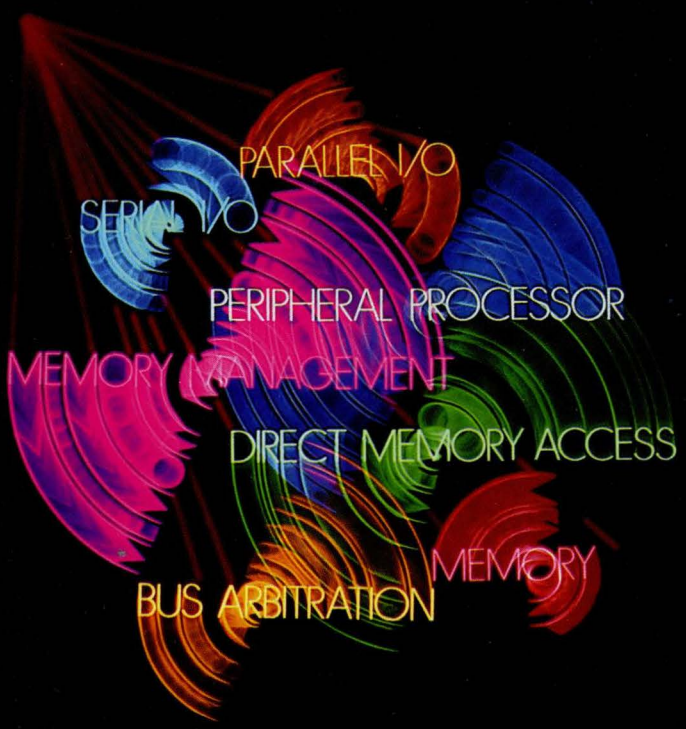
M68KVBS(D4)
MICROSYSTEMS



VERSAbus

Specification Manual

MOTOROLA

M68KVBS(D4)

JULY 1981

VERSAbus

SPECIFICATION MANUAL

Fourth Edition

Third Edition May 1981

PREFACE

WHY A NEW BUS?

The advent of microprocessors in the 1970's has had a revolutionary effect on the thinking of computer system architects. Early system architecture was designed to get the maximum use of a central processing unit. Every effort was made to assure that no expensive processing time was lost. A careful blend of multitasking and batch processing was adopted to accomplish this objective.

The new microprocessor chips, with their lower cost per processing unit, have now caused a significant shift in this thinking. The most expensive portions of a system today are the peripherals and software. Thus, multiprocessor system architectures are now a viable alternative in constructing powerful computer systems.

As a result, in the late 1970's, system architects began to develop computer system architectures which made use of multiple processors. Difficulties in designing these complex systems made mandatory a clearly defined system bus - defined independently of the processors involved. A processor-independent bus also reduces peripheral costs by allowing manufacturers to provide a single type of bus interface.

As new products became available, it was seen that the earlier bus architecture did not have sufficient flexibility to meet the needs of the bigger and more complex systems that systems architects were envisioning. Several major problem areas started to appear.

0.1 Bus Arbitration

Most existing system buses are designed around a single controlling microprocessor. Buses of this type do not easily expand to multiprocessor configurations because they lack a method of bus arbitration to allow multiple processors to access the bus on an interleaved basis.

0.2 Synchronization

Most existing system buses are synchronous. When multiple processors are running on a bus (as is typical with the new multiprocessor architectures), they must be resynchronized with the bus clock each time a bus access is required. This can become very complex, particularly if the processors involved are of different types and/or run at different clock rates. Designs based on synchronous bus structures become more and more costly as the system complexity grows.

Conversely, an asynchronous bus allows the various processors to access the bus without a need to synchronize to a common clock. As newer, faster devices become available, they may be added to the system without necessitating modifications on the existing units.

0.3 Addressability

As hardware costs plummet, while the cost of rotating media storage devices remains relatively fixed, a combination of speed and cost constraints encourages the development of larger and larger memory configurations on systems. Current microprocessor buses tend to restrict addressing to 16 bits. Since all newly announced microprocessors have addressing capability in excess of this limitation, these buses are clearly insufficient for the task at hand.

0.4 User/System Protection

As multi-user systems become increasingly more available in microprocessor systems, the hazards of inadvertent access to and/or destruction of the data intended for other users and the system executive become greater. Because of the growing size of these systems, three concerns come into effect:

a. The sheer size of the system may require that limited subsets be put into operation before the total system is complete.

b. The ability to minimize problems caused by the interaction of non-related pieces in a "live" system becomes a necessity because of the geometrically expanding number of such interactions. The time and equipment required to exhaustively test all possible combinations of conditions will rapidly exceed the available resources of any organization.

c. Modifications required by changing needs, and enhancements made possible by new technologies, need to be added after the system has already been "live" for a significant period.

