M68KVMEB(D1)
MICROSYSTEMS

VME bus

Specification Manual

MOTOROLA

VME bus

Specification Manual

REV. A.
OCT. 1981

MOSTEK CORP.
MOTOROLA INC.
SIGNETICS/PHILIPS

NOTICE

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont'd)

LIST OF TABLES

CHAPTER 1

INTRODUCTION TO THE VME BUS SPECIFICATION

1.1 VME BUS SPECIFICATION OBJECTIVES

The VME bus specification defines an interfacing system for use in interconnecting data processing, data storage, and peripheral data control devices in a closely coupled configuration. The system has been conceived with the following objectives:

a. To provide communication between two devices on the VME bus without disturbing the internal activities of other devices interfaced to the VME bus.

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 VME bus.

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

d. To provide terminology and definitions that precisely describe system protocol.

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 VME BUS INTERFACE SYSTEM ELEMENTS

1.2.1 Basic Definitions

As an aid to understanding the material presented in this document, the following basic definitions are provided. More detailed definitions will be given in subsequent chapters as appropriate.

BACKPLANE	A printed circuit board which provides the interconnection path between other printed circuit cards.
SLOT	A single position at which a card may be inserted into the backplane. One slot may consist of more than one connector.
BOARD/CARD	Interchangeable terms representing one printed circuit board capable of being inserted into the backplane and containing a collection of electronic components.
MODULE	A collection of electronic components with a single functional purpose. More than one module may exist on the same card.

MASTER	A functional module capable of initiating data bus transfers. (Sometimes referred to as a "DTB MASTER" to emphasize its close association with the Data Transfer Bus.)
REQUESTER	A functional module capable of requesting control of the data transfer bus.
INTERRUPT HANDLER	A functional module capable of detecting interrupt requests and initiating appropriate responses.
MASTER SUB-SYSTEM	The combination of a MASTER, REQUESTER, and (optionally) an INTERRUPTER and/or an INTERRUPT HANDLER, which function together.

NOTE

All MASTERS, REQUESTERS, and INTERRUPT HANDLERS must be pieces of a MASTER SUB-SYSTEM.

SLAVE	A functional module capable of responding to data transfer operations generated by a MASTER. (Sometimes referred to as a "DTB SLAVE" to emphasize its close association with the Data Transfer Bus.)
INTERRUPTER	A functional module capable of requesting service from a MASTER SUB-SYSTEM by generating an interrupt request.
SLAVE SUB-SYSTEM	The combination of a SLAVE and INTERRUPTER which function together and which must be on the same card.

NOTE

All INTERRUPTERS must be part of either SLAVE SUB-SYSTEMS or MASTER SUB-SYSTEMS. However, SLAVES may exist as stand-alone elements. Such SLAVES will never be called SLAVE SUB-SYSTEMS.

CONTROLLER SUB-SYSTEM	The combination of modules used to provide utility and emergency signals for the VME bus. There will always be one and only one CONTROLLER SUB-SYSTEM. It may contain the following functional modules:

a. Data transfer bus ARBITER
b. System clock driver
c. System reset driver
d. Power monitor (for AC fail driver)
e. Bus time-out module

In any VME bus system, only one each of the above functional modules will exist. The slot numbered A1 is designated as the controller sub-system slot because the user will typically provide modules _a_ and _b_ on the board residing in this slot. The system reset driver is typically connected to an operator control panel and may be located elsewhere. The power monitor is interfaced to the incoming AC power source and may also be located remotely.

1.2.2 Basic VME Bus Structure

The VME bus interface system consists of four groups of signal lines called "buses", and a collection of "functional modules" which can be configured as required to interface devices to the buses. Figure 1-1 shows the elements of a typical VME bus system. The functional modules communicate with one another by means of bus signal lines provided by a backplane.

<u>NOTE</u>

> The "functional modules" defined in the specification are used as vehicles for discussion of the bus protocol, and need not be considered a constraint to logic design. For example, the designer may choose to design logic which interacts with the VME bus in the manner described, but uses different on-board signals.

The interface functions of the VME bus have been divided into four areas. Each functional area consists of a bus and associated functional modules which work together to perform specific duties within the system interface. Figure 1-2 illustrates the individual functional modules and buses contained within the VME bus definition, and each area is briefly summarized below.

a. Data Transfer

Devices transfer data over the Data Transfer Bus (DTB) which contains the data and address pathways and associated control signals. Functional modules called "DTB MASTERS" and "DTB SLAVES" use the DTB to transfer data between each other.

b. DTB Arbitration

Since the VME bus system may be configured with more than one DTB MASTER, a means must be provided to transfer control of the DTB between MASTERS in an orderly manner and to guarantee that only one MASTER controls the DTB at a given time. Bus arbitration is the area of the VME bus specification which defines the signals (Arbitration Bus) and modules (DTB REQUESTERS and DTB ARBITER) to perform the control transfer.

c. Priority Interrupt

The priority interrupt capability of the VME bus provides a means by which devices can request interruption of normal bus activity and can be serviced by an interrupt handler. These interrupt requests can be prioritized into a maximum of seven levels. The associated functional modules are called INTERRUPTERS and INTERRUPT HANDLERS, which use signal lines called the Interrupt Bus.

d. Utilities

The system clock, initialization, and failure detection have been grouped into the area of utilities. These functions include a clock line, system reset, system fail, and AC fail.

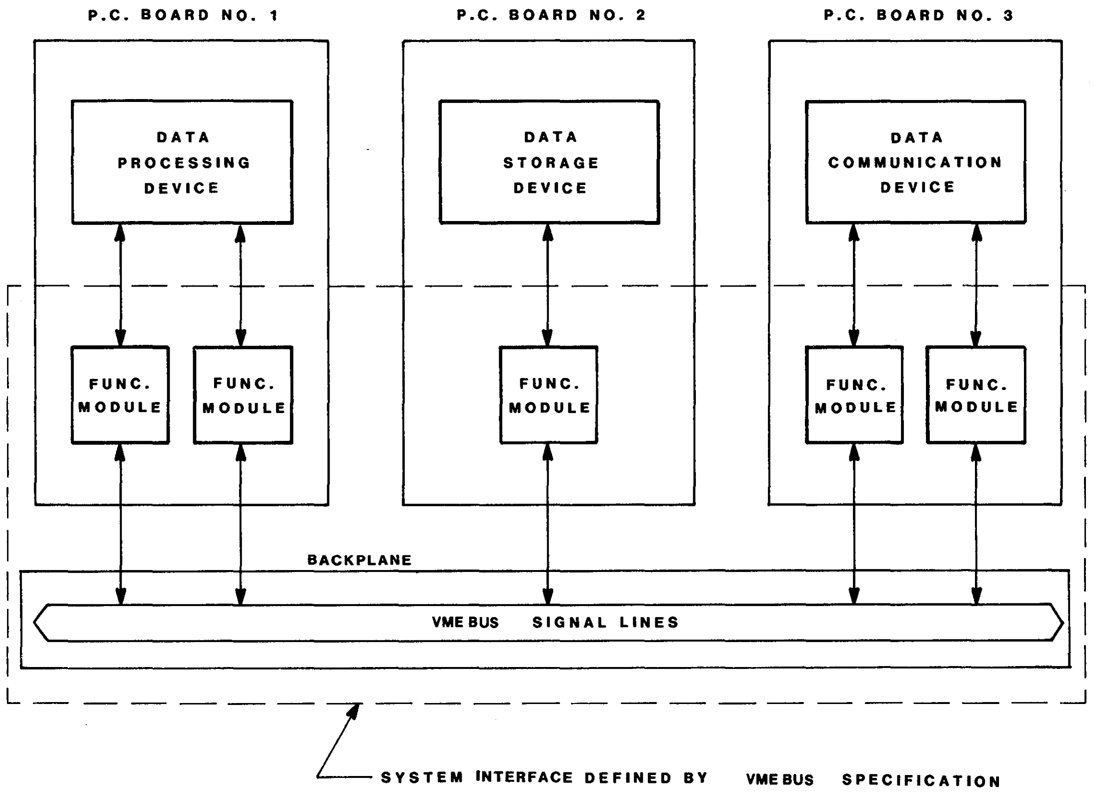


FIGURE 1-1. System Elements Defined by the VME Bus Specification

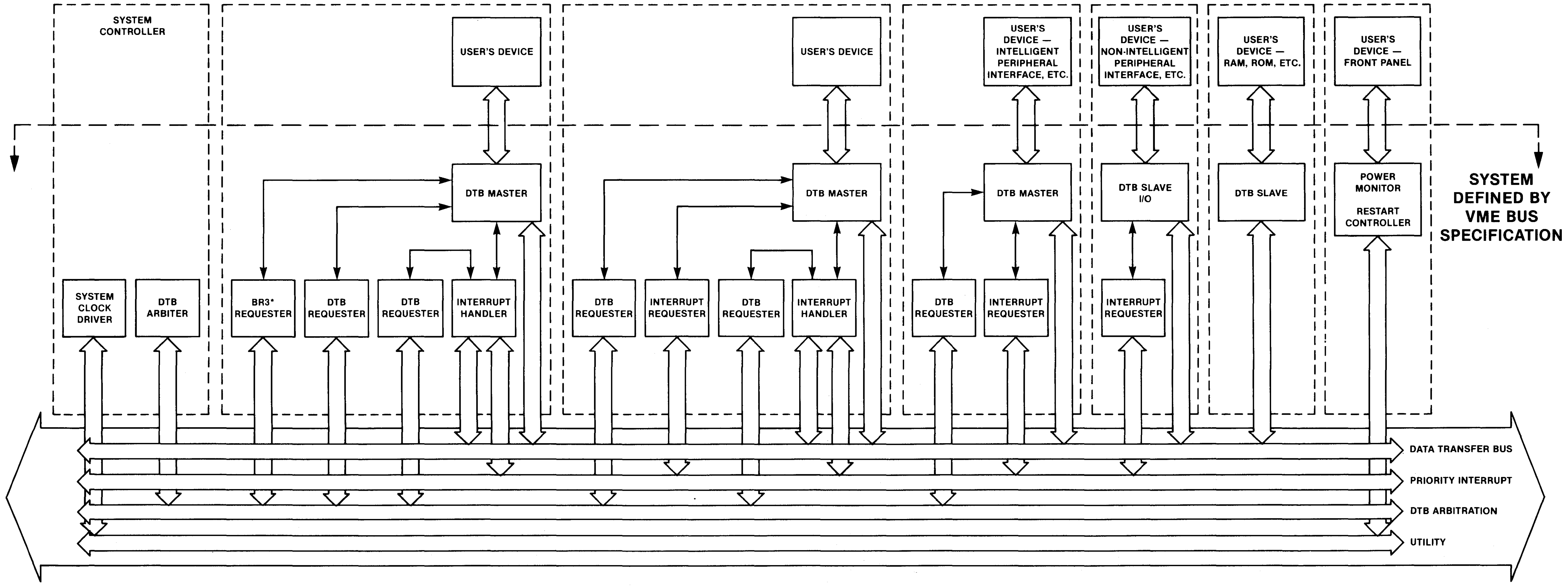


FIGURE 1-2. Functional Modules and Buses Contained Within the VME Bus Definition

1.3 VME BUS SPECIFICATION FORMAT

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

a. Timing diagram - shows the timing relationships of signal transitions. The times involved will have minimum and/or maximum limits associated with them.

b. Sequence diagram - is similar to a timing diagram and shows interlocked relationships of signal line transitions with respect to each other. This diagram is intended to show a sequence of events, rather than to specify the times involved.

c. Flow diagram - shows stream of events as they would occur during a VME bus operation. The events are stated in words and result from interaction of two or more functional modules. The flow diagram describes VME bus operations in a sequential manner and, at the same time, shows interaction of the functional modules.

Additional chapters include electrical specifications, mechanical specifications, and VME bus subset compatibility. Various "options" are defined in the chapters on the DTB, priority interrupt, and bus arbitration, and the compatibility between these options is analyzed in Chapter 6.

1.4 SPECIFICATION TERMINOLOGY

In some bus specifications, the protocol is treated on an abstract level. For example, it might be said that Device A "sends a message" to Device B. While this does allow a protocol to be defined in an application independent manner, the VME bus specification is more closely related to the physical implementation. It describes the protocol in terms of levels and transitions on bus lines.

1.4.1 Signal Line States

A signal line is always assumed to be in one of two <u>levels</u> or in <u>transition</u> between these levels. Whenever the term "<u>high</u>" is used, it refers to a high TTL voltage level ($\geq$ + 2.0 V). The term "<u>low</u>" refers to a low TTL voltage level ($\leq$ + 0.8 V). A signal line is "in transition" when its voltage is moving between + 0.8V and + 2.0V.

There are two possible transitions which can appear on a signal line, and these will be referred to as "<u>edges</u>". A <u>rising edge</u> is defined as the time period during which a signal line makes its transition from a low level to a high level. The <u>falling edge</u> is defined as the time period during which a signal line makes its transition from a high level to a low level.

1.4.2 Use of Asterisk (*)

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

a. An asterisk (*) following the signal name for 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 for signals which are edge significant denotes that the actions initiated by that signal occur on a high to low transition.

NOTE

The asterisk is inappropriate for the asynchronously running clock line SYSCLK. There is no fixed phase relationship between this clock line and other VME bus timing.

1.5 PROTOCOL SPECIFICATION

Each VME bus functional area - such as data transfer, bus arbitration, and priority interrupt - is defined via protocol specifications. Functional modules are defined for each area (Figures 1-1 and 1-2), and a protocol is defined for each module. The protocol is a set of rules governing the interaction of the module with the VME bus. A functional module communicates with another module by driving/receiving bus signals. The protocol governs these communications by determining:

a. when a module may drive and change the level of bus signals, and
b. when and how a module must respond to a bus signal.

Bus signals can be generally discussed in two classifications:

. Interlocked Bus Signals
. Broadcast Bus Signals

1.5.1 Interlocked Bus Signals

An interlocked bus signal is sent from a specific module to another specific module. The signal must be acknowledged by the receiving module. An interlocked relationship exists between the two modules until the signal is acknowledged. For example, an interrupt REQUESTER can send a signal asking for an interrupt. That signal must be answered at some time with an interrupt acknowledge signal (no time limit is prescribed by the VME bus specification).

Interlocked bus signals are dedicated to coordinating internal functions of the VME bus system, as opposed to interacting with external stimuli. Each interlocked signal has an internal source module and destination module. Also, these signals have timing specifications associated with them to assure proper bus operation.

Of significant importance are the interlocked bus signals used to coordinate transfer of addresses and data. Addresses and data cannot be considered "signals" in the strictest sense because they are not "sent" from one device to another. Instead, they are "placed" on a bus, while a separate bus signal (called a strobe) is sent to indicate their presence on the bus. The actual addresses or data have no effect on the protocol - that is, the specific address or data on the bus does not affect the strobe; however, the timing sequence (i.e., set-up time) between the specific address or data being placed on the bus and the sending of the accompanying strobe signal is important. Whenever this relationship is important, it is emphasized in the protocol definition.

An example of a pair of interlocked signals is DS0* (or DS1*) and DTACK*, which provide interlocking between an addressed SLAVE and the active MASTER.

1.5.2 Broadcast Bus Signal

A broadcast bus signal can be placed on the bus by a module in the system in response to an external event. There is no prescribed protocol for acknowledging a broadcast signal. Instead, the broadcast is maintained for a minimum specified time period long enough to assure that all appropriate modules will detect the signal. Broadcast signals may also be monitored from outside the system to gain information about system status. The broadcast signal can be given at any time, irrespective of any other activity taking place on the bus.

Since the broadcast signal has no interlocked relationship with other bus signals, a dedicated line must be provided for each broadcast signal type. These lines are used for functions such as system reset and power failure sequencing. These activities also differ from interlocked signals because the modules that generate broadcast signals do not address another specific module, but announce special conditions to all modules.

1.6 SYSTEM EXAMPLES AND EXPLANATIONS

The protocol specification is, of necessity, centered around specific modules; it describes how the module responds to signals, without discussing where these signals are generated. However, the protocol does not give the reader a good understanding of how various modules interact to accomplish overall system functions. Additional information must be presented to gain a broader perspective of VME bus functions. Therefore, system examples and explanatory descriptions are provided to give the VME bus user an understanding of VME bus capabilities from a system perspective. Care has been taken to differentiate between specification requirements and examples of typical system operation.

Each chapter begins with a discussion of the philosophy behind the particular bus function being discussed (subparagraph X.1). This is to give the reader an idea as to why the function is needed and how it is normally used.

The next subparagraph (X.2) describes the bus lines which are used by the modules to send or broadcast required bus messages.

The third subparagraph (X.3) introduces and defines the specific modules required within the protocol specification.

The next section (subparagraph X.4) provides examples of typical system operation sequences. These sequences do not necessarily outline every possible situation; however, the user can then understand the basic interaction between the functional modules on the bus.

Finally, additional details are provided (subparagraph X.5). These may be in text and/or timing diagram form with explanatory text.

1.7 ELECTRICAL/MECHANICAL SPECIFICATIONS

The electrical and mechanical specifications define the physical implementation of the VME bus. Chapter 7 contains the electrical specifications, including power distribution, signal characteristics, driver specifications, receiver specifications, and bus loading. Chapter 8 contains the mechanical specifications, including backplane, PC board, connectors, and pin references.

CHAPTER 2

VME BUS DATA TRANSFER

2.1 INTRODUCTION

The VME bus contains a high speed asynchronous parallel Data Transfer Bus (DTB), as shown in Figure 2-1. The DTB is used by a processor or Direct Memory Access (DMA) device to select the desired peripheral or memory location and to transfer data to or from that location. The DTB can be logically subdivided into address, data, and control line groups. The number of lines in each of these groups varies with the particular VME bus options selected by the user. There are four sets of DTB related options:

D8, D16, or D32	Data path width
A16, A24, or A32	Address path width
BTO(n)	Bus Time Out
SEQ	Sequential Access

2.1.1 DTB Options - Basic Description

Option D8 specifies that a SLAVE will read or write only eight bits at a time on D00-D07. Option D8 specifies that a MASTER will be capable of reading or writing via all 16 data lines D00-D15, but only eight in any one transfer.

Option D16 specifies that all data transfer activities supported by the module will be restricted to eight or sixteen bits. Option D32 specifies that the module (MASTER or SLAVE) will be capable of doing LONGWORD (32-bit) data transfers. Option D32 also requires an expanded bus system.

Option A16 specifies that a MASTER will place only short address AM codes and 15 bits of address on the bus. Option A16 specifies that a SLAVE will decode only address lines A01-A15, and it will respond only when a short address AM code is presented.

Option A24 specifies that all addresses generated by the MASTER or decoded by the SLAVE will be restricted to no more than 23 bits. Option A32 selection extends the address range to 31 bits. The address modifier lines indicate to SLAVES whether the address is 15, 23, or 31 bits. Option A32 also requires an expanded bus system.

For a MASTER which generates its own bus time-out, or for a bus time-out module, option BTO (n) specifies that the module will abort a data transfer cycle after "n" microseconds if no response is received from a SLAVE. This protects against bus lockups caused by an invalid address or a malfunctioning SLAVE.

Option SEQ specifies that the MASTER may request a sequential access transfer. Option SEQ specifies that the SLAVE will respond to a sequential access transfer.

In the following discussions of the Data Transfer Bus (DTB), the reader should ignore those comments which do not apply due to the particular option being considered.

For a detailed discussion of the constraints placed on user design by the selection of various options, see Chapter 6, VME BUS Options.

2.1.2 DTB Operation

Each data transfer on the DTB occurs between a functional DTB MASTER (see paragraph 2.3, Functional Modules) and a functional DTB SLAVE. Each data transfer is initiated by the DTB MASTER (see Figures 2-2 and 2-3). The addressed SLAVE must then acknowledge the transfer. The asynchronous nature of the DTB allows the SLAVE to control the amount of time taken for the transfer. After receiving the transfer acknowledge, the DTB MASTER terminates the data transfer cycle.

2.2 DATA TRANSFER BUS LINE STRUCTURES

2.2.1 Address Lines

Depending on the options chosen, the MASTER must drive the following lines:

15, 23, or 31	address lines (A01 through A15 if option A16) (A01 through A23 if option A24) (A01 through A31 if option A32)
6	address modifier lines (no change with any option selected)

The smallest addressable unit of storage is capable of storing eight bits of binary data. Each 8-bit group is called a "byte", and the location in which the byte is stored is called a "byte location". Two consecutive byte locations (even byte address and the next higher sequential odd byte address) comprise a "word location". A MASTER accesses a byte or word location by placing its binary "word address" on the address bus. The address lines on the bus are numbered starting with A01 instead of A00 to emphasize the fact that byte location addressing is done with data strobe lines instead of an "A00" line.

The address modifier lines allow the MASTER to pass additional information to the SLAVE during data transfer. This information may be used in several ways.

SYSTEM PARTITIONING

SLAVES in the system may be configured (either dynamically or statically) to respond to a single address modifier code. If there are several MASTERS on the VME bus, each may be assigned a code to be used when accessing the SLAVES. This allows the system to be partitioned and prevents a single malfunctioning MASTER from taking the whole system down.

MEMORY MAP SELECTION

SLAVES may be designed to respond at different addresses, depending upon the address modifier received. This allows the MASTER using the bus to place the system resources in selected map locations (or eliminate them from the map) by providing different address modifier codes.

PRIVILEGED ACCESS

Because SLAVES could be designed to respond to some address modifiers and not to others, it is possible to establish a large number of privilege levels. Each MASTER would provide an address modifier indicating its privilege level when accessing a SLAVE. If the SLAVE did not receive an appropriate AM code, it would not respond.

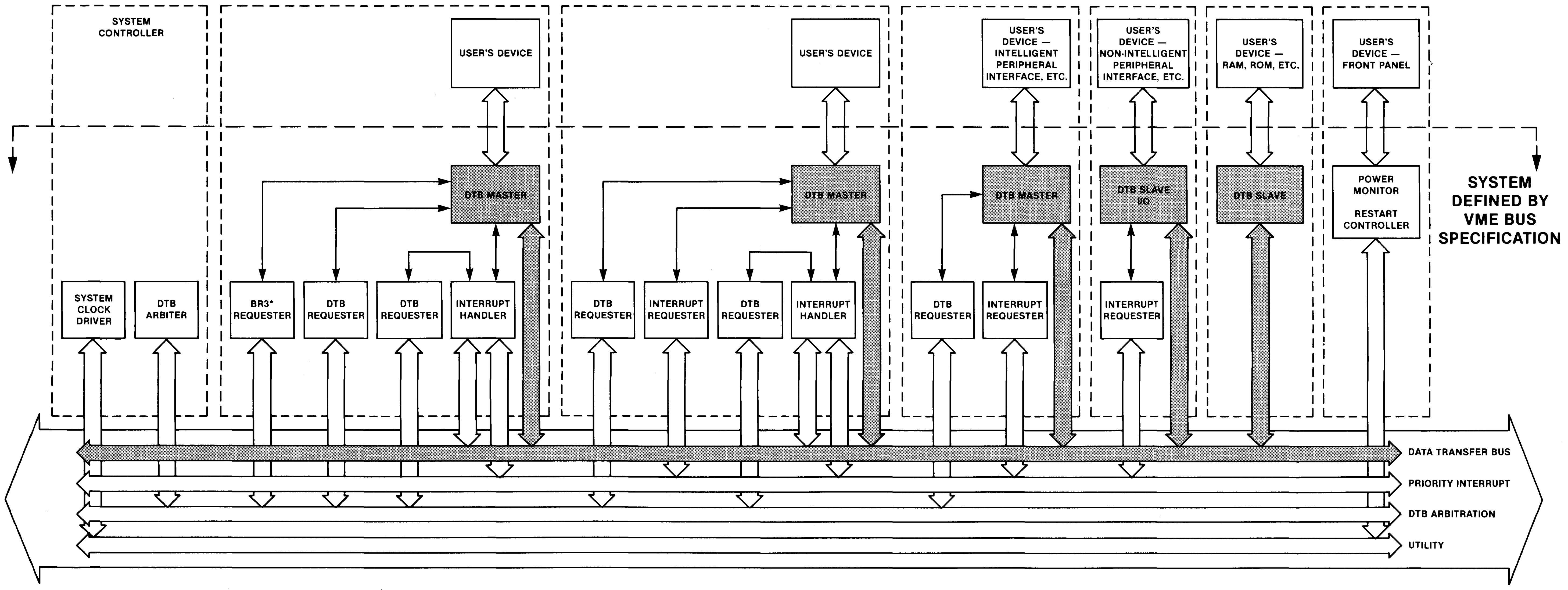
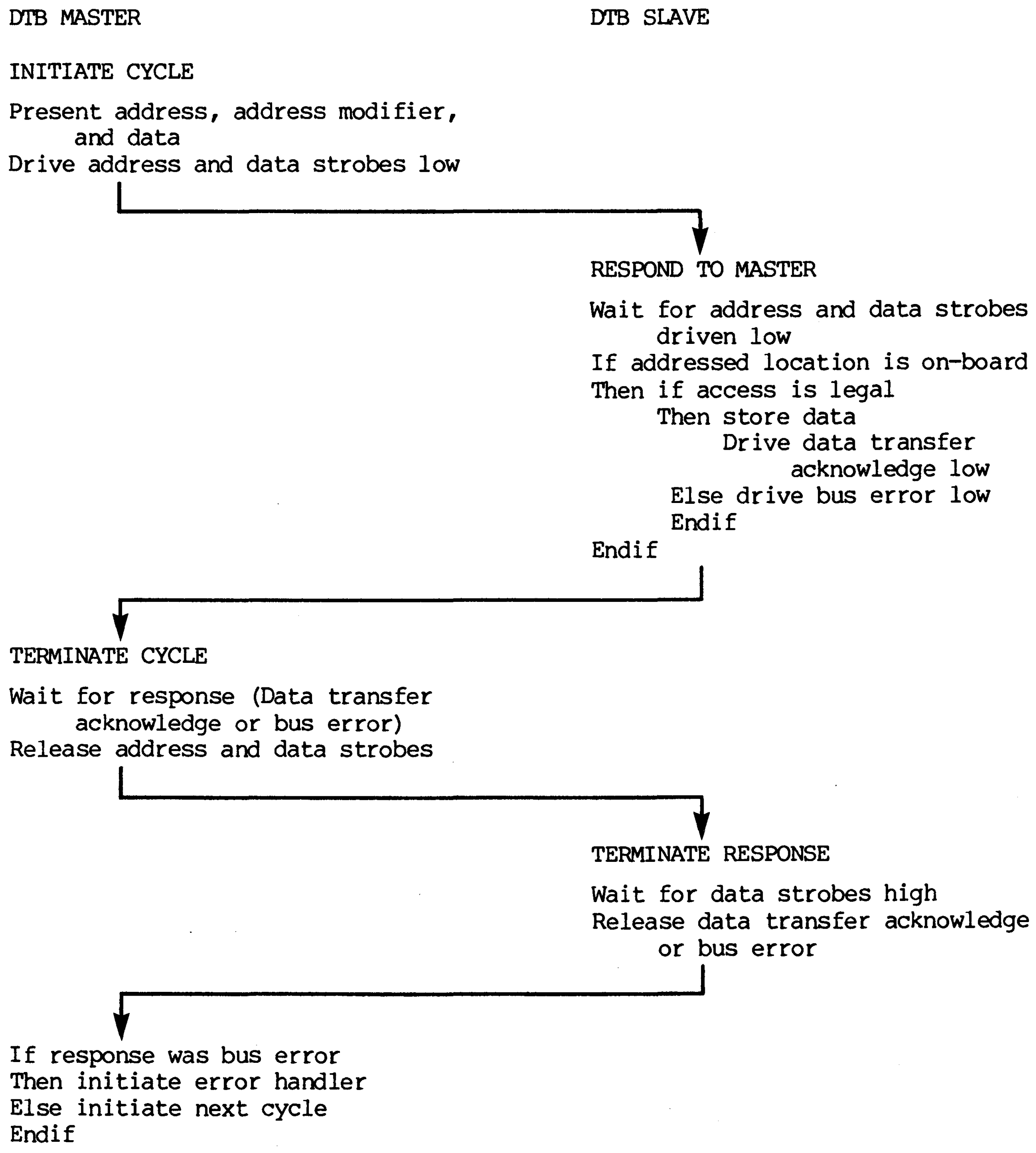


FIGURE 2-1. VME Bus Data Transfer Functional Block Diagram



DTB MASTER

DTB SLAVE

INITIATE CYCLE

```
Present address, address modifier,
     and data
Drive address and data strobes low
```

RESPOND TO MASTER

```
Wait for address and data strobes
     driven low
If addressed location is on-board
Then if access is legal
     Then store data
          Drive data transfer
               acknowledge low
      Else drive bus error low
      Endif
Endif
```

TERMINATE CYCLE

```
Wait for response (Data transfer
     acknowledge or bus error)
Release address and data strobes
```

TERMINATE RESPONSE

```
Wait for data strobes high
Release data transfer acknowledge
     or bus error
```

```
If response was bus error
Then initiate error handler
Else initiate next cycle
Endif
```

FIGURE 2-2. Typical Write

```
DTB MASTER                                   DTB SLAVE

INITIATE CYCLE

Present address and address modifier
Drive address and data strobes low

                                             RESPOND TO MASTER

                                             Wait for address and data strobes
                                                  driven low
                                             If addressed location is on-board
                                             Then if access is legal
                                                  Then present data
                                                       Drive data transfer
                                                            acknowledge low
                                                  Else drive bus error low
                                                  Endif
                                             Endif

TERMINATE CYCLE

Wait for response (Data transfer
     acknowledge or bus error)
If response is data transfer acknowledge
Then store data
Endif
Release address and data strobes

                                             TERMINATE RESPONSE

                                             Wait for data strobes high
                                             If data lines driven
                                             Then release data lines
                                             Endif
                                             Release data transfer acknowledge
                                                  or bus error

If response was bus error
Then initiate error handler
Else initiate next cycle
Endif
```

FIGURE 2-3. Typical Read

CYCLE TYPE

The AM codes can be used to specify a special type of transfer cycle. The VME bus specifies one special cycle type. The user could use additional codes to specify others. This special cycle type is a sequential access cycle. There are four sequential access AM codes, and when one of these is placed on the bus, the memory boards in the system latch the address into a counter and appropriately increment the counter after each odd-byte, word, or LONGWORD transfer.

DISTRIBUTED MEMORY MANAGEMENT

Memory management logic is often used in systems to allocate and translate memory segments dynamically. A collection of these segments is assigned to each active task. Each time the real-time executive switches from one task to the next, it must either change the contents of the segment registers or select another set of segment registers (the latter approach being much faster).

Address modifier codes may be used as segment register selectors. In this case, the MASTER places AM codes on the bus which indicate to memory management logic on the slave boards which set of segment registers should be used.

ADDRESSING RANGE

The VME bus provides 31 address lines to allow direct addressing to over four billion bytes. For most SLAVES, however, the extra logic required to decode all 31 address lines is a needless expense. For this reason, the VME bus defines three address ranges:

Short addressing	64K bytes
Standard addressing	16M bytes
Extended addressing	4g bytes

A group of address modifier codes is set aside for each type of addressing. SLAVES receiving a short address AM code ignore the upper 16 address lines (A16-A31). SLAVES receiving a standard address AM code ignore the upper eight lines (A24-A31). When receiving an extended address AM code, the SLAVE decodes all 31 address lines.

Slave boards which do not decode address lines A24-A31 should not respond to extended address AM codes. Slave boards which do not decode address lines A16-A31 should not respond to either extended or standard AM codes.

Table 2-1 lists all of the 64 possible address modifier codes and classifies each into one of three categories:

DEFINED BY:
- VME Bus Spec.
- USER
- RESERVED

TABLE 2-1. Address Modifier Codes

HEXADECIMAL CODE	ADDRESS MODIFIER 5	4	3	2	1	0	FUNCTION	DEFINED BY
3F	H	H	H	H	H	H	Standard Supervisory Ascending Access	VME Bus Spec.
3E	H	H	H	H	H	L	Standard Supervisory Program access	VME Bus Spec.
3D	H	H	H	H	L	H	Standard Supervisory Data Access	VME Bus Spec.
3C	H	H	H	H	L	L	Undefined	Reserved
3B	H	H	H	L	H	H	Standard Non-Privileged Ascending Access	VME Bus Spec.
3A	H	H	H	L	H	L	Standard Non-Privileged Program Access	VME Bus Spec.
39	H	H	H	L	L	H	Standard Non-Privileged Data Access	VME Bus Spec.
38	H	H	H	L	L	L	Undefined	Reserved
30-37	H	H	L	X	X	X	Undefined	Reserved
2F	H	L	H	H	H	H	Undefined	Reserved
2E	H	L	H	H	H	L	Undefined	Reserved
2D	H	L	H	H	L	H	Short Supervisory I/O Access	VME Bus Spec.
2C	H	L	H	H	L	L	Undefined	Reserved
2B	H	L	H	L	H	H	Undefined	Reserved
2A	H	L	H	L	H	L	Undefined	Reserved
29	H	L	H	L	L	H	Short Non-Privileged I/O Access	VME Bus Spec.
28	H	L	H	L	L	L	Undefined	Reserved
20-27	H	L	L	X	X	X	Undefined	Reserved
10-1F	L	H	X	X	X	X	Undefined	User
0F	L	L	H	H	H	H	Extended Supervisory Ascending Access	VME Bus Spec.
0E	L	L	H	H	H	L	Extended Supervisory Program Access	VME Bus Spec.
0D	L	L	H	H	L	H	Extended Supervisory Data Access	VME Bus Spec.
0C	L	L	H	H	L	L	Undefined	Reserved
0B	L	L	H	L	H	H	Extended Non-Privileged Ascending Access	VME Bus Spec.
0A	L	L	H	L	H	L	Extended Non-Privileged Program Access	VME Bus Spec.
09	L	L	H	L	L	H	Extended Non-Privileged Data Access	VME Bus Spec.
08	L	L	H	L	L	L	Undefined	Reserved
00-07	L	L	L	X	X	X	Undefined	Reserved

TABLE 2-1. Address Modifier Codes (cont'd)

NOTES:

Short address uses 15 address lines (A01-A15).
Standard address uses 23 address lines (A01-A23).
Extended address uses 31 address lines (A01-A31)

Codes defined by the "VME Bus Spec." should not be used for purposes other than those specified.

Codes defined by "User" may be used for any purpose which the VME Bus user (board vendor or customer) deems appropriate (page switching, memory protection, MASTER or task identification, privileged access to resources, etc.).

Codes defined by "Reserved" should not be used by the user; they are reserved for system use and future enhancements.

Those defined by the VME bus Spec are intended for a specific purpose and may not be used for any other. These include codes for sequentially accessing memory (ascending access), and two I/O access codes (one used only by the highest privilege level software and one which may be used by any level).

Those codes which may be defined by the USER may be used for any of the purposes outlined above (e.g., system partitioning, distributed memory management, etc.).

Those codes labeled as RESERVED should not be used. They are intended for use in future system enhancements.

Because system requirements vary, it is important that VME bus compatible Eurocard vendors recognize the need for flexibility in decoding the address modifiers on the slave boards. Decoding should allow the customer to configure the board to respond to the codes required for his system. A simple way to do this is to route the AM lines into the address lines of a PROM (e.g., 82S129). If this part is socketted, the customer can program a PROM which gives the decoding required by his system.

It is recommended that non-I/O SLAVES be shipped in a configuration which responds to AM codes 39, 3A, 3D, and 3E if they decode 23 address lines, and AM codes 09, 0A, 0D, and 0E if they decode 31 address lines. Short addressed I/O SLAVES should be configured to respond at least to AM code 2D.

It would also be possible to design slave boards with segment registers on them to allow dynamic allocation and translation of their on-board resources. If this is done, the segment registers should be addressable only in the supervisory I/O map (i.e., AM code 2D).

Master boards must, of course, drive the AM lines. If the master board has a real-time executive, that executive may need to control the AM code placed on the VME bus during bus accesses. A good way to do this is to provide latches on the master board which may be written into by the executive each time a context switch is made from task to task. The AM code may then be used to partition the system or to indicate the task's privilege level.

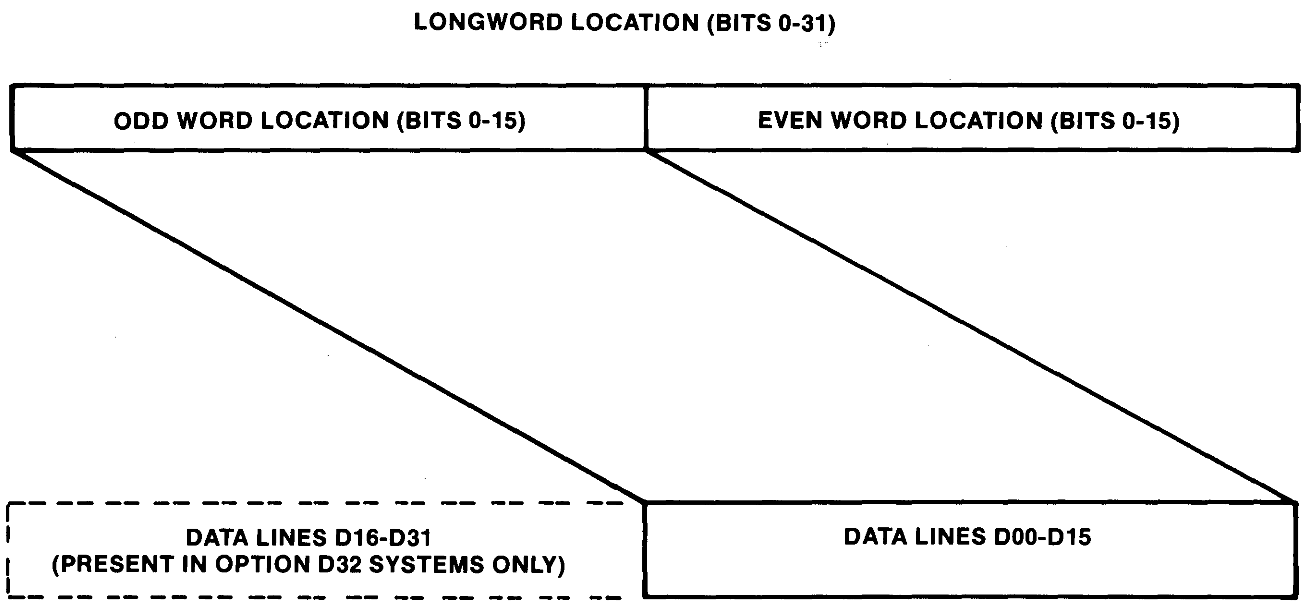
2.2.2 Data Transfer Lines

Depending on the options chosen and the type of data transfer, the source of the data (MASTER or SLAVE) must drive the following data related lines:

```
                                (D00 through D07/D08 through D15 if option D8)
 8, 16, or 32       data lines  (D00 through D15 if option D16)
                                (D00 through D31 if option D32)
```

A word transfer requires the driving of data lines D00 through D15. On a LONGWORD transfer, data lines D00 through D31 require driving. Note that word transfers always use the same 16 lines (i.e., D00 through D15), while byte transfers use different lines for odd and even bytes.

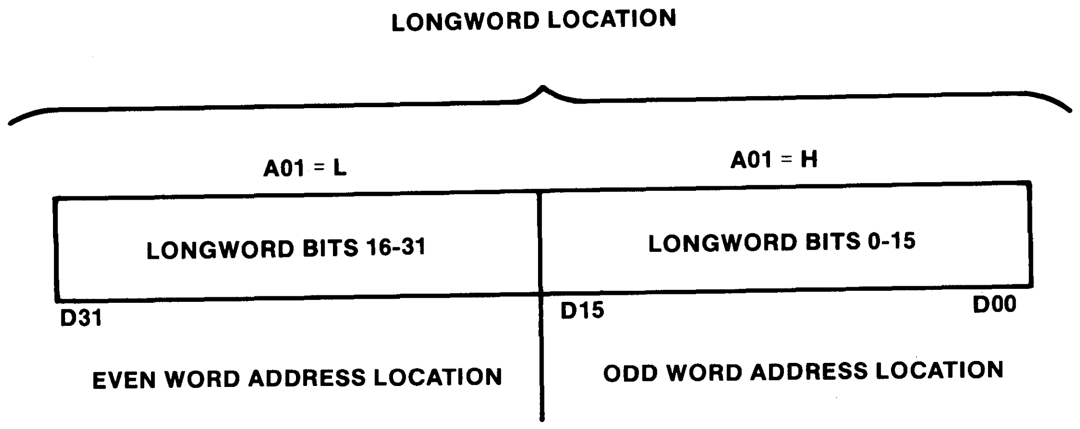
Only LONGWORD transfers use lines D16 through D31. When LONGWORD transfers take place, the address presented is the even word address in the LONGWORD location (A01 is low). There are two word addresses for each LONGWORD. When accessing the data stored in a LONGWORD location on a word basis, the even word address (A01 low) corresponds to the LONGWORD data bits D16 through D31. Likewise, the structure of the bytes within a word have the even byte (selected by DS1*) as the most significant eight bits of the word (see Figures 2-4, 2-5, and 2-6).


LONGWORD LOCATION (BITS 0-31)
ODD WORD LOCATION (BITS 0-15)
EVEN WORD LOCATION (BITS 0-15)
DATA LINES D16-D31
(PRESENT IN OPTION D32 SYSTEMS ONLY)
DATA LINES D00-D15

FOR A WORD ACCESS, THE DATA IS ALWAYS TRANSFERRED ON D00-D15

FIGURE 2-4. Odd Word Location Accesses

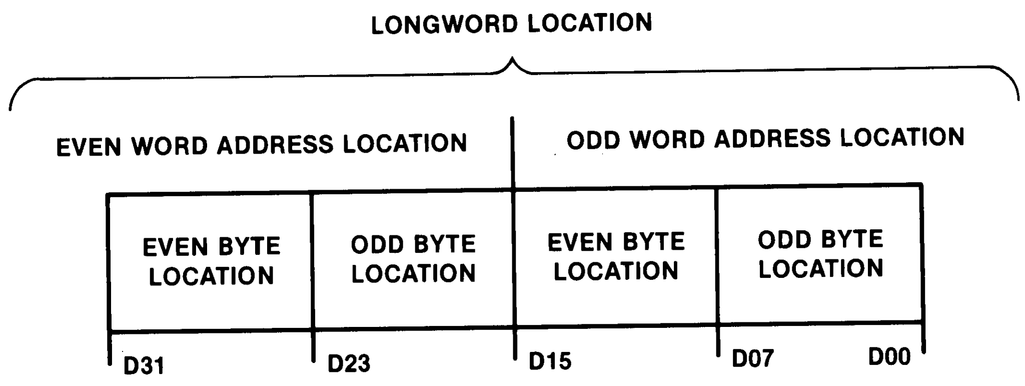
When LONGWORD data is written into memory, the 32 data bits are stored at a LONGWORD location. There are two word locations for each LONGWORD location. When reading the LONGWORD location on a word basis, the even word address (A01=0) corresponds to the LONGWORD data bits D16-D31. See Figure 2-5.



THE ODD WORD LOCATION (A01 = 1) CORRESPONDS
TO THE LONGWORD DATA BITS DOO-D15

FIGURE 2-5. Word Addressing of LONGWORD Locations

Two 8-bit bytes of data may be stored in each word location. The even byte location is defined as the eight most significant data bits of the word location. The odd byte location is defined as the eight least significant data bits of the word location. See Figure 2-6.



Byte Address	XX..X00	XX..X01	XX..X10	XX..X11
Byte Access				
LWORD*	high	high	high	high
A01	low	low	high	high
DS1*	low	high	low	high
DS0*	high	low	high	low
Word Access				
LWORD*	high	Note	high	Note
A01	low	1	high	1
DS1*	low		low	
DS0*	low		low	
LONGWORD Access				
LWORD*	low	Note	Note	Notes
A01	low	2	3	2 & 3
DS1*	low			
DS0*	low			

NOTES:

1. Not legal to access 16 bits of data on an odd byte address.

2. Not legal to access 32 bits of data on an odd byte address.

3. Not legal to access 32 bits of data on an odd word address.

FIGURE 2-6. Byte Location Numbering

2.2.3 Data Transfer Control Lines

The MASTER will drive the following lines:

AS*	Address strobe	(On all transfers)
DS0*	Odd data byte strobe	(Each is operation dependent, but at
DS1*	Even data byte strobe	least one must always be driven)
LWORD*	LONGWORD select	(Operation dependent)
WRITE*	Read/Write select	(Operation dependent)

The SLAVE will always drive the following lines:

BERR*	Bus error	(If error detected)
DTACK*	Data acknowledge to MASTER	(If data transfer was successful)

AS* is the address strobe. It informs all SLAVE modules that the address is now stable and may be clocked into holding registers. This type of operation is essential to certain devices which require setup time after an address has stabilized before the data can be accessed. It is recommended that all module operations be keyed to and timed by an address strobe.

DS0* and DS1* select the data to be transferred and, on a write transfer, strobe the transferred data. DS0* low means that the byte which would be addressed with A00 set high (the odd byte) is to be found on data lines D00 through D07. Likewise, DS1* low means that the byte which would be addressed with A00 set low (the even byte) is to be found on data lines D08 through D15. This explains why A00 does not exist as a signal line. These two lines replace the function that A00 would perform. The sender is not prohibited from driving the data lines which are not being strobed. The receiver should ignore any and all levels and/or transitions which occur on non-strobed data lines.

LWORD* specifies that 32 bits will be transferred on data lines D00 through D31. Certain special constraints are placed on various combinations of these signals. Table 2-2 specifies these constraints.

IACK* is the interrupt acknowledge line. During data transfers, it will be driven high by MASTERS or pulled high by the bus terminators, appearing as a high level to all SLAVES. If IACK* is low, the cycle is not a data transfer cycle and SLAVES should not decode the address or respond. See Chapter 4 on interrupts for more information.

TABLE 2-2. Data Transfer Control Table

LWORD*	A01	DS0*	DS1*	CONSTRAINT/ACTION
L	H	X	X	Illegal (not on proper boundary)
L	L	L	L	Long word transfer
L	L	L	H	Illegal (only one data strobe)
L	L	H	L	Illegal (only one data strobe)
L	L	H	H	Illegal (no data strobe)
H	H	L	L	Transfer both bytes of odd word
H	H	L	H	Transfer odd byte of odd word
H	H	H	L	Transfer even byte of odd word
H	H	H	H	Illegal (no data strobe)
H	L	L	L	Transfer both bytes of even word
H	L	L	H	Transfer odd byte of even word
H	L	H	L	Transfer even byte of even word
H	L	H	H	Illegal (no data strobe)

The reasons for the illegal cases are:

- a LONGWORD must fit on a 4-byte, double-word boundary (it is not possible to address it otherwise),
- a LONGWORD transfer must provide both data strobes, and all transfers must have one or both data strobes.

WRITE* controls the direction of data transfer between MASTER and SLAVE. If WRITE* is high (WRITE false), the operation is a transfer from SLAVE to MASTER. This can be expressed as the MASTER reads from the SLAVE. If WRITE* is low (WRITE true), the operation is a transfer from MASTER to SLAVE. This can be expressed as the MASTER writes to the SLAVE.

BERR* is the signal from the SLAVE to the MASTER which may be used to indicate that an error has been encountered in processing the request at the SLAVE. This could include an address map error, memory protect violation, or an illegal LONGWORD request. It is recommended that the SLAVE respond with BERR*:

- on all requests for LONGWORD data when bit A01 is high,
- on all LONGWORD requests made to a SLAVE incapable of accepting such requests,
- on all LONGWORD requests made without both data strobes.

