMSbus. (Although the VMEbus specification defines a SERIAL CLOCK DRIVER for use with the VMSbus, and although it reserves two backplane signal lines for use by that bus, the VMSbus protocol is completely independent of the VMEbus). A specification for the timing of the signal generated by the SERIAL CLOCK DRIVER is given in appendix C.
SLAVE	A functional module that detects DTB cycles initiated by a MASTER and, when those cycles specify its participation, transfers data between itself and the MASTER.
SLOT	A position where a board can be inserted into a VMEbus backplane. If the VMEbus system has both a J1 and a J2 backplane (or a combination J1/J2 backplane) each slot provides a pair of 96 pin connectors. If the system has only a J1 backplane, then each slot provides a single 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 do not disengage themselves from the backplane due to vibration or shock.
SYSTEM CLOCK DRIVER	A functional module that provides a 16 MHz timing signal on the UTILITY BUS.
SYSTEM CONTROLLER BOARD	A board which resides in slot 1 of a VMEbus backplane and has a SYSTEM CLOCK DRIVER, a DTB ARBITER an IACK DAISY CHAIN DRIVER, and a BUS TIMER. Some also have a SERIAL CLOCK DRIVER, a POWER MONITOR, or both.
UAT	A MASTER that sends or receives data in an unaligned fashion, or a SLAVE that sends and receives data in an unaligned fashion, OR a SLAVE that sends and receives data in an unaligned fashion.
UTILITY BUS	One of the four buses provided by the VMEbus backplane. This bus includes signals that provide periodic timing and coordinate the power-up and power-down of VMEbus systems.
WRITE CYCLE	A DTB cycle used to transfer 1,2,3, or 4 bytes from a MASTER to

	a SLAVE. The cycle begins when the MASTER broadcasts an address and address modifier and places data on the DTB. Each SLAVE captures this address and address modifier, and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The MASTER then terminates the cycle.

APPENDIX B

VMEbus CONNECTOR/PIN DESCRIPTION

INTRODUCTION

This appendix describes the VMEbus signal lines. The following table identifies the VMEbus signals by signal mnemonic, and describes the signal characteristics.

VMEbus Signal Identification

SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
A01-A15	ADDRESS bus (bits 1-15) - Three-state driven address lines that are used to broadcast a short, standard, or extended address.
A16-A23	ADDRESS bus (bits 16-23) - Three-state driven address lines that are used in conjunction with A01-A15 to broadcast a standard or extended address.
A24-A31	ADDRESS bus (bits 24-31) - Three-state driven address lines that are used in conjunction with A01-A23 to broadcast an extended address.
ACFAIL*	AC FAILURE - An open-collector driven signal which indicates that the AC input to the power supply is no longer being provided or that the required AC input voltage levels are not being met.
AM0-AM5	ADDRESS MODIFIER (bits 0-5) - Three-state driven lines that are used to broadcast information such as address size, cycle type, and/or MASTER identification.
AS*	ADDRESS STROBE - A three-state driven signal that indicates when a valid address has been placed on the address bus.
BBSY*	BUS BUSY - An open-collector driven signal driven low by the current MASTER to indicate that it is using the bus. When the MASTER releases this line, the resultant rising edge causes the ARBITER to sample the bus grant lines and grant the bus to the highest priority requester.
BCLR*	BUS CLEAR - A totem-pole driven signal, generated by an ARBITER to indicate when there is a higher priority request for the bus. This signal requests the current MASTER to release the DTB .
BERR*	BUS ERROR - An open-collector driven signal generated by a SLAVE or BUS TIMER. This signal indicates to the MASTER that the data transfer was not completed.
BG0IN*-BG3IN*	BUS GRANT (0-3)IN - Totem-pole driven signals generated by the ARBITER and REQUESTERS. “Bus grant in” and “bus grant out” signals form bus grant daisy chains. The “bus grant in” signal indicates, to the board receiving it, that it may use