   Because of the "evolutionary" nature of systems development, the software and hardware in the system will always have many different degrees of reliability. Provision must be made to ensure that the newly grafted pieces of the system are not allowed to access system resources in ways which might prove detrimental to existing applications and users.

Thus, a method of "privileged access" to system resources must be developed to preclude access to system resources except through previously verified drivers. The bus design must make provisions to detect, inhibit, and report unauthorized access.

0.5 Data Capabilities

As microprocessors grow in capability, one of the major areas of growth is in the number of bits handled concurrently. An enlargement of the data path width multiplies the effective throughput on the bus. As processors are able to perform operations on wider and wider pieces of data, many multi-precision algorithms may be eliminated from the software package. Since these multi-precision operations account for a significant portion of the processing time, this also increases throughput.

As special processors which actually run in a "high level" language mode become available, larger and larger instructions are required. The new processors will require the ability to fetch these wider instructions. The most common bus width of eight bits is totally inadequate for these needs, and even a data width of 16 bits is not adequate for all upward growth needs.

0.6 Malfunction Control

As system complexities grow, the mean time to failure for the system as a whole decreases. Therefore, it becomes essential to provide for system self diagnosis, and to provide the basics for allowing fault isolation and continued operation. Several areas need to be covered in this regard.

A power up self-test of each unit is a necessary ingredient of any fault management package. In a multiple processor system, such a test would be run concurrently by each of the processors. A procedure must be defined to allow each processor to "veto" system start-up until malfunctions are corrected or bypassed. Current buses, designed for single processor operation, have no "ballot box" to allow multiple processors to control start-up.

Once the system is operational, additional procedures are required to detect run-time failures at the earliest possible moment. Early detection is one of the keys to successful isolation. Few buses today incorporate malfunction signal lines or parity check lines.

0.7 Board Size Limitations

To maximize bus availability, it is necessary to design processor boards that are basically self-contained. Self-diagnostics are practical only when the processor board itself contains sufficient ROM and RAM to allow the processor to complete its self-test without requiring the bus. Each processor board must have its own timing logic to allow interfacing to the system bus. Likewise, each processor board must contain the logic necessary to interface properly with the interrupt and bus arbitration bus controls. Even with the increasingly high-density packages coming out of development, PC board area is at a premium. Many present bus standards place over-restrictive limitations on the area possible on any one board. Various design solutions are possible, but each has its costs. A better solution is a bus which allows larger cards.

0.8 Sequential Access Capability

While accessing program storage over a bus, processors exhibit two characteristics:

a. Fetches do not take place on every cycle (i.e., a few fetches may be followed by a period where no fetches take place).

b. Fetches typically take place from sequential memory locations, except when branches are taken.

In the past, the solution has been to add high speed cache memories between the bus memory and the processor. In order to take full advantage of this configuration, the bus memory should have an on-board "address counter" which increments or decrements on each access. The bus must then provide a means for the processor to inform the memory when this counter is to be used and when it should increment. To date, no microprocessor bus provides for this.

0.9 Conclusions

Because of these limitations, and after careful analysis, Motorola felt that a new bus was needed. Such a bus should eliminate or at least minimize the difficulties listed above. It should allow for maximum potential growth in those areas where trends are already established. Not all systems will require every enhancement being provided - but a bus which does not provide them guarantees its early obsolescence.

The VERSAbus, as developed by Motorola, addresses the limitations of existing bus structures and meets the needs of state-of-the-art microprocessor systems. It has been designed with special attention to the following objectives:

- To provide a comprehensive basis for microprocessor systems capable of supporting a wide range of architectures from 8- to 32-bit data paths with up to 5-MHz data transfer rates.
- To provide adequate addressing range and control for large-scale systems.
- To provide for system architectures involving multiple processors.
- To provide sufficient flexibility to exploit the latest technologies without sacrificing ease of use to the designers of future microprocessor-based systems.

VERSAbus has already been implemented in development systems and board-level products. A major advantage to users is the continuity possible from initial development through board products to end-user systems. The bus versatility allows it to serve present products and the future needs of next generation products.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

TABLE OF CONTENTS (cont'd)

Page

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont'd)

LIST OF TABLES

CHAPTER 1

INTRODUCTION TO THE VERSAbus SPECIFICATION

CHAPTER 1

INTRODUCTION TO THE VERSAbus SPECIFICATION

1.1 VERSAbus SPECIFICATION OBJECTIVES

The VERSAbus 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 VERSAbus without disturbing the internal activities of other devices interfaced to VERSAbus.