2.3 FUNCTIONAL MODULES

The modules involved in a data transfer are always classified as a MASTER and a SLAVE. The MASTER is the module controlling the transfer, and the SLAVE is the responding or addressed module. Some boards may be designed with both MASTER and SLAVE modules. For example, a board containing a processor which requires VME bus access would contain a MASTER module. If the same board also contained memory accessible from the VME bus, it would also contain a SLAVE module.

As another example, a dedicated function processor such as a floating point processor or intelligent controller might be designed to receive commands through a SLAVE interface from a general purpose CPU for set-up, but it might then act as a MASTER to access memory as required to complete the command it has been given.

One additional functional module may exist in the system. This bus time-out (BTO) module monitors the DS0*, DS1*, DTACK*, and BERR* lines. When a high to low transition is detected on DS0* or DS1*, this module initiates a delay. If the delay expires without a corresponding high to low transition on either DTACK* or BERR*, or a low to high transition on DS0* or DS1*, this module will drive BERR* low, completing the data transfer handshake and flagging a bus error.

Three requirements are imposed on this module:

a. The BTO module may not drive BERR* to low prior to the falling edge of a data strobe.

b. The BTO module may not drive BERR* to low more than 30 nanoseconds after the fall of DTACK*, the fall of BERR*, or the rise of either data strobe.

c. The BTO module must release BERR* within 30 nanoseconds after the rise of either data strobe.

2.4 TYPICAL OPERATION

2.4.1 Data Transfer Bus Acquisition

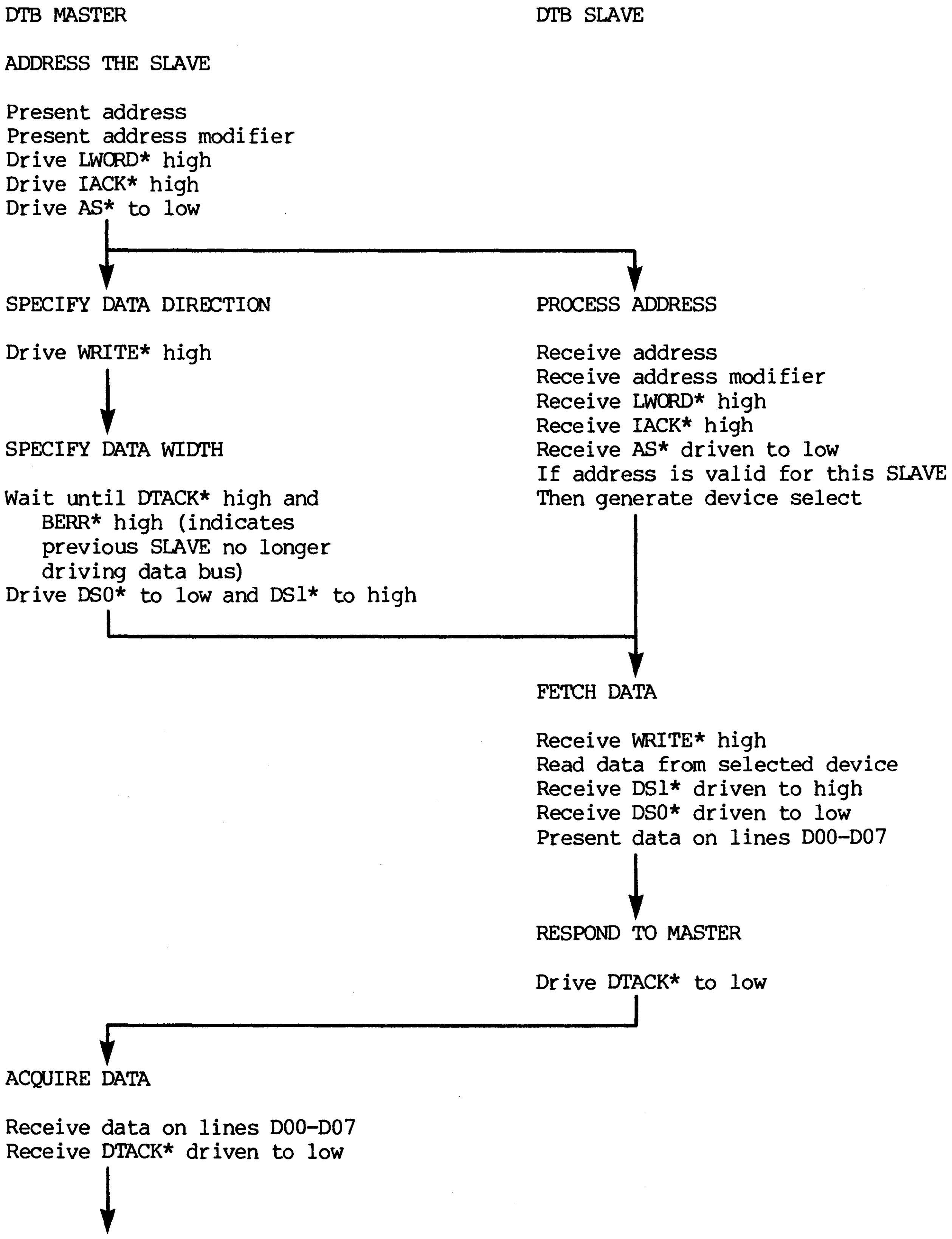
To perform a data transfer, the DTB MASTER must first acquire control of the DTB via its on-board DTB REQUESTER. The DTB REQUESTER requests the DTB ARBITER to grant use of the DTB. (See Chapter 3 for a description of DTB arbitration.) This arbitration is required because several MASTERS might want the DTB at the same time in a multi-processor configuration. The DTB ARBITER grants the DTB to the appropriate REQUESTER. When the DTB REQUESTER has received permission to use the DTB, it will inform its on-board MASTER.

Figure 2-7 shows a typical flow of a DTB byte read cycle, and should be referred to while reading the following cycle flow description.

To start the transfer, the DTB MASTER must first drive the address lines with the desired memory address and address modifier code. The MASTER must also specify a word or LONGWORD transfer. For the byte read cycle shown in Figure 2-7, LWORD* is driven high. Since it is not an interrupt acknowledge cycle, IACK* is driven high. All of these signal lines must be valid before AS* is driven to low.

The SLAVE determines whether the address is its own and whether the address modifier is appropriate. While this occurs, the MASTER drives WRITE* high to indicate a read. The MASTER must then ensure that the last cycle is complete and that the data bus is available by verifying that DTACK* and BERR* are released high. The MASTER may then request the odd byte data in the specified word location by driving DS0* low. (DS1* remains high.)

The SLAVE may then start the transfer because it knows the data word address and that the odd byte of the addressed location is to be read and placed on D00-D07. When the data has been placed on the data bus, the SLAVE acknowledges this by driving DTACK* low. The SLAVE must hold DTACK* low and the data valid as long as the data strobe is driven low.



DTB MASTER

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
previous SLAVE no longer
driving data bus)
Drive DS0* to low and DS1* to high

DTB SLAVE

PROCESS ADDRESS

Receive address
Receive address modifier
Receive LWORD* high
Receive IACK* high
Receive AS* driven to low
If address is valid for this SLAVE
Then generate device select

FETCH DATA

Receive WRITE* high
Read data from selected device
Receive DS1* driven to high
Receive DS0* driven to low
Present data on lines D00-D07

RESPOND TO MASTER

Drive DTACK* to low

ACQUIRE DATA

Receive data on lines D00-D07
Receive DTACK* driven to low

FIGURE 2-7. Data Transfer Bus, Byte Read Cycle (Sheet 1 of 2)

```
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* driven to high
Release D00-D07
```

```
ACKNOWLEDGE TERMINATION

Release DTACK*
```

<u>NOTE</u>

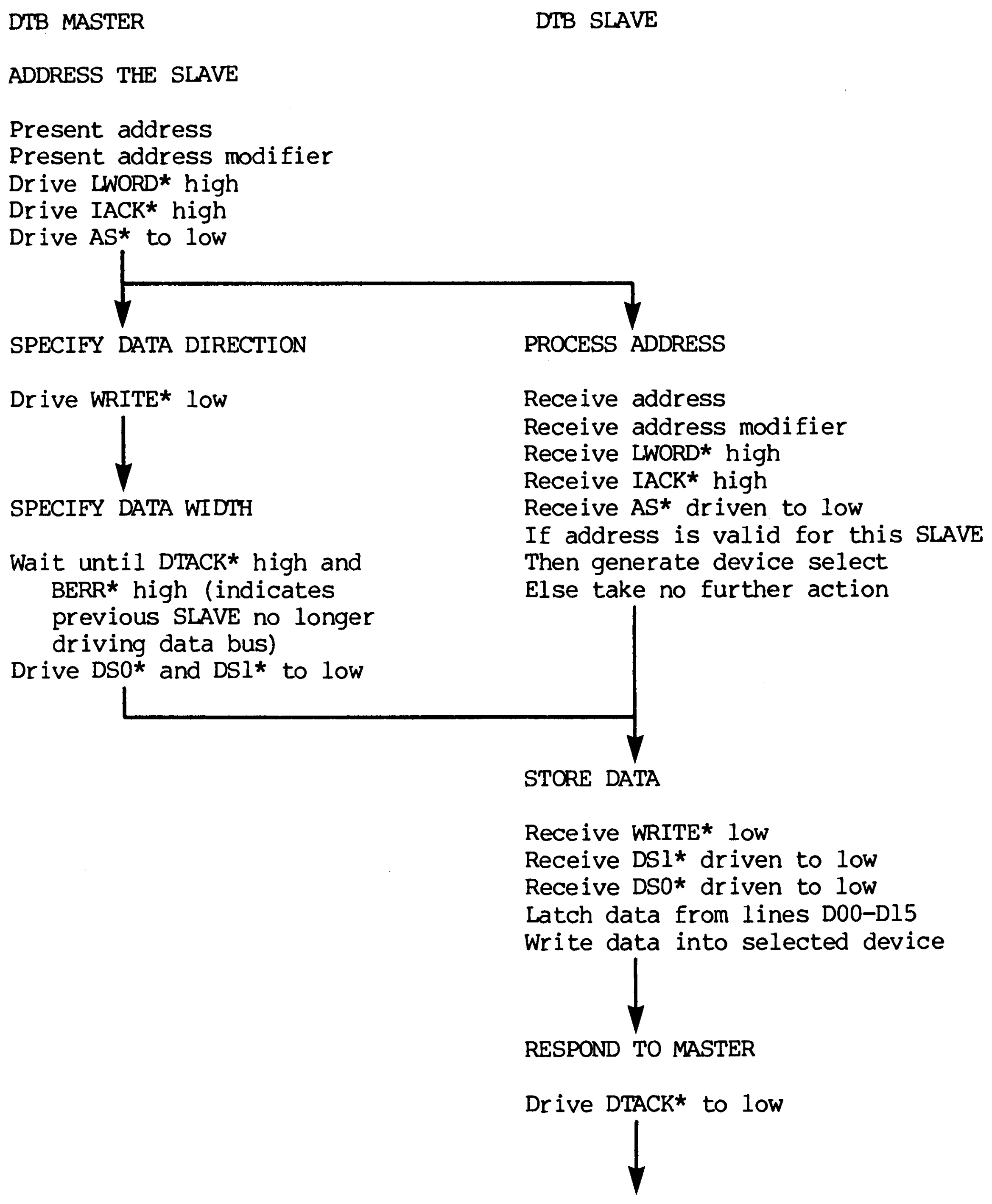
For simplicity, the assumption has been made that no transfer causes a bus error.

FIGURE 2-7. Data Transfer Bus, Byte Read Cycle (Sheet 2 of 2)

When the MASTER receives DTACK* driven to low, it captures the data on D00-D07. When this has been done, the MASTER terminates the cycle by releasing the address lines, driving DS0* and AS* to high, and then releasing DS0*, DS1*, and AS*.

The SLAVE responds to the cycle termination by releasing D00-D07 and releasing DTACK* to high.

The cycle flow for word and LONGWORD data transfers is very similar to the byte cycle. Flow charts for these cycles are shown in Figures 2-8 and 2-9.



DTB MASTER

ADDRESS THE SLAVE

Present address
Present address modifier
Drive LWORD* high
Drive IACK* high
Drive AS* to low

SPECIFY DATA DIRECTION

Drive WRITE* low

SPECIFY DATA WIDTH

Wait until DTACK* high and BERR* high (indicates previous SLAVE no longer driving data bus)
Drive DS0* and DS1* to low

DTB SLAVE

PROCESS ADDRESS

Receive address
Receive address modifier
Receive LWORD* high
Receive IACK* high
Receive AS* driven to low
If address is valid for this SLAVE
Then generate device select
Else take no further action

STORE DATA

Receive WRITE* low
Receive DS1* driven to low
Receive DS0* driven to low
Latch data from lines D00-D15
Write data into selected device

RESPOND TO MASTER

Drive DTACK* to low

FIGURE 2-8. Data Transfer Bus, Word Write Cycle (Sheet 1 of 2)

TERMINATE CYCLE

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

END TERMINATION

```
If last cycle then
   Release DS0* and DS1*
   Release AS*
Else go to ADDRESS THE SLAVE
Endif
```

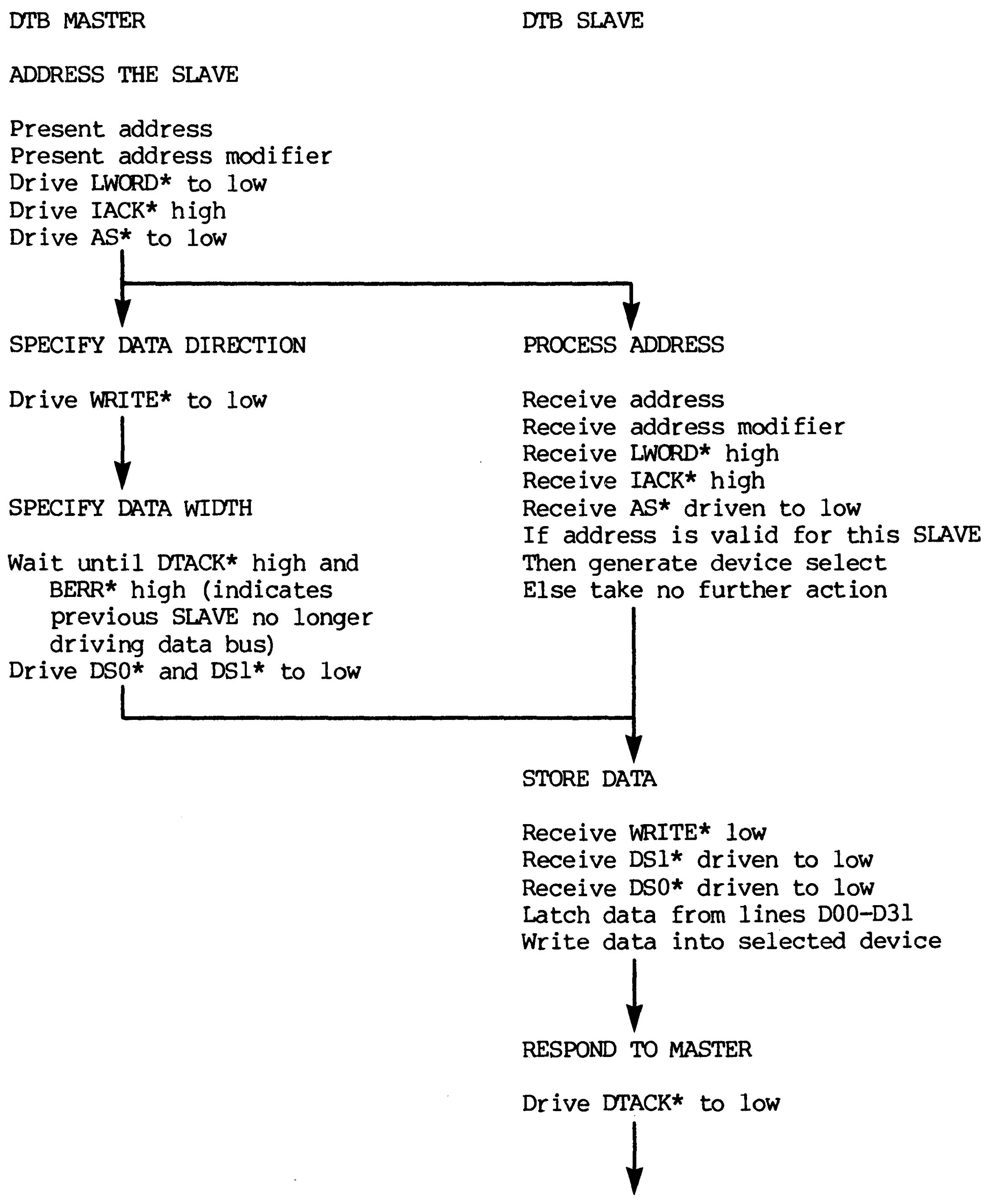
ACKNOWLEDGE TERMINATION

```
Receive AS*, DS0*, and DS1* driven to high
Release DTACK*
```

<u>NOTE</u>

For simplicity, assumption has been made that no transfer causes a bus error.

FIGURE 2-8. Data Transfer Bus, Word Write Cycle (Sheet 2 of 2)



DTB MASTER

ADDRESS THE SLAVE

Present address
Present address modifier
Drive LWORD* to low
Drive IACK* high
Drive AS* to low

SPECIFY DATA DIRECTION

Drive WRITE* to low

SPECIFY DATA WIDTH

Wait until DTACK* high and
BERR* high (indicates
previous SLAVE no longer
driving data bus)
Drive DS0* and DS1* to low

DTB SLAVE

PROCESS ADDRESS

Receive address
Receive address modifier
Receive LWORD* high
Receive IACK* high
Receive AS* driven to low
If address is valid for this SLAVE
Then generate device select
Else take no further action

STORE DATA

Receive WRITE* low
Receive DS1* driven to low
Receive DS0* driven to low
Latch data from lines D00-D31
Write data into selected device

RESPOND TO MASTER

Drive DTACK* to low

FIGURE 2-9. Data Transfer Bus, LONGWORD Write Cycle (Sheet 1 of 2)