	the DTB.
BG0OUT*-BG3OUT*	BUS GRANT (0-3) OUT - Totem-pole driven signals generated by REQUESTERS. The bus grant out signal indicates to the next board in the daisy-chain that it may use the DTB.
BR0*-BR3*	BUS REQUEST (0-3) - Open-collector driven signals generated by REQUESTERS. A low level on one of these lines indicates that some MASTER needs to use the DTB.
D00-D31	DATA BUS - Three-state driven bidirectional data lines used to transfer data between MASTERS and SLAVES.
DS0*, DS1*	DATA STROBE ZERO, ONE - Three-state driven signals used in conjunction with LWORD* and A01 to indicate how many data bytes are being transferred (1 , 2, 3, or 4). During a write cycle, the falling edge of the first data strobe indicates that valid data is available on the data bus. On a read cycle, the rising edge of the first data strobe indicates that data has been accepted from the data bus.
DTACK*	DATA TRANSFER ACKNOWLEDGE - An open-collector driven signal generated by a SLAVE. The falling edge of this signal indicates that valid data is available on the data bus during a read cycle, or that data has been accepted from the data bus during a write cycle. The rising edge indicates when the SLAVE has released the data bus at the end of a READ CYCLE.
GND	The DC voltage reference for the VMEbus system.
IACK*	INTERRUPT ACKNOWLEDGE - An open-collector or three state driven signal used by an INTERRUPT HANDLER acknowledging an interrupt request. It is routed, via a backplane signal trace, to the IACKIN* pin of slot 1 , where it is monitored by the IACK DAISY-CHAIN DRIVER.
IACKIN*	INTERRUPT ACKNOWLEDGE IN - A totem-pole driven signal. The IACKIN* and IACKOUT* signals form a daisy chain. The IACKIN* signal indicates to the VMEbus board receiving it that it is allowed to respond to the INTERRUPT ACKNOWLEDGE CYCLE that is in progress.
IACKOUT*	INTERRUPT ACKNOWLEDGE OUT - A totem-pole driven signal. The IACKIN* and IACKOUT* signals form a daisy-chain. The IACKOUT* signal is sent by a board to indicate to the next board in the daisy-chain that it is allowed to respond to the INTERRUPT ACKNOWLEDGE CYCLE that is in progress.
IRQ1*-IRQ7*	INTERRUPT REQUEST (1-7) - Open-collector driven signals, generated by an INTERRUPTER, which carry interrupt requests. When several lines are monitored by a single INTERRUPT HANDLER the highest numbered line is given the highest priority.

LWORD*	LONGWORD - A three-state driven signal used in conjunction with DS0*, DS1*, and A01 to select which byte location(s) within the 4 byte group are accessed during the data transfer.
RESERVED	RESERVED - A signal line reserved for future VMEbus enhancements. This line MUST NOT be used.
SERCLK	SERIAL CLOCK - A totem-pole driven signal which is used to synchronize the data transmission on the VMSbus.
SERDAT*	SERIAL DATA - An open-collector driven signal which is used for VMSbus data transmission.
SYSCLK	SYSTEM CLOCK - A totem-pole driven signal which provides a constant 16-MHz clock signal that is independent of any other bus timing.
SYSFAIL*	SYSTEM FAIL - An open-collector driven signal that indicates that a failure has occurred in the system. This signal may be generated by any board on the VMEbus.

APPENDIX C

USE OF THE SERCLK AND SERDAT* LINES

Two signal lines on the VMEbus backplane (SERCLK and SERDAT*) are designated for use by the VMSbus and provide a serial communication link between boards. The protocol used on the VMSbus is outside the scope of this document. Since system controller board designers will want to include circuitry on their boards to drive SERCLK, this appendix provides the necessary information.

SERCLK, like SYSCLK has no fixed timing relationship with any VMEbus signal (except SERDAT* which carries data bits that are synchronized to SERCLK).

The drivers and receivers for SERCLK and SERDAT* are specified in Chapter 7.