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

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

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

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 communication speed is primarily device limited - not system interface limited.

1.2 VERSAbus 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 edge 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, but one module should never be spread over multiple cards.

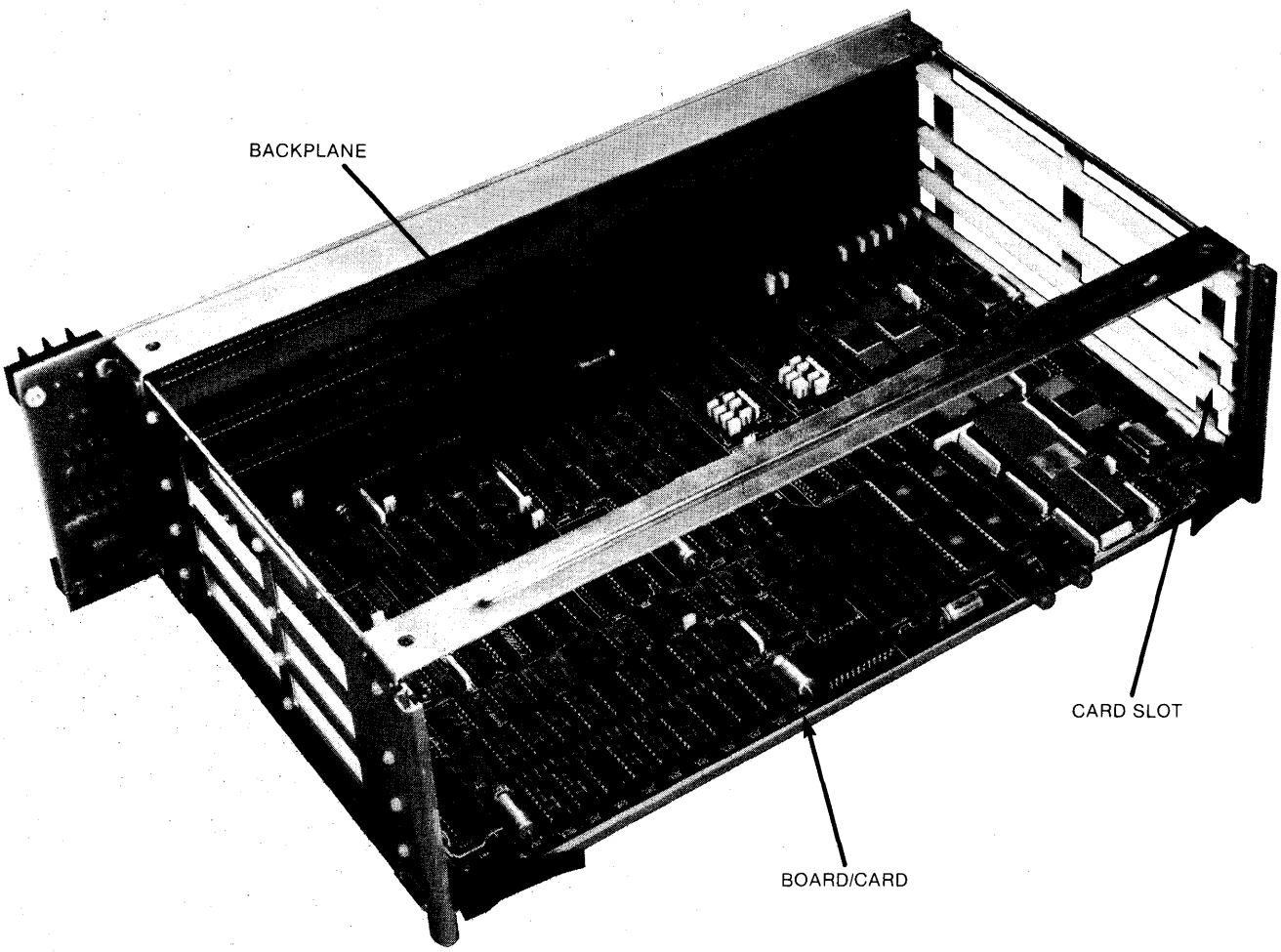


FIGURE 1-1. Typical Multi-Slot System Card Cage with Backplane

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, INTERRUPT HANDLER, and (optionally) an INTERRUPTER, which function together and which must be on the same card.