TERMINATE CYCLE

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

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* driven to high
Release DTACK*
```

NOTE
For simplicity, the assumption has been made that no transfer causes a bus error.

FIGURE 2-9. Data Transfer Bus, LONGWORD Write Cycle (Sheet 2 of 2)

2.5 FORMAL SPECIFICATIONS

The following text defines specifications for data movements on the VME bus.

2.5.1 Data Transfer Bus Acquisition

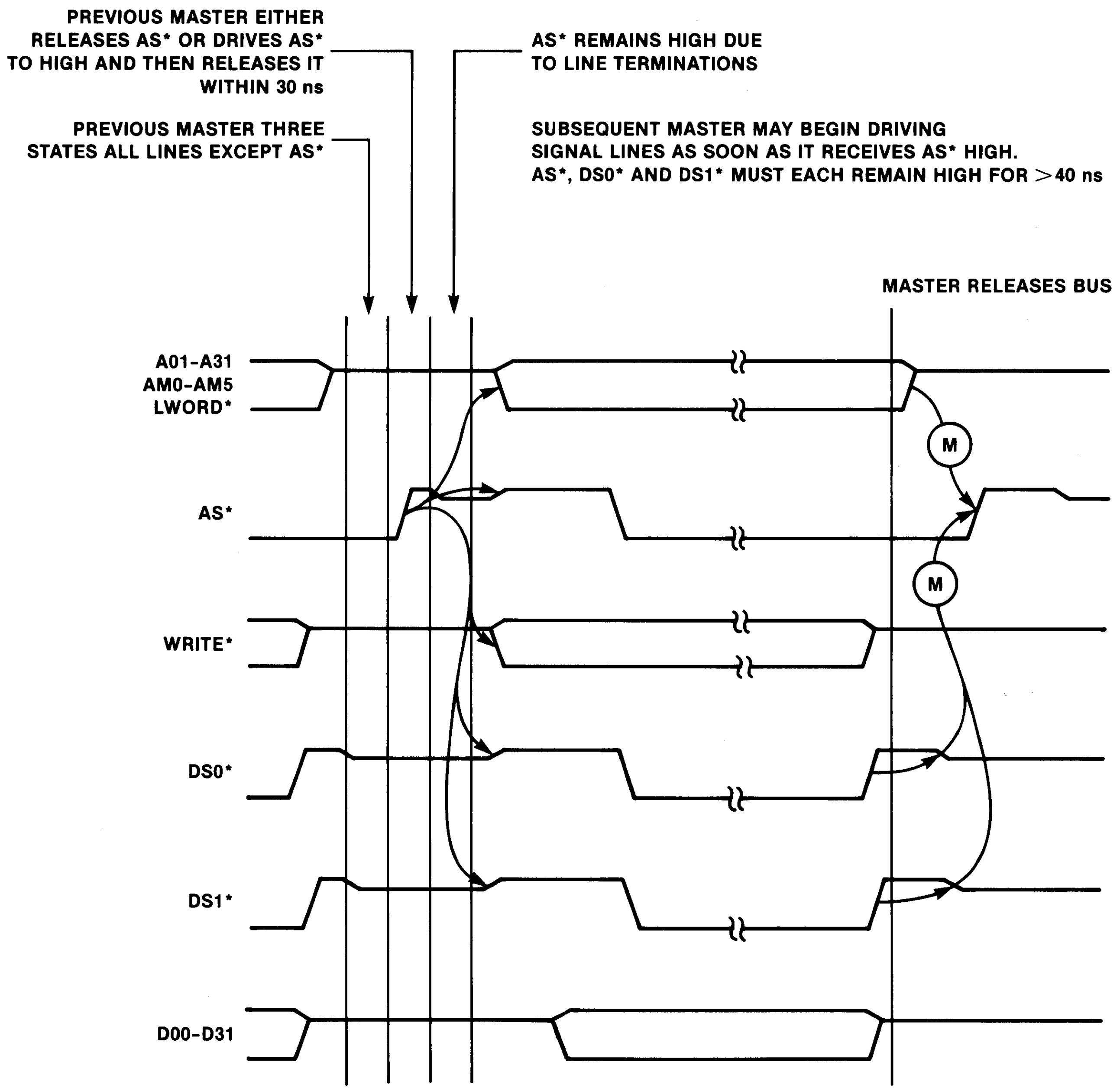
To perform a data transfer, the DTB MASTER must first acquire control of the DTB via its on-board DTB REQUESTER. The DTB REQUESTER will petition the DTB ARBITER to use the DTB. (See Chapter 3 for a description of DTB arbitration.) This is required because in a multi-processor system, several DTB MASTERS may concurrently request use of the DTB. The DTB ARBITER grants the DTB to the highest priority REQUESTER. When the DTB REQUESTER has received permission to use the DTB, it informs its on-board MASTER.

NOTE

> A MASTER may signal to its on-board REQUESTER that it is finished using the DTB prior to the end of its last data transfer cycle. This causes the on-board REQUESTER to release BBSY* and allows DTB arbitration to be done while the last data transfer cycle completes.
>
> When this type of design is used, the MASTER must not signal its REQUESTER until it has driven AS* to low for this last cycle (i.e., until the last cycle has begun). Failure to follow this rule may cause the new MASTER to see AS* high and assume that the DTB is available prematurely.

Since arbitration of the bus may take place during the previous MASTER'S last data transfer, a new DTB MASTER must ensure that the cycle of the previous DTB MASTER is complete -- i.e., that the address bus is no longer being driven, and AS* is high. See Figure 2-10. The new MASTER is then authorized to turn on its three-state output drivers and operate as the current DTB MASTER.

When a DTB MASTER is finished using the bus or when it has been requested to clear the bus by the DTB ARBITER, it must first release all lines except AS* and then drive AS* to high. This procedure guarantees that a DTB driver conflict will not occur. After AS* is driven high, it must be released within a prescribed time limit so that the new MASTER may drive the line without conflict. See Figure 2-10 for a picture of the timing relationships.



PREVIOUS MASTER EITHER RELEASES AS* OR DRIVES AS* TO HIGH AND THEN RELEASES IT WITHIN 30 ns
PREVIOUS MASTER THREE STATES ALL LINES EXCEPT AS*
AS* REMAINS HIGH DUE TO LINE TERMINATIONS
SUBSEQUENT MASTER MAY BEGIN DRIVING SIGNAL LINES AS SOON AS IT RECEIVES AS* HIGH. AS*, DS0* AND DS1* MUST EACH REMAIN HIGH FOR >40 ns
MASTER RELEASES BUS
A01-A31
AM0-AM5
LWORD*
AS*
WRITE*
DS0*
DS1*
D00-D31
M
M

FIGURE 2-10. Data Transfer Bus MASTER Exchange Sequence

2.5.2 Byte Read Sequence

Once the DTB MASTER has use of the DTB, it may perform any number of transfer cycles. Figure 2-11 shows the sequence of events for a typical byte read cycle.

2.5.2.1 Address Sequence

To start the cycle, the address is presented on some or all of A01-A31, and the address modifier code is presented on AM0-AM5. The IACK* line is driven high to indicate a data transfer cycle. The transfer is identified as a non 32 bit transfer by driving LWORD* high.

After all the address information is valid, the MASTER drives AS* to low. A setup time is provided between valid address and AS* driven to low to allow for bus skew and setup time in the input latches on the SLAVES.

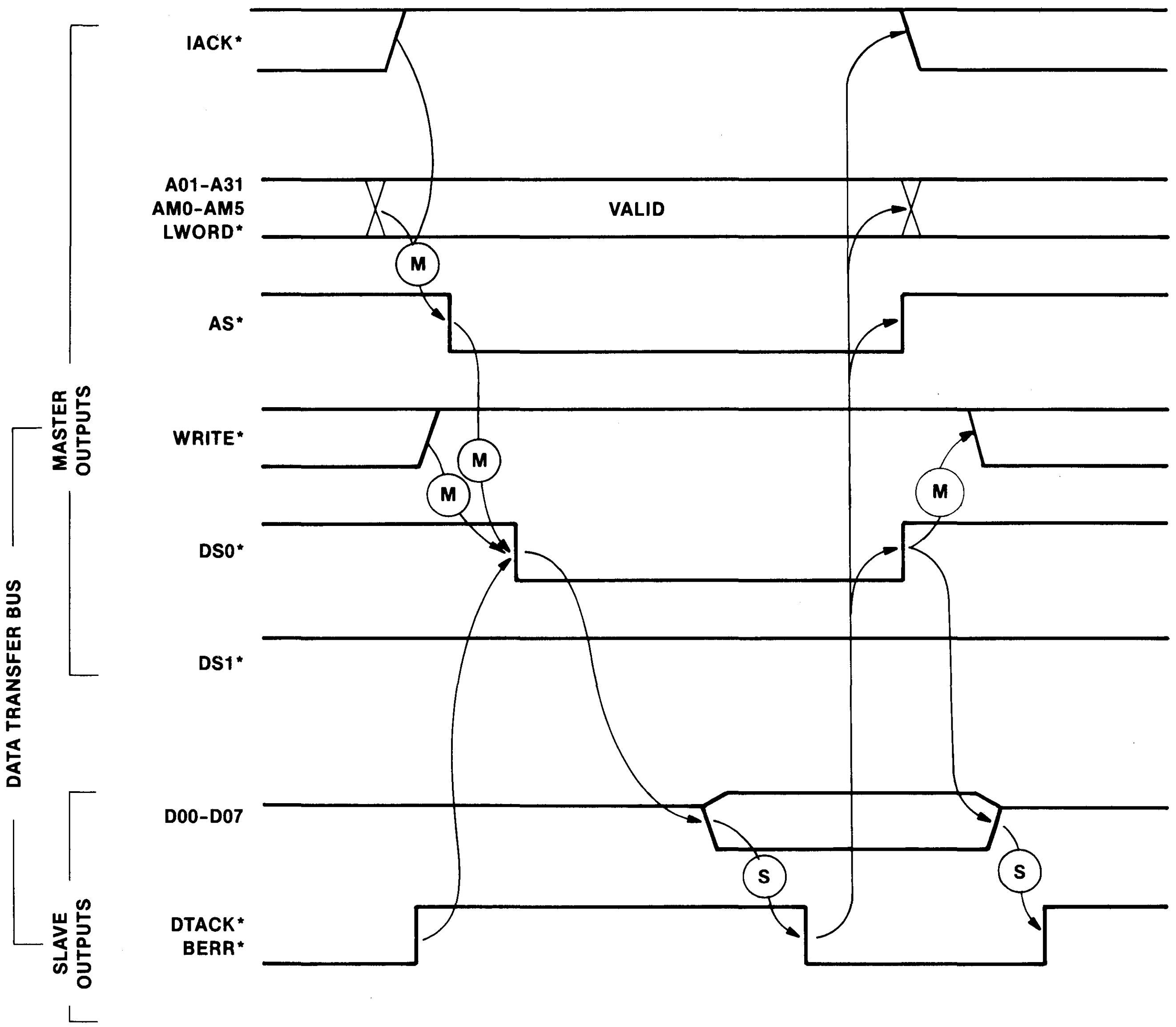
When the SLAVE receives AS* driven to low, its address decoder will compare the incoming address with the pre-assigned SLAVE address. If the address is valid for this device and no error is detected by the SLAVE, a data transfer select is generated internal to the SLAVE.

2.5.2.2 Data Bus Sequencing

Although the address and data timing are largely independent, there are two exceptions. First, the data strobe falling edges may not precede the address strobe falling edge at the MASTER. At the SLAVE, the data strobe falling edges may not precede the address strobe falling edge by more than the bus skew specification. The second exception is that the SLAVE acknowledges both the data strobes and the address strobe with a single signal (DTACK* or BERR*).

As shown in the low order byte read cycle of Figure 2-11, the WRITE* line must be driven high to identify a read cycle <u>before</u> DS0* is driven to low. This delay allows for skew and provides input latch setup time at the SLAVE. The MASTER must also ensure that the data bus is available for the next cycle by detecting that DTACK* and BERR* are high before it drives either data strobe to low. By driving the DS0* line low, the MASTER specifies that the odd byte of the word is to be transferred. The MASTER must then wait until the SLAVE acknowledges the transfer.

If the SLAVE detects an illegal transfer, it will abort the cycle by driving BERR* to low after receiving a data strobe driven to low. The SLAVE <u>must wait</u> for a data strobe before driving BERR* or DTACK* to low to ensure that the previous SLAVE has released these lines. This guarantees a transition of one of the two lines for every transfer cycle. If the addressing is correct, the SLAVE will respond by reading the location. This data is then placed on data lines D00-D07. After valid data has been presented to the bus, the SLAVE drives DTACK* to low. The SLAVE must then maintain the data on the bus until the MASTER drives DS0* to high.



NOTE

Arrows labeled M show timing relationships guaranteed by the internal timing of the MASTER.

Arrows labeled S show timing relationships guaranteed by the internal timing of the SLAVE.

Unlabeled arrows show timing guaranteed by interlocked relationships between the MASTER and SLAVE.

FIGURE 2-11. Data Transfer Bus Byte Read

When the MASTER receives a response of either DTACK* or BERR* driven to low, it begins to terminate the cycle. The SLAVE will continue to drive the data bus until DS0* is driven to high. For optimum performance, the data strobes should be driven to high as soon as possible.

The SLAVE must ensure that as the cycle terminates, valid data is maintained on the data bus until the MASTER drives the data strobe high. This may be accomplished by latching the incoming address or by latching the output data when DTACK* is driven to low.

In general, a MASTER tells a SLAVE that it has read the data from the data bus by driving a data strobe to high. After receiving either strobe driven to high, the SLAVE may release its corresponding output data drivers. Only after the data bus is released may the SLAVE release DTACK* to high. As stated earlier, the MASTER may not use the data bus on the next cycle until DTACK* is released high; therefore, it is important that the SLAVE release the data bus as quickly as possible to allow the maximum data transfer rate on the bus.

2.5.3 Read-Modify-Write Sequence

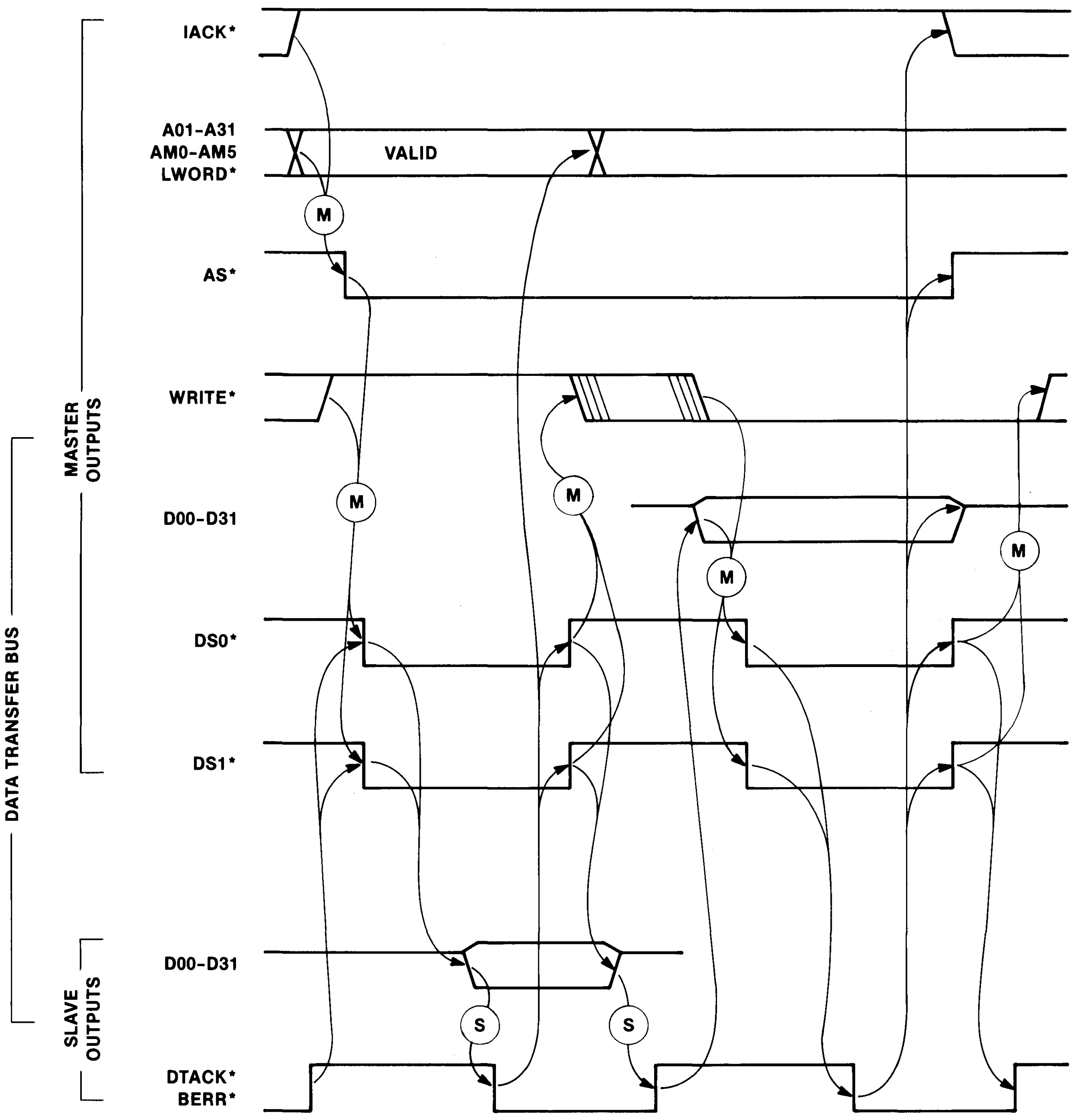
In multiprocessor systems which share resources such as memory and I/O, a method of allocating the resources must be established. The difficulty of any method of allocation is synchronizing the asynchronous requests for that resource. This is best described by the following example.

Two processors in a distributed processing system share a common resource (e.g., a printer). Only one processor may 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 clear, the resource is available. To gain use of the resource, processor A must read the bit and test to determine whether it is cleared. If the bit is cleared, processor A sets the bit to lock out the other processor, B. This operation takes two bus cycles: one to read and test the bit; the other to write the set bit. However, a difficulty may arise if the bus is given to processor B between these two bus cycles. Processor B may also find the bit clear 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 arbitration from taking place between the read and write. The sequence diagram for this type of cycle is defined in Figure 2-12.

The read-modify-write cycle is very similar to a read cycle immediately followed by a write cycle. The difference is that no address is given for the write cycle (it is assumed to be the same as the read cycle), and AS* is continuously driven low during both transfer cycles. All other sequencing and timing is identical to normal read and write cycles.

Unlike a read cycle followed by a write cycle, the read-modify-write cycle cannot be interrupted by the bus arbitration because AS* is driven low continuously through both cycles, and control of the DTB may only be transferred while AS* is high.



NOTE

Arrows labeled M show timing relationships guaranteed by the internal timing of the MASTER.

Arrows labeled S show timing relationships guaranteed by the internal timing of the SLAVE.

Unlabeled arrows show timing guaranteed by interlocked relationships between the MASTER and SLAVE.

FIGURE 2-12. Read-Modify-Write Cycle Sequence

2.5.4 Sequential Access Sequence

Many accesses to bus memory take place in sequence (i.e., memory locations are accessed in ascending order). When this is the case, it is desirable to have a means for accessing several locations without having to provide an address each time. The sequential access sequence allows the MASTER to specify that memory is to be accessed in ascending order by using special address modifier codes.

The MASTER initiates the cycle in the standard way except that it places one of the sequential access AM codes on the address modifier lines. All sequential access SLAVES latch the address into an on-board address counter. The MASTER, upon completing the first data transfer (i.e. drives data strobes high) does <u>not</u> allow AS* to go high. Instead, it repeatedly drives the data strobes low in response to DTACK's from the SLAVE to transfer data to/from sequential memory locations.

The SLAVE increments its on-board address counter as required to access the next location. When accessing memory in ascending order, the counter is incremented on each rising edge of DS0*.

Special attention is drawn to the fact that <u>all</u> sequential access SLAVES should latch the initial address when a sequential access AM code is placed on the bus. In addition, all of these SLAVES should increment the address counter each access cycle. The resulting counter output should then be decoded to see if it falls within the SLAVE'S address boundaries. (This is important because the block of sequential memory location may straddle the boundary between two memory boards, or memory board locations may be interleaved to allow faster access.)

While the sequential access sequence is intended primarily to do a string of reads or a string of writes, there is no practical reason why read and write cycles could not be mixed. When this is the case, WRITE* must be stable and valid prior to the falling edge of the data strobe, and must remain valid until the MASTER drives a data strobe back to high (see timing diagrams, Figures 2-13, 2-14, 2-16, and 2-17).

The sequential access cycle is very similar to a string of normal read/write cycles. The difference is that no address is provided after the first transfer cycle, and AS* is continuously driven low during the remaining transfer cycles. All other sequencing and timing is identical to normal read/write cycles.

The sequential access sequence cannot be interrupted by bus arbitration because AS* is driven low continuously through all cycles, and control of the DTB may only be transferred while AS* is high.

2.6 DETAILED TIMING/STATE DIAGRAMS

This section describes the timing relationships of signals on the DTB. Two separate sets of timing are provided: one for the MASTER and one for the SLAVE. These two sets of timing take into account bus skew time, and provide the designer an exact definition of his DTB timing constraints and guarantees. Because the backplane is a passive device, capacitive loading of signal lines causes a degradation of rise and fall times. This may alter the skew between signals as they propagate down the bus. The worst case skew between two lines on the bus will occur when one line has minimum loading and the other line has maximum loading. (Bus load specifications are discussed in Chapter 7.) For example, consider a DTB MASTER which presents an address on the bus and, after a setup time, drives AS* to low. Capacitive loading may vary significantly among various lines on the bus. As the signal propagates down the bus, the address line transition times could be stretched due to the capacitance of the heavy loading, while AS* is not. As a result, when the signals reach the SLAVE farthest from the MASTER, the relationship of address lines to AS* is the address setup time generated at the MASTER minus the bus skew. The Data Transfer Bus is designed to limit bus skew to a maximum of 10 nanoseconds. This can be seen by observing that each MASTER is required to provide 20 nanoseconds of setup time, while each SLAVE is guaranteed only 10 nanoseconds. If the specified loading limits are obeyed (see Chapter 7), this 10-nanosecond maximum is guaranteed.

The following timing parameters are for the bus pins which plug into the backplane. The designer must guarantee the specified times for output signals so that after any worst case buffer skew, the timing is still met at the VME bus pins. Timing given for input signals to a board include the worst case skew on the bus, and are guaranteed valid at the input pins of the board. On-board input buffers may add additional skew to the times. This additional input buffer skew must be accounted for by the board designer.

2.6.1 DTB MASTER Timing

Three timing diagrams are presented to outline the timing requirements for DTB MASTERS.

Figure 2-13 shows the timing requirements a MASTER must meet when doing a write cycle followed by a read cycle.

Figure 2-14 shows the timing requirements a MASTER must meet when doing a read cycle followed by a write cycle.

Figure 2-15 shows the timing requirements that the MASTER relinquishing control of the DTB and the new MASTER taking control must meet.

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 low 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* may represent DS0* on its falling edge and DS1* on its rising edge.

2.6.1.1 DTB MASTER Timing: Write Cycle Followed by Read Cycle

See Figure 2-13 and Table 2-3. Following is a description of each parameter.

DESCRIPTION OF PARAMETERS

1 This time provides the SLAVE with address setup time. The address lines and IACK* must be stable and valid for the minimum setup time before AS* may be driven across the high level threshold voltage.

2 The address must be held stable until DTACK* is received driven to low, and then may change. Likewise, IACK* must be held high until DTACK* is received driven to low. The address bus need not be released between consecutive cycles of the same MASTER, but may be released, if desired.

3 AS* must be driven high for the minimum time to ensure that all SLAVES detect the end of the bus cycle.

4 AS* must remain low until DTACK* is received driven to low. It is then driven to high.

5 This time applies to whichever data strobe is driven low by the MASTER first. DS "A" corresponds to the first strobe and DS "B" corresponds to the second strobe. The first strobe driven to low may or may not be the first strobe driven to high. The first data strobe may be driven to low concurrently with driving AS* to low, but must not precede it.

6 The WRITE* line must be valid and stable for the minimum setup time before either data strobe is driven across the high level threshold voltage.

7 The WRITE* line must remain valid until the MASTER drives DS"A"* to high.

8 The MASTER must release its output data bus drivers the minimum time before the first data strobe may be driven across the high level threshold voltage.

9 This maximum skew between DS0* and DS1* must not be exceeded for cycles in which both data strobes are driven to low. This time does not apply to byte reads where only one strobe is driven to low.

10 14 Once driven low, a data strobe must be held low until the MASTER receives DTACK* driven to low.

11 12 16 These times require that both data strobes be concurrently driven high for the minimum time.

13 This time requires that the MASTER must receive DTACK* high before either data strobe is driven across the high level threshold voltage.

15 This time guarantees that the data on the data bus lines will remain valid until the MASTER drives the first data strobe across the low level threshold voltage.

17 This read data time guarantees the MASTER that the data bus is valid and stable within the specified time after the received DTACK* crosses the high level threshold voltage on the high to low transition.

18 This time guarantees that the data bus is released by the SLAVE before the received DTACK* crosses the low level threshold voltage on the low to high transition.

TABLE 2-3. DTB MASTER Timing: Write Cycle Followed by Read Cycle

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx and AMx valid and IACK* high to AS* low	20		B
2	DTACK* low to invalid address or IACK* low	0		C
3	AS* High	40		B
4	DTACK* low to AS* high	0		C
5	AS* to DS"A"* skew	0		B
6	WRITE* valid to DS"A"* low	20		B
7	DS"A"* high to invalid WRITE*	10		B
8	DATA release to DS"A" low	0		B
9	DS"A"* to DS"B" skew		10	B
10	DTACK* low to DS"A"* high	0		C
11	DS"A"* high	40		B
12	DS"B"* high to DS"A" low	40		B
13	DTACK* high to DS"A" low	0		C
14	DTACK* low to DS"B" high	0		C
15	DS"A"* high to invalid data	0		D
16	DS"B"* high	40		B
17	Dxx valid to DTACK* low	-10		E
18	Data released to DTACK* high	0		E

NOTES:

A. All times given are in nanoseconds.

B. The MASTER must guarantee this timing between two of its outgoing signal transitions.

C. The MASTER must wait for the incoming signal edge from the SLAVE before changing the level of its outgoing signal.

D. This is a guarantee that the SLAVE will not change the incoming signal until the MASTER changes its outgoing signal.

E. The MASTER is guaranteed this timing between two of its incoming signal transitions.

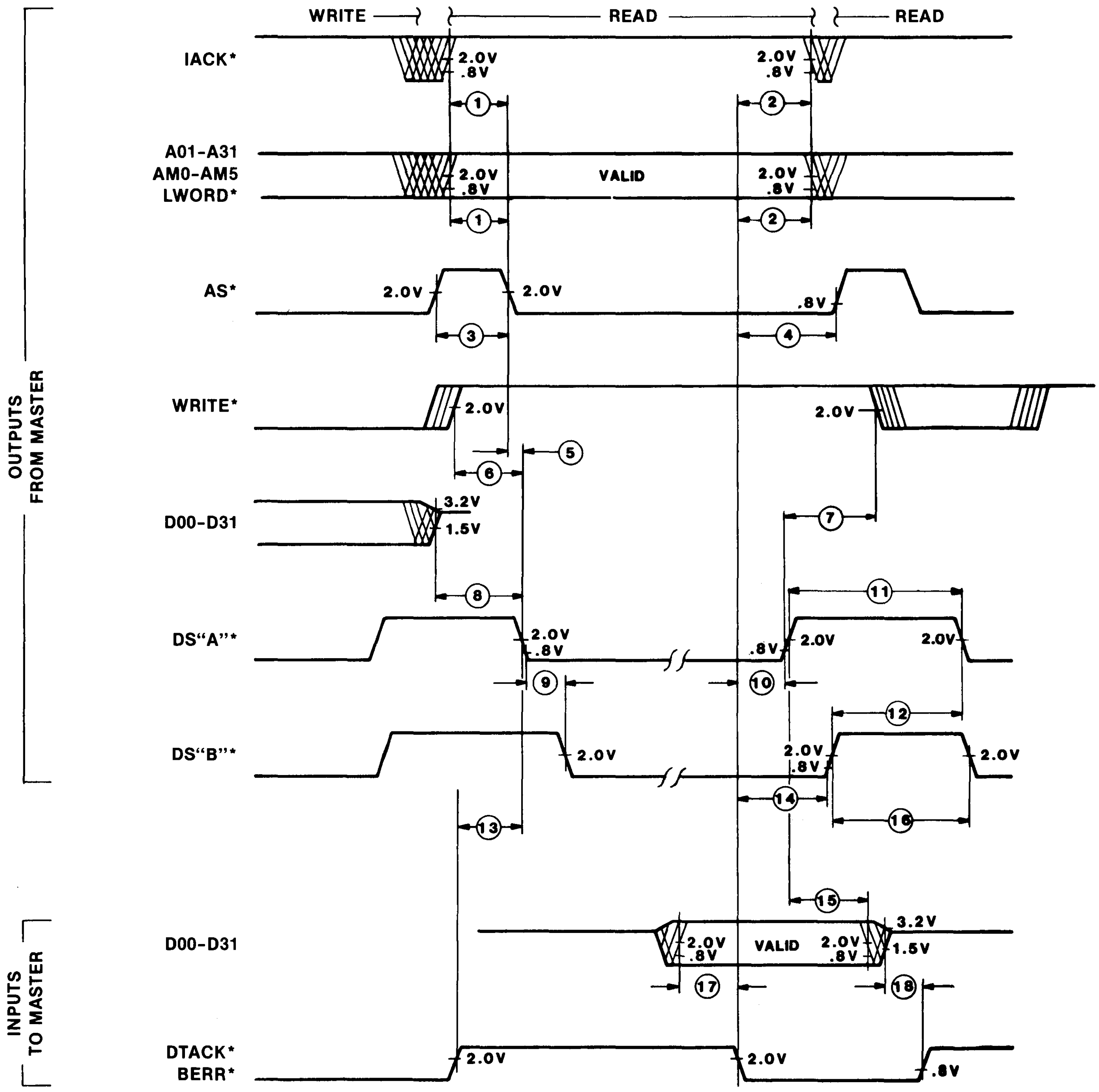


FIGURE 2-13. DTB MASTER Timing: Write Cycle Followed by Read Cycle

2.6.1.2 DTB MASTER Timing: Read Cycle Followed by Write Cycle

See Figure 2-14 and Table 2-4. Following is a description of each parameter.

DESCRIPTION OF PARAMETERS

1 This time provides the SLAVE with address setup time. The address lines and IACK* must be stable and valid for the minimum setup time before AS* may be driven across the high level threshold voltage.

2 The address must be held stable until DTACK* is received driven to low and then may change. Likewise, IACK* must be held high until DTACK* is received driven to low. The address bus need not be released between consecutive cycles of the same MASTER, but may be released, if desired.

3 AS* must be driven high for the minimum time to ensure that all SLAVES detect the end of the bus cycle.

4 AS* must remain low until DTACK* is received driven to low. It may then be driven to high.

5 This time applies to whichever data strobe is driven low by the MASTER first. DS "A" corresponds to the first strobe and DS "B" corresponds to the second strobe. The first strobe driven to low may or may not be the first strobe driven to high. The first data strobe may be driven to low concurrently with driving AS* to low, but must not precede it.

6 The WRITE* line must be valid and stable for the minimum setup time before either data strobe is driven across the high level threshold voltage.

7 The WRITE* line must remain valid until the MASTER drives DS"A"* to high.

8 The MASTER must not drive the data bus until it detects both DTACK* and BERR* high. (This indicates that the SLAVE addressed during the previous read cycle is no longer driving the data bus.)

9 The data bus outputs must be valid and stable a minimum time before the first data strobe may be driven across the high level threshold voltage.

10 The data bus outputs must remain valid and stable until the MASTER receives DTACK* driven to low.

11 This maximum skew between DS0* and DS1* must not be exceeded for cycles in which both data strobes are driven to low. This time does not apply to byte writes.

12 17 Once driven low, a data strobe must be held low until the MASTER receives DTACK* driven to low.

13 14 15 These times require that both data strobes be concurrently driven high for the minimum time.

16 This time guarantees that the SLAVE will not release the DTACK*/BERR* line to high until after the MASTER drives both data strobes high.

TABLE 2-4. DTB MASTER Timing: Read Cycle Followed by Write Cycle

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx and AMx valid and IACK* high to AS* low	20		B
2	DTACK* low to invalid address or IACK* low	0		C
3	AS* High	40		B
4	DTACK* low to AS* high	0		C
5	AS* to DS"A"* skew	0		B
6	WRITE* valid to DS"A"* low	20		B
7	DS"A"* high to invalid WRITE*	10		B
8	DTACK* high to active data bus	0		C
9	Dxx valid to DS"A"* low	20		B
10	DTACK* low to invalid data	0		C
11	DS"A"* to DS"B"* skew	0	10	B
12	DTACK* low to DS"A"* high	0		C
13	DS"A"* high	40		B
14	DS"B"* high to DS"A"* low	40		B
15	DS"B"* high	40		B
16	DS"B"* high to DTACK* high	0		D
17	DTACK* low to DS"B"* high	0		C

NOTES:

A. All times given are in nanoseconds.

B. The MASTER must guarantee this timing between two of its outgoing signal transitions.

C. The MASTER must wait for the incoming signal edge from the SLAVE before changing the level of its outgoing signal.

D. This is a guarantee that the SLAVE will not change the incoming signal until the MASTER changes its outgoing signal.

E. The MASTER is guaranteed this timing between two of its incoming signal transitions.

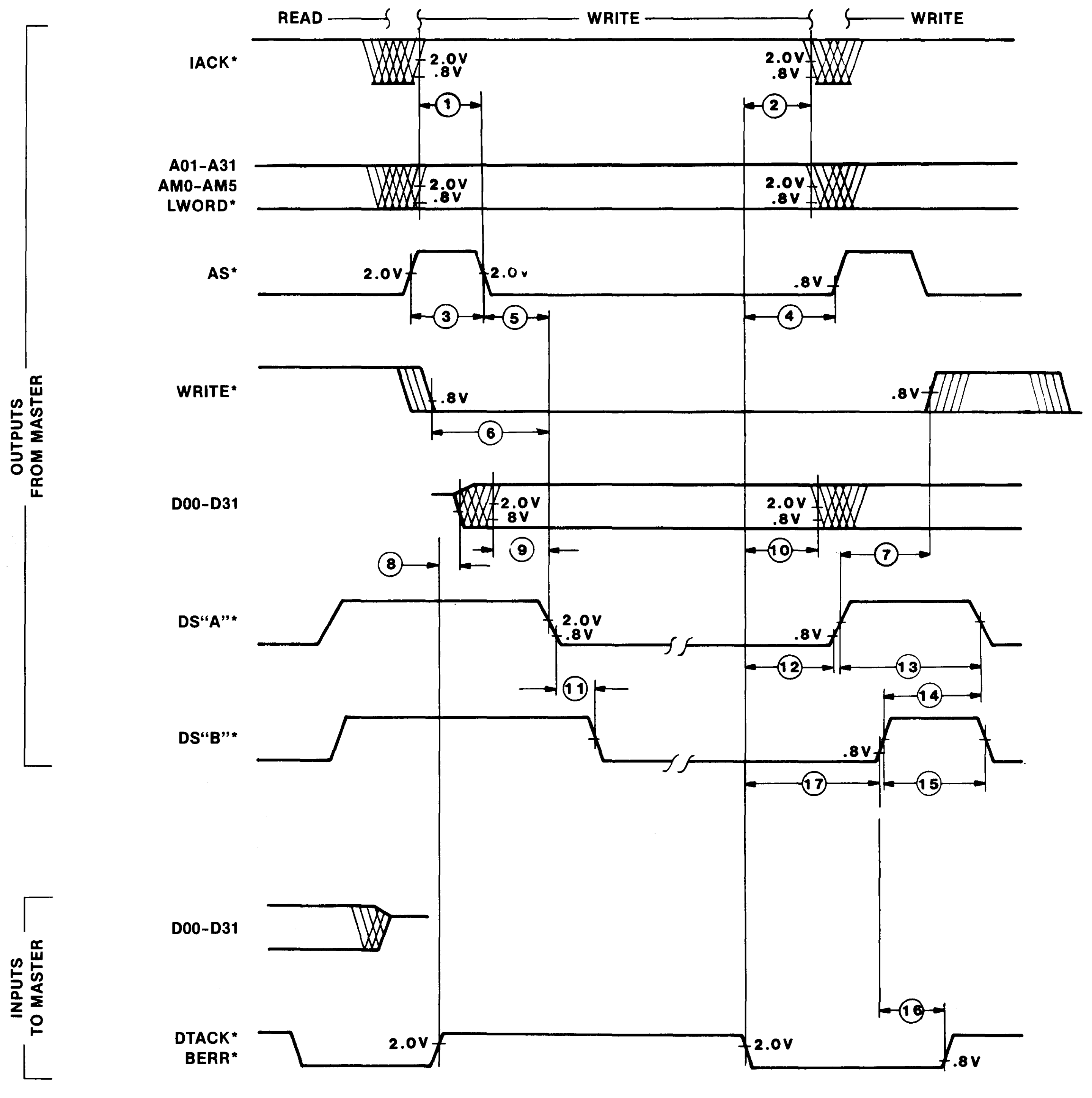


FIGURE 2-14. DTB MASTER Timing: Read Cycle Followed by Write Cycle

2.6.1.3 MASTER Timing: Control Transfer of DTB

When a DTB MASTER has started its last data transfer and has driven AS* to low, it may notify its on-board DTB REQUESTER that it no longer wants the bus. The REQUESTER then releases BBSY*, allowing the ARBITER to arbitrate existing bus requests from other boards in the system and grant use of the DTB to the highest priority REQUESTER. It is vital that control of the DTB be passed smoothly from one MASTER to another. To ensure this control, the following timing requirements must be met.

<u>NOTE</u>

> In the following discussion, the term "MASTER A" will be used to designate the MASTER which has just finished its data transfers and is preparing to give up control of the DTB. "MASTER B" will be used to designate the MASTER which is taking control of the DTB.

The transfer of control takes place in five phases, as shown in Figure 2-15:

PHASE 1	MASTER A is driving the DTB lines.
PHASE 2	MASTER A releases all DTB lines except AS* which is still driven low.
PHASE 3	MASTER A either (a) releases AS* or (b) drives AS* to high and then releases it within 30 ns.
PHASE 4	All DTB lines remain high due to line terminators.
PHASE 5	MASTER B receives AS* driven to high and turns on all of its DTB line drivers except its data bus drivers, ensuring that AS* is driven high (i.e., no falling edge is generated on AS* until the address setup time has been satisfied).

MASTER B then drives the DTB in accordance with the timing given in paragraph 2.6.1. (If the first cycle is to be a write cycle, MASTER B must wait for DTACK* and BERR* to go high before driving the data bus).

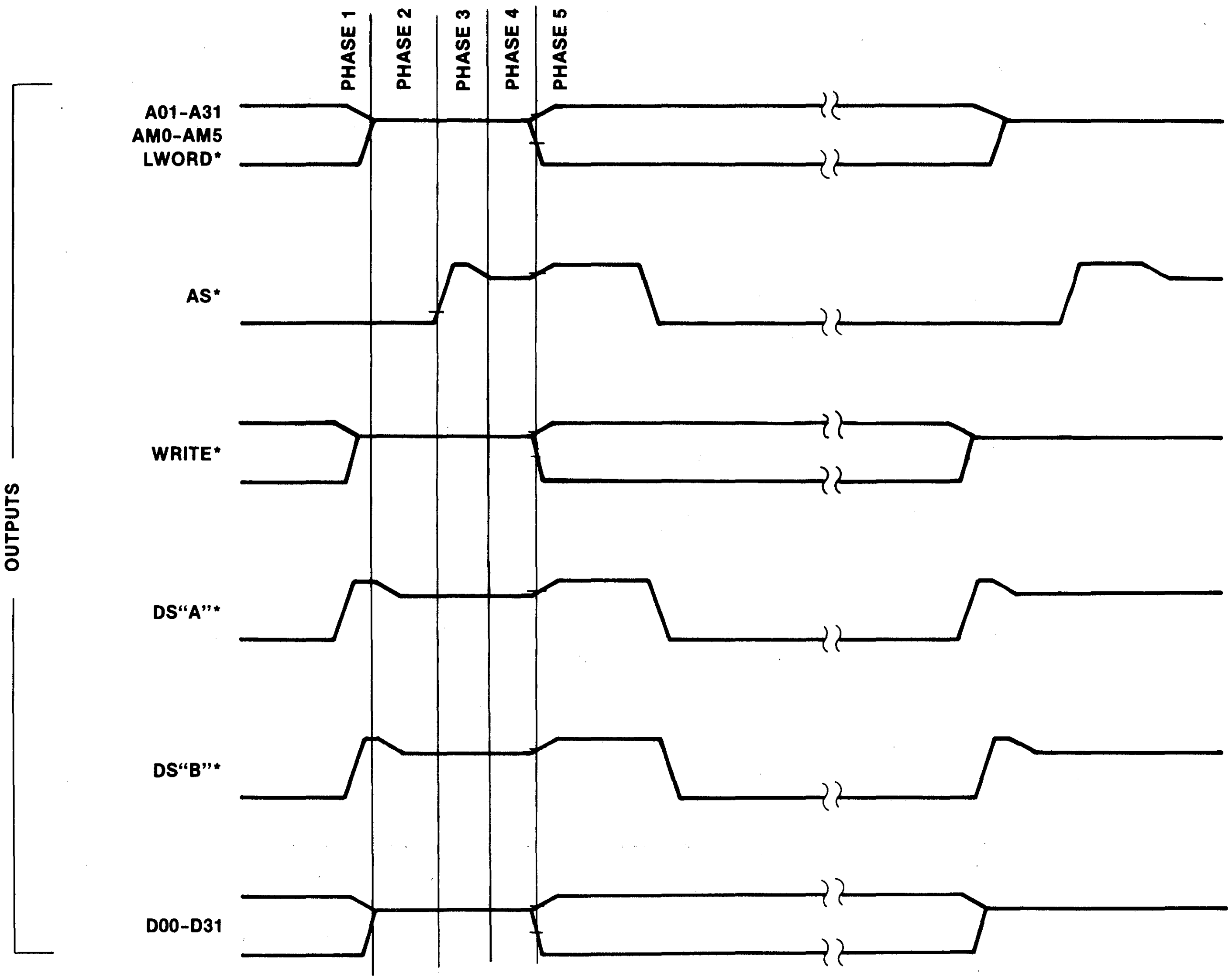


FIGURE 2-15. MASTER Timing: Control Transfers of DTB

2.6.2 DTB SLAVE Timing

Two timing diagrams are presented to outline the timing requirements for DTB SLAVES. These two diagrams describe the timing required when a SLAVE is addressed.

Figure 2-16 shows the timing requirements a SLAVE must meet during two consecutive read cycles.

Figure 2-17 shows the timing requirements a SLAVE must meet during two consecutive write cycles.

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 on which the SLAVE receives a transition (whether it is DS0* or DS1*). The broken line shown while the data strobes are low 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* may represent DS0* on its falling edge and DS1* on its rising edge).

2.6.2.1 DTB SLAVE Timing: Two Consecutive Read Cycles

See Figure 2-16 and Table 2-5. Following is a description of each parameter.

DESCRIPTION OF PARAMETERS

1 This time guarantees the SLAVE a minimum address setup time. The address lines are stable and valid for the minimum setup time before AS* is received driven across the high level threshold voltage.