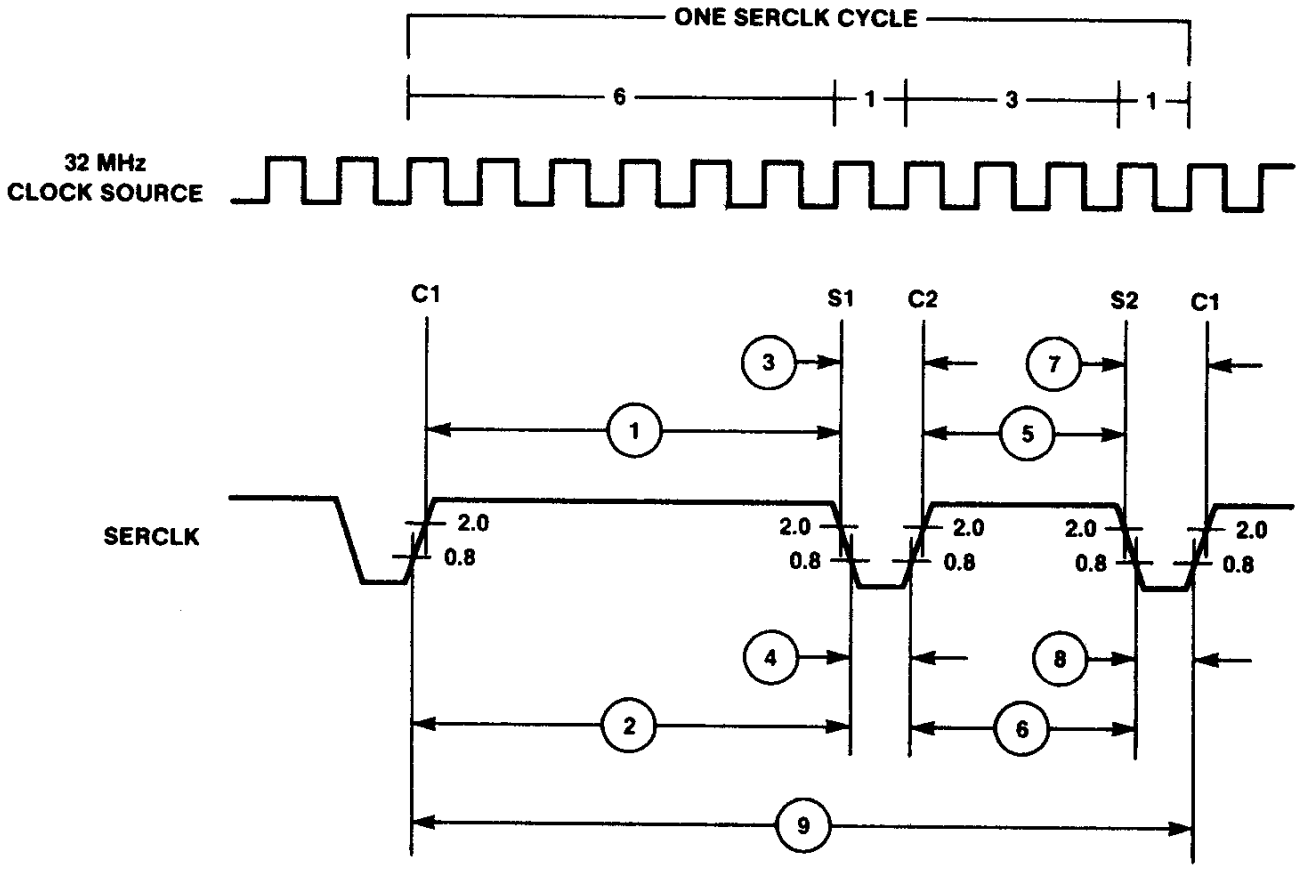
Figure C-1 and Table C-1 show the required timing parameters for the SERCLK signal line. A SERCLK waveform with these timing values can be derived from a 32 MHz clock source. The timing values in Table C-1 are for use when the SERCLK and SERDAT* lines are not extended beyond the VMEbus backplane. If these signals are extended to carry intersystem information, each of the timing values given in Table C-1 should be multiplied by a common scaling factor, greater than 1, to allow for the increased SERCLK and SERDAT* propagation times.

RECOMMENDATION C.1:
When designing the SERIAL CLOCK DRIVER module, take into account the fact that the SERCLK line driver's propagation delays for the rising and falling edges will likely be different. This difference is emphasized when the SERCLK line is heavily loaded. When calculating the driver's propagation delays, use the delays specified on the manufacturer's data sheet for a 300 pf capacitive load. If the only propagation delays given are for a 30 pf load, add 1 0 nSec to each of the propagation delays.

SUGGESTION C.1:
Design the SERIAL CLOCK DRIVER so that it can be jumpered to work from various stages of a binary counter that is driven by a 32 MHz clock source. This allows the selection of 32 MHz , 16 MHz , 8 MHz, etc., as the base frequency for the SERIAL CLOCK DRIVER and makes it easy to select a frequency appropriate for the length of the SERCLK and SERDAT* lines.

OBSERVATION C.1:
If a 32 MHz clock source is used to generate the SERCLK waveform, it can also be used to generate the VMEbus's 16 MHz SYSCLK signal.

SUGGESTION C.2:
To allow multiple boards that include SERIAL CLOCK DRIVER to be installed in the same backplane, design them with a jumper that disconnects the SERIAL CLOCK DRIVER from the SERCLK line.



Note:
See Table C-1 on the following page for timing values.

Figure C-1. SERCLK Timing Diagram

Table C-1. SERCLK Timing Values

PARAMETER NUMBER	MIN	MAX
1	167	
2		194
3		51
4	25	
5	74	
6		100
7		51
8	25	
9	340	347

Note:
All timing values are in nanoseconds.