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 VERSAbus. There will always be one and only one CONTROLLER SUB-SYSTEM. It can contain the following functional modules:

a. Data Transfer Bus ARBITER
b. Emergency Data Transfer Bus REQUESTER
c. Power up/power down MASTER
d. System clock driver
e. System reset driver
f. System test controller
g. Power monitor (for AC clock and AC fail driver)

In any VERSAbus 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 through d on the board residing in this slot. System reset and the system test controller are 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 VERSAbus Structure

The VERSAbus 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 1-2 shows the elements of a typical VERSAbus system. The functional modules communicate with one another by means of bus signal lines provided by a backplane. By defining the module's operational characteristics, the VERSAbus specification defines the design of the bus interface portion of each card to assure reliable system operation.

NOTE

> 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 VERSAbus in the manner described, but uses different on-board signals.

The interface functions of the VERSAbus 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 1-3 illustrates the individual functional modules and buses contained within the VERSAbus 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

   When a VERSAbus system is configured with more than one DTB MASTER, a means must be provided to transfer control of the DTB between these MASTERS in an orderly manner and to guarantee that only one MASTER controls the DTB at a given time. Bus arbitration is that area of VERSAbus whose signals (Arbitration Bus) and modules (DTB REQUESTERS and DTB ARBITER) provide that means.

c. Priority Interrupt

   The priority interrupt capability of VERSAbus 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 general categories of system clocks, initialization, and diagnostics have been grouped into the area of utilities. These functions include clock lines, system reset, system test, etc.