2 The address is guaranteed to remain stable until the SLAVE drives DTACK* to low. The address lines may then change.

3 AS* is driven high for this guaranteed minimum time to ensure that the SLAVE detects the end of the bus cycle.

4 AS* is guaranteed to remain low until the SLAVE drives DTACK* to low.

5 This time applies to whichever data strobe is received low by the SLAVE first. DS"A"* corresponds to the first strobe and DS"B"* corresponds to the second strobe. The first strobe received low may or may not be the first strobe received high. Because of bus skew, the first data strobe falling edge may slightly precede the AS* falling edge, but is guaranteed not to precede it by more than the specified time.

6 The WRITE* line is guaranteed valid and stable for the minimum setup time before either data strobe is received driven across the high level threshold voltage.

7 The WRITE* line is guaranteed to remain valid until DS"A"* is received driven to high.

8 The SLAVE is guaranteed that this maximum skew between DS0* and DS1* will not be exceeded for cycles in which both data strobes are driven to low. This time does not apply to byte reads where only one data strobe is drive to low.

9 17 Once driven low, a data strobe is guaranteed to remain low until the SLAVE drives DTACK* to low.

10 11 12 These times guarantee that both data strobes will be concurrently driven high for the minimum time.

13 The SLAVE must not drive BERR* low until DS"A"* is received driven to low. If BERR* is driven low, then the SLAVE need not provide read data setup time, since no valid data is placed on the data bus.

14 This time guarantees that a new data strobe will not be received driven through the high level threshold voltage until the SLAVE releases DTACK* to high.

15 The SLAVE must not drive the data bus until the first data strobe is driven low.

16 The SLAVE must provide the minimum read data setup time before it drives DTACK* across the high level threshold voltage.

18 The SLAVE must hold the data valid and stable until it receives either data strobe driven to high.

19 The SLAVE must release the output data bus drivers before it releases DTACK* across the low level threshold voltage to high.

20 The SLAVE must not release BERR* prior to receiving either data strobe driven to high.

TABLE 2-5. DTB SLAVE Timing: Two Consecutive Read Cycles

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx and AMx valid and IACK* high to AS* low	10		E
2	DTACK* low to invalid address or IACK* low	0		D
3	AS* High	30		E
4	DTACK* low to AS* high	0		D
5	AS* to DS"A"* skew	-10		E
6	WRITE* valid to DS"A"* low	10		E
7	DS"A"* high to invalid WRITE*	0		E
8	DS"A"* to DS"B"* skew		20	E
9	DTACK* low to DS"A"* high	0		D
10	DS"A"* high	30		E
11	DS"B"* high to DS"A"* low	30		E
12	DS"B"* high	30		E
13	DS"A"* low to DTACK* low	0		C
14	DTACK* high to DS"A"* low	0		D
15	DS"A"* low to Active data bus	0		C
16	Data valid to DTACK* low	0		B
17	DTACK* low to DS"B"* high	0		D
18	DS"A"* high to invalid data	0		C
19	Data bus released to DTACK* high	0		B
20	DS"A"* high to BERR* high	0		C

NOTES:

A. All times given are in nanoseconds.

B. The SLAVE must guarantee this timing between two of its outgoing signal transitions.

C. The SLAVE must wait for the incoming signal edge from the MASTER before changing the level of its outgoing signal.

D. This is a guarantee that the MASTER will not change the incoming signal until the SLAVE changes its outgoing signal.

E. The SLAVE is guaranteed this timing between two of its incoming signal transitions.

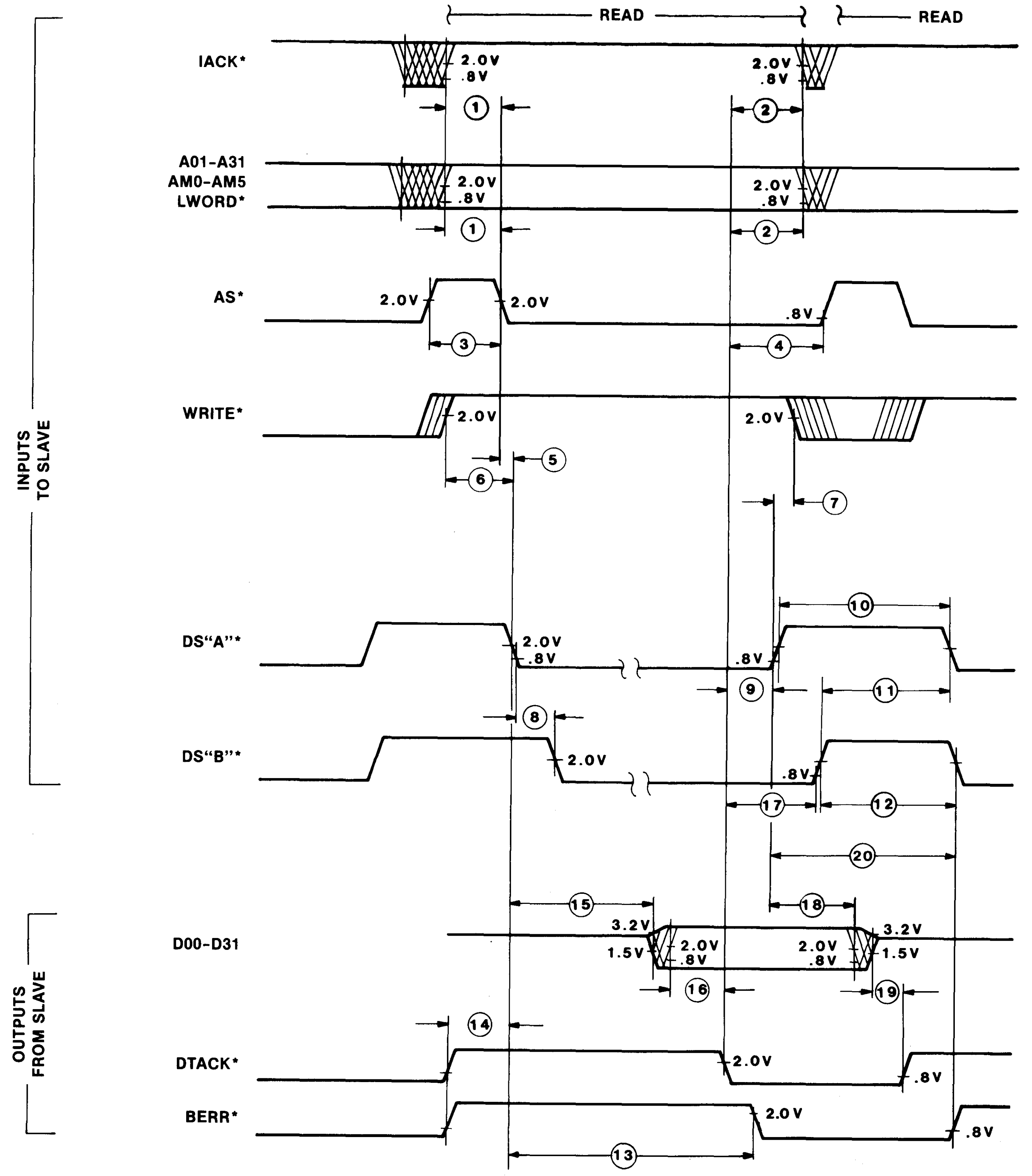


FIGURE 2-16. Data Transfer Bus SLAVE Read Cycle

2.6.2.2 DTB SLAVE Write Cycle Timing

See Figure 2-17 and Table 2-6. Following is a description of each parameter.

DESCRIPTION OF PARAMETERS

1 This time guarantees the SLAVE a minimum address setup time. The address lines are stable and valid for the minimum setup time before AS* is received driven across the high level threshold voltage.

2 The address is guaranteed to remain stable until the SLAVE drives DTACK* to low. The address lines may then change.

3 AS* is driven high for this guaranteed minimum time to ensure that the SLAVE detects the end of the bus cycle.

4 AS* is guaranteed to remain low until the SLAVE drives DTACK* to low.

5 This time applies to whichever data strobe is received low by the SLAVE first. DS"A"* corresponds to the first strobe and DS"B"* corresponds to the second strobe. The first strobe received low may or may not be the first strobe received high. Because of bus skew, the first data strobe falling edge may slightly precede the AS* falling edge, but is guaranteed not to precede it by more than the specified time.

6 The WRITE* line is guaranteed valid and stable the minimum time before either data strobe is received driven across the high level threshold voltage.

7 The WRITE* line is guaranteed to remain valid until DS"A"* is received driven to high.

8 The data bus is guaranteed to be valid and stable for the minimum setup time before the SLAVE will receive the first data strobe driven across the high level threshold voltage.

9 The data bus is guaranteed to remain valid and stable until the SLAVE drives DTACK* to low.

10 The SLAVE is guaranteed that this maximum skew between DS0* and DS1* will not be exceeded for cycles in which both data strobes are driven to low. This time does not apply to byte writes when only one data strobe is driven to low.

11 16 Once driven low, a data strobe is guaranteed to remain low until the SLAVE drives DTACK* to low.

12 13 These times guarantee that both data strobes will be concurrently driven
14 high for the minimum time.

15 The SLAVE must wait the minimum time after it receives the first data strobe driven to low before it may drive the acknowledge signal to low. (This assures that an acknowledge will not be given prior to the second data strobe going low.)

17 The SLAVE must drive low the acknowledge signals until it receives either data strobe driven to high.

TABLE 2-6. DTB SLAVE Timing: Two Consecutive Write Cycles

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx and AMx valid and IACK* high to AS* low	10		D
2	DTACK* low to invalid address or IACK* low	0		C
3	AS* High	30		D
4	DTACK* low to AS* high	0		C
5	AS* to DS"A" skew	-10		D
6	WRITE* valid to DS"A"* low	10		D
7	DS"A"* high to invalid WRITE*	0		D
8	Data valid to DS"A"* low	10		D
9	DTACK* low to invalid data	0		C
10	DS"A"* to DS"B"* skew		20	D
11	DTACK* low to DS"A"* high	0		C
12	DS"A"* high	30		D
13	DS"B"* high to DS"A"* low	30		D
14	DS"B"* high	30		D
15	DS"A"* low to DTACK*/BERR* low	30		B
16	DTACK* low to DS"B"* high	0		C
17	DS"B"* high to DTACK* high	0		B

NOTES:

A. All times given are in nanoseconds.

B. The SLAVE must wait for the incoming signal edge from the MASTER before changing the level of its outgoing signal.

C. This is a guarantee that the MASTER will not change the incoming signal until the SLAVE changes its outgoing signal.

D. The SLAVE is guaranteed this timing between two of its incoming signal transitions.

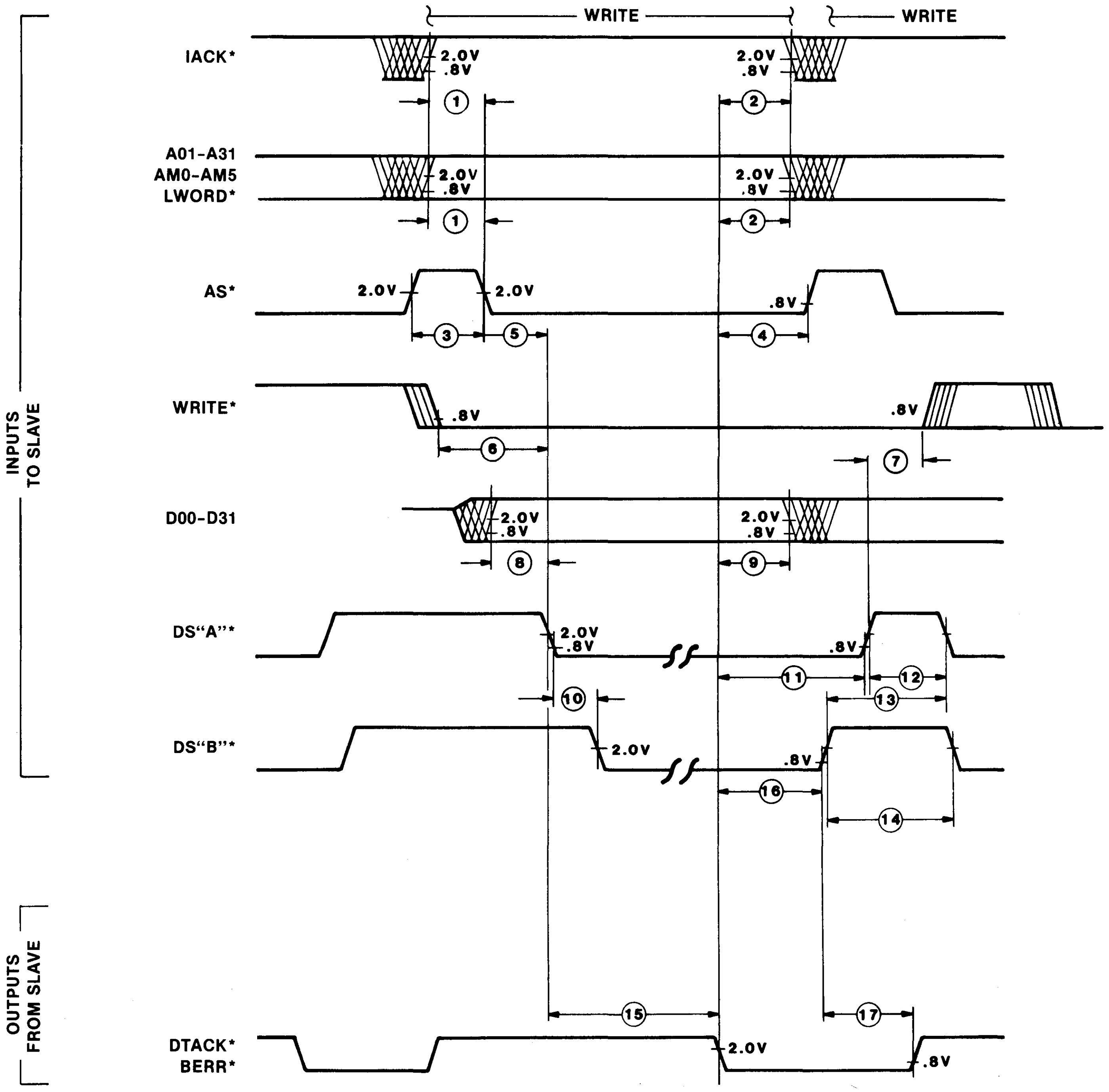


FIGURE 2-17. Data Transfer Bus SLAVE Write Cycle

CHAPTER 3

VME BUS DATA TRANSFER BUS ARBITRATION

3.1 BUS ARBITRATION PHILOSOPHY

As microprocessor costs decrease, more systems become cost-effective with multiple processors sharing global resources over a system bus. Each of these processors will have its own task or tasks, and its own need for resources.

The most fundamental of these global resources is the data transfer bus through which all other global resources are accessed. Therefore, any system supporting multiprocessing must provide an allocation method for the data transfer bus. Because each processor may have its own real-time executive, and because speed of allocation of the data transfer bus is vital, a hardware allocation scheme must be provided. The VME bus meets this need with its Bus Arbitration subsystem. (See Figure 3-1).

The VME bus arbitration subsystem is designed to:

. Prevent simultaneous access of the bus by two MASTERS.
. Schedule requests from multiple MASTERS for optimum resource use.

The logic used to implement the bus allocation algorithm is called the ARBITER. It is the ARBITER'S responsibility to respond to requests for the bus and to optimize usage by proper control of the allocation process.

3.1.1 ARBITER Options

The ARBITER used to control the arbitration system may be one of three options. An option PRI (Priority) ARBITER always assigns the bus on a fixed priority basis wherein each bus request line is assigned a fixed priority from highest (BR3*) to lowest (BR0*). An option RRS (Round Robin Select) ARBITER assigns the bus on a rotating priority basis. If the current bus MASTER is level "n", the highest priority will be given to level "n-1" and proceed sequentially from there. An option ONE (Single level) ARBITER only honors requests on BR3* and relies on the daisy-chain structure for priority determination.

In order to understand the value of the first two arbitration methods, consider two types of bus MASTERS. The first is a fixed rate transfer device. This means it transfers data over the bus at a fixed rate. This type of device maintains a constant data rate despite the fluctuating bus usage by other devices. It does this by one of two methods.

a. Variable length bursts
b. Variable frequency bus requests

When a device uses the first method, it adjusts the number of transfer cycles per bus grant to maintain the data rate. If it is forced to wait longer for the bus to be granted, it will execute a larger number of transfer cycles. A typical example of this method is a device with a 16-word FIFO, which requests the bus whenever eight or more words exist in the FIFO, and it empties the FIFO before relinquishing the bus.

When a device uses the second method, it adjusts the frequency of its bus requests in order to maintain the required transfer rate. A typical example of this method is a device with a 32-word FIFO, which transfers eight words per bus request and requests the bus whenever the FIFO contains at least eight words. In this case, the effect of the delay in the first bus grant is that the FIFO is closer to eight words after the first burst is sent. This will result in a reduced time before the next bus request is issued.

The fixed data rate device may be said to require a certain percentage of bus time. Long-term failure to provide that percentage will result in data overrun.

The second type of MASTER does not have any such constraints. If it does not receive the bus within any given time or at any fixed percentage, it will simply wait, resulting in reduced performance, but no overruns. Because it does not adjust the number of transfers per burst (typically one) or its rate of request, its bus usage may be reduced by delaying its bus grant. A typical example of such a device is a general purpose processor.

System bus usage fluctuates widely during normal system operation. In order to maximize system throughput, it is important to maximize bus usage without allowing peak periods to cause buffer overruns on fixed data rate MASTERS. A simple way to verify that peak bus loading will not cause buffer overruns is to add up the percentage of bus time required by all fixed data rate masters. If this percentage exceeds 100, then overruns will likely occur. If it does not exceed 100, then it will be possible to prevent overruns by reducing the number of bus grants given to the variable data rate devices during peak bus usage.

If the percentage of bus usage by fixed rate devices is very close to 100, it will probably be necessary to use prioritized arbitration and assign the fixed rate MASTERS to the higher priority request levels. When all fixed rate devices are using the bus concurrently, the lower priority variable data rate MASTERS may then be locked out of the bus temporarily.

If the percentage of bus usage by fixed rate MASTERS is significantly lower than 100 percent, the user now has a choice. If the user wishes to grant preference to certain variable rate devices, he will choose a PRI system. If it is his desire to uniformly distribute the bus usage among the variable rate devices, he will choose the round robin system. The fixed data rate devices (DMA), by their design, will take their required share of the bus. The balance of the bus will be distributed on a relatively uniform manner among the co-equal tasks. The purpose of a round-robin arbitration scheme is to uniformly distribute the burden experienced by variable data rate devices during heavy loading periods. This is most often found in interactive time-shared systems, where a heavy load should be visible to all users as a poorer response time, rather than as a selectively applied penalty.

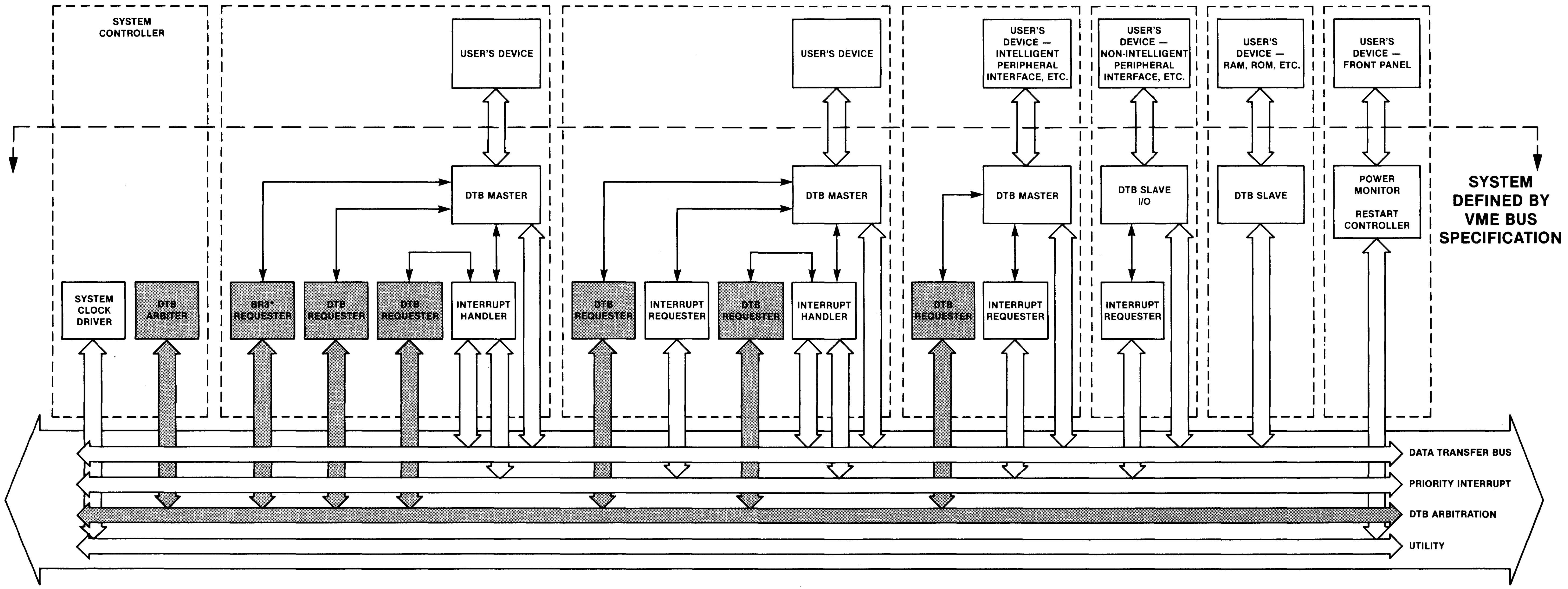


FIGURE 3-1. VME Bus Arbitration Functional Block Diagram

3.1.2 ARBITER Operation

Except for option ONE ARBITERS, the bus arbitration subsystem accepts requests for the bus on four request lines, which are driven by open-collector drivers so that several MASTERS can share a common request line. Each request line has a corresponding grant daisy-chain line (BG3IN*/OUT* through BG0IN*/OUT*). If the bus is idle when a request is received, the ARBITER will immediately respond on the grant line corresponding to the level of request pending. When the current MASTER relinquishes the bus, the ARBITER will respond to the highest pending level of request with a grant line. If no requests are pending at the time the currently active MASTER relinquishes control, the ARBITER will wait in the idle state until a bus request is received.

In addition to the ARBITER allocating the bus on an assigned basis, a secondary level of prioritization is built into the bus itself. The bus grant signals are daisy-chained in such a way that REQUESTERS sharing a common request line are prioritized by slot position. The REQUESTER closest to slot one has the highest priority.

Option ONE ARBITERS respond only to bus request 3 (BR3*) and depend on the daisy-chain of BG3IN*/BG3OUT* to do the prioritizing.

3.2 ARBITRATION BUS LINE STRUCTURES

The arbitration bus consists of six bussed VME bus lines and four broken or daisy-chained lines. These daisy-chained lines require special signal names. The signals entering each REQUESTER are identified as "Bus Grant IN" lines (BGxIN*), while the signals leaving the REQUESTER are identified as the "Bus Grant OUT" lines (BGxOUT*). Therefore, the lines which leave slot N as BGxOUT* enter slot N+1 as BGxIN*. This is illustrated in Figure 3-2.

<u>NOTE</u>

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 may have any value from zero to three.

In the VME bus arbitration system, a REQUESTER will drive the following lines:

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

If the REQUESTER does not use some levels of bus requests, it may jumper the respective bus grant in lines to their respective bus grant out lines instead of driving those bus grant out lines.

The ARBITER will drive the following:

1	bus clear line	(BCLR*) (Option PRI ARBITER only)
4	bus grant in lines	(BG0IN* through BG3IN*)

An option ONE ARBITER drives only BG3IN*.

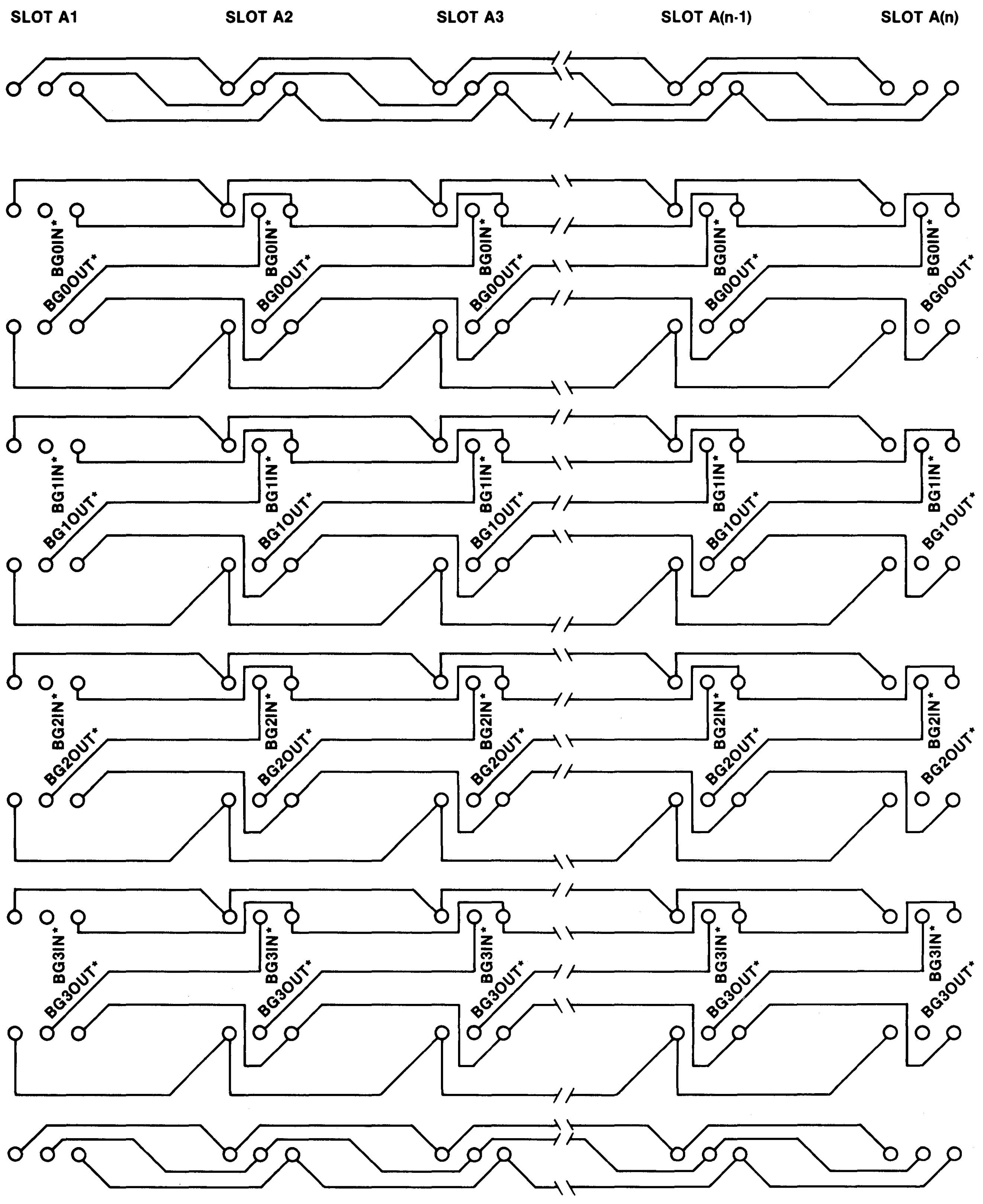


FIGURE 3-2. Illustration of the Daisy-Chained Bus Grant Lines

Two additional lines are intimately connected with the power-up and power-down sequencing of the arbitration system. These are:

1	system reset line	(SYSRESET*)
1	AC power fail	(ACFAIL*)

While their impact on the arbitration system is included in this chapter, these lines will be discussed further in the chapter on UTILITY bus lines.

3.2.1 Bus Request and Bus Grant Lines

The bus request lines are used by each REQUESTER to ask for use of the data transfer bus. The bus grant lines are the ARBITER'S means of awarding that use. However, at any given point on the bus, the signal on the BGxIN* lines may no longer be driven to the same level as the BGxIN* lines driven by the ARBITER. Each of these lines enters a given slot on its BGxIN* pins, but leaves for the next board on this slot's BGxOUT* pins. This type of structure is called daisy-chaining. (See Figure 3-2).

If a board is not using a particular request/grant level (identified by the 'x' in the signal name), the signal is passed through by a jumper. In the case where the board uses the request/grant level being examined, the signal BGxIN* will be gated on board. If this REQUESTER is currently asking for the bus, it will not pass on the low level grant via BGxOUT*. If it is not requesting the bus, a low level will appear on BGxOUT* (with a maximum delay of 70 nanoseconds) after receipt of the low BGxIN*. If a slot does not contain a board, it is necessary that these bus grant signals be jumpered around the slot. The backplane mechanical specification will define a provision for the installation of jumpers at each slot.

This daisy-chain structure allows two levels of prioritization for VME bus access. The four bus request lines are prioritized so that a PRI or RRS ARBITER will issue a grant to the "highest" level of BR signal, based on the prioritizing options selected (fixed or rotating).

Within a given level, the prioritization is accomplished by the daisy chain. The slot closest to the ARBITER will have the highest priority, and the priority will decrease with distance along the chain. Because of this physical structure, THE ARBITER MUST BE LOCATED IN SLOT 1. The ARBITER actually drives the pins BGxIN* in slot 1, so that any REQUESTERS on the board in slot 1 may follow the same process as they would on any other board. Such a design creates uniformity in the structure of the VME bus interface on each board, and modularizes the ARBITER and REQUESTER functions.

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 will drive BBSY* low. Control of the bus may not be taken from this REQUESTER until it releases BBSY*.

3.2.3 Bus Clear Line (BCLR*)

Bus clear is the line used by a PRI ARBITER to inform the MASTER currently in control of the DTB that a higher priority request is now pending. The current MASTER is not required to relinquish control immediately. Typically, it will continue transferring data until it reaches an appropriate break-off point, and then allow its on-board REQUESTER to release BBSY*.

Bus clear is driven only by option PRI ARBITERS. Because the bus request lines have no fixed priority in a round robin arbitration scheme, an RRS ARBITER does not drive bus clear (BCLR*). Because of the bus terminations, BCLR* will always be high.

3.3 FUNCTIONAL MODULES

The arbitration subsystem is composed of several modules:

- One data transfer bus ARBITER (Option PRI, RRS, or ONE)
- One or more data transfer bus REQUESTERS
- One or more data transfer MASTERS

<u>NOTE</u>

Although SYSRESET* and ACFAIL* are not specified as part of the arbitration bus, it is necessary to look at the arbitration subsystem response to these signal lines. (SYSRESET* and ACFAIL* are driven by the power monitor module which is discussed in Chapter 5.)

3.3.1 Data Transfer Bus ARBITER

The tasks of the data transfer bus ARBITER are to prioritize the incoming bus requests and grant the bus to the appropriate REQUESTER by generating the matching BGxIN* signal for that level. Where a fixed priority (PRI) request scheme is used, the ARBITER also informs any MASTER currently in control of the bus when a higher level request is pending by driving BCLR* to low.

A block diagram of a PRI ARBITER is given in Figure 3-3. It uses as its prioritization inputs the four bus request lines BR0* through BR3*, and responds with BG0IN* through BG3IN*, as appropriate.

To visualize a round robin (RRS) ARBITER, consider a mechanical switch being driven by a stepping motor. Each position on the switch corresponds to one level of bus request and bus grant. When the bus is busy, the switch is stopped on the level which is currently using the bus. Upon release of the bus, the switch will step one position lower (i.e., from BR(n)* to BR(n-1)*) and test for request. It will then continue this scanning operation until a request is found and send a bus grant over the appropriate line.

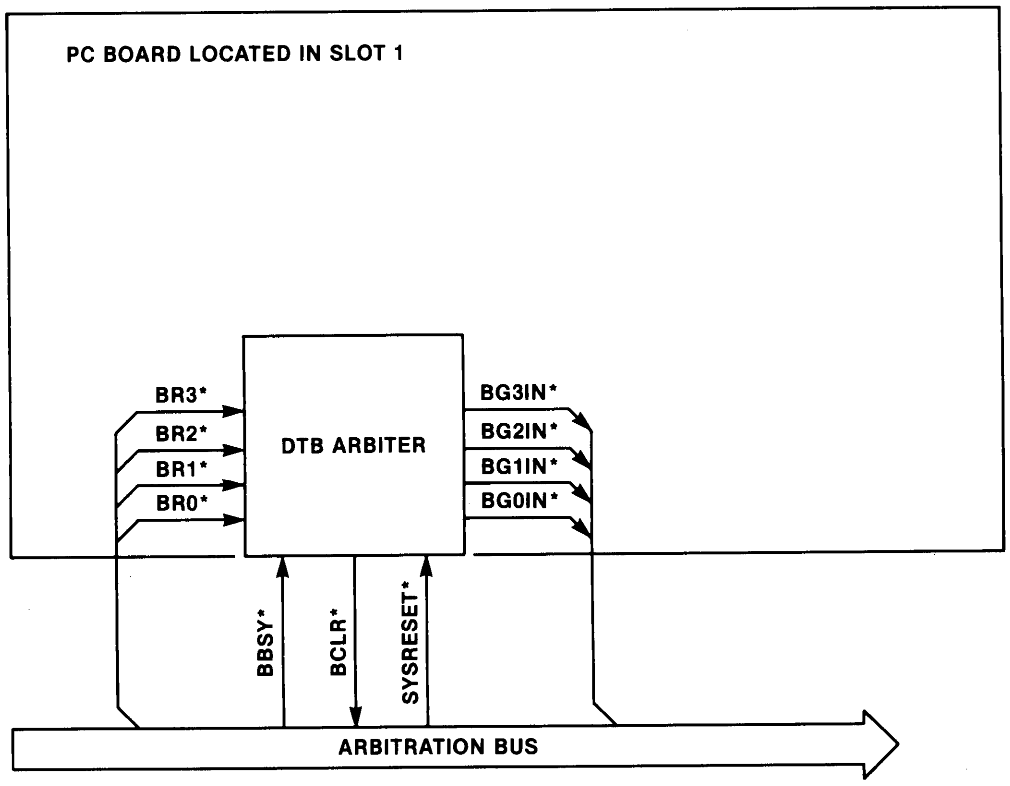


FIGURE 3-3. Block Diagram: Option PRI DTB ARBITER

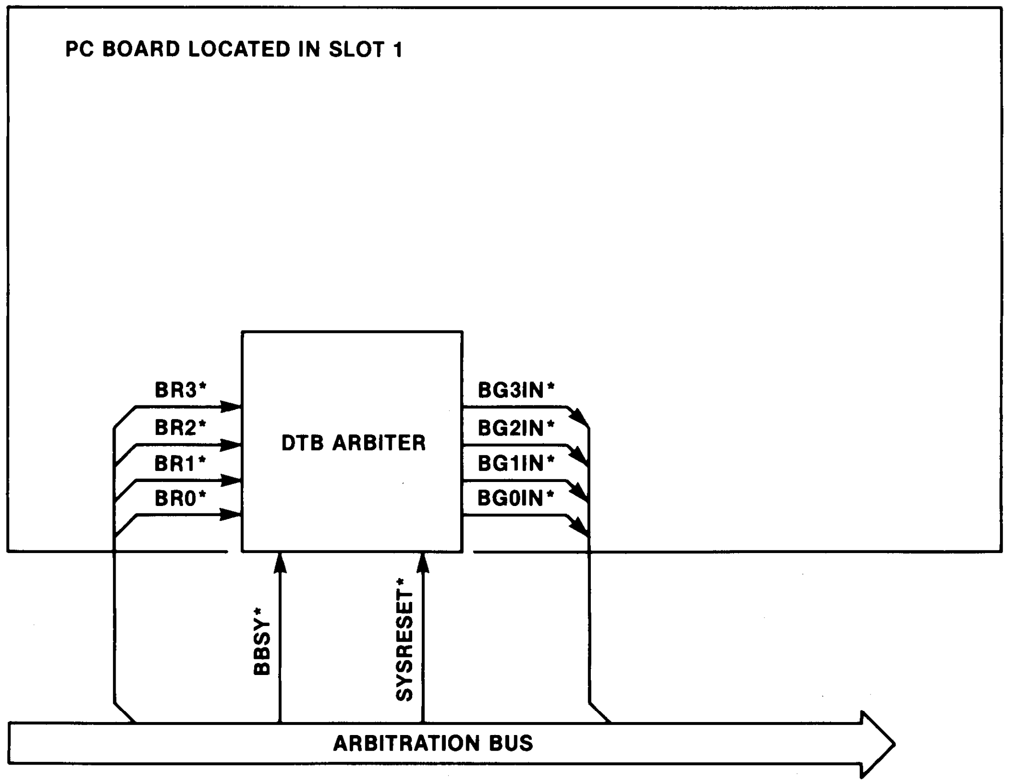


FIGURE 3-4. Block Diagram: Option RRS DTB ARBITER

3.3.2 Data Transfer Bus REQUESTER

Each REQUESTER in the system is required to:

- translate the MASTER WANTS BUS on-board condition to an appropriate bus request,
- accept the incoming bus grant signal for that same level and, if the on-board MASTER does not want the bus, provide the same level outgoing bus grant signal, or
- if the on-board MASTER does want the bus, translate the bus grant to an internal latch which provides the response MASTER GRANTED BUS and the signal Bus Busy (BBSY*) as long as the MASTER is indicating it wants the bus.

In the simplest REQUESTER, (Option RWD (Release When Done)), BBSY* will be released when the MASTER stops indicating that it wants the bus. In systems where maximum data transfer rate is critical, a slightly more complex REQUESTER (Option ROR (Release on Request)) would be implemented that does not drop BBSY* upon loss of the MASTER'S request. This REQUESTER would monitor the four bus request lines and drop BBSY* only if another bus request is pending. Use of this latter option would reduce the number of arbitrations initiated by a MASTER which is generating a large percentage of the bus traffic. See Figures 3-5 and 3-6.

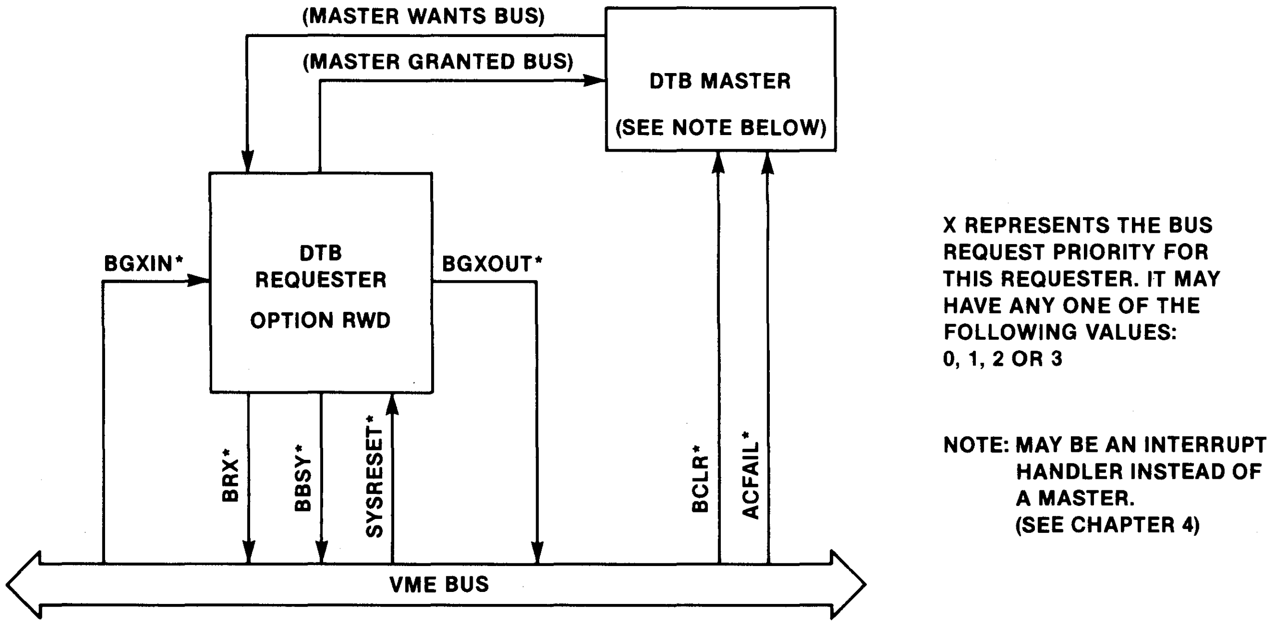

(MASTER WANTS BUS)
(MASTER GRANTED BUS)
DTB MASTER
(SEE NOTE BELOW)
BGXIN*
DTB REQUESTER
OPTION RWD
BGXOUT*
BRX*
BBSY*
SYSRESET*
BCLR*
ACFAIL*
VME BUS
X REPRESENTS THE BUS REQUEST PRIORITY FOR THIS REQUESTER. IT MAY HAVE ANY ONE OF THE FOLLOWING VALUES: 0, 1, 2 OR 3
NOTE: MAY BE AN INTERRUPT HANDLER INSTEAD OF A MASTER. (SEE CHAPTER 4)

FIGURE 3-5. Block Diagram: Option RWD REQUESTER

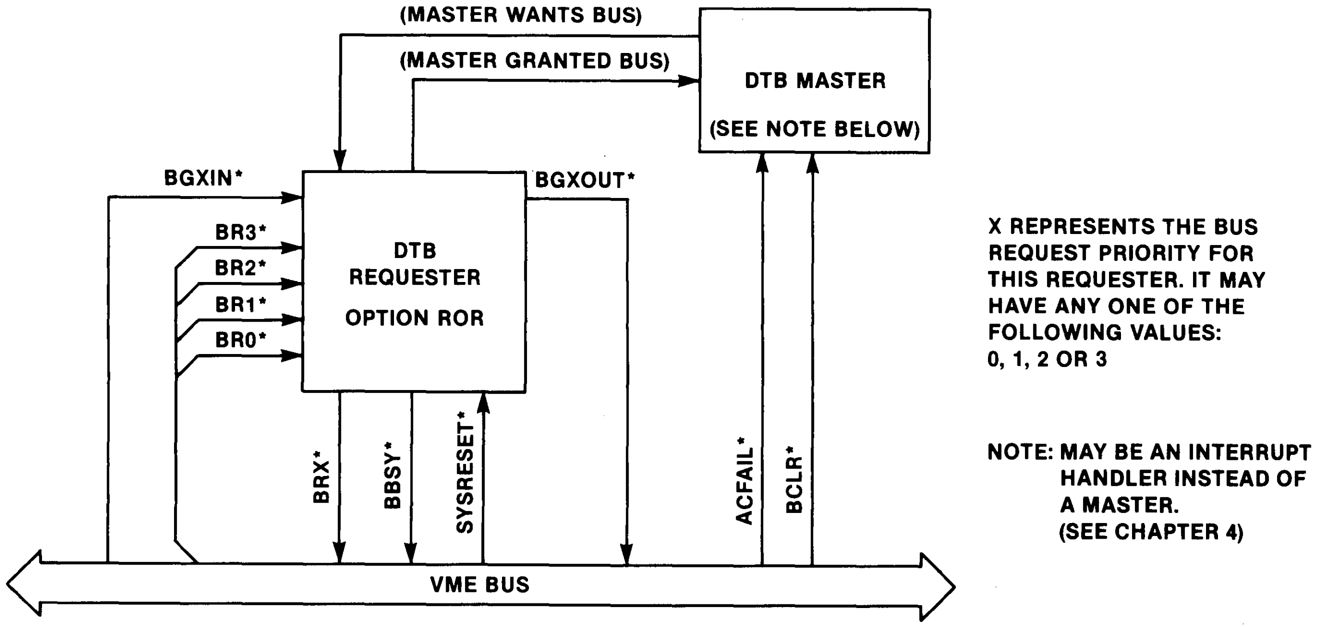

(MASTER WANTS BUS)
(MASTER GRANTED BUS)
DTB MASTER
(SEE NOTE BELOW)
BGXIN*
BGXOUT*
BR3*
BR2*
BR1*
BR0*
DTB
REQUESTER
OPTION ROR
BRX*
BBSY*
SYSRESET*
ACFAIL*
BCLR*
VME BUS
X REPRESENTS THE BUS REQUEST PRIORITY FOR THIS REQUESTER. IT MAY HAVE ANY ONE OF THE FOLLOWING VALUES: 0, 1, 2 OR 3
NOTE: MAY BE AN INTERRUPT HANDLER INSTEAD OF A MASTER. (SEE CHAPTER 4)

FIGURE 3-6. Block Diagram: Option ROR REQUESTER

3.3.3 Data Transfer Bus MASTER Considerations

While the data transfer characteristics of two classes of MASTER have been covered in a previous chapter, it is now necessary to look at additional controls and responses which the MASTER must deal with to use the arbitration subsystem. Each MASTER should monitor the two VME bus signals ACFAIL* and BCLR*. Both of these signals inform the MASTER that another need for the bus exists which is greater than its own. In the case of BCLR*, the MASTER'S design determines how long it will maintain control of the bus. For example, if a MASTER provides a DMA interface for a very high-speed device, such as a hard disk, the MASTER may not be able to relinquish the bus during a sector transfer without loss of data. Therefore, the MASTER might keep the bus for as long as the sector transfer takes. ACFAIL* differs from BCLR* in that it 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. Even in this case, the MASTER is allowed 200 microseconds to relinquish the DTB. This should normally be sufficient to allow an orderly termination of activity.

In examining the relationship between REQUESTER and MASTER, we see that each REQUESTER is associated with a particular MASTER and acts as its interface to the arbitration bus. While the normal relationship is one to one, it would be possible for a MASTER which requires multiple levels of bus request to have multiple REQUESTERS. In this case, each REQUESTER would request the DTB on a different level. Higher priority data transfers might then be done at one request level and lower priority transfers at a lower level. This would be effective only with option PRI ARBITERS.

3.4 TYPICAL OPERATION

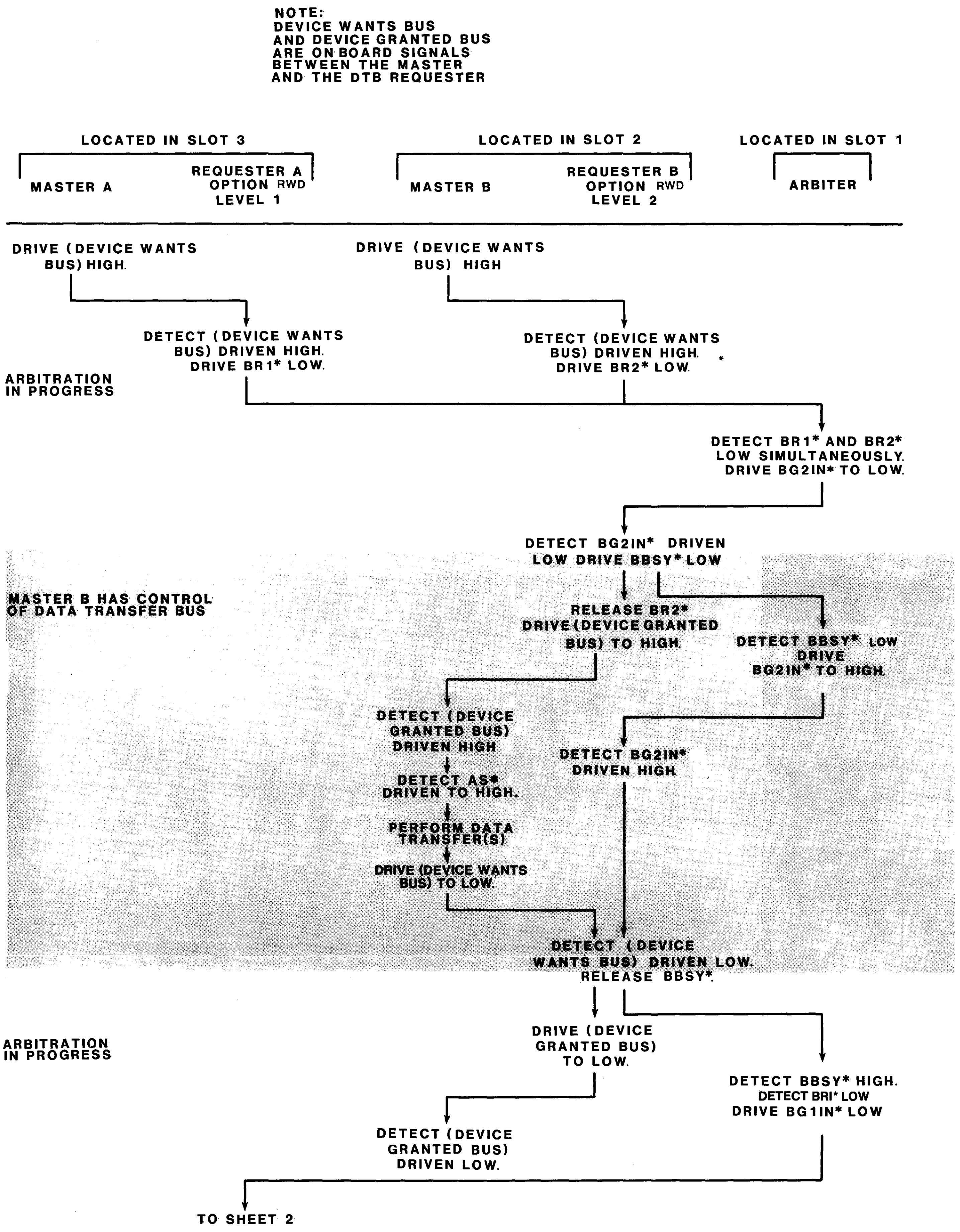
3.4.1 Arbitration of Two Different Levels of Bus Request

Figures 3-7 and 3-8 illustrate the sequence of events which take place when two REQUESTERS send simultaneous bus requests to an ARBITER on different bus request lines. When the sequence begins, each of the REQUESTERS is driving its respective bus request line low (REQUESTER A drives BR1* and REQUESTER B drives BR2*). Assuming that the ARBITER detects BR1* and BR2* low simultaneously, it will drive BG2IN* of slot 1 low because BR2* has the highest priority. When the signal has propagated through to REQUESTER B, REQUESTER B will respond to the low BG2IN* level by driving BBSY* low. It then releases the BR2* line and informs its own MASTER (MASTER B) that the DTB is available.