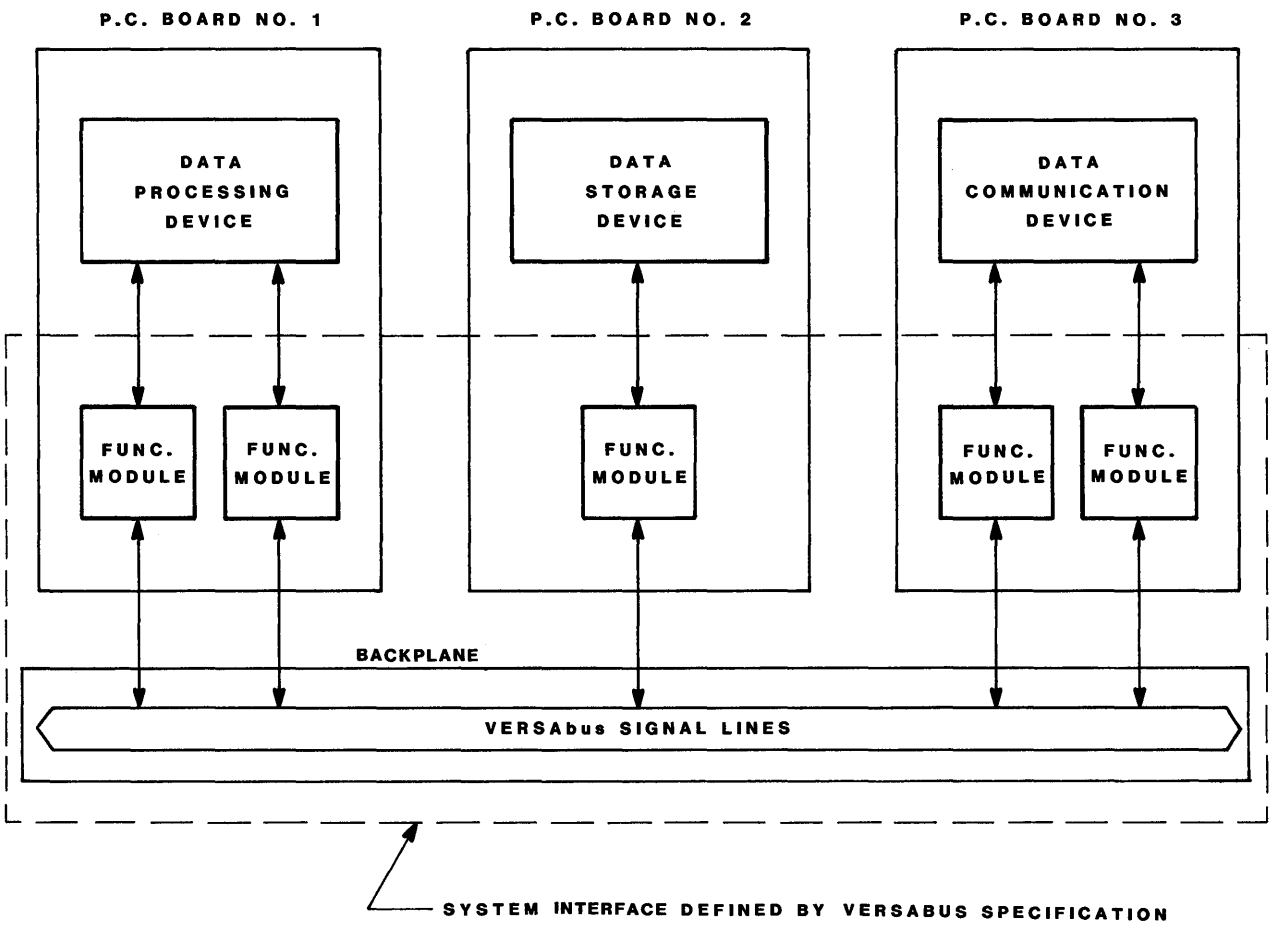


FIGURE 1-2. System Elements Defined by the VERSAbus Specification

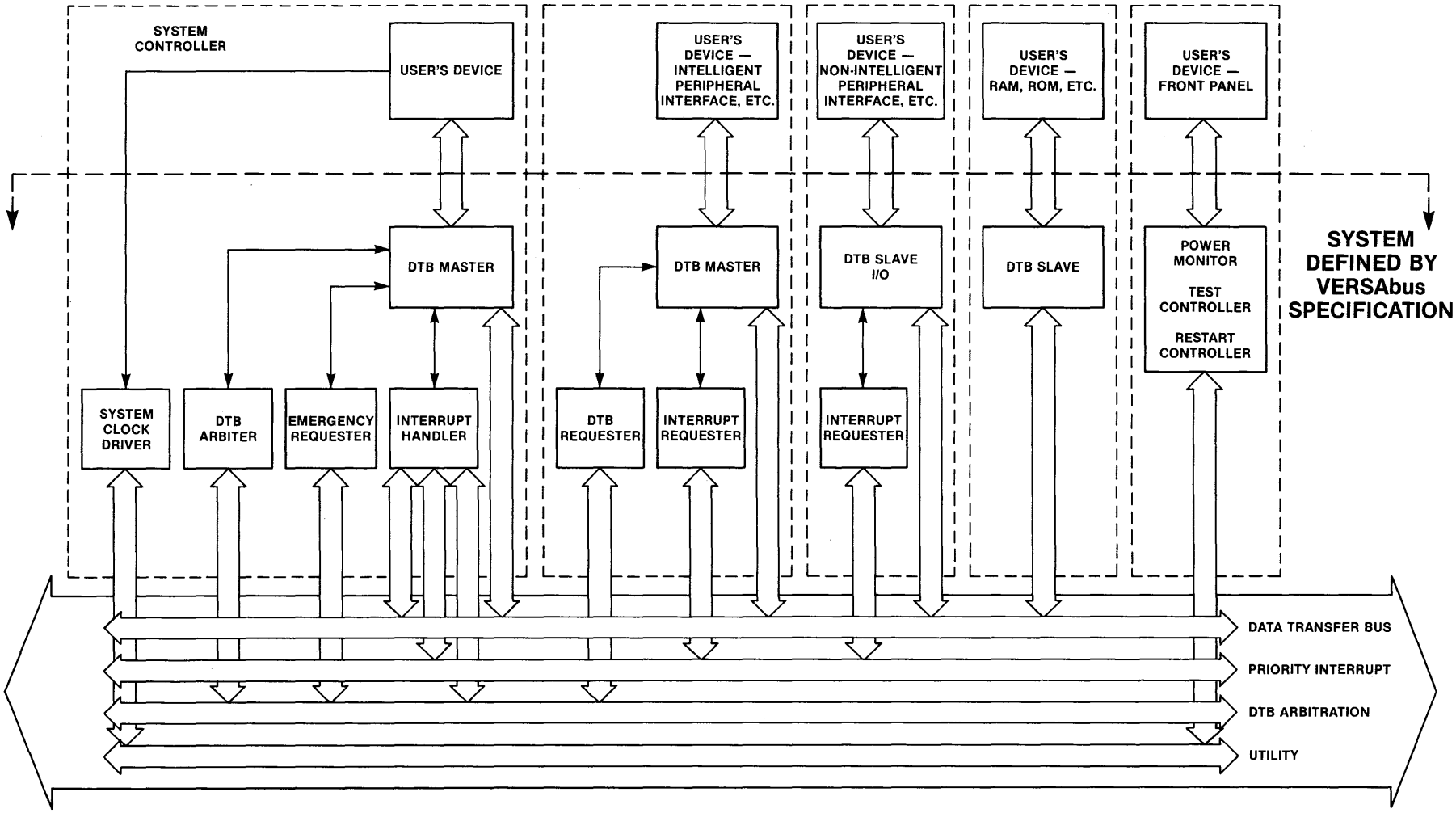


FIGURE 1-3. Functional Modules and Buses contained within the VERSAbus Definition

1.3 VERSAbus SPECIFICATION FORMAT

As aids to defining or describing VERSAbus 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 VERSAbus operation. The events are stated in words and result from interaction of two or more functional modules. The flow diagram describes VERSAbus operations in a sequential manner and, at the same time, shows interaction of the functional modules.

d. State diagram - shows all possible allowed states for a functional module. Also presented are conditions for changing states and all allowed paths between states. The state diagram is the most rigorous definition of a functional module. Appendix B defines state diagram notation and usage.

Additional chapters include electrical specifications, mechanical specifications, and VERSAbus 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 VERSAbus 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 levels or in transition between these levels. Whenever the term "high" is used, it refers to a high TTL voltage level ($\geq$ + 2.0 V). The term "low" 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 "edges". A rising edge is defined as the time period during which a signal line makes its transition from a low level to a high level. The falling edge 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, single line 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.

Because there are several types of signal line functions, the asterisk is defined more exactly in following subparagraphs. The key idea is to associate the low level or falling edge with use of the signal line.

NOTE

The asterisk is inappropriate for asynchronously running clock lines (i.e., ACCLK or SYSCLOCK). There are no fixed phase relationships between these clock lines and other VERSAbus timing.

1.4.3 Summary for Signal Line Terminology

Table 1-1 summarizes the terminology associated with driving and sensing signal line conditions. The terminology reflects the actual conditions on the signal lines and shows how the protocol is dependent on levels or transitions. Basically,

a. If a signal line event is signified by a transition, a module is said to drive the line to high or to low, and a detecting module is said to receive this condition.

b. If a signal line event or condition is signified by a level, a module is said to drive the line high or low, and a detecting module is said to receive this condition.

c. Open collector shared lines are treated as a special case because a module can drive a line low, but cannot drive it high (another module can be driving the line low). On a shared line, a module is said to hold a line low and then release the line. A detecting module is said to detect a high or low level.

The signal line categories shown in Table 1-1 are treated in detail in the following subparagraphs. The terminology is discussed and the asterisk usage is also explained. Whenever a signal line is taken to a level (during a protocol discussion), it is assumed to remain at that level until stated otherwise.

TABLE 1-1. VERSAbus Signal Line Terminology

SIGNAL LINE CATEGORY	SIGNAL LINE MNEMONICS (1)	TERMINOLOGY		
		OUTPUT	INPUT	ASTERISK
BUS LINES (Three-state)	A01*-A31* D00*-D31* APARITY0*-APARITY1* DPARITY0*-DPARITY3* AM0*-AM7* TEST0*-TEST1* WRITE*, LWORD*	Drive X Drive X high Drive X low Place valid X Remove X Release X	 Receive X driven high Receive X driven low Receive X	Indicates a low level Equals a logic 1
Drive X defines the point at which the three-state drivers are enabled. Place valid X defines the point at which the levels on the bus are valid. Remove X defines the point at which the levels on the bus are invalid. Release X defines the point at which the three-state drivers are no longer enabled.				
STROBE LINES (Three-state)	AS* DS0* DS1*	Drive X to low Drive X to high	Receive X driven to low Receive X driven to high	Indicates the information on the strobed bus is valid on the falling edge of the strobe line.
STROBE RESPONSE LINES (Open Collector)	DTACK* BERR*	Drive X to low Release X to high	Receive X driven to low Receive X high	Indicates the strobe response is valid on the falling edge of the signal line.
SHARED LINES (Open Collector)	IRQ1*-IRQ7* BR0*-BR4* SYSFAIL*	Hold X low Release X	Detect X low Detect X high (only if no drivers holding line low)	Indicates this line is activated in the low state.
(1) For other signal lines, see subparagraphs 1.4.3 and 1.4.8.				

1.4.4 Bus Lines (Three-State Level Significant)

The following VERSAbus lines are members of this category:

A01*-A31*	Address Sub-bus
D00*-D31*	Data Sub-bus
AM0*-AM7*	Address Modifier Sub-bus
TEST0*-TEST1*	System Test Lines
APARITY0*-APARITY1*	Address Parity Lines
DPARITY0*-DPARITY3*	Data Parity Lines
WRITE*	Data Transfer Control Line
LWORD*	Longword Transfer Control Line

These lines are all driven by three-state drivers; transitions on them carry no useful timing information. Their levels are important because they carry binary information (Low = 1, High = 0). Different functional modules may drive these lines at different times, although only one module can drive them at any one time. Before a module may use the bus lines, it must "turn on" its drivers and, when finished, the module must "turn off" its drivers.