After detecting BBSY* low, the ARBITER drives BG2IN* high. Note that BBSY* and the bus grants are interlocked as shown in Figure 3-8 (i.e., the ARBITER is not permitted to drive the grant high until it detects BBSY* low). When MASTER B completes its data transfer(s), REQUESTER B releases BBSY*, provided BG2IN* has been received high and 30 nanoseconds have elapsed since the release of BR2*. This 30-ns delay ensures that the ARBITER will not interpret the old low BR2* level as another request. REQUESTER B will wait until the 30 nanoseconds have elapsed, and will then release BBSY*.

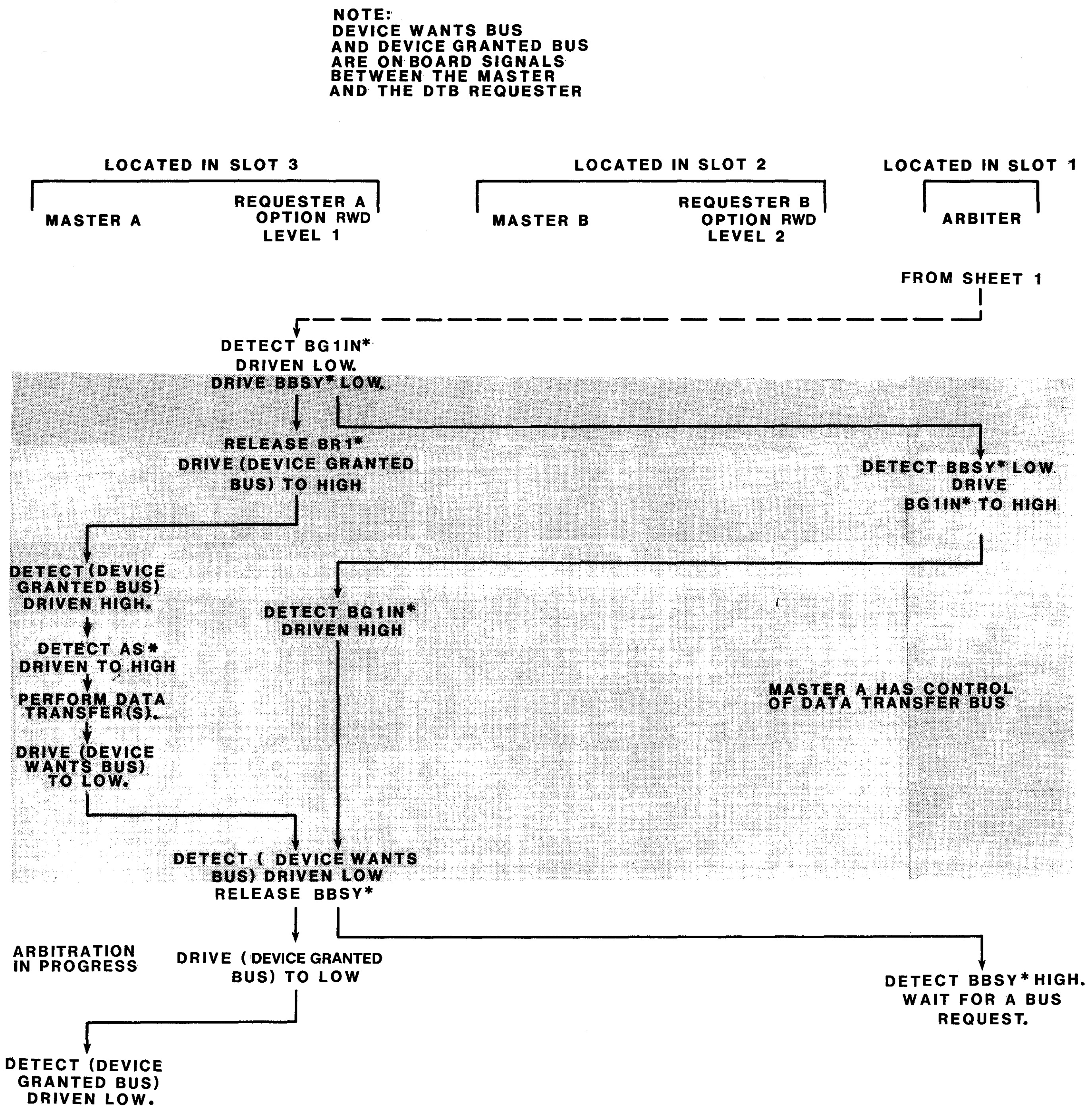
The ARBITER interprets the release of BBSY* as a signal to arbitrate the bus requests. Since BR1* is low (the only bus request being driven low), the ARBITER grants the DTB to REQUESTER A by driving BG1IN* low. REQUESTER A responds by driving BBSY* low. When MASTER A completes its data transfer(s), REQUESTER A releases BBSY*, provided BG1IN* has been received high and 30 nanoseconds have elapsed since the release of BR1*. In this example, since no bus request lines are driven low when REQUESTER A releases BBSY*, the ARBITER will be idle until a new request is made.

Note that this description would hold for both PRI and RRS option 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.



NOTE: DEVICE WANTS BUS AND DEVICE GRANTED BUS ARE ON BOARD SIGNALS BETWEEN THE MASTER AND THE DTB REQUESTER
LOCATED IN SLOT 3
MASTER A
REQUESTER A OPTION RWD LEVEL 1
LOCATED IN SLOT 2
MASTER B
REQUESTER B OPTION RWD LEVEL 2
LOCATED IN SLOT 1
ARBITER
DRIVE (DEVICE WANTS BUS) HIGH.
DRIVE (DEVICE WANTS BUS) HIGH
DETECT (DEVICE WANTS BUS) DRIVEN HIGH. DRIVE BR1* LOW.
DETECT (DEVICE WANTS BUS) DRIVEN HIGH. DRIVE BR2* LOW.
ARBITRATION IN PROGRESS
DETECT BR1* AND BR2* LOW SIMULTANEOUSLY. DRIVE BG2IN* TO LOW.
DETECT BG2IN* DRIVEN LOW DRIVE BBSY* LOW
MASTER B HAS CONTROL OF DATA TRANSFER BUS
RELEASE BR2* DRIVE (DEVICE GRANTED BUS) TO HIGH.
DETECT BBSY* LOW DRIVE BG2IN* TO HIGH.
DETECT (DEVICE GRANTED BUS) DRIVEN HIGH
DETECT AS* DRIVEN TO HIGH.
PERFORM DATA TRANSFER(S)
DRIVE (DEVICE WANTS BUS) TO LOW.
DETECT BG2IN* DRIVEN HIGH
DETECT (DEVICE WANTS BUS) DRIVEN LOW. RELEASE BBSY*.
ARBITRATION IN PROGRESS
DRIVE (DEVICE GRANTED BUS) TO LOW.
DETECT BBSY* HIGH. DETECT BRI* LOW DRIVE BG1IN* LOW
DETECT (DEVICE GRANTED BUS) DRIVEN LOW.
TO SHEET 2

FIGURE 3-7. Arbitration Flow Diagram: Two REQUESTERS, Two Request Levels (Sheet 1 of 2)


NOTE:
DEVICE WANTS BUS AND DEVICE GRANTED BUS ARE ON BOARD SIGNALS BETWEEN THE MASTER AND THE DTB REQUESTER
LOCATED IN SLOT 3
MASTER A
REQUESTER A OPTION RWD LEVEL 1
LOCATED IN SLOT 2
MASTER B
REQUESTER B OPTION RWD LEVEL 2
LOCATED IN SLOT 1
ARBITER
FROM SHEET 1
DETECT BG1IN* DRIVEN LOW. DRIVE BBSY* LOW.
RELEASE BR1* DRIVE (DEVICE GRANTED BUS) TO HIGH
DETECT BBSY* LOW. DRIVE BG1IN* TO HIGH
DETECT (DEVICE GRANTED BUS) DRIVEN HIGH.
DETECT BG1IN* DRIVEN HIGH
DETECT AS* DRIVEN TO HIGH
PERFORM DATA TRANSFER(S).
MASTER A HAS CONTROL OF DATA TRANSFER BUS
DRIVE (DEVICE WANTS BUS) TO LOW.
DETECT (DEVICE WANTS BUS) DRIVEN LOW RELEASE BBSY*
ARBITRATION IN PROGRESS
DRIVE (DEVICE GRANTED BUS) TO LOW
DETECT BBSY* HIGH. WAIT FOR A BUS REQUEST.
DETECT (DEVICE GRANTED BUS) DRIVEN LOW.

NOTE

The on-board signals between the MASTER and REQUESTER are interlocked. The MASTER may not drive MASTER WANTS BUS low until MASTER GRANTED BUS has gone high from the previous cycle.

FIGURE 3-7. Arbitration Flow Diagram: Two REQUESTERS, Two Request Levels (Sheet 2 of 2)

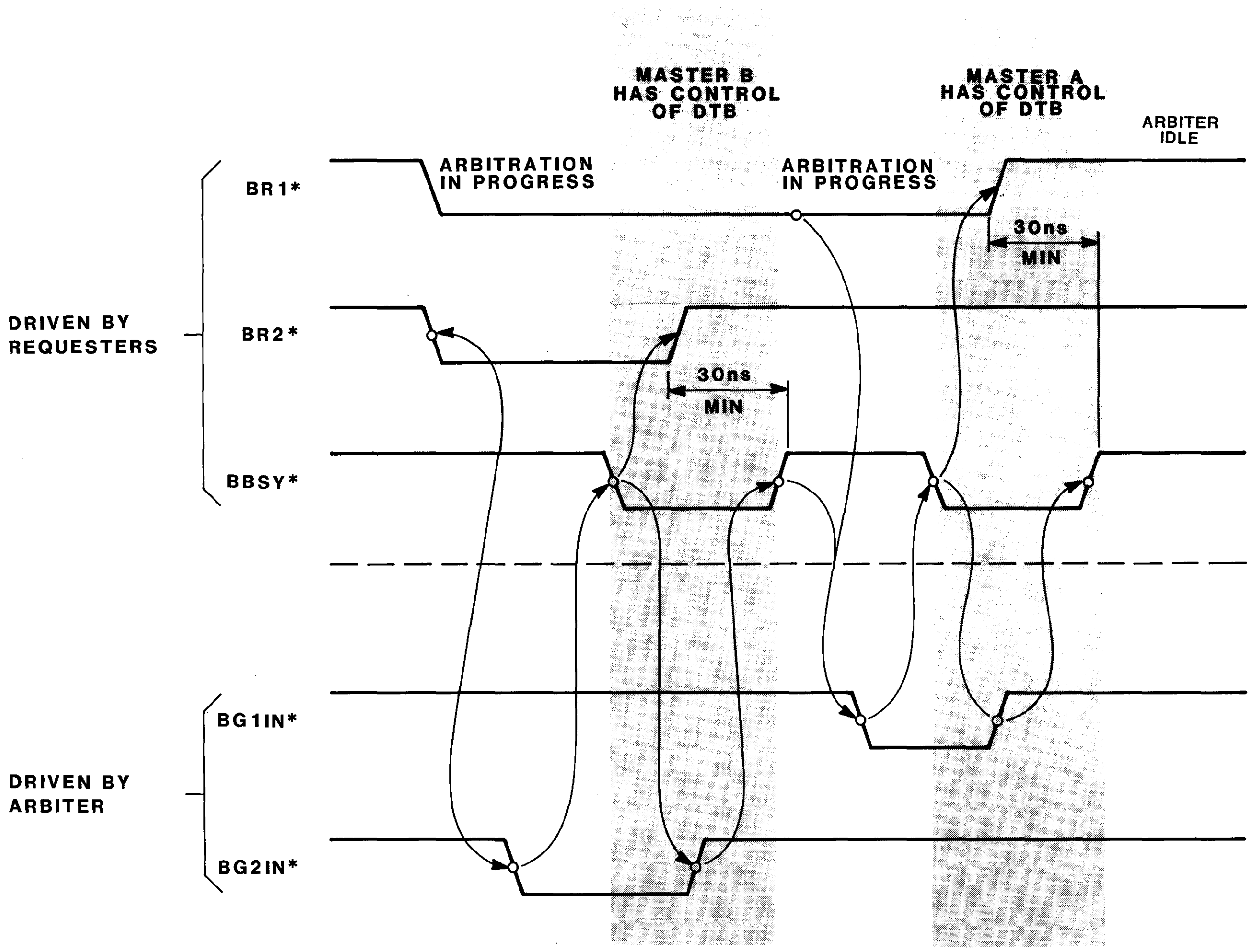


FIGURE 3-8. Arbitration Sequence Diagram: Two REQUESTERS, Two Request Levels

3.4.2 Arbitration of Two Bus Requests on the Same Bus Request Line

Figures 3-9 and 3-10 illustrate the sequence of events which take place when an option ROR REQUESTER and an option RWD REQUESTER send simultaneous requests to an ARBITER on a common bus request line. In this example, the ARBITER and option RWD REQUESTER are located on the system controller board in the first board slot, with the option ROR REQUESTER located in the second board slot. When the sequence begins, each of the REQUESTERS is requesting the DTB by driving BR1* low. The ARBITER in board slot 1 detects the BR1* low and since BBSY* is high, it drives BG1IN* low to its own slot. BG1IN* is monitored by REQUESTER A (also in slot 1). When REQUESTER A in slot 1 detects BG1IN* low, it responds by driving BBSY* low. At the same time, it informs its own DTB MASTER (MASTER A) that the DTB is available. It also releases BR1*. (Note: BR1* remains low because REQUESTER B is still driving it low.)

After detecting BBSY* low, the ARBITER drives BG1IN* high. When DTB MASTER A has completed its data transfer(s), it drives MASTER WANTS BUS low. When REQUESTER A detects MASTER WANTS BUS low, and the minimum delay since the release of BR1* has been satisfied, REQUESTER A releases BBSY*.

The ARBITER interprets the release of BBSY* as a signal to arbitrate the 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 ARBITER detects the low BBSY*, it drives BG1IN* high, which causes REQUESTER A to drive its BG1OUT* high. After detecting its BG1IN* high, REQUESTER B receives a low DEVICE WANTS BUS on-board signal, indicating that its MASTER has finished using the DTB.

Since REQUESTER B is an option ROR REQUESTER, it does not release BBSY*, but keeps it driven low. In the event its MASTER wishes 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 to allow the ARBITER to grant the bus to REQUESTER A. Figure 3-10 is a sequence diagram illustrating this.

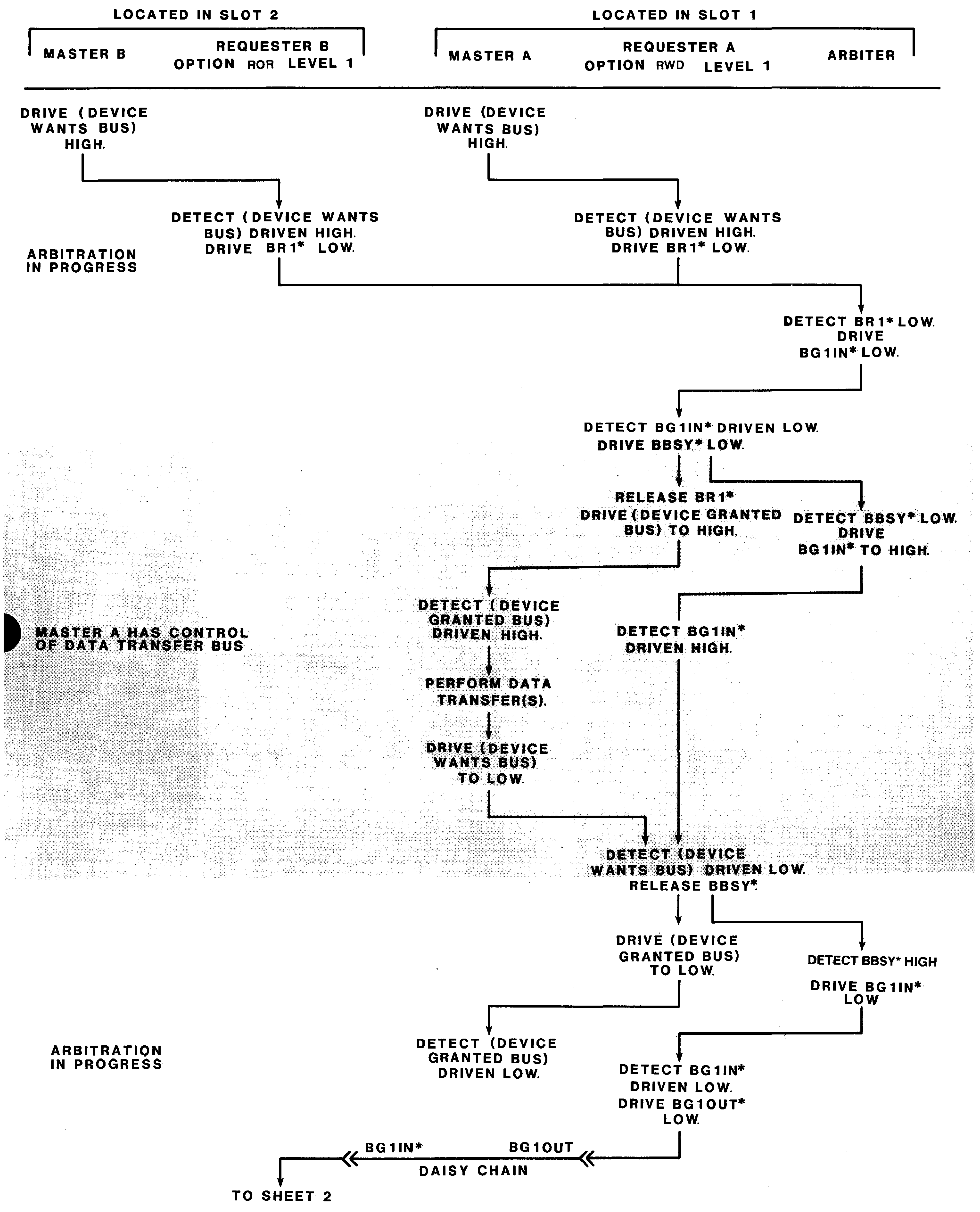

LOCATED IN SLOT 2
MASTER B
REQUESTER B OPTION ROR LEVEL 1
LOCATED IN SLOT 1
MASTER A
REQUESTER A OPTION RWD LEVEL 1
ARBITER
DRIVE (DEVICE WANTS BUS) HIGH.
DRIVE (DEVICE WANTS BUS) HIGH.
DETECT (DEVICE WANTS BUS) DRIVEN HIGH. DRIVE BR1* LOW.
DETECT (DEVICE WANTS BUS) DRIVEN HIGH. DRIVE BR1* LOW.
ARBITRATION IN PROGRESS
DETECT BR1* LOW. DRIVE BG1IN* LOW.
DETECT BG1IN* DRIVEN LOW. DRIVE BBSY* LOW.
RELEASE BR1* DRIVE (DEVICE GRANTED BUS) TO HIGH.
DETECT BBSY* LOW. DRIVE BG1IN* TO HIGH.
MASTER A HAS CONTROL OF DATA TRANSFER BUS
DETECT (DEVICE GRANTED BUS) DRIVEN HIGH.
DETECT BG1IN* DRIVEN HIGH.
PERFORM DATA TRANSFER(S).
DRIVE (DEVICE WANTS BUS) TO LOW.
DETECT (DEVICE WANTS BUS) DRIVEN LOW. RELEASE BBSY*.
DRIVE (DEVICE GRANTED BUS) TO LOW.
DETECT BBSY* HIGH
DRIVE BG1IN* LOW
ARBITRATION IN PROGRESS
DETECT (DEVICE GRANTED BUS) DRIVEN LOW.
DETECT BG1IN* DRIVEN LOW. DRIVE BG1OUT* LOW.
BG1IN*
BG1OUT
DAISY CHAIN
TO SHEET 2

FIGURE 3-9. Arbitration Flow Diagram - Two REQUESTERS/Same Request Level (Sheet 1 of 2)

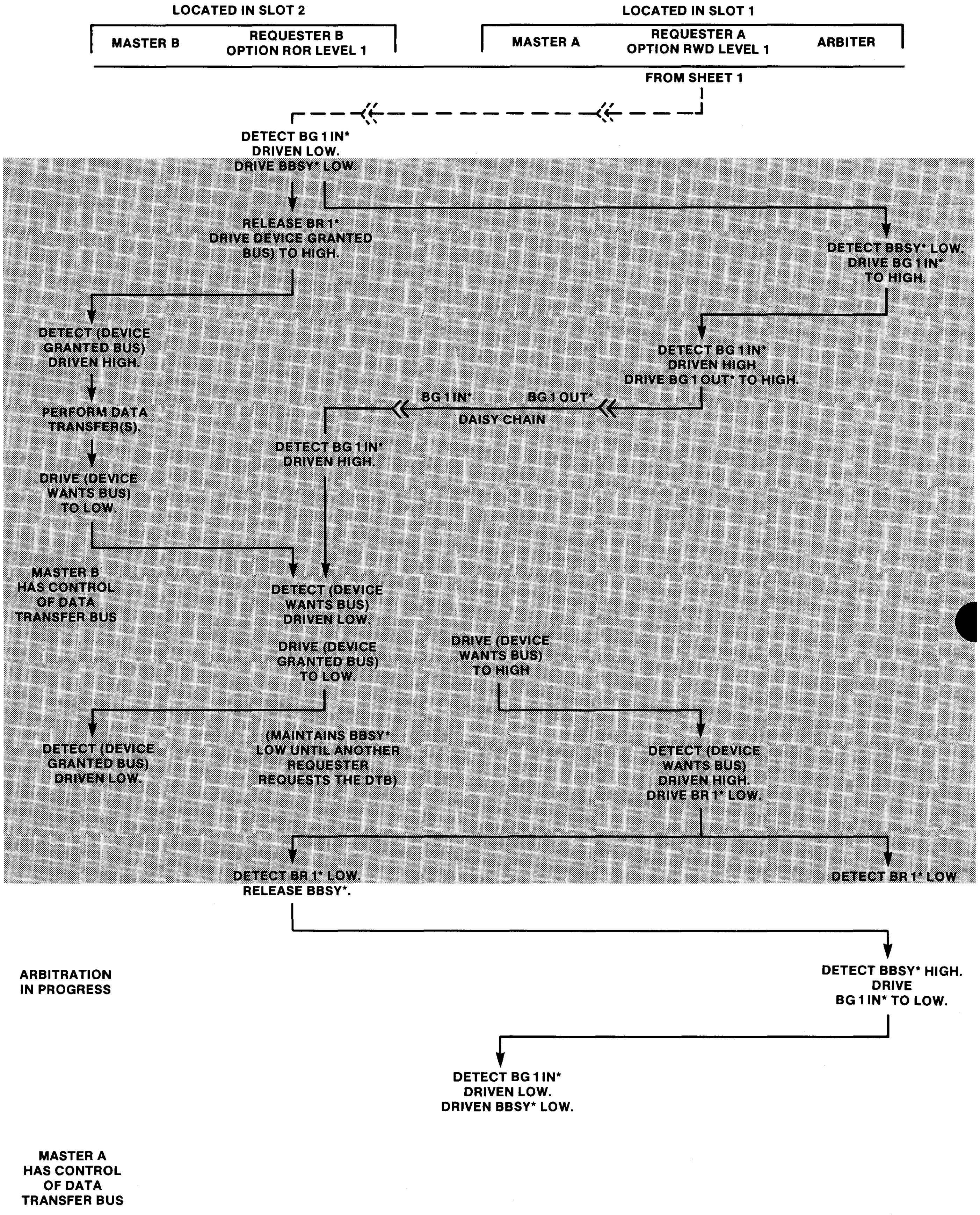

LOCATED IN SLOT 2
MASTER B
REQUESTER B OPTION ROR LEVEL 1
LOCATED IN SLOT 1
MASTER A
REQUESTER A OPTION RWD LEVEL 1
ARBITER
FROM SHEET 1
DETECT BG 1 IN* DRIVEN LOW. DRIVE BBSY* LOW.
RELEASE BR 1* DRIVE DEVICE GRANTED BUS) TO HIGH.
DETECT BBSY* LOW. DRIVE BG 1 IN* TO HIGH.
DETECT (DEVICE GRANTED BUS) DRIVEN HIGH.
DETECT BG 1 IN* DRIVEN HIGH DRIVE BG 1 OUT* TO HIGH.
BG 1 IN*
BG 1 OUT*
DAISY CHAIN
PERFORM DATA TRANSFER(S).
DETECT BG 1 IN* DRIVEN HIGH.
DRIVE (DEVICE WANTS BUS) TO LOW.
MASTER B HAS CONTROL OF DATA TRANSFER BUS
DETECT (DEVICE WANTS BUS) DRIVEN LOW.
DRIVE (DEVICE GRANTED BUS) TO LOW.
DRIVE (DEVICE WANTS BUS) TO HIGH
DETECT (DEVICE GRANTED BUS) DRIVEN LOW.
(MAINTAINS BBSY* LOW UNTIL ANOTHER REQUESTER REQUESTS THE DTB)
DETECT (DEVICE WANTS BUS) DRIVEN HIGH. DRIVE BR 1* LOW.
DETECT BR 1* LOW. RELEASE BBSY*.
DETECT BR 1* LOW
ARBITRATION IN PROGRESS
DETECT BBSY* HIGH. DRIVE BG 1 IN* TO LOW.
DETECT BG 1 IN* DRIVEN LOW. DRIVEN BBSY* LOW.
MASTER A HAS CONTROL OF DATA TRANSFER BUS

FIGURE 3-9. Arbitration Flow Diagram - Two REQUESTERS/Same Request Level (Sheet 2 of 2)

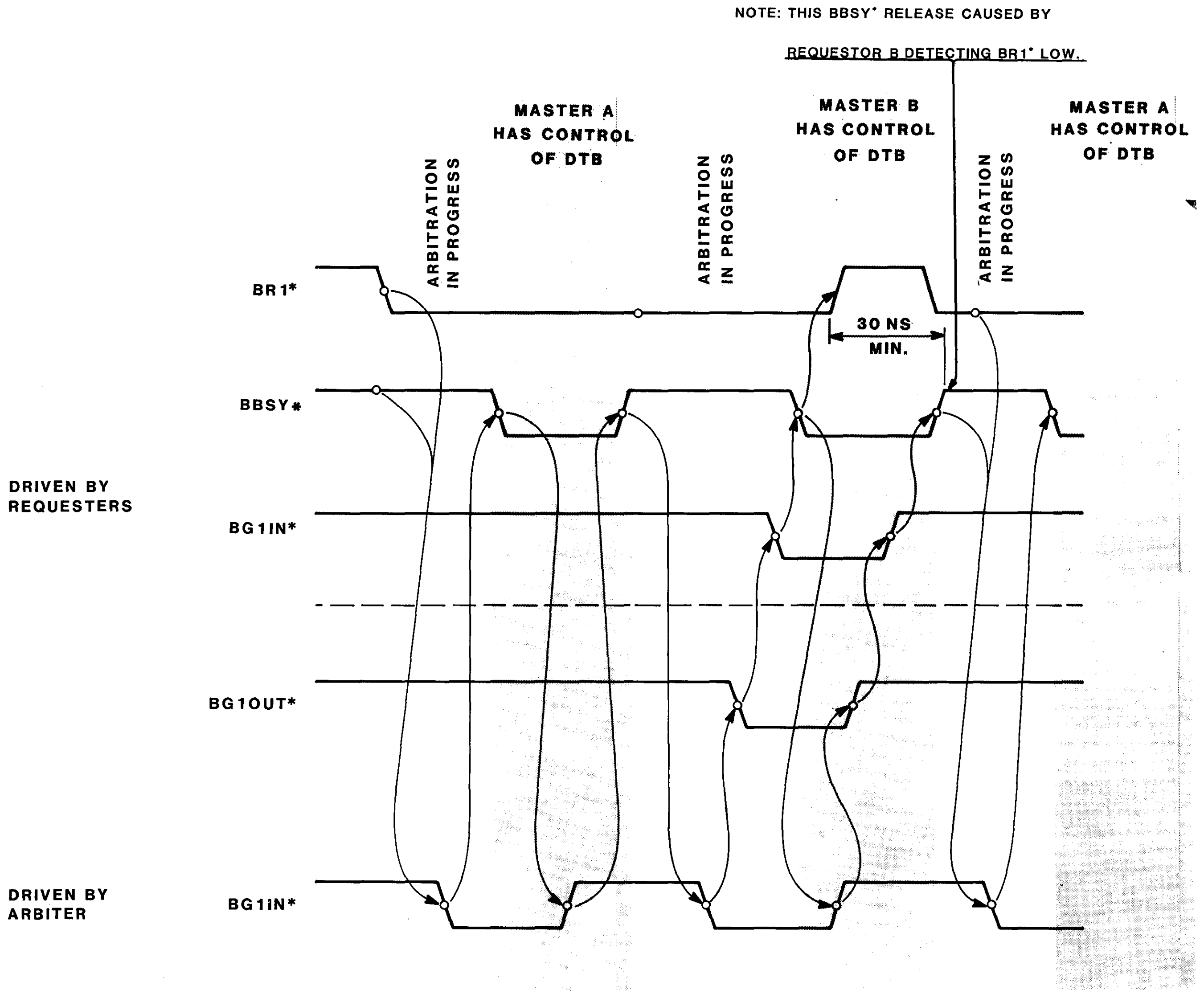

NOTE: THIS BBSY* RELEASE CAUSED BY REQUESTOR B DETECTING BR1* LOW.
MASTER A HAS CONTROL OF DTB
MASTER B HAS CONTROL OF DTB
MASTER A HAS CONTROL OF DTB
ARBITRATION IN PROGRESS
ARBITRATION IN PROGRESS
ARBITRATION IN PROGRESS
30 NS MIN.
BR1*
BBSY*
BG1IN*
BG1OUT*
BG1IN*
DRIVEN BY REQUESTERS
DRIVEN BY ARBITER

FIGURE 3-10. Arbitration Sequence Diagram: Two REQUESTERS, Same Request Level

3.4.3 Power-Down and Power-Up Processing

When an AC power failure occurs, some procedure must be followed to shut the system down in an orderly manner. In addition, some procedure must be defined for restoring data on power-up prior to system operation. There are two signal lines used in the power-down and power-up sequence: ACFAIL* and SYSRESET*. The timing of these signals will be discussed further in Chapter 5. The following section discusses the effect these signal lines have upon MASTERS and SLAVES in the system.

Each MASTER in the system must conform to the follow rules on power-down:

a. Within 200 microseconds of ACFAIL* going low, MASTERS must stop requesting the bus for any activity except power fail response.

b. Any MASTER which had a request pending when ACFAIL* went low must limit its subsequent non-power fail activity to 200 microseconds.

On a power-down sequence, each SLAVE in the system must ignore all data transfer requests initiated more than 30 nanoseconds after SYSRESET* goes low.

On a power-up sequence, MASTERS may not initiate any data transfers until 30 nanoseconds after SYSRESET* goes high, and SLAVES may not respond to data transfers until after the same 30-nanosecond delay.

The actual bus accesses required to restore data within the system will depend upon the application, and are not specified here. It is necessary, however, for the user's software operating system to assure that the appropriate system data is restored prior to system operation. This may involve some processor-to-processor communication in the case of a multiprocessing system.

3.5 DESIGN NOTES

Certain additional points need to be discussed for clarity of understanding.

3.5.1 Race Conditions Between MASTER Requests and ARBITER Grants

In the case where two REQUESTERS operate on the same level, the downstream REQUESTER will often request the bus. It is then possible that a grant to this level of request may be arriving at the first REQUESTER just as it is detecting that its own MASTER wants the bus. In the process of deciding to forward the BUS GRANT downstream to the second REQUESTER or to provide a bus granted response to its own MASTER, it is the responsibility of the first REQUESTER to ensure that no small window occurs where the bus is granted to the other party. As an example, consider a design which latches the state of the on-board MASTER WANTS BUS line upon detection of the BUS GRANT IN signal going low. Since most latches may generate oscillating outputs for some small interval if the signal being latched was in transition at the time of the clock, a period of indecision will occur. Obviously, this signal needs additional processing to provide a valid signal with which to control further action. This specification does not define any requirements on the REQUESTER'S decision of what to do with BUS GRANT IN. The specification requires only that the design ensure that the decision is made in such a manner that no pulse is accidentally generated which might be misinterpreted by another REQUESTER further down the daisy-chain. The specification does provide that, if BUS GRANT OUT is going to occur, it should occur within 70 nanoseconds after receipt of BUS GRANT IN. Failure to meet this requirement will not result in improper operation, but will result in a failure to provide a certain level of performance.

3.5.2 REQUESTER Signal Sequence

When the REQUESTER receives a BUS GRANT IN, and its MASTER receives the bus, three events occur:

a. The REQUESTER drives BBSY* to low.
b. The REQUESTER releases its BUS REQUEST level.
c. The DTB MASTER initiates bus activities.

The MASTER and REQUESTER are not constrained to generate any fixed sequence of these events. It is even possible, although meaningless, that the MASTER does not use the bus in response to this particular grant.

The specific constraints that do apply are:

a. The REQUESTER does drive BBSY* to low.

b. The REQUESTER does release BUS REQUEST to high.

c. The REQUESTER holds BBSY* low for at least 30 nanoseconds after BUS REQUEST has gone high to ensure that the ARBITER does not mistakenly start a new arbitration cycle based on that bus request.

d. The REQUESTER holds BBSY* low until BUS GRANT IN goes high to ensure the BBSY* transition to low has been seen by the ARBITER.

In addition, this specification details some considerations and restrictions on the MASTER driving AS* to low in the last cycle of bus usage and the REQUESTER'S release of BBSY*. See Chapter 2 for details.

CHAPTER 4

PRIORITY INTERRUPT

4.1 INTERRUPT PHILOSOPHY

Interrupt subsystems may be divided into two groups:

a. Single handler systems - have a supervisory processor which receives and services all bus interrupts.

b. Distributed systems - have two or more processors which receive and service bus interrupts.

The single handler system architecture is, perhaps, easier to understand because of its similarity to single processor systems. Any system which has interrupt capability must have a set of interrupt servicing routines in its executive software. Each of the routines may be thought of as a task which is activated by an interrupt. If the system has a real-time executive, these interrupt routines would then operate as tasks under this executive.

In a single processor or single handler system, the executive software and all of the interrupt routines are executed by one processor.

4.1.1 Single Handler Systems

Figure 4-1 shows the interrupt structure of a single handler system. This type of architecture is well suited to machine or process control applications. The dedicated processors are the ones typically interfaced to the machine or process being controlled, so it is important that their processing is interrupted as little as possible by bus activity.

As shown in Figure 4-1, the dedicated processors in the system are typically controlling some external machine or process. The task of controlling this machine or process may consist of several subtasks, some of which are non-interruptable (i.e., loss of control may result if the task once started is not finished within a specific time). Therefore, the dedicated processor may mask some or all of its interrupts while executing these non-interruptable subtasks.

To summarize, in a dedicated system, the supervisory processor is the destination for all bus interrupts. This allows it to service all interrupts in a prioritized manner. The dedicated processors are not required to service interrupts from the bus, but give primary attention to the interrupts received from the machine or process which they control.

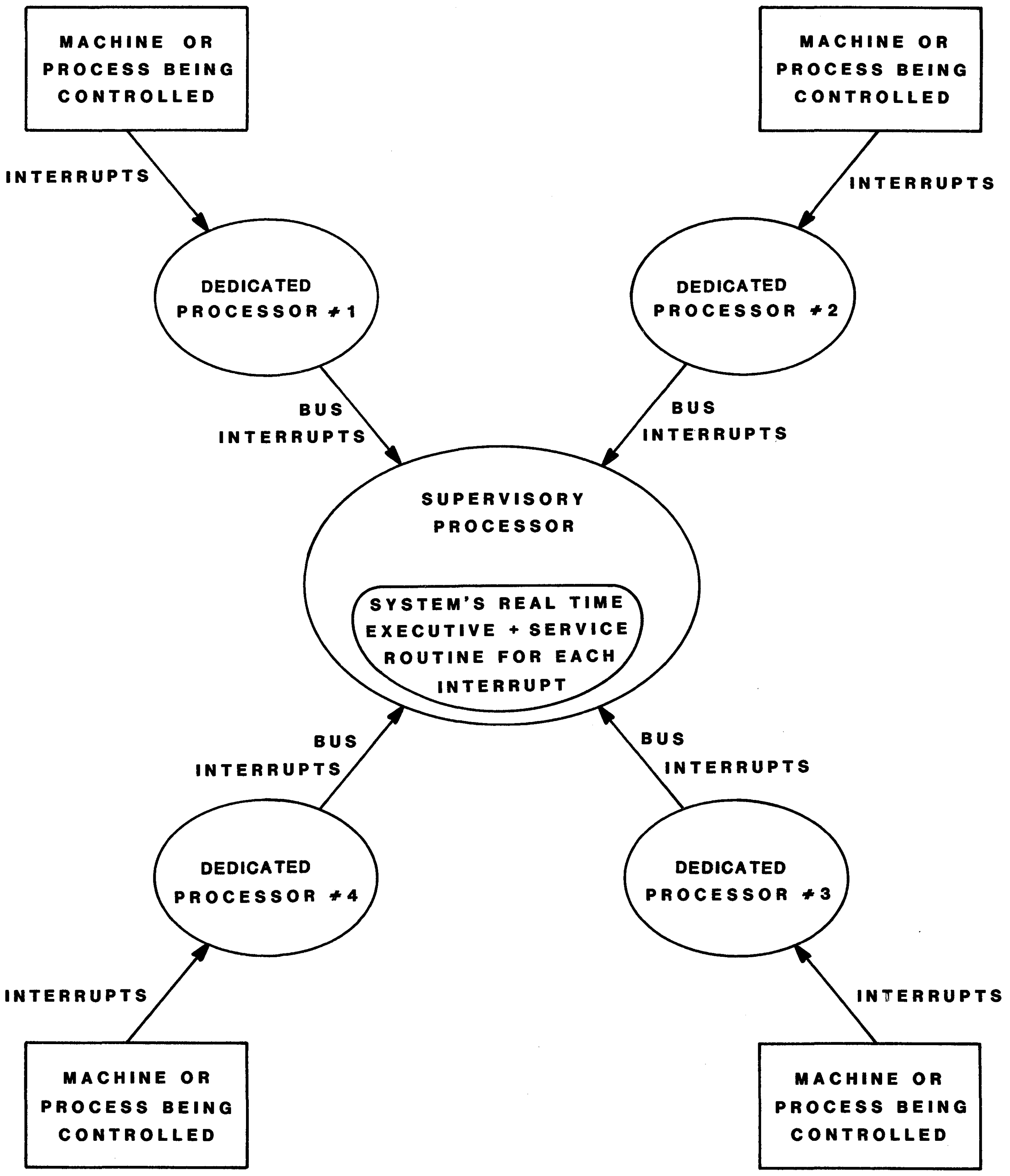

MACHINE OR PROCESS BEING CONTROLLED
MACHINE OR PROCESS BEING CONTROLLED
INTERRUPTS
INTERRUPTS
DEDICATED PROCESSOR #1
DEDICATED PROCESSOR #2
BUS INTERRUPTS
BUS INTERRUPTS
SUPERVISORY PROCESSOR
SYSTEM'S REAL TIME EXECUTIVE + SERVICE ROUTINE FOR EACH INTERRUPT
BUS INTERRUPTS
BUS INTERRUPTS
DEDICATED PROCESSOR #4
DEDICATED PROCESSOR #3
INTERRUPTS
INTERRUPTS
MACHINE OR PROCESS BEING CONTROLLED
MACHINE OR PROCESS BEING CONTROLLED

FIGURE 4-1. Interrupt Subsystem Structure: Single Handler System

4.1.2 Distributed Systems

Figure 4-2 shows the interrupt structure of a distributed system. This type of architecture is well suited to batch processing applications, where incoming tasks may be assigned to the next available processor. Each of the co-equal processors executes part of the system executive software, and services only those interrupts directed to it by other processors within the system. Since the servicing of some of these interrupts may require access to system resources, the parts of the executive software must communicate through globally accessed memory in order to allocate resources and resolve lock-ups.

4.2 SIGNAL LINES USED IN HANDLING INTERRUPTS

The data transfer bus, the arbitration bus, and the interrupt bus are all used in the process of generating and handling bus interrupts.

The following discussion of the priority interrupt subsystem assumes that the reader understands the operation of both the data transfer bus described in Chapter 2, and the arbitration bus described in Chapter 3.

4.2.1 Interrupt Bus Signal Lines

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

IRQ1*
IRQ2*
IRQ3*
IRQ4*
IRQ5*
IRQ6*
IRQ7*
IACK*
IACKIN*/IACKOUT*

Each interrupt request line may be driven low by an INTERRUPTER to request an interrupt. In a single handler system, these interrupt request lines are prioritized, with IRQ7* having the highest priority (see Figure 4-3).

The IACK* line runs the full length of the bus and is connected to the IACKIN* pin of slot A1. When it is driven low, it initiates a low-going transition down the INTERRUPT ACKNOWLEDGE DAISY-CHAIN. This may not occur immediately, since additional constraints are place on the propogation of IACKIN*/IACKOUT* (see following paragraph).

4.2.2 INTERRUPT ACKNOWLEDGE DAISY-CHAIN - IACKIN*/IACKOUT*

Each of the seven interrupt request lines may be shared by two or more INTERRUPTER modules. Because of this, some method must be provided to assure that only one of the modules is acknowledged. This is done by means of the INTERRUPT ACKNOWLEDGE DAISY-CHAIN. This daisy-chain line passes through each board on the VME bus. When an interrupt is acknowledged, IACKIN* is driven low at slot 1. Each module which is driving an interrupt request line low must wait for the low level to arrive at its board slot before accepting the acknowledge. The module accepting the acknowledge does not pass the low level down the daisy-chain, thereby guaranteeing that only one module will be acknowledged.

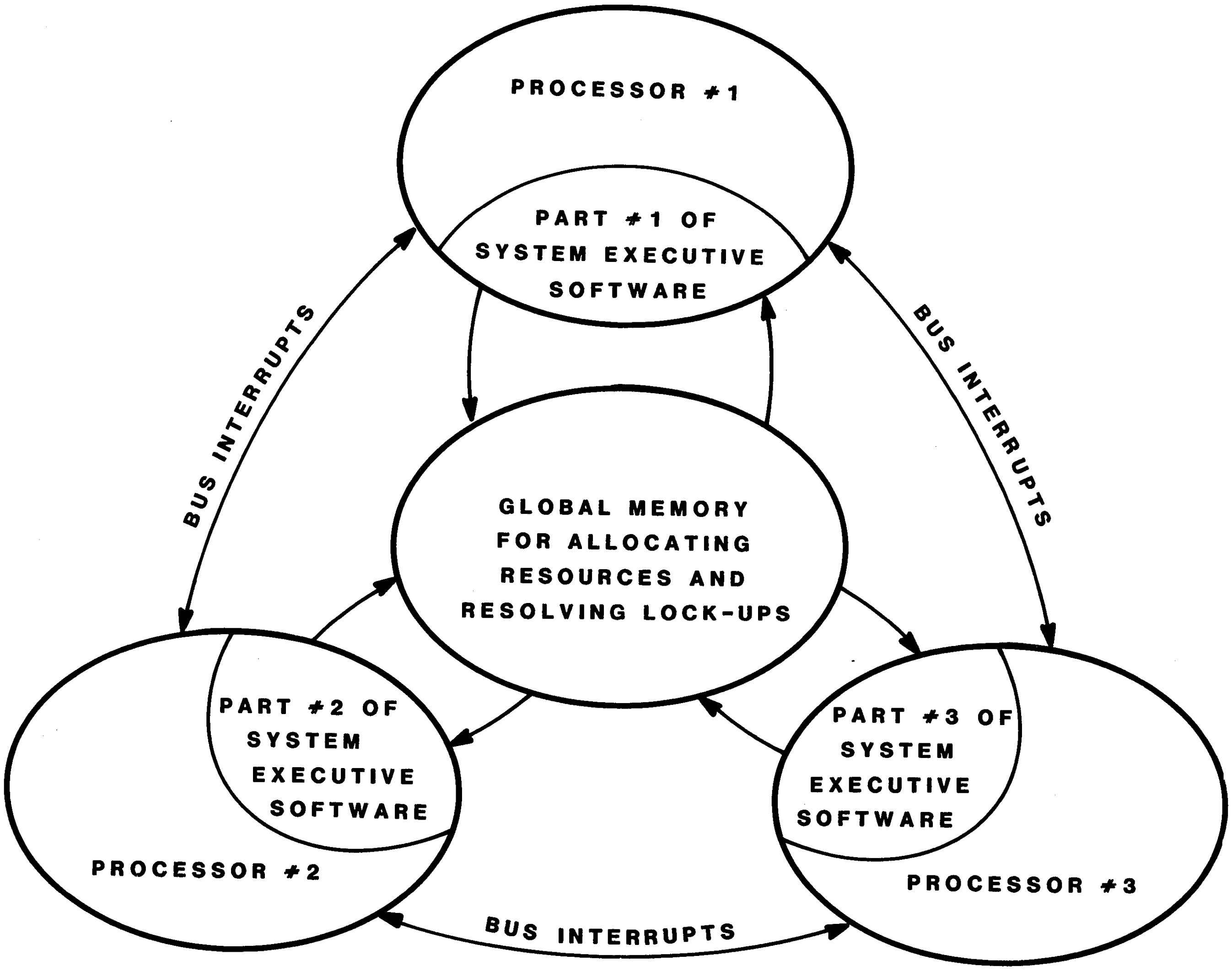

PROCESSOR #1
PART #1 OF SYSTEM EXECUTIVE SOFTWARE
BUS INTERRUPTS
BUS INTERRUPTS
GLOBAL MEMORY FOR ALLOCATING RESOURCES AND RESOLVING LOCK-UPS
PART #2 OF SYSTEM EXECUTIVE SOFTWARE
PROCESSOR #2
PART #3 OF SYSTEM EXECUTIVE SOFTWARE
PROCESSOR #3
BUS INTERRUPTS

FIGURE 4-2. Interrupt Subsystem Structure: Distributed System

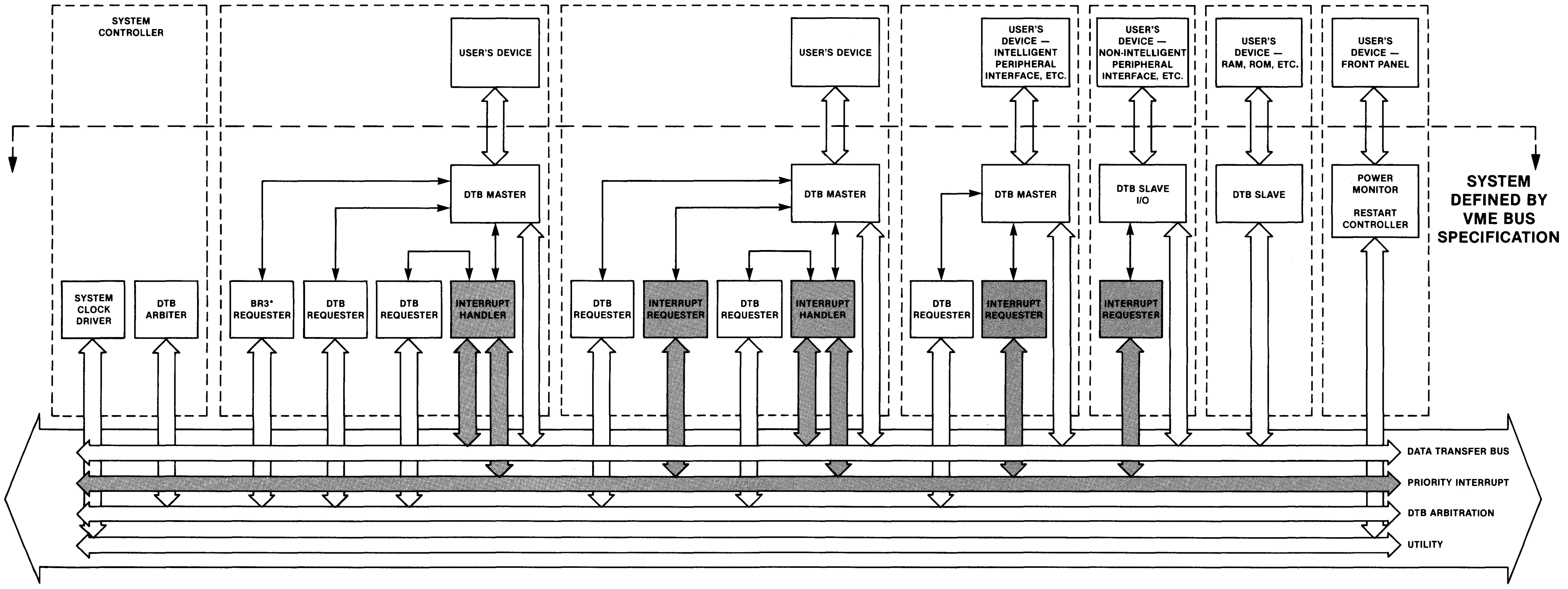


FIGURE 4-3. VME Bus Priority Interrupt Functional Block Diagram

4.3 FUNCTIONAL MODULES

Section 4.2 discussed the additional lines on the VME bus that are used to accomplish interrupt handling. This section discusses in detail the two types of modules which make up a priority interrupt subsystem and how these modules use the VME bus lines.

4.3.1 INTERRUPT HANDLER

The INTERRUPT HANDLER is used to accomplish several tasks:

a. It prioritizes the incoming interrupt requests within its assigned range (max IRQ1*-IRQ7*) from the requests on the interrupt bus.

b. It uses its associated REQUESTER to request the DTB and, when granted use of the DTB, acknowledges the interrupt.

c. It reads the status/ID byte from the INTERRUPTER being acknowledged.

d. Based upon the information received in the status/ID byte, it initiates the appropriate interrupt servicing sequence.

NOTE

No attempt is made in this bus specification to specify what will happen during the interrupt servicing sequence. Servicing of the interrupt may or may not involve use of the VME bus.

INTERRUPT HANDLERS can be identified in a system by the number and range of interrupt request lines that they service. The option notation is IH(a-b), where 'a' is the lowest line serviced and 'b' is the highest. It is a VME bus requirement that AN INTERRUPT HANDLER MAY ONLY SERVICE A CONTIGUOUS SEQUENCE OF INTERRUPT LEVELS.

Figure 4-4 shows an option IH(1-7) INTERRUPT HANDLER and the signal lines which it uses to interact with INTERRUPTERS on the VME bus.

When the INTERRUPT HANDLER receives one or more interrupt requests from the INTERRUPTERS on the bus, it causes its on-board DTB REQUESTER to gain control of the data transfer bus. It then uses the data transfer bus to acknowledge the highest priority INTERRUPTER and read that INTERRUPTER'S status/ID byte.

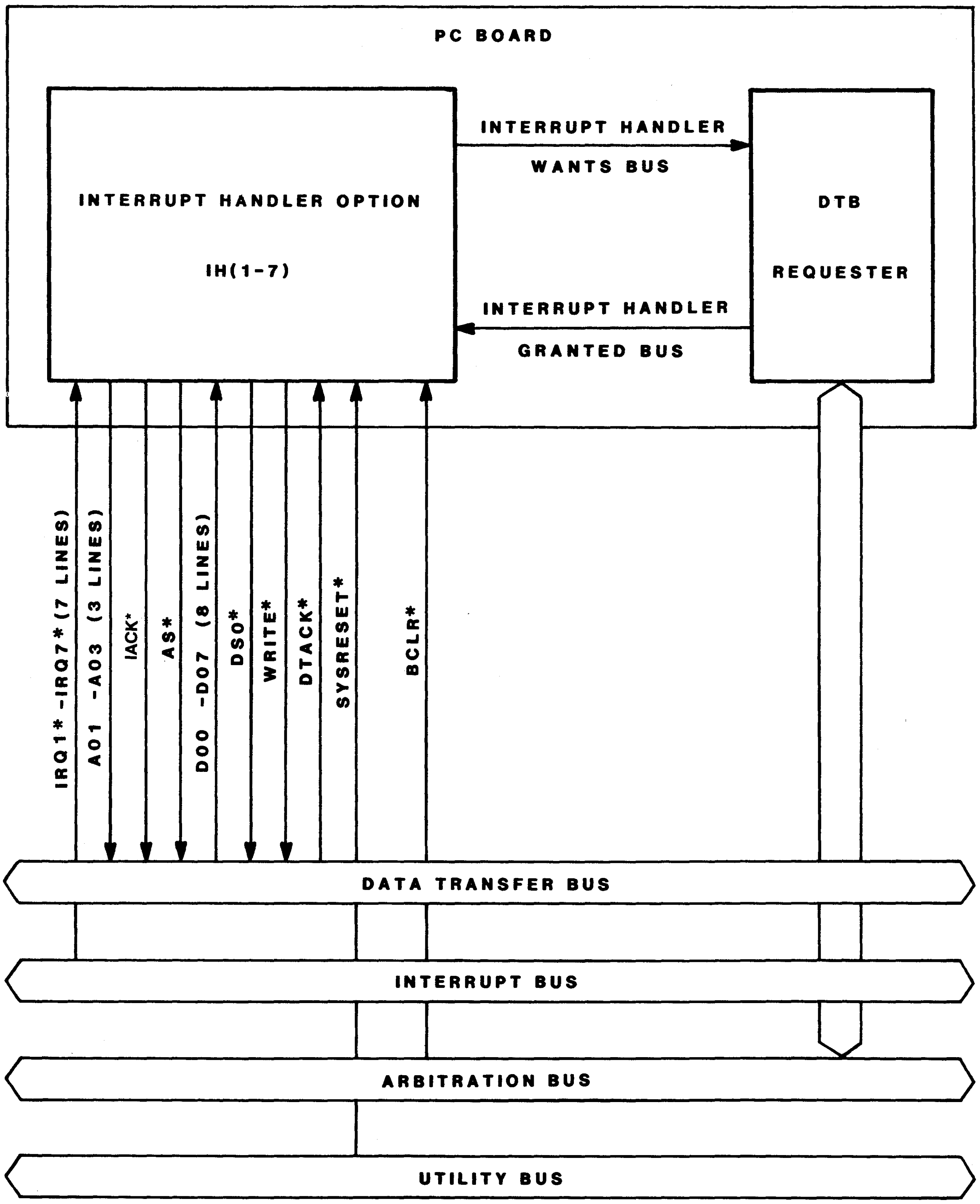


FIGURE 4-4. Signal Lines Used by an IH(1-7) INTERRUPT HANDLER

4.3.2 INTERRUPTER

The INTERRUPTER is used to accomplish three tasks:

a. It requests an interrupt from the INTERRUPT HANDLER which monitors its interrupt request line.

b. It supplies a status/ID byte to the INTERRUPT HANDLER when its interrupt request is acknowledged.

c. It passes through an interrupt acknowledge daisy-chain signal if it is not requesting that level of interrupt.

INTERRUPTERS can be identified in a system by the interrupt request line they utilize. The option notation is I(n), where 'n' is the interrupt request line number. Since the 'INTERRUPTER module' is a concept and not a design constraint, it is possible to visualize a complex logic unit which interrupts on several interrupt lines as a set of 'INTERRUPTER modules'.

Figure 4-5 shows an option I(4) INTERRUPTER and the signal lines which it uses to interact with its INTERRUPT HANDLER on the VME bus.

The INTERRUPTER uses an IRQx* line (in this case, IRQ4*) to request an interrupt. It then monitors the DTB address bus, IACK*, and the IACKIN*/ IACKOUT* daisy-chain to determine when its interrupt is being acknowledged. When acknowledged, it places its status/ID byte on the lower eight lines of the data bus and signals the byte's validity to the INTERRUPT HANDLER via the DTACK* line.

4.3.3 Comparison of Interrupt Bus Functional Modules to DTB Functional Modules

The INTERRUPT HANDLER uses the DTB to read a status/ID byte from the INTERRUPTER. In this respect, the INTERRUPT HANDLER acts like a MASTER and the INTERRUPTER acts like a SLAVE. However, the following differences are important to note.

4.3.3.1 INTERRUPT HANDLER vs MASTER: Differences

There are four primary differences in the use of the DTB by the INTERRUPT HANDLER and the MASTER:

a. The state of the address modifier bus is unspecified.

b. Number of lines driven on the address bus.

c. Use of lines on the data bus.

d. The state of IACK*.

The INTERRUPT HANDLER is required to drive only the lowest three address lines (A01-A03) and is not required to drive the address modifier lines. The levels of these three address lines indicate which of the seven interrupt request lines is being acknowledged. A MASTER is required to drive 15, 23, or 31 address lines (depending upon which address modifier code it has placed on the bus) with the address of the SLAVE being accessed.

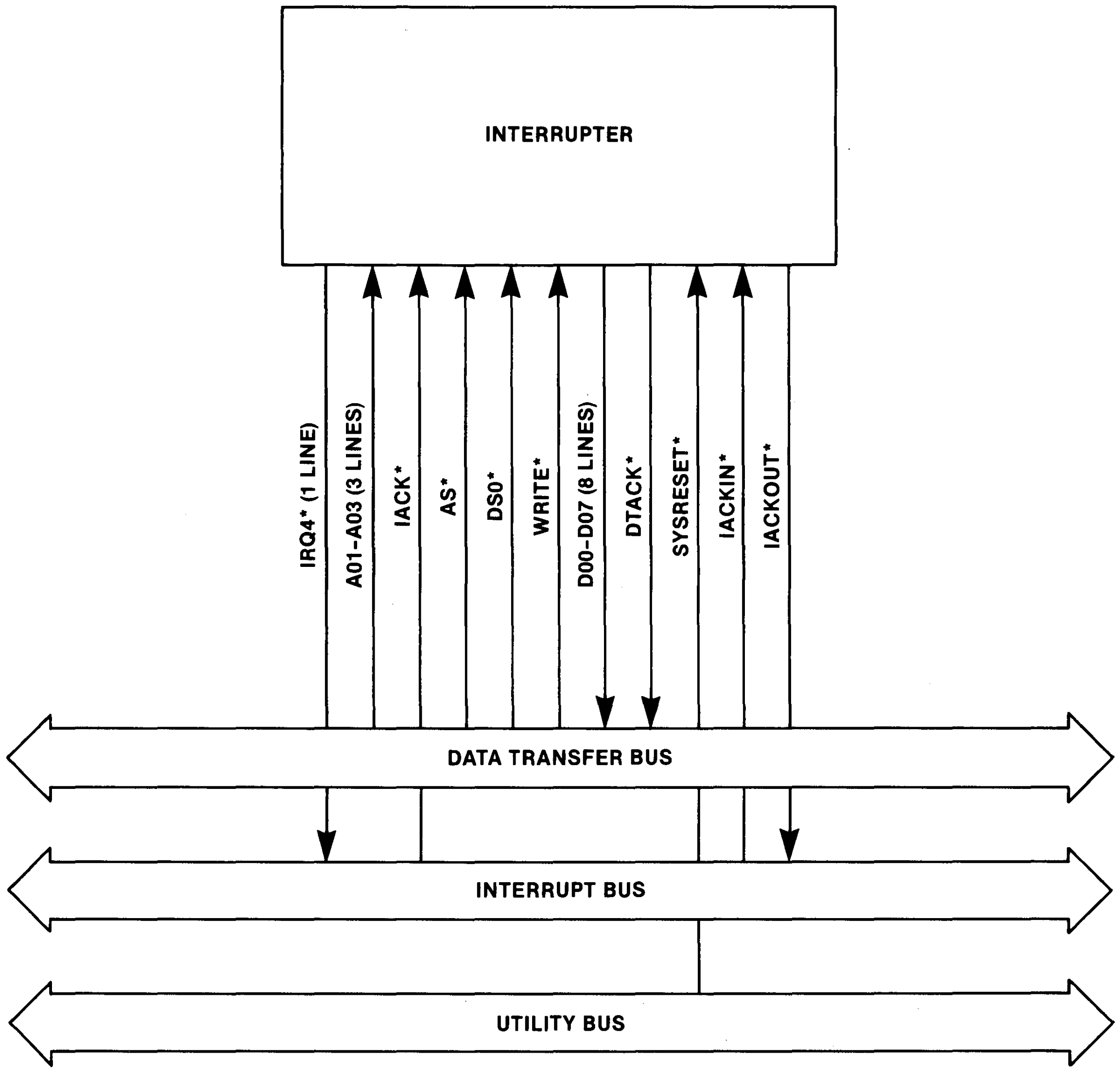


FIGURE 4-5. Signal Lines Used by an I(4) INTERRUPTER

The INTERRUPT HANDLER uses the lower eight data lines (D00-D07) to read the status/ID byte. It is never permitted to drive these lines (i.e., it is not allowed to "write" to the INTERRUPTER) and it must, therefore, always drive WRITE* high when using the DTB. A MASTER uses the data lines to a SLAVE bidirectionally and, during normal use, will drive WRITE* low or high as required. Likewise, the INTERRUPT HANDLER must always drive DS0* to low and DS1* to high. The MASTER is only constrained to drive at least one of the data strobes to low.

The INTERRUPT HANDLER is required to drive IACK* low when it accesses the bus. The MASTER may either drive it high or not drive it at all.

4.3.3.2 INTERRUPTER vs SLAVE: Differences

There are four primary differences in the use of the DTB by the INTERRUPTER and the SLAVE:

a. The monitoring of the address modifier bus.

b. Interpreting the address bus.

c. Using the data bus.

d. The monitoring of IACK*.

The INTERRUPTER ignores the address modifier bus, but uses IACK* as an interrupt acknowledge. A SLAVE never responds with a data transfer when IACK* is low.

The SLAVE decodes the appropriate number of address lines (15, 23, or 31), and determines from the levels of these lines, the current address modifier code, and the state of the IACK* line if it will respond. The INTERRUPTER, however, decodes only the lowest three address lines (A01-A03), and must also check the following conditions before responding.

a. It must have an interrupt request pending.

b. The level of that request must match the level indicated on these lower three address lines.

c. It must receive an incoming low level on its IACKIN* daisy-chain line.

If any of these three conditions is not met, it does not respond to the acknowledge. Instead, the INTERRUPTER passes the low level of IACKIN* to the next module in the daisy-chain via IACKOUT*.

The INTERRUPTER is required to drive only the lowest eight data lines and, therefore, it is not required to monitor DS1*. It is also not required to monitor WRITE*, since it is never written to.

Every SLAVE, however, must monitor the level of WRITE* to prevent it from driving the data bus when it is written to. It must also monitor both DS0* and DS1*.

The INTERRUPTER responds only when IACK* is low. SLAVES respond only when IACK* is high.

4.4 TYPICAL OPERATION

A typical interrupt sequence may be divided into three phases:

a. The interrupt request phase.

b. The interrupt acknowledge phase.

c. The interrupt servicing phase.

Figure 4-6 illustrates the timing relationships between the three phases.

The interrupt request phase (phase 1) is the time between when an INTERRUPTER drives an interrupt request line low and when the INTERRUPT HANDLER gains control of the DTB. The interrupt acknowledge phase (phase 2) is the time during which the INTERRUPT HANDLER uses the DTB to read the REQUESTER'S status/ID byte. The interrupt servicing phase (phase 3) is the time period required to execute a prescribed interrupt servicing routine. This may or may not involve data transfers on the VME bus.

The protocol for the priority interrupt subsystem describes the module interaction required during phase 1 and phase 2. Phase 3 may not require any module interaction. Any data transfers which take place during phase 3 will follow the data transfer bus and arbitration bus protocols described in Chapters 2 and 3. Therefore, phase 3 is not discussed in this chapter.

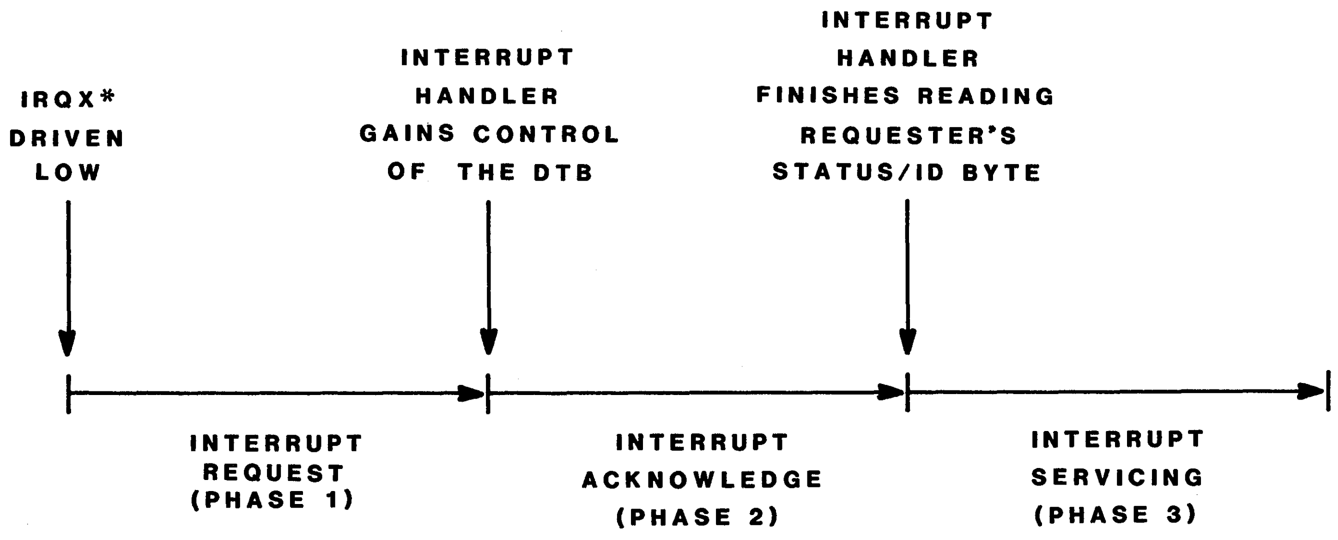

IRQX* DRIVEN LOW
INTERRUPT HANDLER GAINS CONTROL OF THE DTB
INTERRUPT HANDLER FINISHES READING REQUESTER'S STATUS/ID BYTE
INTERRUPT REQUEST (PHASE 1)
INTERRUPT ACKNOWLEDGE (PHASE 2)
INTERRUPT SERVICING (PHASE 3)

FIGURE 4-6. 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. In this case, the interrupt request lines are prioritized (IRQ7* = highest priority), and when simultaneous requests are detected on two interrupt request lines, the status/ID byte of the higher priority request is read first.

4.4.2 Distributed Interrupt Operation

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

1. distributed interrupt systems with seven INTERRUPT HANDLERS,
2. distributed interrupt systems with two to six INTERRUPT HANDLERS.

4.4.2.1 Distributed Interrupt Systems with Seven INTERRUPT HANDLERS

See Figure 4-7. In a bussed system, each of the interrupt request lines may be monitored by a separate INTERRUPT HANDLER. Each INTERRUPT HANDLER must gain control of the data transfer bus before it can read the status/ID byte from an INTERRUPTER driving its own interrupt request line. When two interrupt request lines on the bus are driven low simultaneously, bus arbitration may result in a sequence of service different from that provided in a single handler system.

Figure 4-8 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*. If two INTERRUPTERS on the interrupt bus simultaneously drive IRQ2* and IRQ5* low, the two INTERRUPT HANDLERS might cause their on-board REQUESTERS to drive BR2* and BR3* low simultaneously. (This is by no means certain, since either INTERRUPT HANDLER may wait a considerable period of time before attempting to handle the interrupt.) If priority arbitration is used and if both bus requests are low at the moment of arbitration, the ARBITER will first grant control of the DTB to the INTERRUPT HANDLER B REQUESTER, and INTERRUPT HANDLER A must wait until B has finished using the DTB. If round-robin arbitration is used, either might be granted the bus first.

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 4-9 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 would be prioritized (IRQ4* = highest priority for INTERRUPT HANDLER A), and the IRQ5*-IRQ7* lines would be prioritized (IRQ7* = highest priority for INTERRUPT HANDLER B). The DTB arbitration would still determine which INTERRUPT HANDLER would be allowed to use the DTB first.

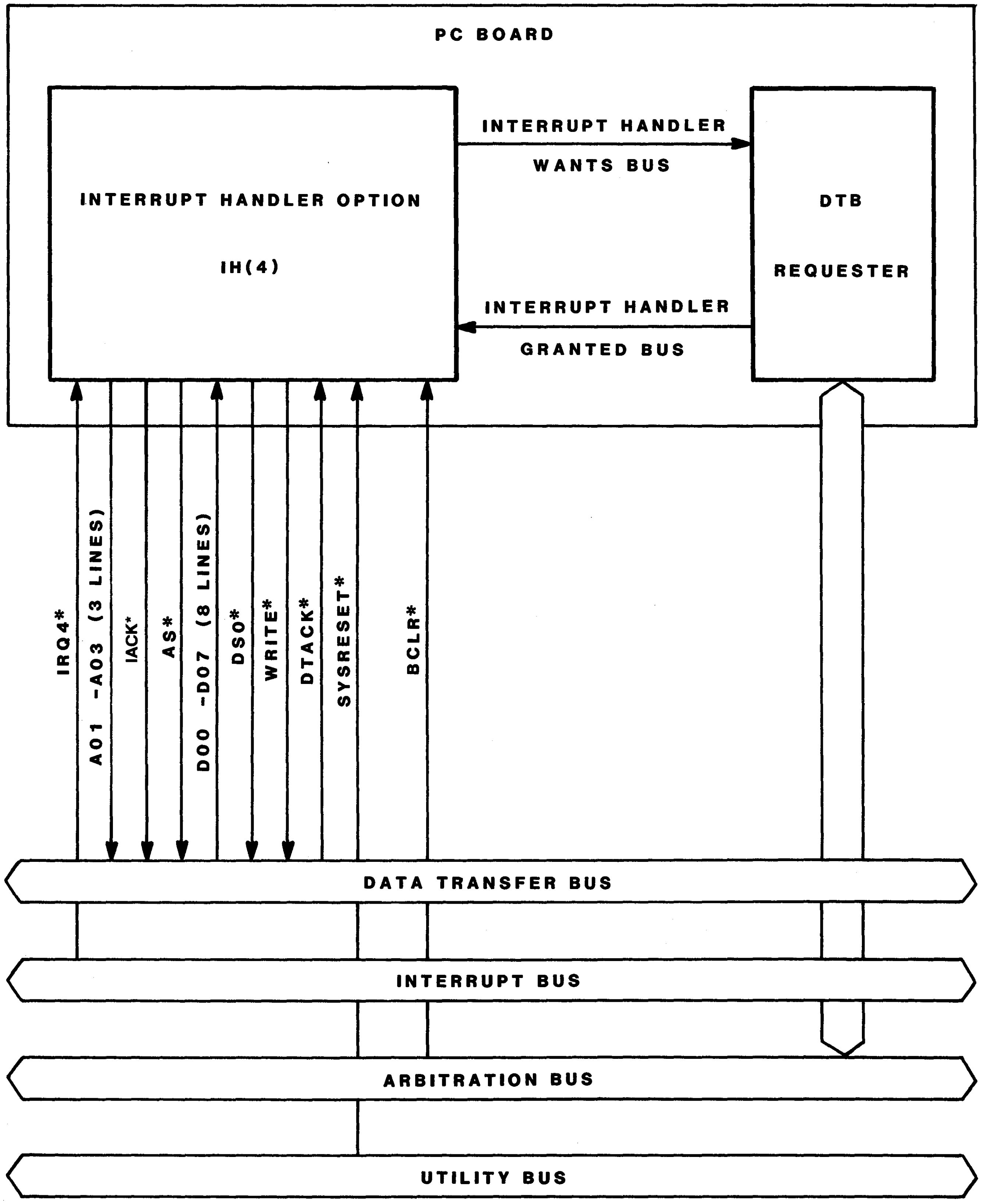


FIGURE 4-7. INTERRUPT HANDLER Monitoring Only IRQ4*

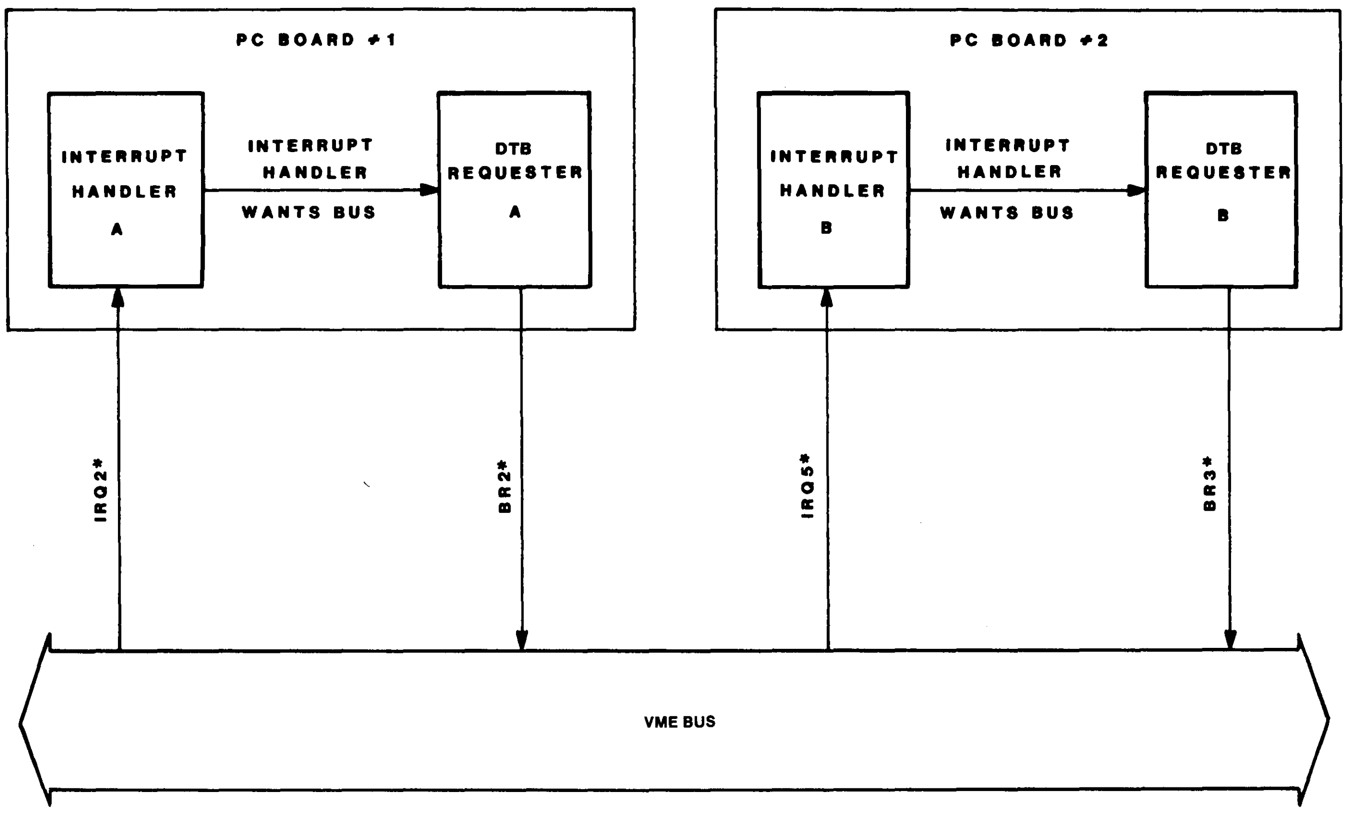


FIGURE 4-8. Two INTERRUPT HANDLERS, Each Monitoring One Interrupt Request Line

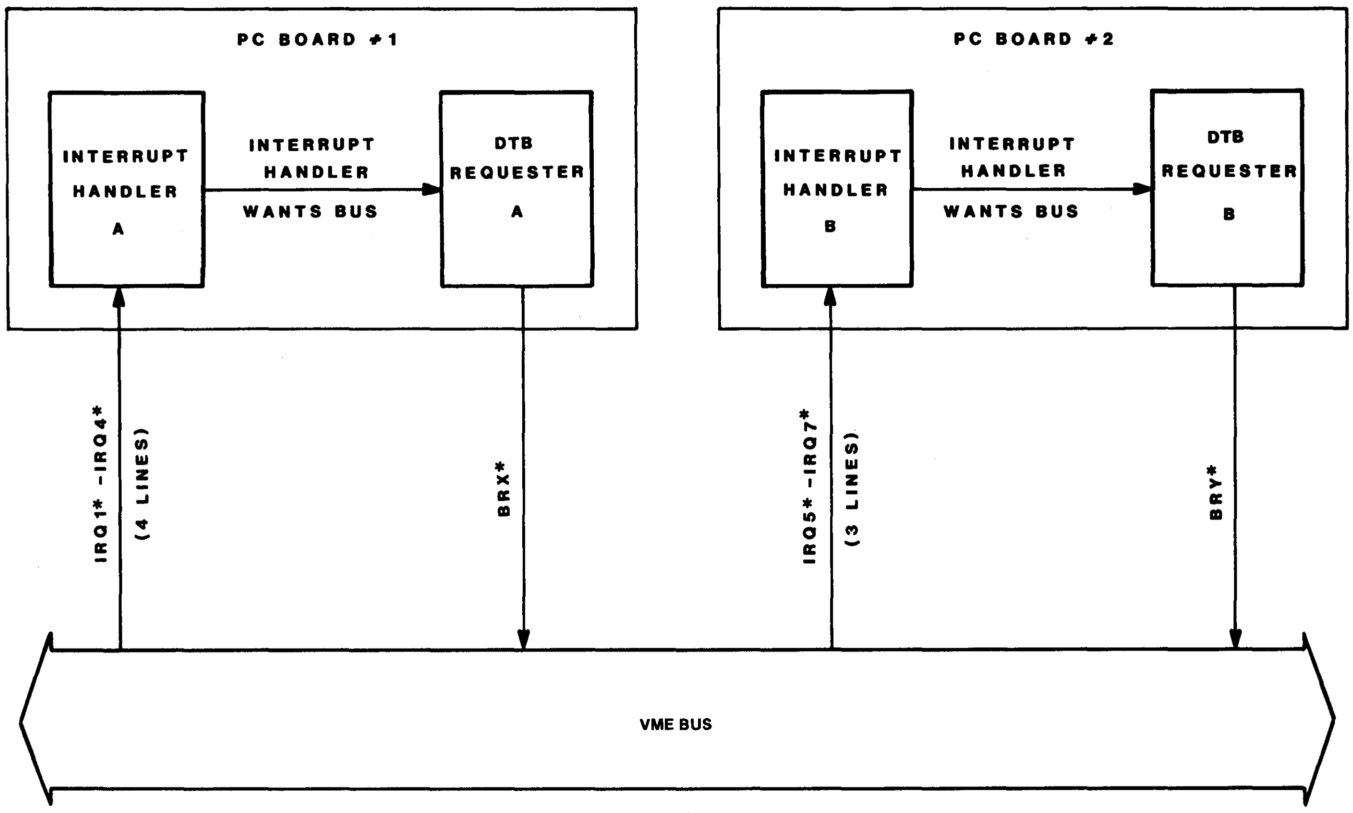


FIGURE 4-9. Two INTERRUPT HANDLERS, Each Monitoring Several Interrupt Request Lines

4.4.3 Example: Typical Single Handler Interrupt System Operation

Figure 4-10 illustrates the operation of a single handler interrupt system whose INTERRUPT HANDLER monitors and prioritizes all seven interrupt lines (IRQ7 = highest priority). At the top of the diagram, a DTB MASTER is using the DTB to move data within the system. An INTERRUPTER requests an interrupt by driving IRQ4* low. When the INTERRUPT HANDLER detects the low IRQ4*, it sends a signal to its on-board DTB 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 an option PRI ARBITER.) When the DTB MASTER detects the low BCLR*, it stops moving data and allows its own on-board REQUESTER to relinquish control of the DTB. (The REQUESTER does this by releasing BBSY*.)

NOTE

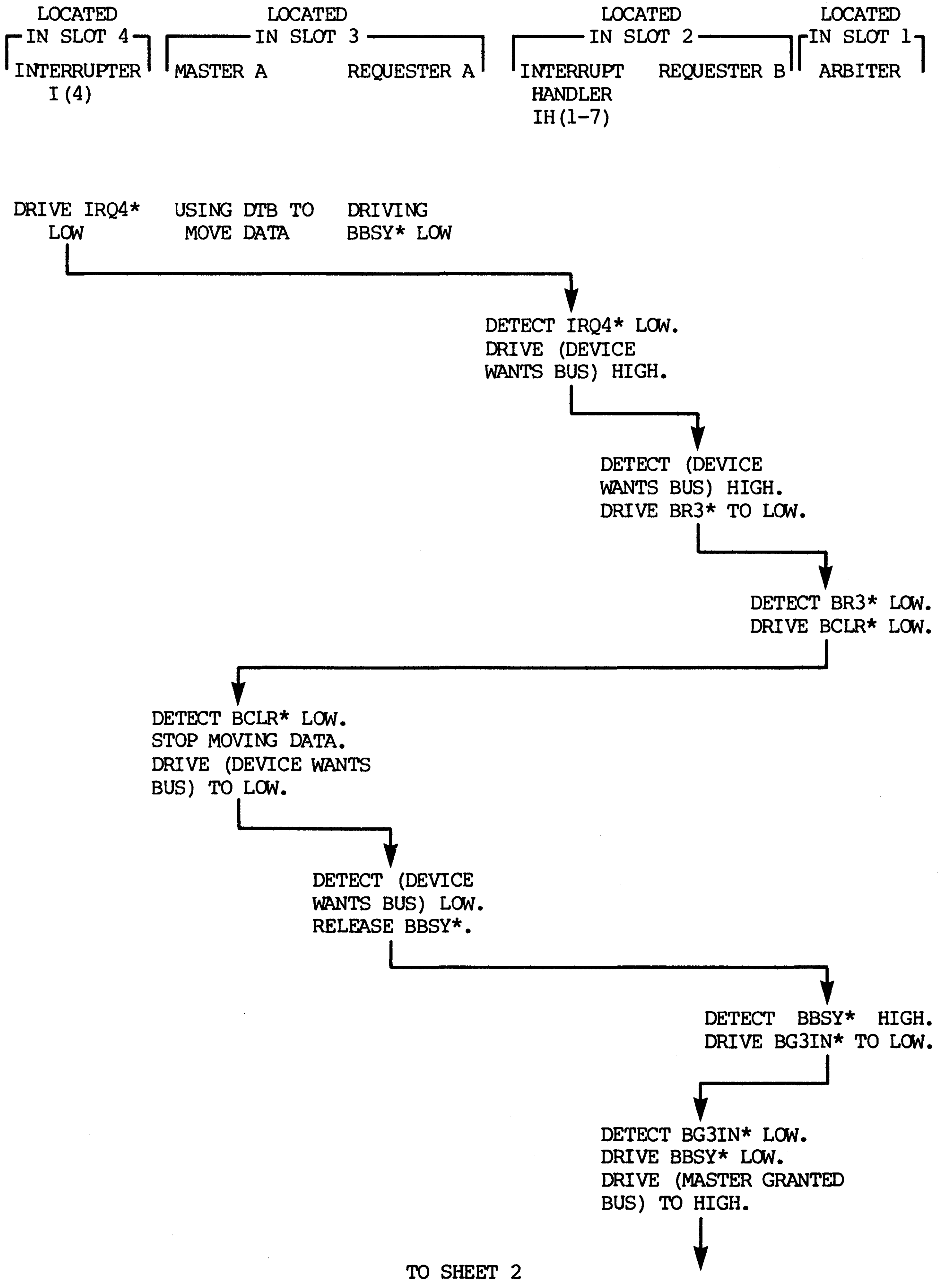
> The active DTB MASTER is not required to relinquish the DTB within a specified time period, but a prompt response to the BCLR* line allows a system to run more efficiently. Since an RRS ARBITER does not generate BCLR*, systems using an RRS ARBITER may have a longer interrupt response time.

When the DTB ARBITER detects BBSY* high, it grants the DTB to the INTERRUPT HANDLER'S on-board REQUESTER, which informs the INTERRUPT HANDLER that the DTB is available. 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 4-1). 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 to the IACKIN* pin of slot A1 and causes a low level to be propagated down the acknowledge daisy-chain on the bus (IACKIN*/IACKOUT*).

When the INTERRUPTER detects a low level on its incoming daisy-chain line, it ensures that it has received address strobe, and then 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, the INTERRUPTER places its 8-bit status/ID byte on the data bus and drives the DTACK* line low. When the INTERRUPT HANDLER detects the low DTACK*, it reads the status byte and initiates the appropriate interrupt service sequence.

NOTE

> No attempt is made in this VME bus specification to define what must take place during an interrupt service sequence. This may or may not involve use of the data transfer bus.


LOCATED IN SLOT 4
INTERRUPTER I(4)
LOCATED IN SLOT 3
MASTER A
REQUESTER A
LOCATED IN SLOT 2
INTERRUPT HANDLER IH(1-7)
REQUESTER B
LOCATED IN SLOT 1
ARBITER
DRIVE IRQ4* LOW
USING DTB TO MOVE DATA
DRIVING BBSY* LOW
DETECT IRQ4* LOW. DRIVE (DEVICE WANTS BUS) HIGH.
DETECT (DEVICE WANTS BUS) HIGH. DRIVE BR3* TO LOW.
DETECT BR3* LOW. DRIVE BCLR* LOW.
DETECT BCLR* LOW. STOP MOVING DATA. DRIVE (DEVICE WANTS BUS) TO LOW.
DETECT (DEVICE WANTS BUS) LOW. RELEASE BBSY*.
DETECT BBSY* HIGH. DRIVE BG3IN* TO LOW.
DETECT BG3IN* LOW. DRIVE BBSY* LOW. DRIVE (MASTER GRANTED BUS) TO HIGH.

TO SHEET 2

FIGURE 4-10. Typical Single Handler Interrupt System Operation Flow Diagram (Sheet 1 of 2)

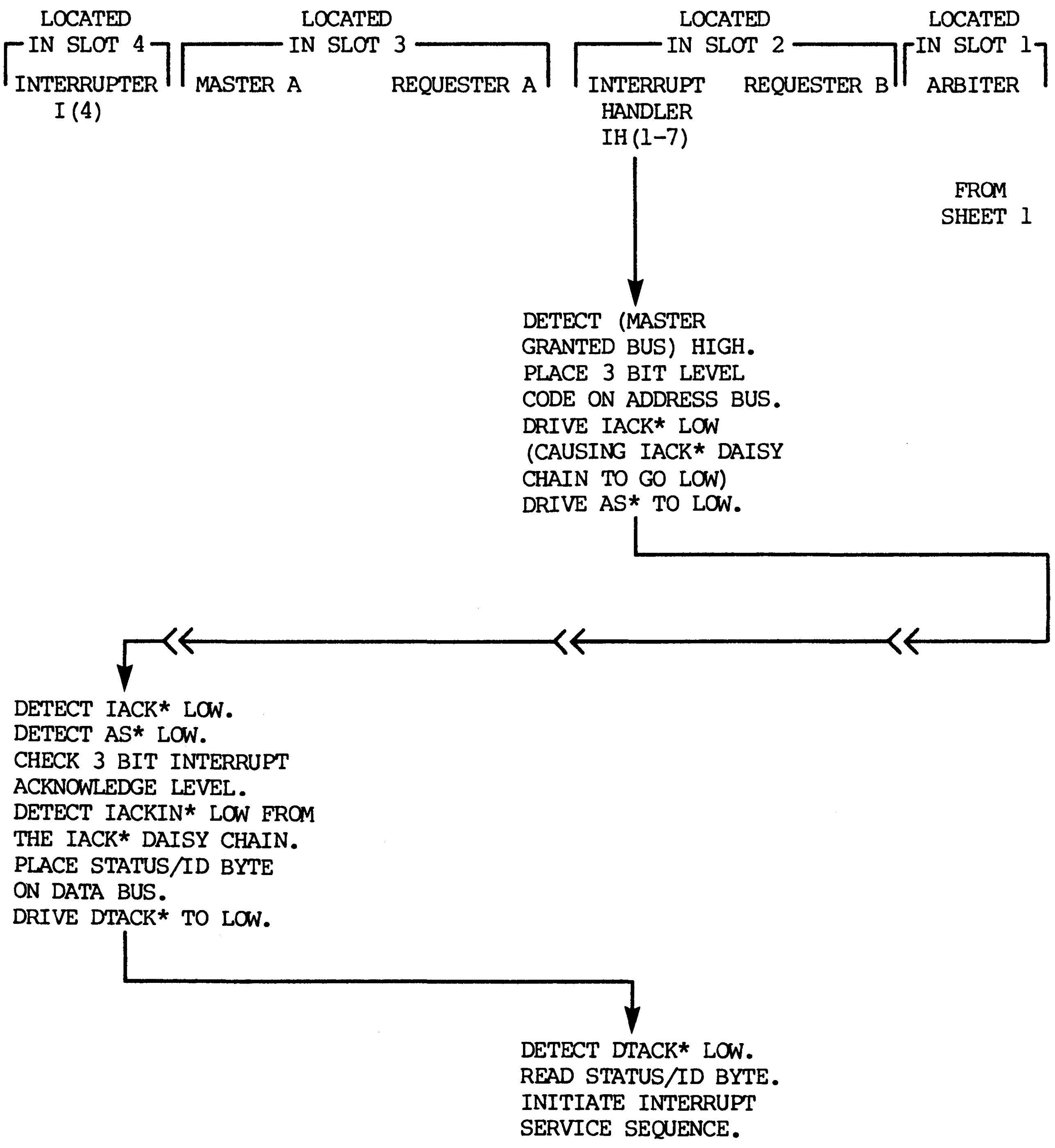

LOCATED IN SLOT 4
INTERRUPTER I(4)
LOCATED IN SLOT 3
MASTER A
REQUESTER A
LOCATED IN SLOT 2
INTERRUPT HANDLER IH(1-7)
REQUESTER B
LOCATED IN SLOT 1
ARBITER
FROM SHEET 1
DETECT (MASTER GRANTED BUS) HIGH. PLACE 3 BIT LEVEL CODE ON ADDRESS BUS. DRIVE IACK* LOW (CAUSING IACK* DAISY CHAIN TO GO LOW) DRIVE AS* TO LOW.
DETECT IACK* LOW. DETECT AS* LOW. CHECK 3 BIT INTERRUPT ACKNOWLEDGE LEVEL. DETECT IACKIN* LOW FROM THE IACK* DAISY CHAIN. PLACE STATUS/ID BYTE ON DATA BUS. DRIVE DTACK* TO LOW.
DETECT DTACK* LOW. READ STATUS/ID BYTE. INITIATE INTERRUPT SERVICE SEQUENCE.

FIGURE 4-10. Typical Single Handler Interrupt System Operation Flow Diagram (Sheet 2 of 2)

TABLE 4-1. 3-Bit Interrupt Acknowledge Code

LOW INTERRUPT LINE BEING ACKNOWLEDGED	ADDRESS BUS INTERRUPT LEVEL CODE A23 - A04	A03	A02	A01
IRQ1*	X - X	L	L	H
IRQ2*	X - X	L	H	L
IRQ3*	X - X	L	H	H
IRQ4*	X - X	H	L	L
IRQ5*	X - X	H	L	H
IRQ6*	X - X	H	H	L
IRQ7*	X - X	H	H	H

4.4.4 Example: Prioritization of Two Interrupts in a Distributed Interrupt System

Figure 4-11 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 DTB REQUESTER that they need the DTB. Both the INTERRUPT HANDLERS drive BR3* low. Upon detecting BR3* low, the DTB ARBITER drives BG3IN* to low on slot 1. This low signal is passed down the BG3IN*/BG3OUT* daisy chain until it is detected by the REQUESTER in slot 4. This REQUESTER then signals its on-board INTERRUPT HANDLER B that the DTB is available for acknowledging the interrupt. INTERRUPT HANDLER B then drives IACK* low to indicate that it is acknowledging an interrupt, and a 3-bit code (see Table 4-1) is placed on the address bus.

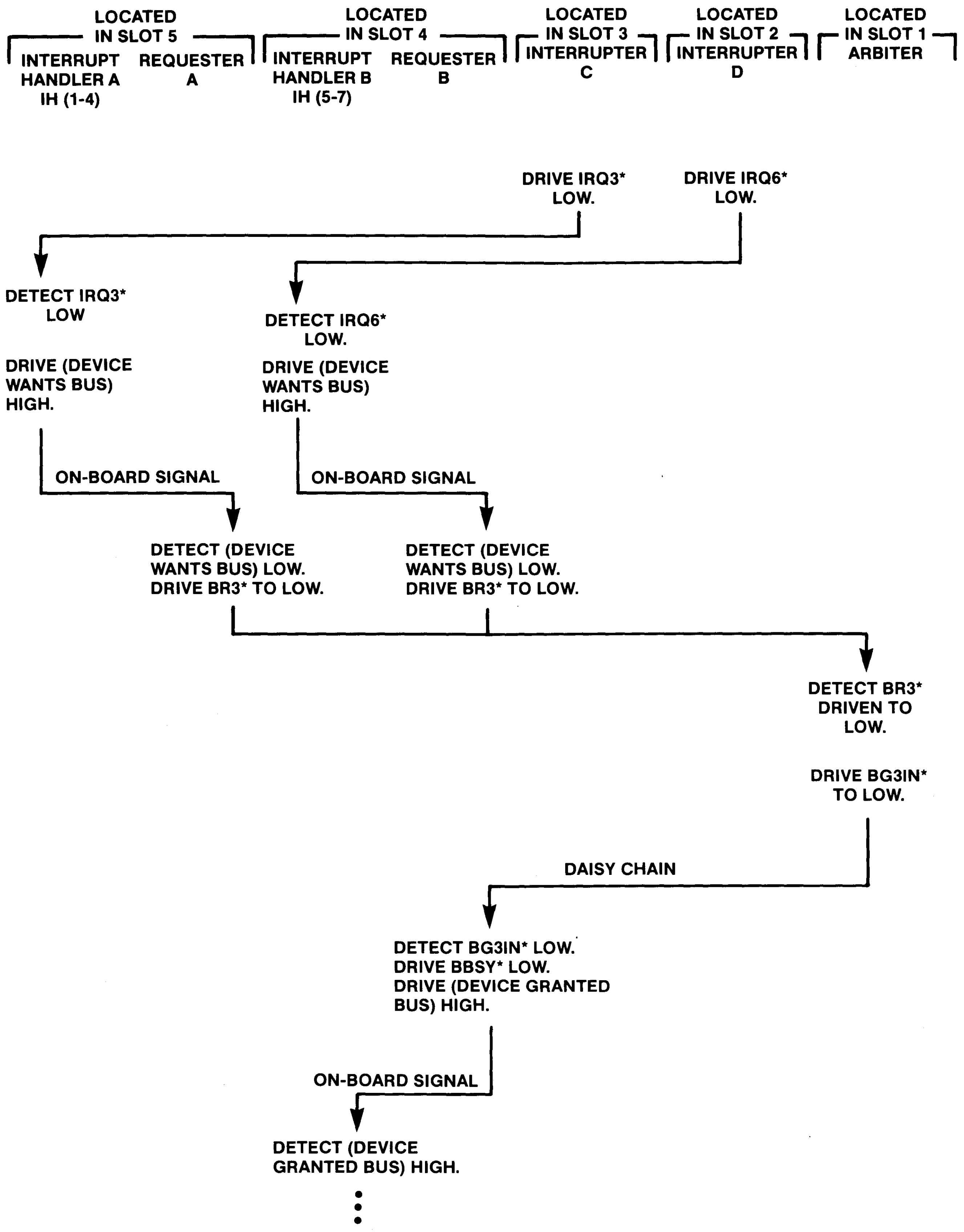

LOCATED IN SLOT 5
INTERRUPT HANDLER A IH (1-4)
REQUESTER A
LOCATED IN SLOT 4
INTERRUPT HANDLER B IH (5-7)
REQUESTER B
LOCATED IN SLOT 3
INTERRUPTER C
LOCATED IN SLOT 2
INTERRUPTER D
LOCATED IN SLOT 1
ARBITER
DRIVE IRQ3* LOW.
DRIVE IRQ6* LOW.
DETECT IRQ3* LOW
DRIVE (DEVICE WANTS BUS) HIGH.
DETECT IRQ6* LOW.
DRIVE (DEVICE WANTS BUS) HIGH.
ON-BOARD SIGNAL
ON-BOARD SIGNAL
DETECT (DEVICE WANTS BUS) LOW. DRIVE BR3* TO LOW.
DETECT (DEVICE WANTS BUS) LOW. DRIVE BR3* TO LOW.
DETECT BR3* DRIVEN TO LOW.
DRIVE BG3IN* TO LOW.
DAISY CHAIN
DETECT BG3IN* LOW. DRIVE BBSY* LOW. DRIVE (DEVICE GRANTED BUS) HIGH.
ON-BOARD SIGNAL
DETECT (DEVICE GRANTED BUS) HIGH.

FIGURE 4-11. Distributed Interrupt System with Two INTERRUPT HANDLERS

4.5 DETAILED TIMING/STATE DIAGRAMS/ADDITIONAL NOTES

This section describes the timing relationships of signals used for interrupt processing. Two separate sets of timing are provided: one for the INTERRUPT HANDLER and DTB driver and one for the INTERRUPTER. These two sets of timing take into account bus skew time, and provide the designer an exact definition of timing constraints and guarantees. Because the backplane is a passive device, capacitive loading of signal lines causes a degradation of rise and fall times. This may alter the skew between signals as they propagate down the bus. The worst case skew between two lines on the bus will occur when one line has minimum loading and the other line has maximum loading. (Bus load specifications are discussed in Chapter 7.)

NOTE

In the following discussions, several on-board signals are defined to allow discussion of the interaction between the INTERRUPTER/INTERRUPT HANDLER module and other on-board logic. These signals are not intended to place restrictions on the board designer, but do illustrate the information which must be passed to and from the modules.

Figure 4-12 shows the block diagram of an INTERRUPT HANDLER, while Figure 4-13 shows an INTERRUPTER.

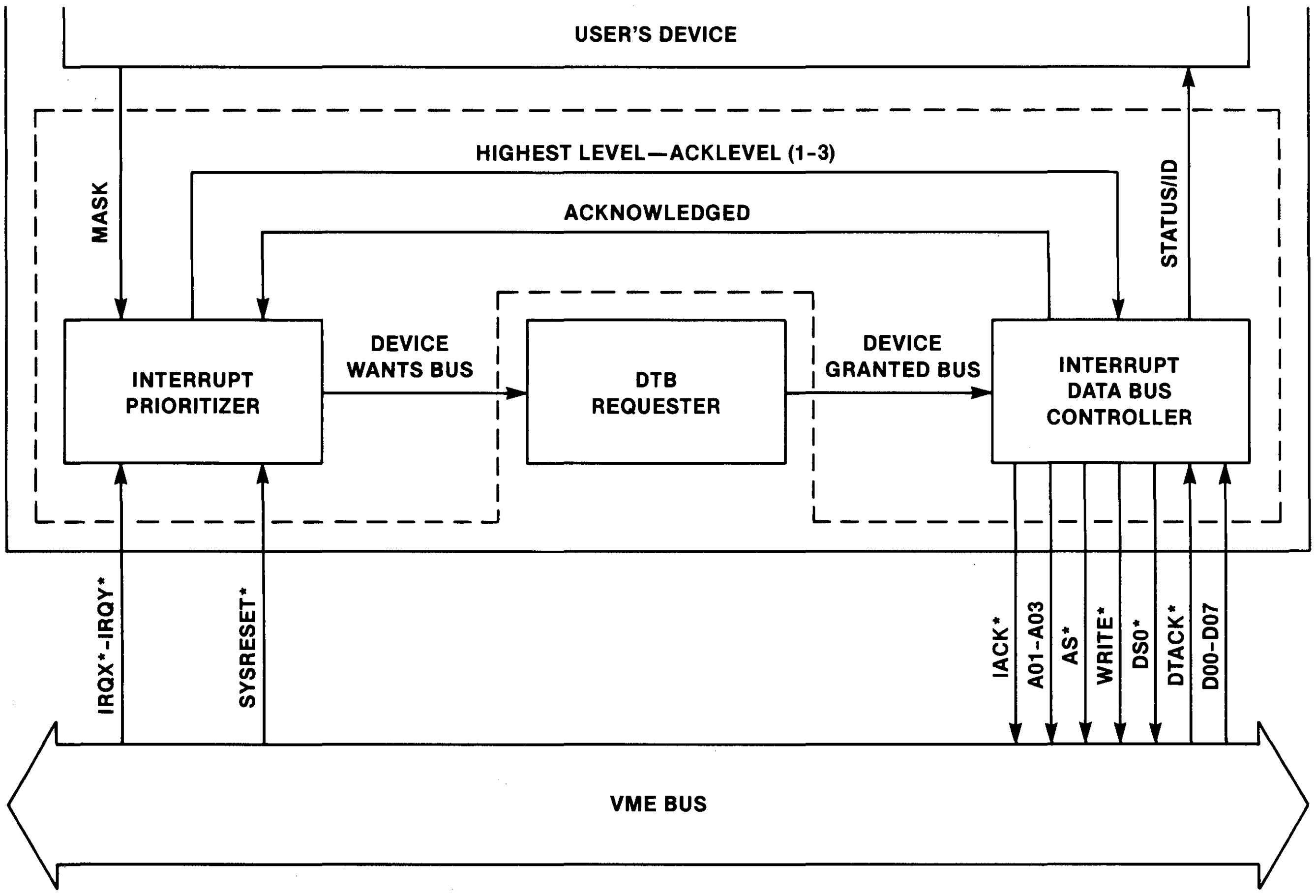


FIGURE 4-12. Block Diagram: INTERRUPT HANDLER

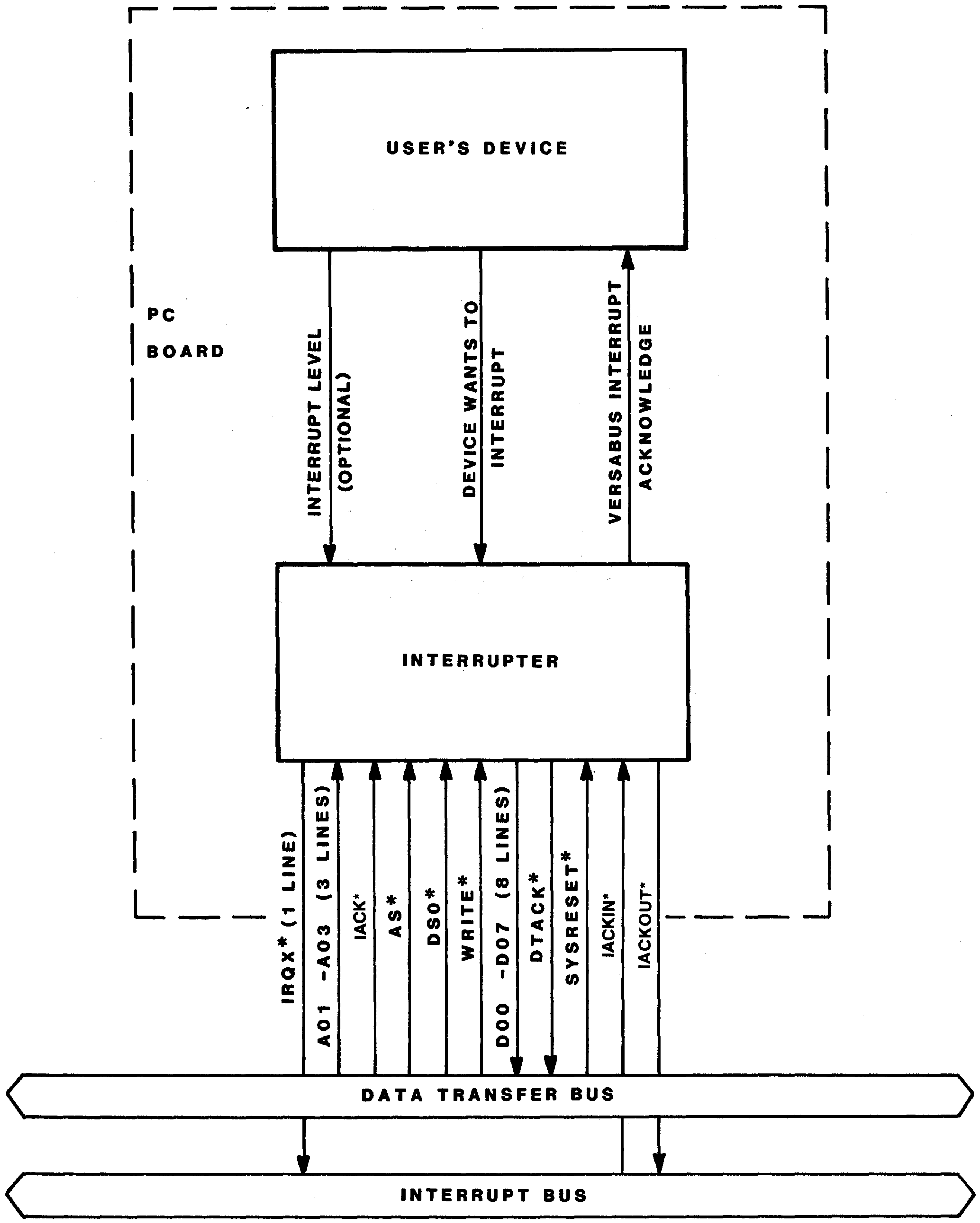


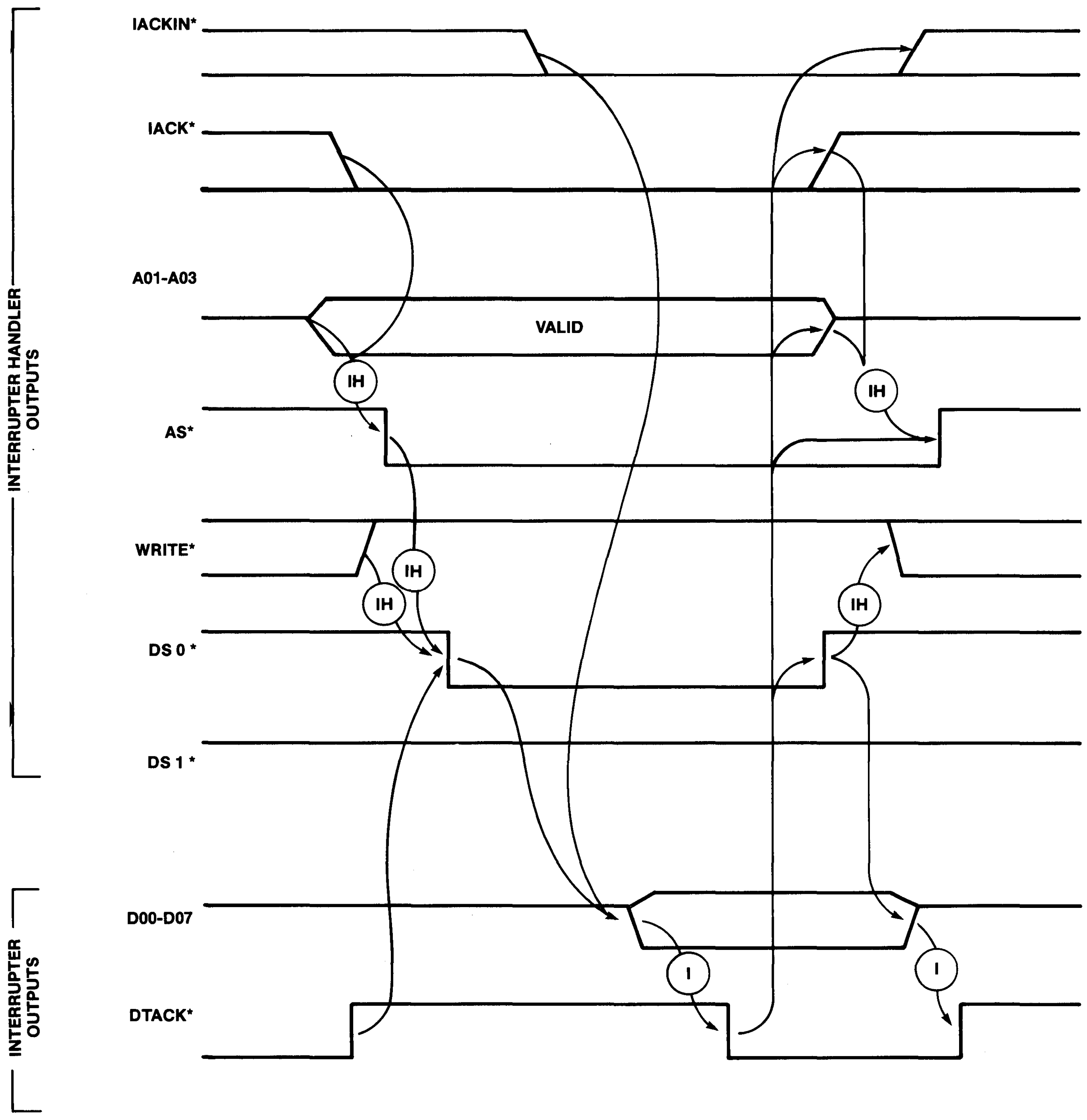
FIGURE 4-13. Block Diagram: INTERRUPTER

Since an INTERRUPT HANDLER must make use of the DTB to acknowledge each interrupt, it always has an on-board DTB REQUESTER. Although the REQUESTER is not part of the INTERRUPT HANDLER, it is shown on the block diagram to illustrate how it is used by the INTERRUPT HANDLER to gain control of the DTB. The seven-level handler must determine the highest priority interrupt on the VERSAbus, while the single-level handler treats its only interrupt level as the highest level. In some cases, the handler may compare the highest level interrupt to a mask level. Only if the interrupt level exceeds the mask level will it indicate a need for the bus to the DTB REQUESTER. The prioritizer, when it requests the bus, must also provide signal levels to the on-board DTB bus driver to indicate what 3-bit code should be placed on the address lines during the interrupt acknowledge. When the DTB bus driver receives a DEVICE GRANTED BUS signal from the DTB REQUESTER indicating that the bus is available, it drives IACK* low, places the interrupt acknowledge level on the lower three address lines, and drives address strobe (AS*) to low.

The DTB bus driver then drives WRITE* high to indicate that a READ will be done, and drives DS0* low, allowing the status/ID byte to be placed on the data bus by the INTERRUPTER being acknowledged.

When the INTERRUPTER has placed its status/ID byte on the bus, it drives DTACK* low. The INTERRUPT HANDLER'S DTB bus driver reads the byte and releases WRITE*, DS0*, the address lines, and IACK*, but does not release AS* until it receives the signal from the data bus controller indicating that all other bus lines have been released. The rising edge of AS* is interpreted by the next MASTER granted the bus as an indication that the INTERRUPT HANDLER has stopped driving the DTB.) (See Figure 4-14.)

The following tables and diagrams give the timing constraints for an INTERRUPT HANDLER (Table 4-2 and Figure 4-15) and for an INTERRUPTER (Table 4-3 and Figure 4-16).



NOTE

Arrows labeled IH show timing relationships guaranteed by the internal timing of the INTERRUPT HANDLER.

Arrows labeled I show timing relationships guaranteed by the internal timing of the INTERRUPTER.

Unlabeled arrows show timing guaranteed by interlocked relationships between the INTERRUPTER and the INTERRUPT HANDLER.

FIGURE 4-14. Interrupt Acknowledge Cycle

DESCRIPTION OF PARAMETERS

1 This time provides the INTERRUPTER with address setup time. The address lines must be stable and valid for the minimum setup time before AS* may be driven across the high level threshold voltage.

2 The lowest three address lines (A01-A03) must be held stable until DTACK* is received driven to low, and then may change. Likewise, IACK* must be held low until DTACK* is received driven to low. The address bus need not be released between consecutive cycles of the same INTERRUPT HANDLER, but may be released, if desired.

3 AS* must be driven high for the minimum time to ensure that the INTERRUPTER detects the end of the bus cycle.

4 AS* must remain low until DTACK* is received driven to low. It is then driven to high.

5 This time sets a minimum time between the driving of AS* low and DS0* low.

6 The WRITE* line must be high for the minimum setup time before DS0* is driven below the high level threshold voltage.

7 The WRITE* line must remain high until the INTERRUPT HANDLER drives DS0* to high.

8 This time guarantees that the data bus is released by the INTERRUPTER before the received DTACK* crosses the low level threshold voltage on the low to high transition.

9 This read data setup time guarantees the INTERRUPT HANDLER that the data bus is valid and stable within the specified time after the received DTACK* crosses the high level threshold voltage on the high to low transition.

10 Once driven low, a data strobe must be held low until the INTERRUPT HANDLER receives DTACK* driven to low.

11 These times require that both data strobes be concurrently driven high for the minimum time.

12 This time guarantees that the data on the data bus lines will remain valid until the INTERRUPT HANDLER drives the first data strobe across the low level threshold voltage.

13 This time requires that the INTERRUPT HANDLER must receive DTACK* high before either data strobe is driven across the high level threshold voltage.

TABLE 4-2. INTERRUPT HANDLER DTB Bus Driver Timing

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx valid and IACK* low to AS* low	20		B
2	DTACK* low to invalid address or IACK* high	0		C
3	AS* High	40		B
4	DTACK* low to AS* high	0		C
5	AS* to DS0* skew	0		B
6	WRITE* valid to DS0* low	20		B
7	DS0* high to invalid WRITE*	10		B
8	Data released to DTACK* high	0		E
9	Dxx valid to DTACK* low	-10		E
10	DTACK* low to DS0* high	0		C
11	DS0* high	40		B
12	DS0* high to invalid data	0		D
13	DTACK* high to DS0* low	0		C

NOTES:

A. All times given are in nanoseconds.

B. The INTERRUPT HANDLER must guarantee this timing between two of its outgoing signal transitions.

C. The INTERRUPT HANDLER must wait for the incoming signal edge from the INTERRUPTER before changing the level of its outgoing signal.

D. This is a guarantee that the INTERRUPTER will not change the incoming signal until the INTERRUPT HANDLER changes its outgoing signal.

E. The INTERRUPT HANDLER is guaranteed this timing between two of its incoming signal transitions.

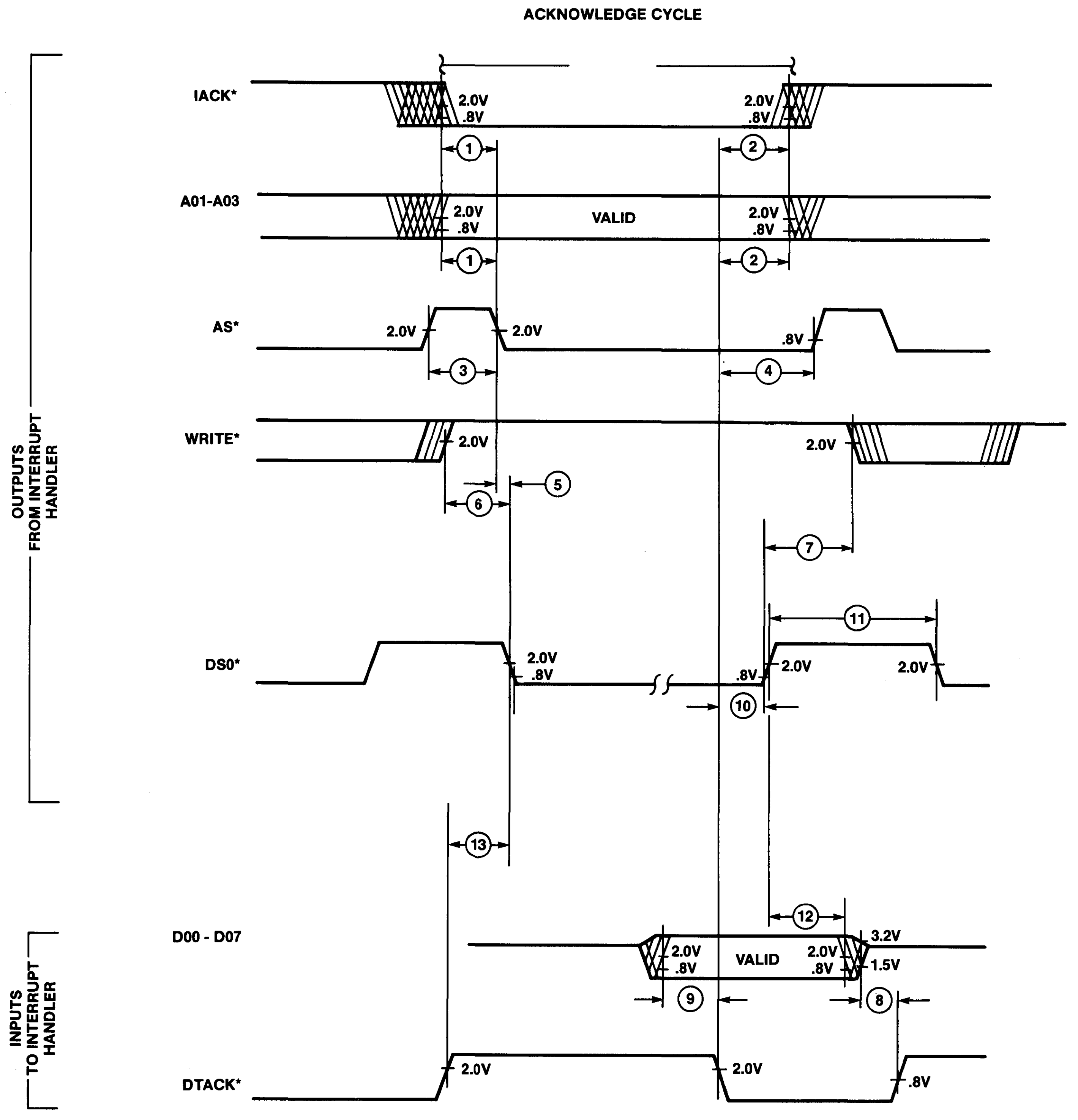


FIGURE 4-15. INTERRUPT HANDLER Timing

DESCRIPTION OF PARAMETERS

1 This time guarantees the INTERRUPTER a minimum address and IACK* setup time. The address lines are stable and valid and IACK* is low for the minimum setup time before AS* is received driven across the high level threshold voltage.

2 The address is guaranteed to remain stable until the INTERRUPTER drives DTACK* to low. The address lines and IACK* may then change.

3 AS* is driven high for this guaranteed minimum time to ensure that the INTERRUPTER detects the end of the bus cycle.

4 AS* is guaranteed to remain low until the INTERRUPTER drives DTACK* to low.

5 This read data time guarantees the INTERRUPT HANDLER that the data bus is valid and stable within the specified time after the received DTACK* crosses the high level threshold on the high to low transition.

6 The WRITE* line is guaranteed high for the minimum setup time before DS0* is received driven across the high level threshold voltage.

7 The WRITE* line is guaranteed to remain high until DS0* is received driven to high.

8 The INTERRUPTER must not drive the data bus until it receives IACKIN* low.

9 Once driven low, a data strobe is guaranteed to remain low until the INTERRUPTER drives DTACK* to low.

10 This time guarantees that the data strobe will be driven high for the minimum time.

11 The INTERRUPTER must release the output data bus drivers before it releases DTACK* across the low level threshold voltage to high.

12 The INTERRUPTER must hold the data valid and stable until it receives DS0* driven to high.

13 The INTERRUPTER must not drive DTACK* low until DS0* is received driven to low.

14 This time guarantees that a new data strobe will not be received driven through the high level threshold voltage until the INTERRUPTER releases DTACK* to high.

15 The INTERRUPTER must not drive the data bus until the first data strobe is driven low.

16 The INTERRUPTER must not drive DTACK* across the high level threshold voltage until it has placed the status/ID byte on the data bus.

TABLE 4-3. INTERRUPTER Timing

		(NOTE A)		
NUMBER	PARAMETER	MIN.	MAX.	NOTES
1	Axx valid and IACK* low to AS* low	10		E
2	DTACK* low to invalid address or IACK* high	0		D
3	AS* High	30		E
4	DTACK* low to AS* high	0		D
5	AS* to DS0* skew	-10		E
6	WRITE* high to DS0* low	10		E
7	DS0* high to invalid WRITE*	0		E
8	IACKIN* low to active data bus	0		C
9	DTACK* low to DS0* high	0		D
10	DS0* high	30		E
11	Data bus released to DTACK* high	0		B
12	DS0* high to invalid data	0		C
13	DS0* low to DTACK* low	0		C
14	DTACK* high to DS0* low	0		D
15	DS0* low to Active data bus	0		C
16	Data valid to DTACK* low	0		B

NOTES:

A. All times given are in nanoseconds.

B. The INTERRUPTER must guarantee this timing between two of its outgoing signal transitions.

C. The INTERRUPTER must wait for the incoming signal edge from the INTERRUPT HANDLER before changing the level of its outgoing signal.

D. This is a guarantee that the INTERRUPT HANDLER will not change the incoming signal until the INTERRUPTER changes its outgoing signal.

E. The INTERRUPTER is guaranteed this timing between two of its incoming signal transitions.

TABLE 4-3. INTERRUPTER Timing

NUMBER	PARAMETER	(NOTE A) MIN.	MAX.	NOTES
1	Axx valid and IACK* low to AS* low	10		E
2	DTACK* low to invalid address or IACK* high	0		D
3	AS* High	30		E
4	DTACK* low to AS* high	0		D
5	AS* to DS0* skew	-10		E
6	WRITE* high to DS0* low	10		E
7	DS0* high to invalid WRITE*	0		E
8	IACKIN* low to active data bus	0		C
9	DTACK* low to DS0* high	0		D
10	DS0* high	30		E
11	Data bus released to DTACK* high	0		B
12	DS0* high to invalid data	0		C
13	DS0* low to DTACK* low	0		C
14	DTACK* high to DS0* low	0		D
15	DS0* low to Active data bus	0		C
16	Data valid to DTACK* low	0		B

NOTES:

A. All times given are in nanoseconds.

B. The INTERRUPTER must guarantee this timing between two of its outgoing signal transitions.

C. The INTERRUPTER must wait for the incoming signal edge from the INTERRUPT HANDLER before changing the level of its outgoing signal.

D. This is a guarantee that the INTERRUPT HANDLER will not change the incoming signal until the INTERRUPTER changes its outgoing signal.

E. The INTERRUPTER is guaranteed this timing between two of its incoming signal transitions.

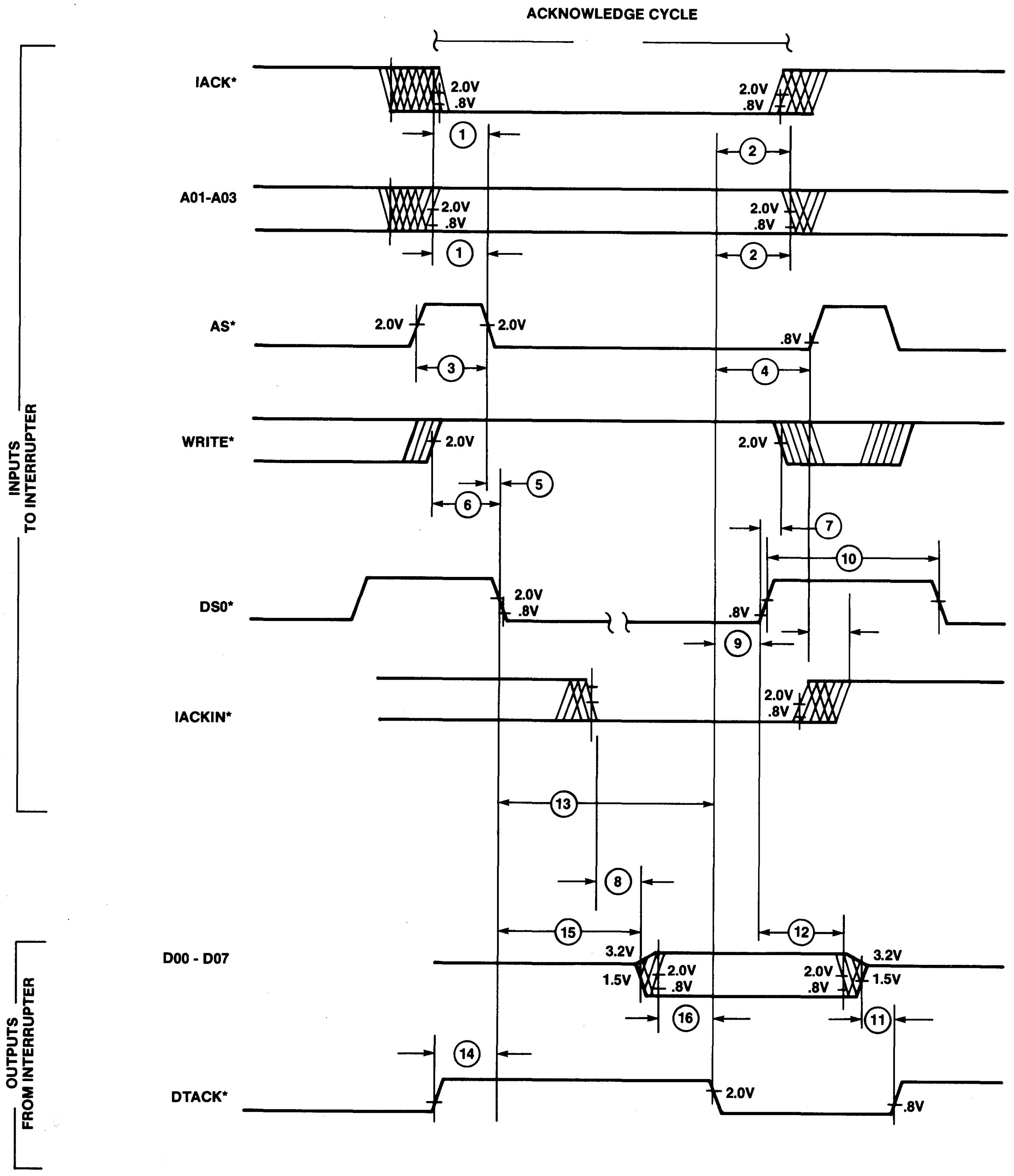


FIGURE 4-16. INTERRUPTER Timing

CHAPTER 5

VME BUS UTILITIES

5.1 INTRODUCTION

This chapter identifies and defines the signal lines and modules which serve utility-type functions for the VME bus. Utility lines supply periodic timing signals and provide initialization and diagnostic capability for the VME bus. Utility lines include:

System Clock	(SYSCLK)
AC Fail	(ACFAIL*)
System Reset	(SYSRESET*)
System Test	(SYSFAIL*)

ACFAIL* and SYSRESET* are typically driven by a POWER MONITOR module. Its purpose is to detect power failures and reset the system upon power up, initiating a power-up self-test. (See Figure 5-1.)

5.2 UTILITY SIGNAL LINES

5.2.1 System Clock (SYSCLK) Specification

The system clock is an independent, non-gated, fixed frequency, 16 megahertz, 50 percent (nominal) duty cycle signal. It can be used to generate on-board delays or timing functions where a fixed duration delay will be generated by counting off a known time base. <u>SYSCLK has no fixed phase relationships with other VME bus timing</u>. Figure 5-2 shows the system timing diagram.

The SYSCLK driver is normally located on the system controller located in board slot one (see Chapter 1). The driver is a high current totem-pole device and only one receiver per card is allowed.

5.2.2 System Initialization and Diagnostics

System Reset (SYSRESET*) is an open collector line driven by a POWER MONITOR module and/or by a manual switch (such as from an operator's panel). Failure of a system test is shown via the system fail line (SYSFAIL*). It is recommended but not required that all boards within the VME bus system drive this system fail line low upon power up, and maintain it low until they have passed their respective self tests. Non-intelligent boards which are incapable of self test could maintain this line low until a MASTER in the system writes to their on-board test register. Whenever SYSRESET* is driven to low, it must be held there for a minimum period of 200 milliseconds. Figure 5-3 shows the required timing relationship which must be maintained between the system reset and the system fail line.

After SYSRESET* is released, the system software executes a test sequence. Upon completion of testing, the system releases SYSFAIL* and goes into the normal operating mode if functional tests show no failures. The SYSFAIL* line is not allowed to go high if any failures occur.

SYSFAIL* can also be driven low at any time during normal operation as the result of a detected system failure. As an example, an intelligent board may periodically perform an on-board self-test and activate SYSFAIL* if a failure occurs.

5.3 POWER MONITOR MODULE

This module is usually an external PC board (Figure 5-4) upon which the ACFAIL* and (optionally) SYSRESET* drivers are located. Logic to detect a reset command from an operator's panel and to detect an AC power failure may also be placed on this circuit card.

The ACFAIL* and SYSRESET* transitions, and the point at which the system DC voltages violate specification, have certain timing relationships. These relationships are spelled out in Figures 5-5 and 5-6.

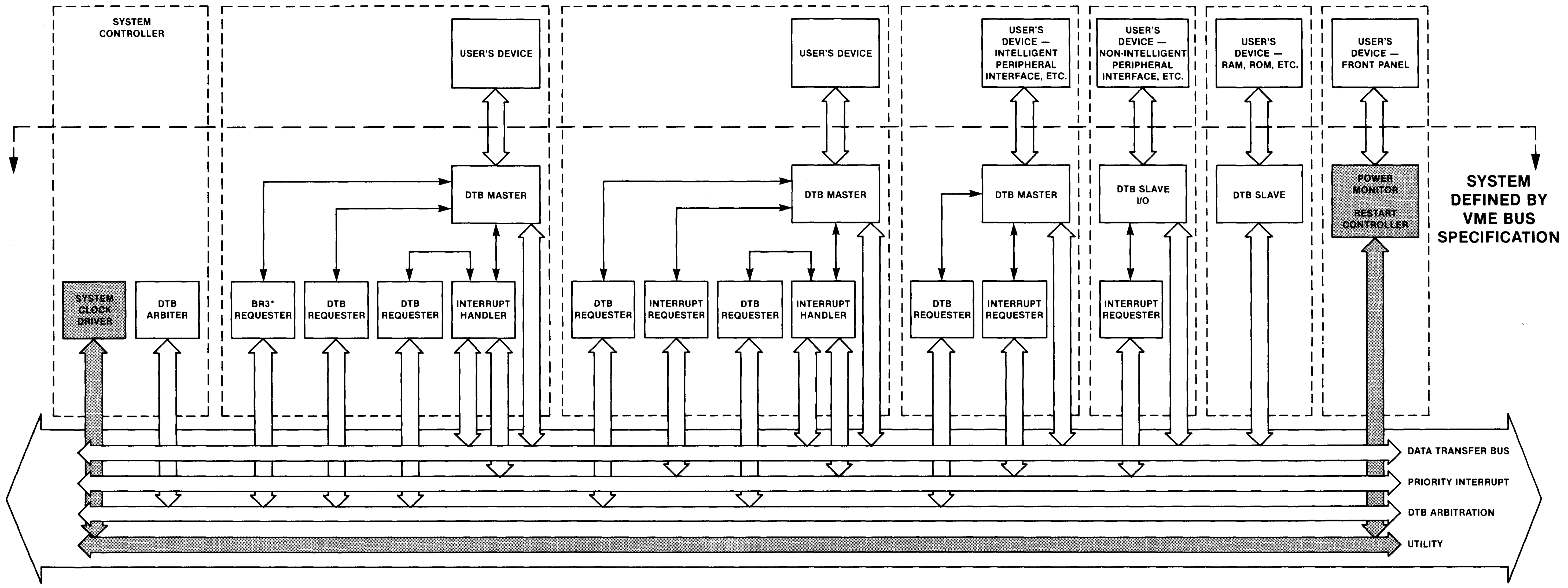


FIGURE 5-1. VME Bus Utility Block Diagram

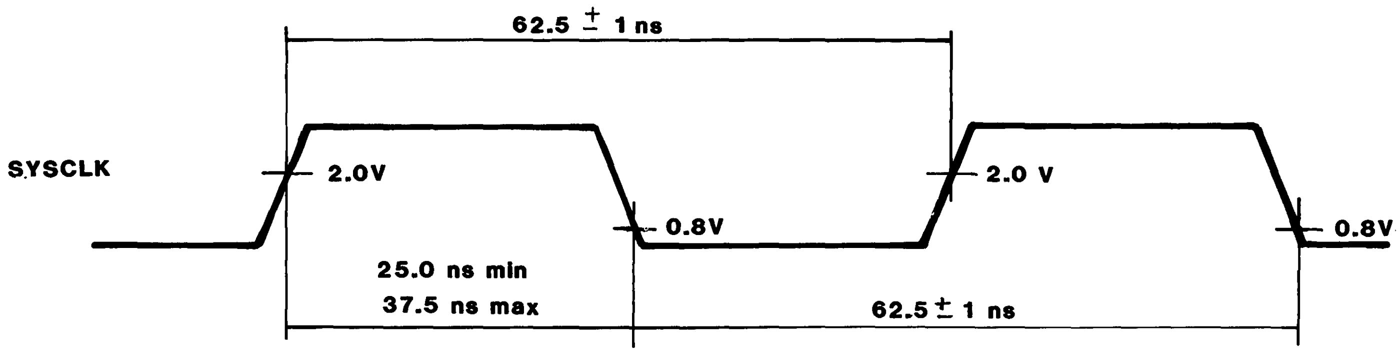


FIGURE 5-2. System Clock Timing Diagram

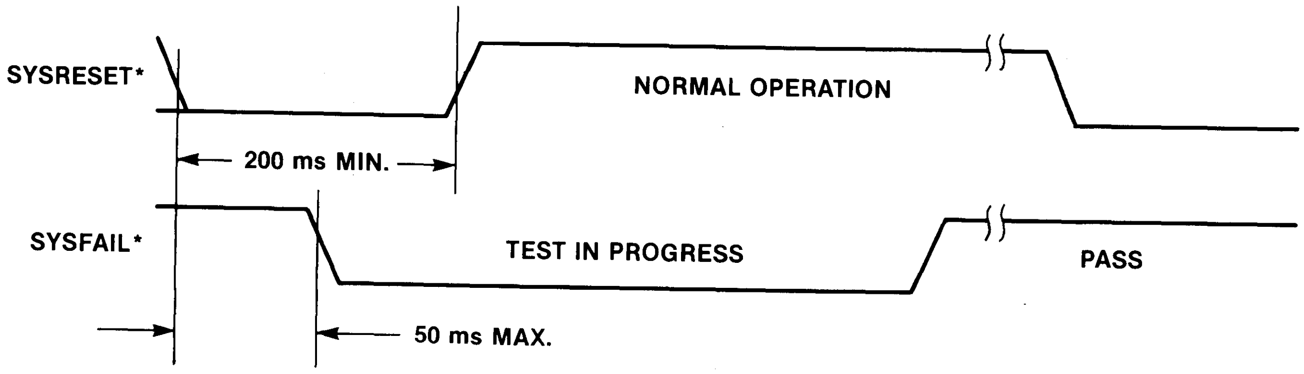


FIGURE 5-3. System Reset and Test Timing Diagram

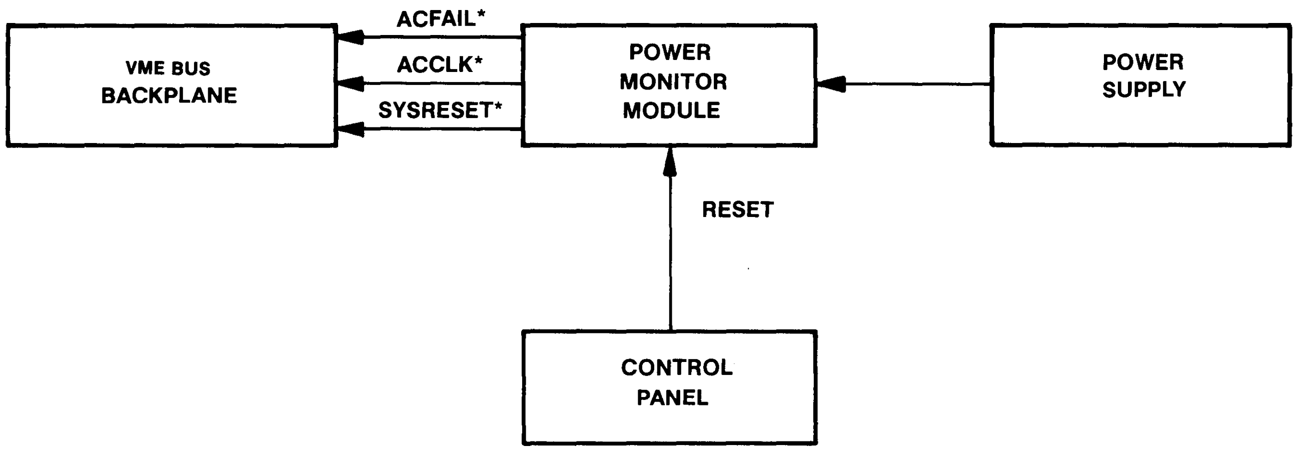


FIGURE 5-4. Block Diagram of POWER MONITOR Module

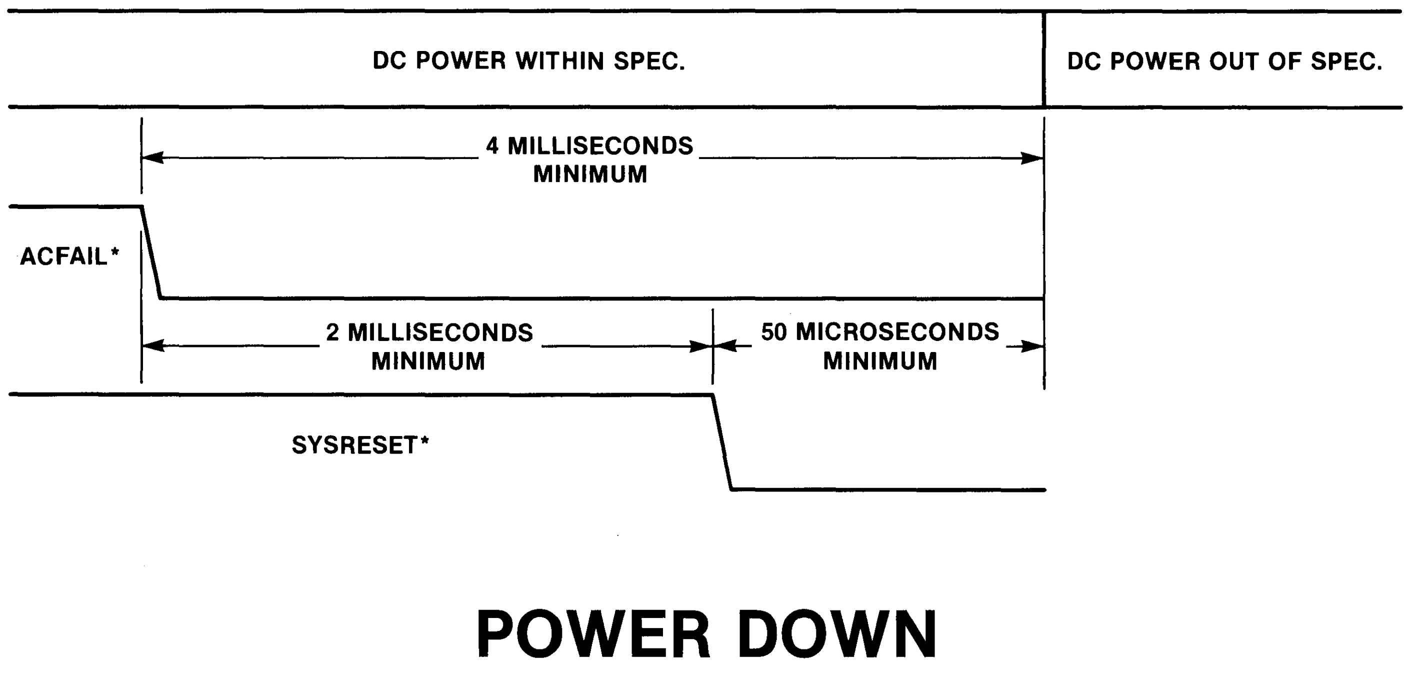


FIGURE 5-5. System Power Fail Timing

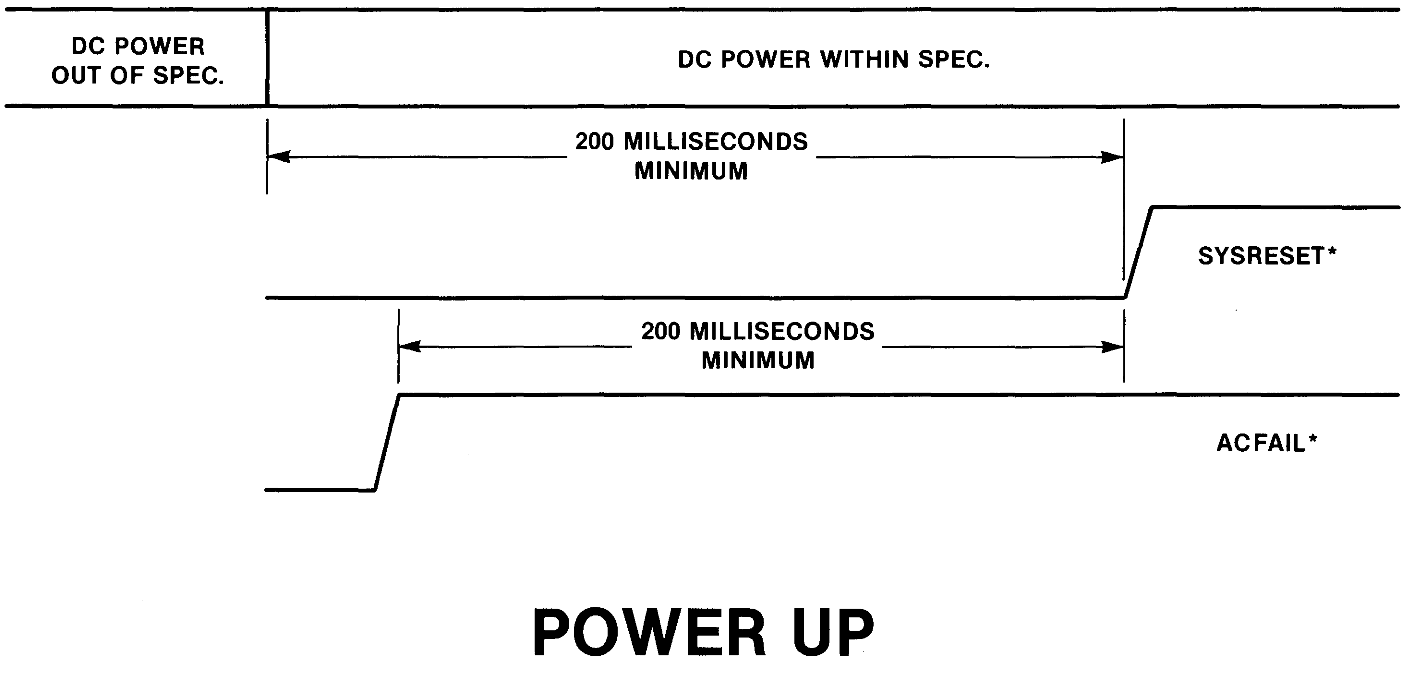


FIGURE 5-6. System Power Restart Timing

5.4 POWER PINS

Although individual pins on the VME bus cards are specified as having 1.5 ampere capacity, the slightly varying pin contact resistance produces uneven current flow in pins which are bussed common on both the backplane and card. For this reason, recommended limits on power consumption are less than 1.5 ampere times the number of pins. Following is a table of current limits for each Eurocard.

+5 Volts	3 A
+5 Standby	1 A
+12 Volts	1 A
-12 Volts	1 A