BUS TERMINOLOGY

In the following statements, "X" can represent any of the above sets of VERSAbus lines (Address, Data, Address modifier, Test code, Address parity, Data parity).

a. When a module on VERSAbus presents binary information to the X bus, the module turns on its X bus drivers and it is said to drive the X bus or present the X. For individual lines, a module will drive X low or drive X high. Levels of all bus lines are assumed to remain constant until the module removes the X from the bus. When the module turns off its X drivers, it releases the X bus.

b. When another module inputs binary information from the X bus, it is said to receive the X. For individual lines, it may receive X driven low or receive X driven high.

c. For most of these lines, the asterisk suffix indicates that a low level on the signal line equals a logic one. A high level, in turn, is a logic zero. WRITE* and LWORD*, however, have dual meanings. A high level on WRITE* indicates a read operation; a low indicates a write operation. A high level on LWORD* indicates an 8- or 16-bit transfer; a low indicates a 32-bit transfer. For these two lines, the asterisk indicates that the named function (write operation and longword operation) occurs when the signal line is at a low level.

1.4.5 Strobe Lines (Three-State Edge Significant)

In order to read binary information on bus lines, timing information must be provided indicating when the level significant lines may be read. Also, timing information must be provided indicating when a driving module has released the bus. Three strobe lines provide the required timing information. They are named:

AS*	Address Strobe
DS0*	Data Byte Zero Strobe
DS1*	Data Byte One Strobe

The strobe lines provide timing information on their rising and falling edges (transitions), but they provide no information on their levels. Each strobe is associated with a specific set of bus lines. The strobe falling edge indicates when the bus lines should be read; the strobe rising edge indicates when the bus lines are released.

In all cases, the strobe falling edge of AS* indicates when the address line should be read. The rising edge of AS* indicates that the data transfer is complete and that the MASTER has released the address lines.

In the case where WRITE* is low, the falling edge of DS0* and/or DS1* indicates that the corresponding data lines should be read by the addressed SLAVE. The rising edge indicates that the transfer is complete.

In the case where WRITE* is high, the falling edge of DS0* and/or DS1* authorizes the SLAVE to drive the corresponding data lines. The rising edge indicates that the MASTER has received the SLAVE'S response and that the SLAVE should release the data lines.

Similar to the bus lines, strobe lines are driven by three-state drivers. Different functional modules may drive them at different times. Because only one module can drive bus lines and strobes at any one time, some method must be provided to indicate when they are released. On VERSAbus, each strobe line is terminated with a resistor network that maintains a high level when the strobe line is released. The strobe lines provide a high level indication whenever they are not being driven.

When control of a strobe line is passed from one module to another, the next driving module is required to wait a specified period after receiving the strobe line high before driving it to the low level. Therefore, the previous driver is permitted to drive the strobe line to the high level before releasing it, but this is not required.

STROBE LINE TERMINOLOGY

In the following statements, "X" can represent AS*, DS0*, or DS1*.

a. When a VERSAbus module turns on its X driver, it is said to drive X. When it causes a falling edge to occur on X, the module is said to drive X to low. When the module causes a rising edge to occur on X, it is said to drive X to high. When the driver turns off, it releases X.

b. When another module senses a falling edge on X, it is said to receive X driven to low. When the module senses a rising edge on X, it is said to receive X driven to high.

c. The asterisk suffix indicates that the binary information on the address lines and the binary information on the data lines (if a write operation) is valid on the falling edge of the strobe line. It also indicates that on a read operation, the falling edges of DS0* and DS1* request the SLAVE to present its data on the data lines.

1.4.6 Strobe Response Lines (Open Collector)

The VERSAbus lines forming this category are:

DTACK*	Data Acknowledge
BERR*	Bus Error

Similar to strobe lines, strobe response lines provide timing information on their transitions but provide no timing information on their levels. Unlike the strobe lines, they are driven by open collector devices instead of three-state devices. The terminology reflects this difference in describing transitions on these lines. Also, open collector lines depend on the resistor terminations to establish a high level.