A double-wide card which incorporates the optional 32-line bus expansion would be allowed to draw six amperes of +5 Vdc because three additional +5 Vdc pins are available.

5.5 RESERVED LINES

The VME bus specification labels one signal lines as RESERVED. This line is set aside for future expansion of VME bus functional capabilities, and <u>should not</u> be used in any Eurocard designs.

CHAPTER 6

VME BUS OPTIONS

6.1 INTRODUCTION

The VME bus specification allows a broad range of design latitude, permitting the designer to optimize cost and/or performance. This is done by defining options. Options allow the user to trade one feature for another or save cost by eliminating some feature. In most cases, these options are intended for use in a particular combination (e.g., a MASTER designed for 32-bit data transfers is intended for use with a SLAVE capable of 32-bit data transfers). There will be cases, however, where the user will mix options. It is important in these cases that the user understand the performance limitations imposed by the mix. This chapter highlights the limitations which will be encountered.

In order for the user to evaluate the limitations of a system prior to buying Eurocards from multiple vendors, he must have access to information which completely describes the options of each board. This chapter provides a standard notation which may be used by board vendors on the product specification sheets.

6.1.1 Hardware vs Dynamic Option Selectivity

Many Eurocards will be capable of operating in several configurations (e.g., an INTERRUPTER on a slave board might be capable of generating interrupts on any one of seven levels.) In some cases, the user will select the option required through the use of switches, jumpers, or PROM's, thus configuring the system prior to power-up. In this case, the option selected remains fixed as long as the system continues operating. In order to change it, the user must power-down the system and reconfigure it.

Other Eurocards will incorporate dynamic option selectivity. These cards allow their options to be changed by on-board control registers while the system is running through data transfers (e.g., if a MASTER assigns a task to an intelligent peripheral, it may wish to configure the peripheral to interrupt on a particular IRQ line upon task completion).

In cases where an optional configuration is possible, the vendor should list the possible options as follows:

ANY ONE OF xxx, yyy, or zzz (DYN)
ANY ONE OF xxx, yyy, or zzz (STAT)

(DYN) indicates that the selection is done dynamically. (STAT) indicates that the selection is fixed while the system is in operation.

6.2 OPTION DEFINITIONS

The options for the VME bus may be divided into six basic categories:

- . Data transfer
- . Arbitration
- . Interrupt
- . Environmental
- . Power
- . Physical configuration

6.2.1 Data Transfer Options

The data transfer options define:

- . Number of address lines used
- . Number of data lines used
- . Data transfer DTACK* timing
- . Number of microseconds before MASTER bus timeout
- . Capability of sequential access operations

6.2.1.1 Address Bus Options

There are three address bus options:

- . A32
- . A24
- . A16

An A32 MASTER is capable of driving 31 address lines (A01-A31). It drives 31 lines with a valid address whenever it places an extended address AM code on the bus, 23 lines whenever it places a standard AM code on the bus, and 15 lines when it places a short address AM code on the bus.

An A32 SLAVE must be capable of decoding up to 31 address lines (A01-A31). It decodes 31 lines when an extended address AM code is on the bus, 23 when a standard address code is present, and 15 when a short address code is present.

An A24 MASTER is capable of driving 23 address lines (A01-A23). It drives 23 lines with a valid address whenever it places a standard address AM code on the bus, and 15 lines when it places a short address code on the bus. It is never allowed to place an extended address code on the bus.

An A24 SLAVE must be capable of decoding up to 23 address lines (A01-A23). It decodes 23 lines when a standard address code is present, and 15 lines when a short address code is present. It must not respond when an extended address code is present.

An A16 MASTER is capable of driving 15 address lines (A01-A15). It must always place a short address AM code on the bus when addressing a SLAVE. It is never allowed to place an extended address code or standard address code on the bus.

An A16 SLAVE must be capable of decoding 15 address lines (A01-A15). It decodes 15 lines when a short address code is present on the bus. It must not respond when an extended or standard address code is present.

6.2.1.2 Data Bus Size Options

There are three data transfer bus word size options:

. D8
. D16
. D32

A D8 MASTER is capable of driving and monitoring 16 data lines, although it will drive or monitor only eight at any given time. This allows it to do 8-bit transfers on either (D00-D07) or (D08-D15). It is never allowed to drive LWORD* low.

A D8 SLAVE is capable of driving and monitoring eight data lines (D00-D07). This allows it to do 8-bit transfers only.

A D16 MASTER is capable of driving and monitoring 16 data lines (D00-D15). This allows it to do either 16-bit data transfers on (D00-D15) or 8-bit transfers on (D00-D07) or (D08-D15). It is never allowed to drive LWORD* low.

A D16 SLAVE is capable of driving or monitoring 16 data lines (D00-D15). It is capable of 16-bit transfers on (D00-D15) or 8-bit transfers on (D00-D07) or (D08-D15).

A D32 MASTER is capable of driving and monitoring 32 data lines (D00-D31). This allows it to do 32-bit data transfers on (D00-D31) while driving LWORD* low, and 16-bit data transfers on (D00-D15), or 8-bit data transfers on (D00-D07) or (D08-D15) while driving LWORD* high.

A D32 SLAVE is capable of driving and monitoring 32 data lines (D00-D31). When LWORD* is low, it transfers data on all 32 data lines. When LWORD* is high, it must be capable of transferring all internally stored data on the lower 16 data lines (D00-D15).

6.2.1.3 Time-Out Options

Each MASTER may have a timer which will terminate a data transfer cycle if a response (DTACK* or BERR*) is not received from a SLAVE within a specified time. The DTB timing which results looks the same as it would if the MASTER had received a BERR* response at the time-out point (i.e., the MASTER will remove address and data from the DTB, drive strobes high, etc.). If the MASTER has the time-out option, it should be specified by the board vendor as follows:

TOUT = 'x' us

where 'x' is the number of microseconds that the MASTER will wait from the falling edge of the last data strobe.

The optional bus time-out module would perform the same function for the entire system. By monitoring DS0*, DS1*, DTACK*, and BERR*, the BTO module can also generate BERR* if a delay expires with no module on the bus responding to a data strobe. A BTO module's time-out is specified as TOUT = 'x' us, just like the option as a MASTER.

6.2.1.4 Sequential Access Options

Each MASTER which is capable of generating the full protocol for sequential access should be defined as option MSEQ. Each SLAVE which contains all of the registers and logic to respond to a sequential access should be defined as option SSEQ.

6.2.2 Arbitration Options

The arbitration options define:

- what basis the ARBITER uses to grant the DTB,
- what basis the REQUESTER uses to release the DTB.

6.2.2.1 ARBITER options

There are three ARBITER options:

. PRI	Fixed Priority
. RRS	Round Robin Sequence
. ONE	Single level arbiter

A PRI ARBITER uses each request level as a priority number, and will always grant the bus to the highest level number pending. This is useful in systems requiring a hierarchical assignment.

An RRS ARBITER shifts its priority sequence based on the level of the last previous bus user. This provides a relatively uniform distribution, useful in systems containing multiple co-equal MASTERS.

A ONE ARBITER monitors only BR3* and will respond only via the BG3IN*/BG3OUT* daisy chain. The only arbitration done is accomplished by the daisy chain.

6.2.2.2 REQUESTER Options

There are two REQUESTER options:

- Release When Done (RWD)
- Release On Request (ROR)

Each option reflects the basic criteria that the REQUESTER uses when determining when to release the DTB for arbitration.

An RWD REQUESTER releases the BBSY* line each time its on-board MASTER indicates it no longer wants the bus. This option is beneficial where multiple MASTERS share the use of the bus equally and where data transfers are done mostly on a cycle by cycle basis.

An ROR REQUESTER does not release the BBSY* line each time its on-board MASTER indicates it no longer wants the bus. Instead, it waits until some other REQUESTER requests the DTB. The ROR option is beneficial in systems where maximizing data transfer rate for a particular MASTER is desired and where other MASTERS have a comparatively low bus usage.

In addition to RWD and ROR options, the request level which the REQUESTER uses is also specified as an option:

R(x)

where 'x' is the number of the request line used.

6.2.3 Interrupt Options

The interrupt options are used to describe:

- INTERRUPT HANDLERS
- INTERRUPTERS

6.2.3.1 INTERRUPT HANDLER Options

The INTERRUPT HANDLER options are used to describe which interrupt request lines a given INTERRUPT HANDLER will respond to. The notation used is shown below:

IH(x-y)

where 'x' is the lowest numbered interrupt request line number and 'y' is the highest. (x may be equal to y when the INTERRUPT HANDLER responds to only one level.) As is evidenced by this notation, if an INTERRUPT HANDLER responds to any two interrupt request lines x and y, it must also respond to all lines numbered between x and y. The INTERRUPT HANDLER should respond to these lines in prioritized manner with the highest priority assigned to the highest numbered line.

When the user configures a system, he must ensure that no two INTERRUPT HANDLERS will respond to the same line (e.g., an IH(2-4) could not be mixed with an IH(3-6) because levels 3 and 4 are responded to by both INTERRUPT HANDLERS.

If each INTERRUPT HANDLER responds to only one request line, it is possible to configure a system with seven INTERRUPT HANDLERS. This configuration is most useful in multiprocessing systems where processor-to-processor interrupts are required.

6.2.3.2 INTERRUPTER Options

The INTERRUPTER options are used to describe which interrupt request line a given INTERRUPTER uses to generate its interrupt. The notation used is shown below:

I(x)

where 'x' is the number of the interrupt request line used.

Since each interrupt request line is driven by open-collector drivers, there is no specific limit on the number of requesters which may use a single line (e.g., a system could be configured with five I(4) INTERRUPTERS).

Special note should be made that INTERRUPTERS are often designed for dynamic option selection. This is especially useful in a multiprocessor system where a processor may assign a task to an intelligent device and configure the device's INTERRUPTER to use the appropriate interrupt line upon task completion.

6.2.4 Environmental Options

Two environmental options should be specified:

. TEMPERATURE
. HUMIDITY

The minimum and maximum operating temperature should be specified in centigrade. The minimum and maximum storage temperature may be specified if the vendor deems it to be important. When deriving the operating temperature, it should be assumed that the board lies in a vertical plane with convection cooling only.

The maximum recommended operating humidity should be given. It should be relative humidity and should be expressed as a percent. It may be assumed that the mixing of the air in the proximity of the board is sufficient to prevent condensation. A typical value for most boards is 90%.

6.2.5 Power Options

The power requirements for each Eurocard should be specified as follows for each voltage:

x mA max (y mA typ) at z VDC

6.2.6 Physical Configuration Options

Three optional configurations are allowed:

EXP (expanded bus-double Eurocard)
NEXP (non-expanded bus-double Eurocard)
SINGLE (single Eurocard)

6.2.6.1 Expanded Configuration

An expanded MASTER or SLAVE is an option A32 and/or D32 double width (see Chapter 8) Eurocard which is capable of driving/monitoring signal pins B1-32 of the J2 connector on the backplane.

An expanded system requires that J2 be a 96-pin DIN 41612 connector, and provides bussed interconnects for signal pins B1-32 of the J2 edge connectors. This allows expanded MASTERS and SLAVES to make use of the extended address and data bus. This also requires a second backplane for those signals bussed on J2.

See Appendix E.

6.2.6.2 Non-Expanded Configuration

A non-expanded MASTER or SLAVE is an option (A24 or A16)/(D16 or D8) double width (see Chapter 8) Eurocard which does not use signal pins B1-32.

A non-expanded system may or may not have any given DIN connector as a secondary connector, and it does not require a second backplane.

6.2.6.3 Single-Size Configuration

A single-size MASTER or SLAVE is an option (A24 or A16)/(D16 or D8) single width (see Chapter 8) Eurocard. Since it has no P2 edge connector, any I/O must be from the front edge of the card.

A single-size card cage has no J2 connector. The bussing provided for the signal lines on the J1 connector is identical to that of the double Eurocard system.

6.2.6.4 Mixing Expanded, Non-Expanded, and Single-Size Options

Expanded Eurocards <u>may</u> be installed in a non-expanded system, but they will not be capable of doing extended addressing or LONGWORD transfers. Since A32/D32 MASTERS and SLAVES are capable of placing standard and short addresses on the VME bus and since they are able to do word transfers, they will work satisfactorily with the non-expanded boards which share the system.

Single-size Eurocards may be installed in double Eurocard slots. However, the card cage must provide support for both edges of the board.

Double-size Eurocards cannot be installed in a single-size system because of the obvious mechanical incompatibility.

It would be possible to construct a backplane/cardcage which would support all three Eurocard configurations by having a certain number of each type of slot. Where this is the case, the vendor should specify the number of slots provided for each configuration supported.

6.2.6.5 Examples of Vendor Specification Sheets

The following two examples show how a vendor of VME bus compatible products may use the option notation described in this chapter to concisely describe product features.

VME Bus Card Rack

SPECIFICATION

ENVIRONMENTAL OPTIONS:

OPERATING TEMPERATURE: 0° C to 70° C

STORAGE TEMPERATURE: -65° C to $+150^{\circ}$ C

MAXIMUM OPERATING HUMIDITY: 90%

PHYSICAL CONFIGURATION OPTIONS

EXP (4 SLOTS)

NEXP (4 SLOTS)

VME Eurocard
SPECIFICATION

MASTER DATA TRANSFER OPTIONS

A24:D16
TOUT = ANY ONE OF 4, 8, 16, or 32 us (STAT)

ARBITER OPTIONS

ANY ONE OF PRI, or RRS (STAT)

REQUESTER OPTIONS

ANY ONE OF R(0), R(1), R(2), or R(3) (STAT)
ROR

INTERRUPT HANDLER OPTIONS

ANY ONE OF IH(x-y) (STAT)
where $1 \le x \le 7$ and $x \le y \le 7$

INTERRUPTER OPTIONS

ANY ONE OF I(1), I(2), I(3), I(4), I(5), I(6), or I(7) (DYN)

ENVIRONMENTAL OPTIONS

OPERATING TEMPERATURE: 0° C to 70° C
MAXIMUM OPERATING HUMIDITY: 90%

POWER OPTIONS

1.2 A MAX (900 mA typ) at +5 VDC
300 mA MAX (250 mA typ) at +12 VDC
150 mA MAX (120 mA typ) at -12 VDC

PHYSICAL CONFIGURATION OPTIONS

NEXP

CHAPTER 7

VME BUS ELECTRICAL CONSIDERATIONS

7.1 INTRODUCTION

This chapter defines the non-timing electrical specifications for proper VME bus interfacing. VME bus signal levels are normally TTL generated, although any technology which complies with the specification may be used. In addition, recommendations and examples are included in many sections to aid the designer in obtaining optimum system performance.

7.2 POWER DISTRIBUTION

Power in a VME bus system is distributed on the backplane as regulated direct current (DC) voltages. The available voltages are described as follows.

+5 Vdc is the main logic level and normally has the largest associated current requirement. The bulk of the system circuitry - including TTL logic, MOS microprocessors, and memories - requires this voltage.

±12 Vdc represent the auxiliary digital logic supplies. They supply the needs for MOS memories and I/O circuitry requiring multiple voltages. They may also be used for analog purposes. A -5 Vdc bias voltage and a -5.2 Vdc ECL voltage may also be derived from -12 Vdc, as needed. These supplies normally have lower current requirements than the +5 Vdc.

+5 Vdc Standby is used for distributing battery backup power. The standby voltage is maintained during system power loss to sustain memory and time-of-day clocks. If the user is not concerned with power fail protection, this supply line should be supported by the normal +5 Vdc supply.

7.2.1 Bus Voltage/Current Specifications

Table 7-1 summarizes the bus voltage/current specifications. The listed specifications are the maximum allowed variance as measured at the connector of any card plugged into the backplane.

The percentage listed under VARIATION is the total DC tolerance allowed at the Eurocard connector. This percentage is the sum expressed as:

VARIATION = DISTRIBUTION + LINE-REGULATION + LOAD-REGULATION

where:

DISTRIBUTION	is defined as the percent variation caused by backplane effects (resistive losses and differences from power supply sense point).
LINE-REGULATION	is defined as provided in the vendor's power supply specification.
LOAD-REGULATION	is defined as provided in the vendor's power supply specification.

TABLE 7-1. Bus Voltage Specifications

MNEMONIC	DESCRIPTION	VARIATION (see NOTE)	RIPPLE & NOISE BELOW 10 MHz (PK-PK)	CONNECTOR PIN NUMBERS	MAXIMUM CURRENT DRAW PER SLOT
+5V	+5 Vdc power	+5.0%/-2.5%	50 mV	A32,B32,C32	3 A
+12V	+12 Vdc power	+5.0%/-3.0%	50 mV	C31	1 A
-12V	-12 Vdc power	+3.0%/-5.0%	50 mV	A31	1 A
+5V STDBY	+5 Vdc standby	+5.0%/-2.5%	50 mV	B31	1 A
GND	ground	Ref.	Ref.	A9,11,15,17,19 B20,23 C9,13	

NOTE: The non-symmetric variation spec is given to ensure that the DC power will remain within the ±5% tolerance required by most IC's despite any drops resulting from power distribution on individual Eurocards.

The voltage tolerances listed apply to steady state conditions in the system. If very high current fluctuations occur due to system operation (such as might be caused by memory refresh), the response time of the voltage distribution system becomes important. Sufficient bypass capacitance and adequate power supply response time must be provided for such cases.

7.2.2 Pin and Socket Connector Electrical Ratings

The pin and socket connector mechanical specifications are listed in Chapter 8. The electrical specifications are listed below:

a. Voltage rating: 200 volts DC, minimum pin to pin
b. Current rating: 1 A per contact
c. Contact resistance: 50 milliohms maximum at rated current
d. Insulation resistance: 1000 megohms, minimum pin to pin

7.3 ELECTRICAL SIGNAL CHARACTERISTICS

Other than power supply lines, all VME bus signals are limited to positive levels between 0 and 5.0 volts. As described in paragraph 1.4.1, the signal levels are:

a. $0.0\ V \leq \text{Low level} \leq 0.8\ V$
b. $2.0\ V \leq \text{High level} \leq 5.0\ V$

Figure 7-1 gives a simple graphic representation of these levels.

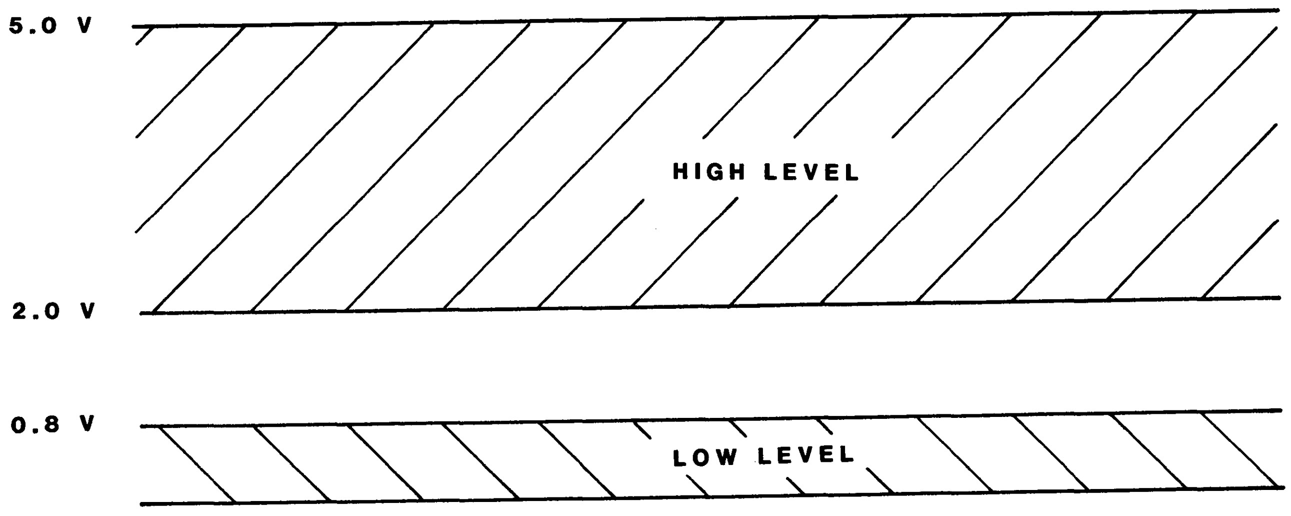


FIGURE 7-1. VME Bus Signal Levels

Depending on the function required, the VME bus uses three-state, wired-OR, and totem-pole drivers. The drivers are specified in paragraph 7.4, and the receivers are specified in paragraph 7.5. Appendix C lists signal lines by function, and gives their associated characteristics.

7.4 DRIVER SPECIFICATIONS

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

a. 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).

b. Three-state - similar to a totem-pole driver except that it can go to a high impedance state (drivers turned off) in addition to the 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. Only one driver can be active at any one time (e.g., data transfer bus).

c. Open-collector - sinks current in the low state but sources no 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 7-2 lists driver specifications. As examples, 74LS640-1, 74LS645-1, 74S240, or 74S241 chips can be used for totem-pole or three-state drivers requiring 48 mA current sink capability. All standard 74LS outputs can drive unterminated line totem-pole applications using 8 mA current sink. Open-collector applications can use the 74S38.

TABLE 7-2. Bus Driver Specifications

DRIVER TYPE	PARAMETERS	MIN.	MAX.	UNIT	TEST CONDITION
Totem-pole	Low state (V_{OL})		0.6	V	Sink 48 mA
(High current)	High state (V_{OH})	2.4		V	Source 3 mA
Totem-pole	Low state (V_{OL})		0.6	V	Sink 8 mA
(Low current)	High state (V_{OH})	2.7		V	Source 400 uA
Three-state	Low state (V_{OL})		0.6	V	Sink 48 mA
	High state (V_{OH})	2.4		V	Source 3 mA
	Off-state output current (I_{OZ})		$\pm$ 50	uA	2.4V or 0.5V applied
Open Collector	Low state (V_{OL})		0.7	V	Sink 40 mA
	High state output current (I_{OH})		50	uA	5.0V applied

NOTE: For output current, a positive value indicates current flow into the driver. A negative value indicates current flow out of the driver.

7.5 RECEIVER SPECIFICATIONS

Table 7-3 lists the standard receiver specifications. Bus receivers should have input diode clamp circuits to prevent excessive negative voltage excursions.

The same specifications apply to all signal lines with the exception of the low current drive daisy-chain lines. A standard 74LS receiver meets this specification. The standard specification of 50 uA can be met by using devices with standard PNP inputs.

TABLE 7-3. Bus Receiver Specifications

PARAMETER	MIN.	MAX.	UNIT	INPUT VOLTAGE
Low state input voltage (V_{IL})		0.8	V	
High state input voltage (V_{IH})	2.0		V	
Low state input current (I_{IL})		-400	uA	$0 \leq V \leq 0.5$
High state input current (I_{IH})		* 50	uA	$5.0 \geq V \geq 2.7$

* High state input current I_{IH} should be limited to 20 uA for low current totem-pole drive lines. (20 uA represents a standard LS TTL input.)

7.6 BACKPLANE SIGNAL LINE INTERCONNECTIONS

The VME bus is an asynchronous, high speed bus intended for high performance systems. Generally, the signal line distance on the backplane will be short, and signal line noise will be low. The following paragraphs specify and describe signal line characteristics and elements that influence backplane signals.

7.6.1 Termination Networks

A standard termination is used on each end of all VME bus signal lines except daisy-chain lines. The termination serves two purposes:

a. reflection and ringing reduction,
b. high state pullup for open collector drivers.

The Thevenin equivalent of the standard termination is shown in Figure 7-2. One possible resistor network configuration which achieves this is also shown. Resistor values and source voltage of the Thevenin equivalent must be 5% tolerance maximum.

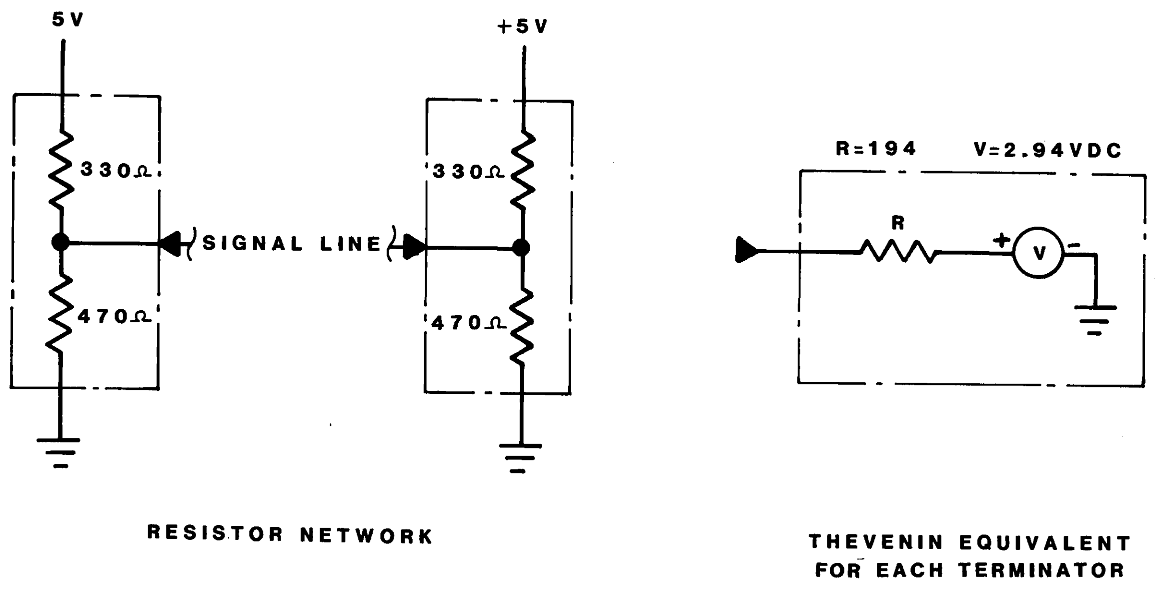


FIGURE 7-2. Termination Network

The signal line usage of termination networks is summarized below:

a. High current totem-pole drive lines, three-state lines, and open-collector drive lines all use terminations at both ends of the line.

b. Low current (8 mA) totem-pole drive lines (daisy-chain lines) use <u>no</u> terminations.

7.6.2 Characteristic Impedance

Each signal in the backplane (normally multilayer) has an associated characteristic impedance Z_o. This characteristic impedance is important because discontinuities in Z_o (due to capacitive effects and loads on the bus) and mismatches between Z_o and the terminations can cause reflections and ringing on signal waveforms.

Figure 7-3 shows a microstrip signal line cross section which is the normal configuration for a multilayer backplane signal line. The Z_o is a function of the width and thickness of the line, the thickness of the dielectric, and the relative dielectric constant (e_r). Figure 7-4 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, 1980.

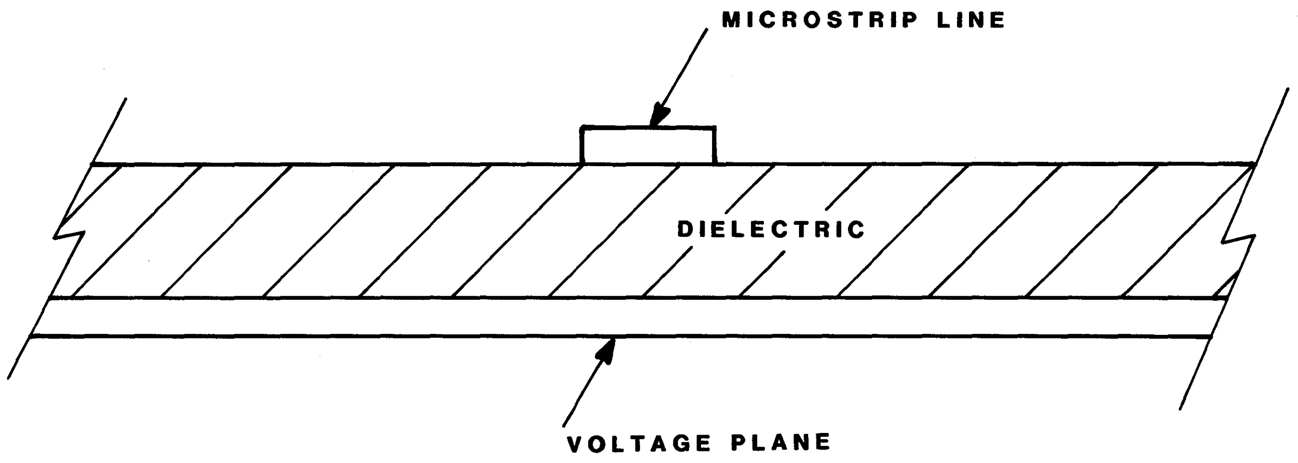


FIGURE 7-3. Backplane Microstrip Signal Line Cross Section

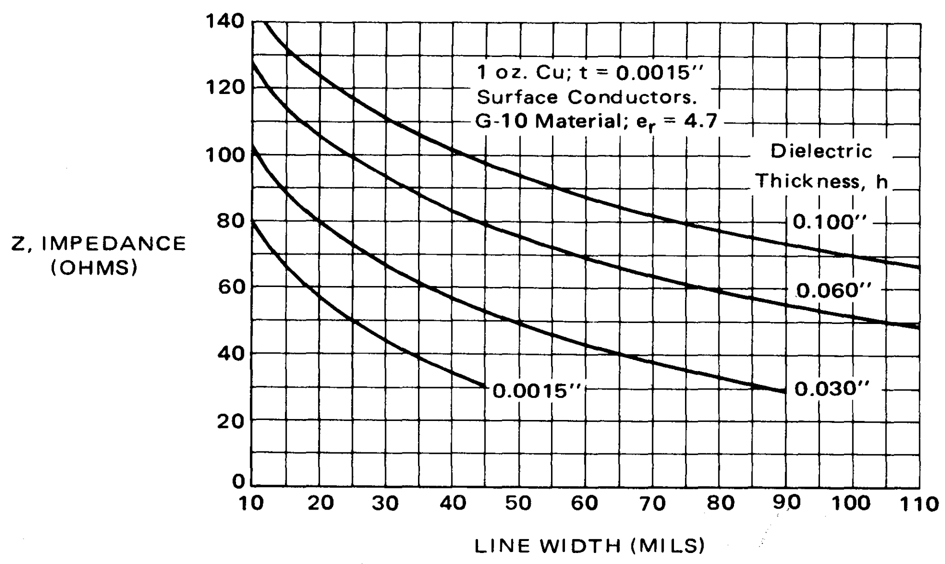


FIGURE 7-4. Impedance versus Line Width and Dielectric Thickness for Microstrip Lines

As stated in section 7.6.1, the terminations on the VME bus serve to help minimize ringing. Although an impedance match (the best case) is not maintained between the termination networks and the signal line impedance, it is important not to allow the signal line Z_o value to become too low. The calculated Z_o of the microstrip line itself - e.g., its <u>unloaded</u> Z_o (considering <u>no</u> loading effects of pc boards, connectors, and plated-through holes) - should be no lower than 100 ohms.

The actual characteristic impedance of a backplane signal line is called the <u>effective characteristic impedance</u> Z_o' and will be lower than the Z_o due to capacitive effects of plated-through holes and connectors. The resulting capacitance of holes and connectors makes the Z_o' go below the calculated 100 ohm value that is discussed in the preceding paragraph. Although plated-through holes are necessary to accommodate connectors, additional holes should be kept to a minimum.

The designer can calculate the effects of these capacitances. Typically, an empty plated-through hole has one picofarad of capacitance. A plated-through hole with a connector pin has two picofarads of capacitance. The resulting effective impedance Z_o' can be calculated by adding up the total hole/connector pin capacitances and using the following equation:

$$Z_o' = \frac{Z_o}{1 + C_h/C_o}$$

where: Z_o' = effective impedance (line with holes)

Z_o = impedance of just the microstrip line

C_h = sum of the hole capacitance

C_o = intrinsic line capacitance of microstrip line without holes (capacitance/ft x ft). See Figure 7-4.

The resulting Z_o' should not go below 60 ohms.

When PC boards are in the system, they add more capacitive load to the bus lines. The following paragraph gives limits on card loading.

7.6.3 Board Level Loading

For any Eurocard, the following rules apply:

a. Total capacitive load on any VME bus line shall not exceed 25 picofarads. Typically, a driver has a 10 to 15 picofarad capacitance and a receiver has a 3 to 5 picofarad capacitance. The capacitance of the signal line connecting these to the VME bus can be calculated using the capacitance per foot shown in Figure 7-4.

b. Any signal line input to a Eurocard cannot source more than 850 uA at 0.55 V nor sink more than 150 uA at 2.4 volts.

The system clock (SYSCLK) line loading is subject to more stringent requirements. Only one driver is used per system, and the driver should be located at one end of the backplane. Only one receiver can be used per card, and total card input capacitive load on the SYSCLK line shall not exceed 30 pf.

7.7 I/O SIGNALS

Systems using the optional expanded bus and its required second backplane and 96-pin DIN 41612 connectors are required to limit the I/O signals on rows A and C to ± 15 volts to minimize potential short damage.

CHAPTER 8

MECHANICAL SPECIFICATIONS

8.1 INTRODUCTION

Information in this chapter is provided to assure that VME bus backplanes, card racks, and PCB's are mechanically compatible. Throughout this chapter, the following terminology is used:

Backplane - A board into which 96-pin connectors are installed. This board interconnects some of the pins of these connectors to provide a bus.

Eurocard - A PC board which is plugged into the backplane and communicates with other Eurocards installed in the same backplane.

The "front" of a backplane is the side from which the Eurocards are inserted into the connectors.

The "front" of a Eurocard is the side on which the components are mounted.

The "rear edge" of a Eurocard is the edge which provides the on-board connector for mating with the connector on the backplane.

8.2 VME BUS BACKPLANE

This portion of text provides the following VME bus backplane information:

a. Construction techniques

b. Reference designations and pin numbering standards

c. PCB relationships

d. Sockets

e. I/O connectors

f. Socket/connector-pin assignments.

8.2.1 Backplane Construction Techniques

Many backplane construction techniques are available to the designer. Backplane construction can be of a multilayer laminated or unlaminated design. Following is a descriptive example of a multiple layer design composed of multiple discrete, two-sided, glass epoxy boards separated by mylar insulators. The boards and mylar insulators are held together, utilizing high force press fit socket contacts. This process provides a gas-tight connection between the contact and the plated-through hole in each board. Each side or layer of a board provides signal conductors or a voltage/ground plane. Several layers of the backplane contain the signal conductors. Another layer provides the voltage plane, and still another layer is the ground plane. Connectors are press fitted into the boards and the mylar insulators, forming the multi-layer backplane.

A second possible backplane construction technique would be to laminate the boards. This lamination process would bond the boards and insulators together. Then connectors are inserted into the backplane and soldered.

8.2.2 Reference Designations and Pin Numbering Standards

The following standards are recommended for backplane identification purposes (see Figure 8-1):

a. Card slots are designated A1, A2, A3 through A"n". Slots A0 and A"n+1" are not card slots, but represent space for the terminating resistor networks. Slot numbering goes from left to right when viewing the front of the backplane. The daisy-chain wiring must be laid out so the logical flow is from slot A1 to A2 to ... .

b. The primary or only backplane connector is designated as J1, as in slot A2 connector J1. If the expanded system bus is used, a second backplane would exist below the first, utilizing the same style 96-pin DIN 41612 connectors, and bussing pins B1 through B32. These connectors are designated as J2, as in slot A2 connector J2.

c. Additional connectors on either backplane, such as power connectors, are designated P1, P2, P3.

d. If the J1 connector has wire wrap pins, these may be used to jumper the daisy-chain signals for slots which are currently empty. Otherwise, pins must be provided on the backplane for this purpose, and should be designated JB21 for pin B21 jumper, etc. Thus, to bypass a slot on the interrupt daisy-chain requires a jumper from JB21 to JB22. It is recommended that all of the daisy-chain jumpers for a slot be arranged as a 2 x 5 matrix right next to the connector for that slot.

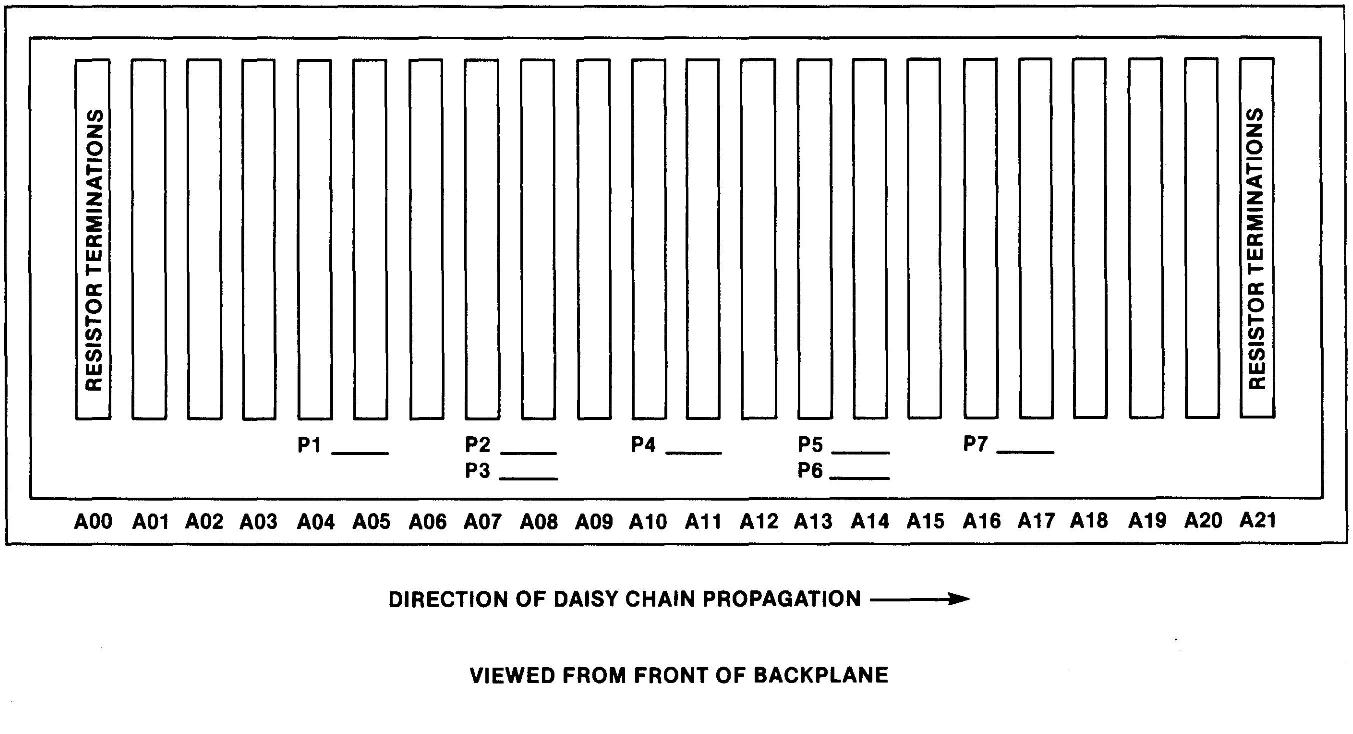

RESISTOR TERMINATIONS
RESISTOR TERMINATIONS
P1
P2
P3
P4
P5
P6
P7
A00 A01 A02 A03 A04 A05 A06 A07 A08 A09 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21
DIRECTION OF DAISY CHAIN PROPAGATION
VIEWED FROM FRONT OF BACKPLANE

FIGURE 8-1. Backplane Reference Designations

8.2.3 Backplane/Eurocard Dimensional Requirements

Certain specifications must be adhered to when designing a VME bus backplane and compatible Eurocards. Figure 8-2 illustrates the following relationships:

a. Board Spacing (B_S) - center-to-center spacing of the connectors J1 and J2 are as follows:

1. Printed Circuit Board spacing	- .800 minimum
2. Wire Wrap Board spacing	- 1.600 minimum

b. Board Thickness (BT)- maximum board thickness is .062 $\pm$ .005 inch.

c. Component Lead Length (LL) - length of the component leads protruding through the back of the Eurocards must not exceed .100 inch.

d. Component Height (CH) - height of the components on the front of each Eurocard must not exceed .600 inch.

e. Board Warpage (BW) - maximum allowable Eurocard warpage is .025 inch.

f. Wire Wrap Pin Height (WW) - length of the wire wrap pins protruding through the back of the Eurocards must not exceed .800 inch.

8.2.4 Eurocard Connectors

Eurocards require a 96-pin DIN41612 standard connector on the top back of the Eurocard which is identified as P1 (see Figure 8-3). If a connector exists on the bottom back of the Eurocard, it is designated as P2 and should be a DIN standard connector. If this Eurocard is for the expanded system, P2 must also be a 96-pin DIN 41612 connector utilizing the pin assignments given in Appendix D.

VIEW FROM FRONT OF BACKPLANE

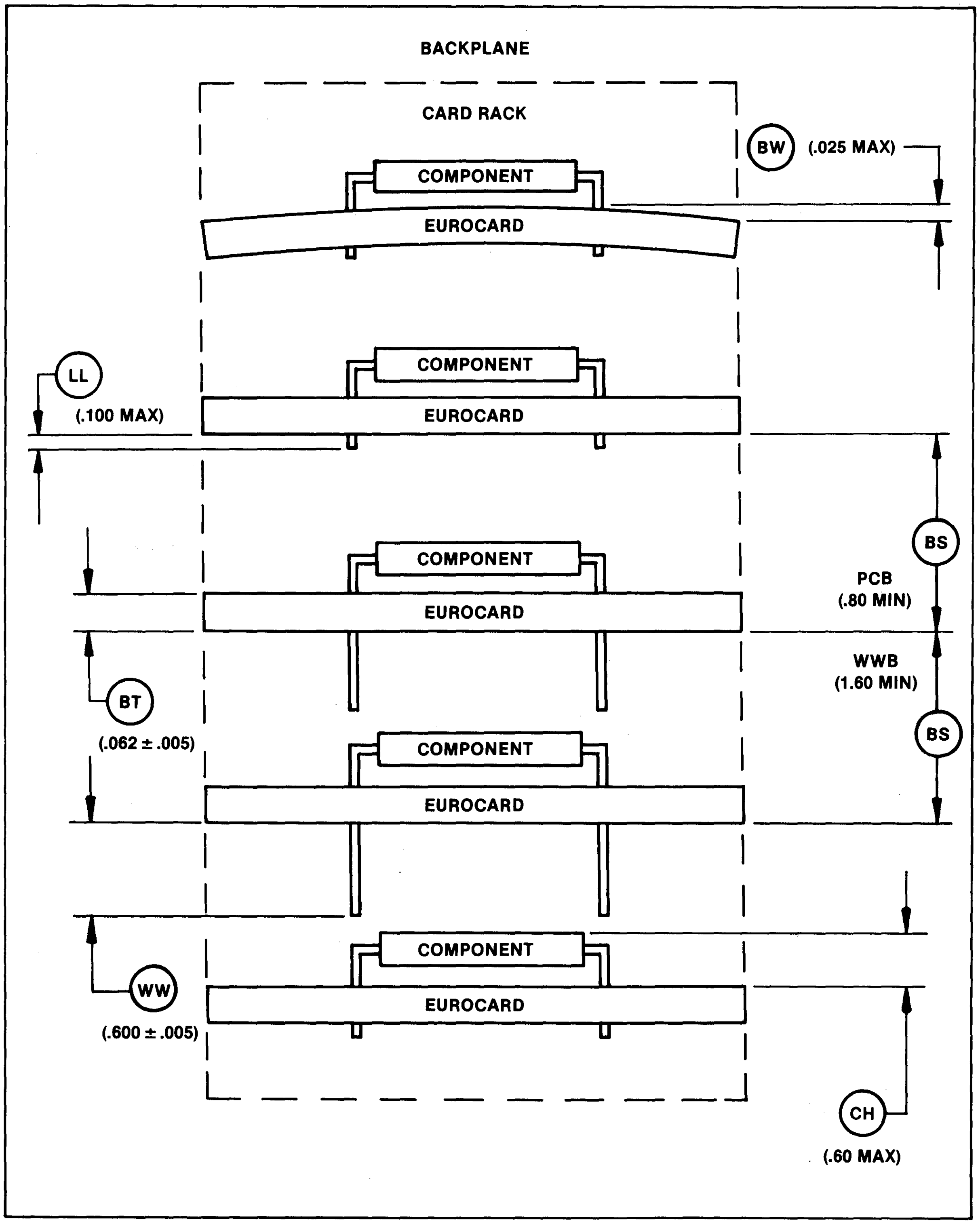
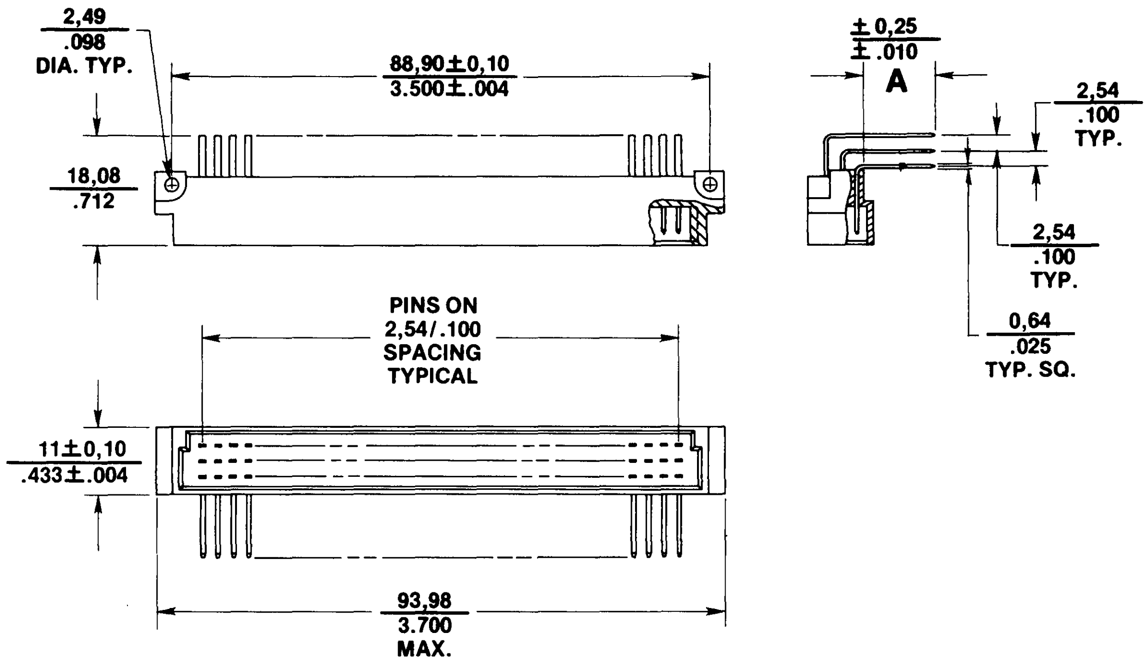
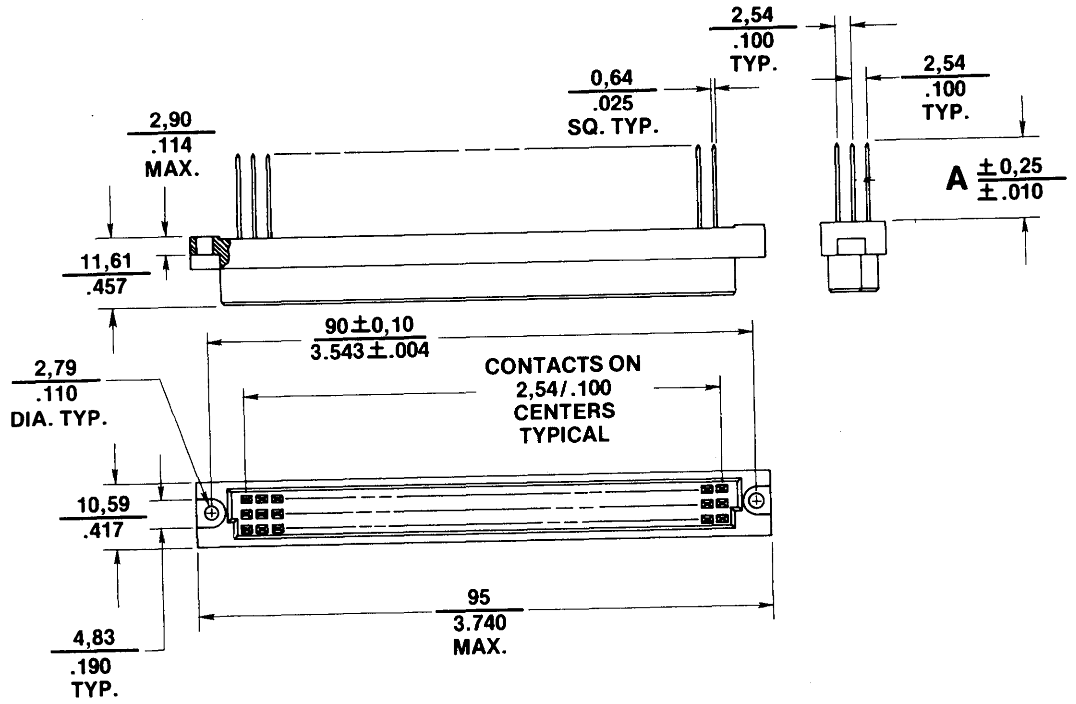


FIGURE 8-2. Backplane/Eurocard Dimensional Requirements



FEMALE CONNECTOR



MALE CONNECTOR

FIGURE 8-3. Typical Backplane/Card Mating Connectors

8.3 EUROCARDS

This portion of text provides the following Eurocard information:

a. Construction techniques
b. Reference designations and pin numbering standards
c. Overall dimensions
d. Connector information

8.3.1 Eurocard Construction Techniques

Many construction techniques are available to the designer. To provide optimum power and ground distribution, a multilayer PC design is recommended; however, this is not a requirement for VME bus compatibility. If a multilayer design is not used (i.e., no ground plane is provided), care should be taken to assure that an adequate ground grid is provided on the board.

8.3.2 Reference Designations and Pin Numbering Standards

Refer to Figure 8-4. The following standards are recommended for identification purposes:

a. Connectors on the back edge of the Eurocard are designated Pl and P2. Pl is the 96-pin primary connector, and P2 is an optional secondary connector.

b. Numbering of the 96-pin connector is depicted in Figure 8-4.

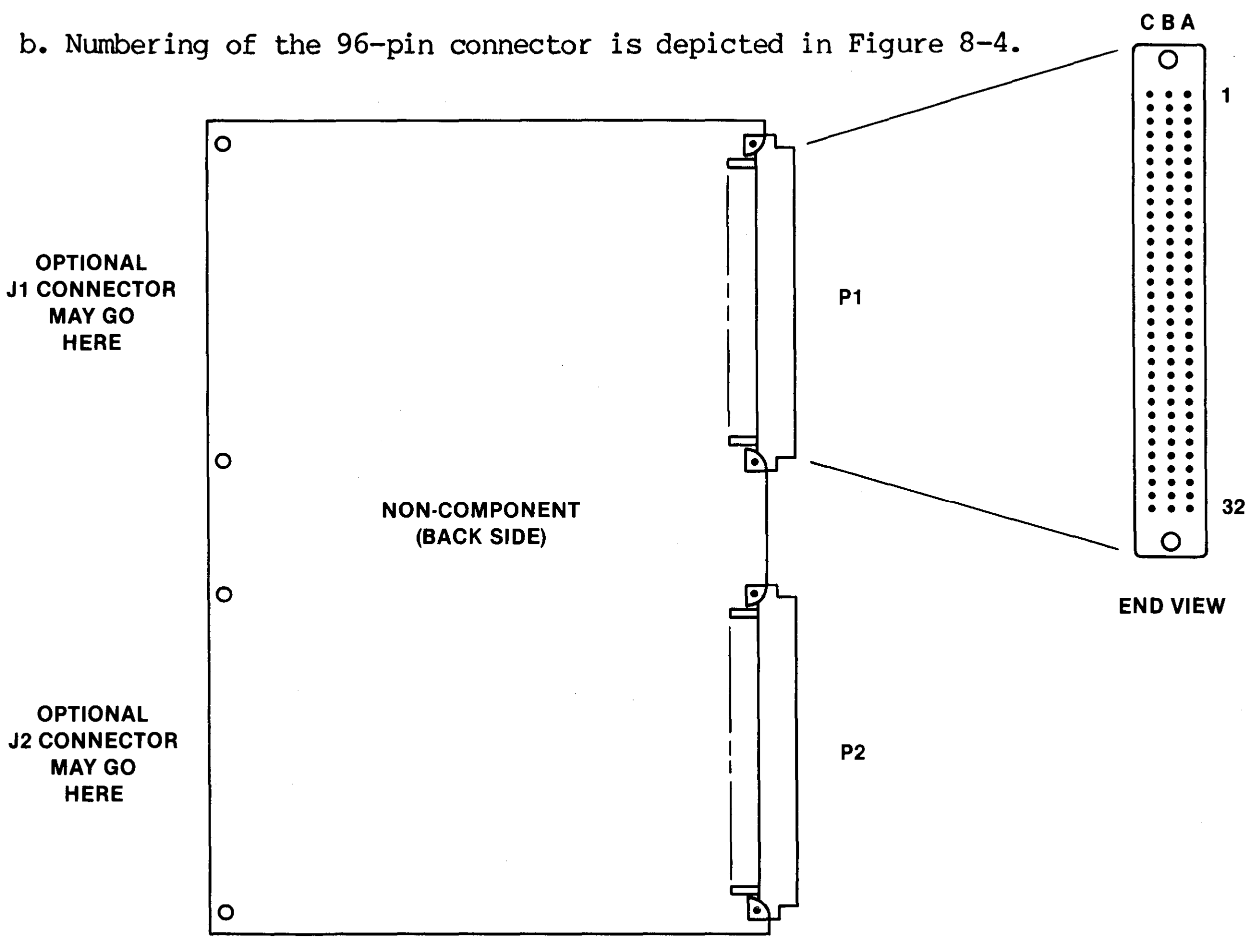


FIGURE 8-4. Eurocard Reference Designations and Pin Numbering Standards

8.3.3 Eurocard Dimensions

Figure 8-5 illustrates the double size dimensions applicable to Eurocards. In addition to the outside dimensions, connector mounting holes are given.

Figure 8-6 illustrates the single size Eurocard dimensions. These dimensions enable a designer to manufacture a board that mates with only the 96-pin VME bus backplane socket J1. Unless the free edge is supported, it is not recommended that this size board be used in a card cage with double size boards.

8.3.4 Eurocard Non-Bus Edge Connectors

While it is recommended for ease of access and stability that all boards requiring I/O be double-width boards and utilize the optional secondary J2 connector for I/O, this is not a requirement for VME bus compatibility. Where I/O is taken off the front of the board, it is recommended that the connector be mounted in a supportive housing for mechanical stability.

NOTE

It is recommended that use of cable connections to the front edge of the board be minimized, since it makes the job of installing and removing Eurocards more difficult.

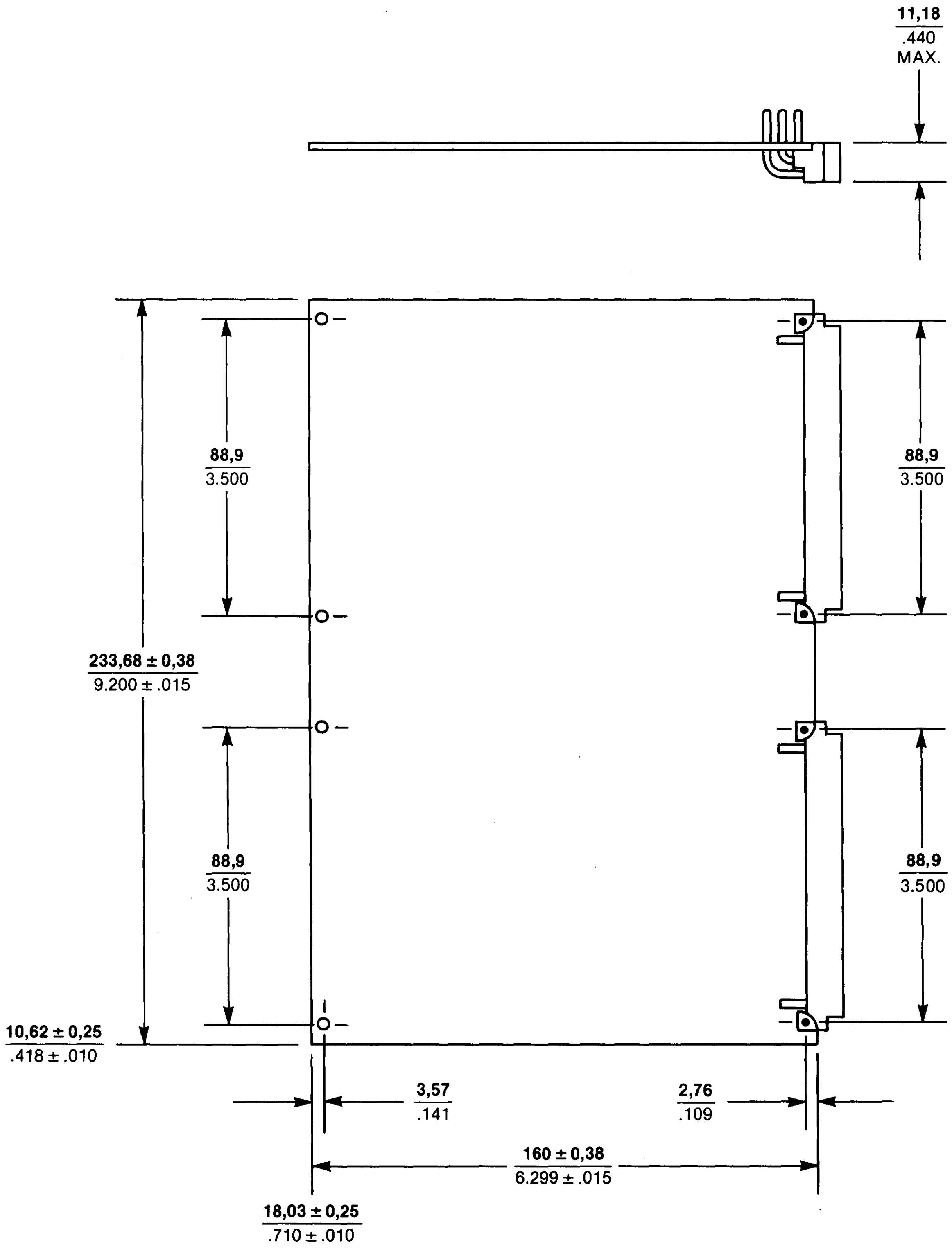


FIGURE 8-5. Double Size Eurocard

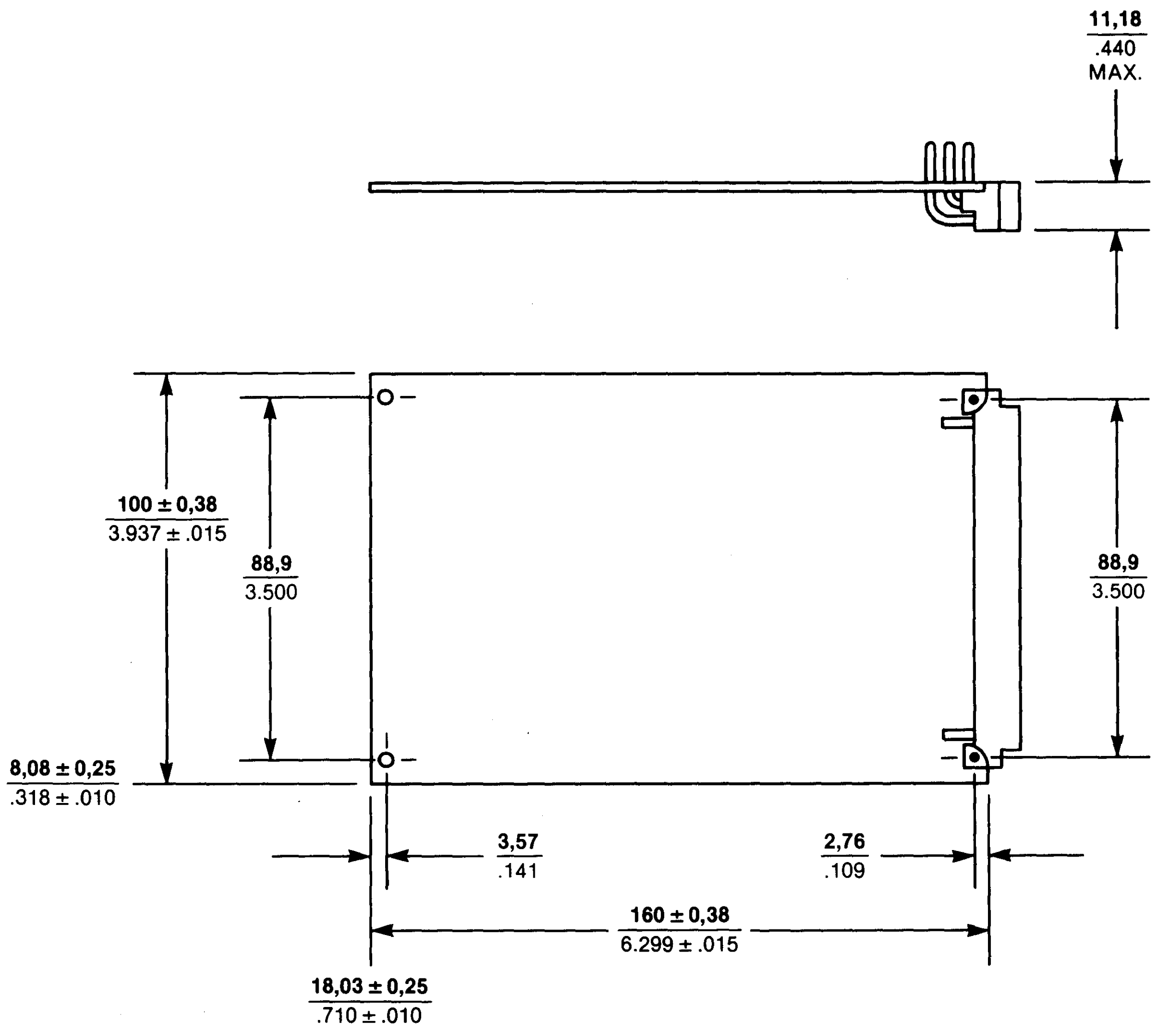


FIGURE 8-6. Single Size Eurocard

APPENDIX A

GLOSSARY OF VME BUS TERMS

Axx	- the symbolic notation for a particular address line on the VME bus, where 'xx' may have the values 01 through 31. These lines are driven by the module currently designated as the VME bus MASTER to selectively address one and only one other module. The module is designated as the addressed SLAVE for the purpose of initiating a data transfer between two modules.
A24	- the VME bus option which identifies a particular module as capable of driving or responding to only 23 address lines. (See the section on SUBSET compatibility for further information.)
A32	- the VME bus option which identifies a particular module as capable of driving or responding to all 31 address lines. (See the section on SUBSET compatibility for further information.)
ARBITER	- the term used to reference the logic circuitry connected to the VME bus at slot 1 to perform the task defined as ARBITRATION. (See ARBITRATION for additional definition. See the chapter on bus arbitration for detailed description.)
ARBITRATION	- the task of assigning control of the data transfer bus on a priority basis to a requester. (For detailed material, see the chapter on bus arbitration.)
AS*	- the address strobe line of the VME bus. When a MASTER has assumed control of the bus and has driven this line low, it is a signal to all SLAVES that the address bus is now valid and that a data transfer is initiated. The last act of a given MASTER in a data transfer cycle must be to release this line to the TRI-STATE condition, so that all units may know that this data transfer is over and that, if the 'bus busy' line is also released, the DTB is now available to be reassigned to the next MASTER. For detailed information on timing considerations, see the timing section of the Data Transfer Bus chapter.
BGxIN*/ BGxOUT*	- the symbolic notation used for a particular bus grant line on the VME bus, where 'x' may have the values 0 through 3. These lines are used to grant a particular level of data transfer bus access from the bus ARBITER, and represent a set of lines on the bus which are not bused, but daisy-chained. In particular, this nomenclature refers to the signal being forwarded to the next module in the chain. If this module does not use the particular level in question, the path from BGxIN* to BGxOUT* will probably be a jumper; but if this board uses the level, the signal will be passed through an AND gate, so that if the board receives the signal, and has a request pending, it may inhibit the signal from continuing down the bus and triggering activity on two MASTERS simultaneously. Not only does this methodology provide a means for having more MASTERS than levels of bus request, but it provides a secondary level of prioritization within a given level of bus request, which is physically determined by proximity to slot 1. (See also the chapter on bus arbitration.)

BRx*	- the symbolic notation used for a particular bus request line on the VME bus, where 'x' may have the values 0 through 3. These lines are used to request control of data transfer bus access from the bus ARBITER.
Dxx*	- the symbolic notation for a particular data line on the VME bus, where 'xx' may have the values 00 through 31. These lines are used by one module to selectively transfer data to one and only one other module.
DTB	- an acronym for DATA TRANSFER BUS. This is the particular subset of VME bus lines involved in a data transfer, consisting of the address lines, the data lines, and the lines WRITE*, LWORD*, AS*, DS0*, DS1*, DTACK*, and BERR*.
LWORD*	- the signal on the VME bus used to invoke 32-bit data transfers. (See also LONGWORD.)
LONGWORD	- a data transfer operation involving 32 bits of transferred data, which is invoked by a MASTER module driving the signal LWORD* to the low state. Note that only modules classified as having option D32 can be expected to transfer long words. (See the section on SUBSET compatibility for further material.)
MASTER	- a module capable of requesting control of the VME data transfer bus via its associated REQUESTER. Upon being signaled by its REQUESTER that the data transfer bus has been granted, the MASTER is capable of addressing another module by driving the address lines and sending data to, or receiving data from, the module so addressed.
READ	- a data transfer initiated by a MASTER, with the data flow from SLAVE to MASTER.
SLAVE	- a module capable of decoding the address lines of the VME bus, and properly responding to a MASTER by accepting or rejecting data transfers via the DTACK* and BERR* response lines when the address presented matches one recognized by this SLAVE as within its range.
System MASTER	- a designation for that MASTER which has the responsibility for saving and restoring data at system power up, system power down, and other emergency handling.
WRITE	- a data transfer initiated by a MASTER, with the data flow from MASTER to SLAVE.

APPENDIX B

VME BUS CONNECTOR/PIN DESCRIPTION

INTRODUCTION

This appendix describes the VME bus pin connections. The following table identifies the VME bus signals by signal mnemonic, connector and pin number, and signal characteristic.

VME Bus Signal Identification

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
ACFAIL*	1B: 3	AC FAILURE - Open-collector driven signal which indicates that the AC input to the power supply is no longer being provided or that the required input voltage levels are not being met.
IACKIN*	1A: 21	INTERRUPT ACKNOWLEDGE IN - Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKIN* signal indicates to the Eurocard that an acknowledge cycle is in progress.
IACKOUT*	1A: 22	INTERRUPT ACKNOWLEDGE OUT - Totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chained acknowledge. The IACKOUT* signal indicates to the next board that an acknowledge cycle is in progress.
AM0-AM5	1A: 23 1B: 16,17, 18,19 1C: 14	ADDRESS MODIFIER (bits 0-5) - Three-state driven lines that provide additional information about the address bus, such as size, cycle type, and/or DTB master identification.
AS*	1A: 18	ADDRESS STROBE - Three-state driven signal that indicates a valid address is on the address bus.
A01-A23	1A: 24-30 1C: 15-30	ADDRESS bus (bits 1-23) - Three-state driven address lines that specify a memory address.
A24-A31	2B: 4-11	ADDRESS bus (bits 24-31) - Three-state driven bus expansion address lines.
BBSY*	1B: 1	BUS BUSY - Open-collector driven signal generated by the current DTB master to indicate that it is using the bus.
BCLR*	1B: 2	BUS CLEAR - Totem-pole driven signal generated by the bus arbitrator to request release by the current DTB master in the event that a higher level is requesting the bus.

VME Bus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
BERR*	1C: 11	BUS ERROR - Open-collector driven signal generated by a slave. This signal indicates that an unrecoverable error has occurred and the bus cycle must be aborted.
BG0IN*-BG3IN*	1B: 4,6, 8,10	BUS GRANT (0-3) IN - Totem-pole driven signals generated by the Arbiter or Requesters. Bus grant in and out signals form a daisy-chained bus grant. The bus grant in signal indicates to this board that it may become the next bus master.
BG0OUT*-BG3OUT*	1B: 5,7, 9,11	BUS GRANT (0-3) OUT - Totem-pole driven signals generated by Requesters. Bus grant in and out signals form a daisy-chained bus grant. The bus grant out signal indicates to the next board that it may become the next bus master.
BR0*-BR3*	1B: 12-15	BUS REQUEST (0-3) - Open-collector driven signals generated by Requesters. These signals indicate that a DTB master in the daisy-chain requires access to the bus.
DS0*	1A: 13	DATA STROBE 0 - Three-state driven signal that indicates during byte and word transfers that a data transfer will occur on data bus lines (D00-D07).
DS1*	1A: 12	DATA STROBE 1 - Three-state driven signal that indicates during byte and word transfers that a data transfer will occur on data bus lines (D08-D15).
DTACK*	1A: 16	DATA TRANSFER ACKNOWLEDGE - Open-collector driven signal generated by a DTB 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.
D00-D15	1A: 1-8 1C: 1-8	DATA BUS (bits 0-15) - Three-state driven bidirectional data lines that provide a data path between the DTB master and slave.
D16-D31	2B: 14-21 2B: 23-30	DATA BUS (bits 16-31) - Three-state driven bi-directional lines for data bus expansion.
GND	1A: 11,15, 17,19 1B: 20,23 1C: 9 2B: 3,12, 22,31	GROUND

VME Bus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
IACK*	1A: 20	INTERRUPT ACKNOWLEDGE - Signal from any MASTER processing an interrupt request. Routed via backplane to Slot 1, where it is looped back to become Slot 1 IACKIN* to start the interrupt acknowledge daisy-chain.
IRQ1*-IRQ7*	1B: 24-30	INTERRUPT REQUEST (1-7) - Open-collector driven signals, generated by an interrupter, which carry prioritized interrupt requests. Level seven is the highest priority.
LWORD*	1C: 13	LONGWORD - Three-state driven signal to indicate that the current transfer is a 32-bit transfer.
[RESERVED]	2B: 3	RESERVED - Signal line reserved for future VME bus enhancements. This line must not be used.
SERCLK	1B: 21	A reserved signal which will be used as the clock for a serial communication bus protocol which is still being finalized.
SERDAT	1B: 22	A reserved signal which will be used as the transmission line for serial communication bus messages.
SYSCLK	1A: 10	SYSTEM CLOCK - A constant 16-MHz clock signal that is independent of processor speed or timing. This signal is used for general system timing use.
SYSFAIL*	1C: 10	SYSTEM FAIL - Open-collector driven signal that indicates that a failure has occurred in the system. This signal may be generated by any module on the VME bus.
SYSRESET*	1C: 12	SYSTEM RESET - Open-collector driven signal which, when low, will cause the system to be reset.
WRITE*	1A: 14	WRITE - Three-state driven signal that specifies the data transfer cycle in progress to be either read or write. A high level indicates a read operation; a low level indicates a write operation.

VME Bus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
+5V STDBY	1B: 31	+5 Vdc STANDBY - This line supplies +5 Vdc to devices requiring battery backup.
+5V	1A: 32 1B: 32 1C: 32 2B: 1,13,32	+5 Vdc Power - Used by system logic circuits.
+12V	1C: 31	+12 Vdc Power - Used by system logic circuits.
-12V	1A: 31	-12 Vdc Power - Used by system logic circuits.

APPENDIX C

VME BUS BACKPLANE CONNECTORS AND EUROCARD CONNECTORS

INTRODUCTION

This appendix identifies the VME bus backplane J1/P1 connector pin assignments. The following table lists the pin assignments by pin number order. (The connector consists of three rows of pins labeled rows A, B, and C.)

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	BG2OUT*	GND
10	SYSCLK	BG3IN*	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	AM4	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	+5V STDBY	+12V
32	+5V	+5V	+5V

NOTE:

(1) SERCLK and SERDAT represent provision for a special serial communication bus protocol still being finalized.

APPENDIX D

VME BUS BACKPLANE CONNECTORS AND EUROCARD CONNECTORS (OPTIONAL EXPANSION)

INTRODUCTION

This appendix identifies the optional expanded VME bus backplane J2/P2 connector pin assignments for the DIN 41612 96-pin connector required for expanded bus systems. The following table lists the pin assignments by pin number order. (The connector consists of three rows of pins labeled rows A, B, and C.)

J2/P2 Pin Assignments

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	User I/O	+5 Volts	User I/O
2	User I/O	GND	User I/O
3	User I/O	RESERVED	User I/O
4	User I/O	A24	User I/O
5	User I/O	A25	User I/O
6	User I/O	A26	User I/O
7	User I/O	A27	User I/O
8	User I/O	A28	User I/O
9	User I/O	A29	User I/O
10	User I/O	A30	User I/O
11	User I/O	A31	User I/O
12	User I/O	GND	User I/O
13	User I/O	+5 Volts	User I/O
14	User I/O	D16	User I/O
15	User I/O	D17	User I/O
16	User I/O	D18	User I/O
17	User I/O	D19	User I/O
18	User I/O	D20	User I/O
19	User I/O	D21	User I/O
20	User I/O	D22	User I/O
21	User I/O	D23	User I/O
22	User I/O	GND	User I/O
23	User I/O	D24	User I/O
24	User I/O	D25	User I/O
25	User I/O	D26	User I/O
26	User I/O	D27	User I/O
27	User I/O	D28	User I/O
28	User I/O	D29	User I/O
29	User I/O	D30	User I/O
30	User I/O	D31	User I/O
31	User I/O	GND	User I/O
32	User I/O	+5 Volts	User I/O

APPENDIX E

DC SIGNAL SPECIFICATION

This appendix provides a summary showing which signal lines on the VME bus are driven/received by each functional module, and the type of driver each uses.

In order to simplify the table, 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 (low current)	- TP LC
Three-state	- THREE
Open-collector	- OC

For the driver specifications, see Table 7-2 in Chapter 7.

All functional modules use the same type of receiver. (For the receiver specifications, see Table 7-3 in Chapter 7.)

BUS DRIVER AND RECEIVER SUMMARY

SIGNAL MNEMONIC	SIGNAL NAME	DRIVER		RECEIVER MODULE	TERMINATION NETWORK
		TYPE	MODULE		
A01-A31 (31 lines)	ADDRESS BUS				
AM0*-AM5* (6 lines)	ADDRESS MODIFIER				
AS*	ADDRESS STROBE	THREE	MASTERS, INTERRUPT HANDLERS	SLAVES, INTERRUPTERS	YES
WRITE*	WRITE				
DS0*-DS1* (2 lines)	DATA STROBES				
LWORD*	LONGWORD				
D00-D31 (32 lines)	DATA BUS	THREE	MASTERS, SLAVES, INTERRUPTERS	SLAVES, MASTERS, INTERRUPT HANDLERS	YES
DTACK*	DATA TRANSFER ACKNOWEDGE	OC	SLAVES, INTERRUPTERS	MASTERS, INTERRUPT HANDLERS	YES
BERR*	BUS ERROR	OC	SLAVES	MASTERS	YES

BUS DRIVER AND RECEIVER SUMMARY (cont'd)

SIGNAL MNEMONIC	SIGNAL NAME	DRIVER TYPE	DRIVER MODULE	RECEIVER MODULE	TERMINATION NETWORK
BR0*-BR3* (4 lines)	BUS REQUEST	OC	REQUESTERS	ARBITER	YES
BG0IN*-BG3IN* (4 lines)	BUS GRANT	TP LC	ARBITER	REQUESTERS	NO
BG0OUT*-BG3OUT* (4 lines)			REQUESTERS	(NOT APPLICABLE)	NO
BBSY*	BUS BUSY	OC	REQUESTER	ARBITER	YES
BCLR*	BUS CLEAR	TP HC	ARBITER	MASTERS	YES
IRQ1*-IRQ7* (7 lines)	INTERRUPT REQUEST	OC	INTERRUPTERS	INTERRUPT HANDLERS	YES
IACKIN*/IACKOUT*	INTERRUPT ACKNOWLEDGE DAISY CHAIN	TP LC	INTERRUPT HANDLERS, INTERRUPTERS	INTERRUPTERS INTERRUPTERS	NO
IACK*	INTERRUPT ACKNOWLEDGE	OC or THREE		SLOT 1	NO
SYSRESET*	SYSTEM RESET	OC	POWER MONITOR, MANUAL SWITCH, SYSTEM CONTROLLER	ANY ANY	YES
ACFAIL*	AC FAILURE	OC	POWER MONITOR	MASTERS	YES
SYSCLK	SYSTEM CLOCK	TP HC	CLOCK DRIVER	ANY	YES
SYSFAIL*	SYSTEM FAIL	OC	MASTERS, SLAVES	ANY ANY	YES

MOTOROLA *Semiconductor Products Inc.*

BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

14826 PRINTED IN USA 9-81 IMPERIAL LITHO 99127 3500