STROBE RESPONSE LINE TERMINOLOGY

In the following statements, "X" can represent DTACK* or BERR*.

a. When a VERSAbus module causes a falling edge to occur on X, it is said to drive X to low. When the module causes a rising edge to occur on X, it is said to release X to high.

b. When another module senses a falling edge on X, it is said to receive X driven to low. When this module senses a rising edge on X, it is said to receive X high.

c. The asterisk suffix indicates that the strobe response is signified by the falling edge of the strobe response signal line.

For example, the falling edge of DTACK* on a write operation indicates the SLAVE'S acceptance of the data. The falling edge of DTACK* on a read operation is the SLAVE'S signal to the MASTER that the SLAVE is presenting valid data. The rising edge of DTACK* on a read operation indicates that the SLAVE has released the data lines.

1.4.7 Shared Lines (Open Collector)

Although strobe response lines are open collector driven, they are normally driven by only one driver. Shared lines allow multiple drivers to be activated coincidently. These lines are:

IRQ1*-IRQ7*	Interrupt Request
BR0*-BR4*	Bus Request
SYSFAIL*	System Fail

Because driving modules cannot control transitions on these open collector shared lines, edges are not used to carry information. (If one driver releases the line, another driver may still be driving the line low.) Therefore, modules sensing a shared line look for a low level to convey information.

SHARED LINE TERMINOLOGY

In the following statements, "X" can represent one level of interrupt request, one level of bus request, or system failure.

a. When a VERSAbus module turns on its X driver, it is said to hold X low. When the module turns off its X driver, it is said to release X.

NOTE

It should never be assumed that a shared line will go high when a module releases the line. The line will go high only if no other modules are driving the line.

b. When a module senses a low level on X, it is said to detect X low. When a module senses a high level on X, it is said to detect X high.

c. The asterisk suffix indicates that the low state is a request for service on the IRQX* or BRX* lines, and indicates by the low level that a system failure has occurred for the System Fail line.

1.4.8 Other VERSAbus lines

Other signal lines have a wide range of characteristics which place them outside of the categories described in subparagraphs 1.4.4 through 1.4.7. When discussing these signal line functions, they will be described in terms consistent with statements in subparagraph 1.4.3, "Summary for Signal Line Terminology".

1.5 PROTOCOL SPECIFICATION

Each VERSAbus 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 VERSAbus and its on-board data device. A functional module communicates with another module by driving/receiving bus signals, and communicates with its on-board device by driving/receiving on-board signals. The protocol governs these communications by determining:

a. when a module may drive and change the level of bus signals,
b. when a module may change the level of its on-board signals,
c. when and how a module must respond to a bus signal, and
d. when and how a module must respond to an on-board 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 VERSAbus Specification).

Interlocked bus signals are dedicated to coordinating internal functions of the VERSAbus 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 their 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 are 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 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. Broadcast signals from outside the system cause a set of modules to enter a known state and to remain there until the broadcast is terminated.

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 present to gain a broader perspective of VERSAbus functions. Therefore, system examples and explanatory descriptions are provided to give the VERSAbus user an understanding of VERSAbus 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 describes the bus lines which are used by the modules to send or broadcast required bus messages (subparagraph X.2).

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

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, the formal protocol specification for each module is given (subparagraph X.5). The protocol is defined in state diagram and/or timing diagram form with explanatory text. To assure that the user can fully understand the state diagrams used in the specification, Appendix B, State Diagram Notation and Usage, should be read and assimilated.

1.7 ELECTRICAL/MECHANICAL SPECIFICATIONS

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

CHAPTER 2

VERSAbus DATA TRANSFER

CHAPTER 2

VERSAbus DATA TRANSFER

2.1 INTRODUCTION

VERSAbus 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 VERSAbus options selected by the user. There are four sets of DTB related options:

D16 or D32	Data path width
NDP or DP	Data parity
A24 or A32	Address path width
NAP or AP	Address parity

2.1.1 DTB Options - Basic Description

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 backplane.

Option NDP specifies that data parity will not be checked or generated by the module. Option DP specifies that the module will provide even data parity when transmitting, and verify even data parity when receiving.

Option A24 specifies that all addresses generated by the MASTER or decoded by the SLAVE will be restricted to 23 bits. Option A32 selection extends the address range to 31 bits. The address modifier lines indicate to A24 SLAVES whether the address is 23 or 31 bits. Option A32 also requires an expanded bus system backplane.

Option NAP specifies that address parity will not be generated by the MASTER or verified by the SLAVE. Option AP specifies that the MASTER module will generate even address parity and that the addressed SLAVE module will verify proper address parity.

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. As an example, ignore all references to data parity error processing if NDP is chosen as the option.

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

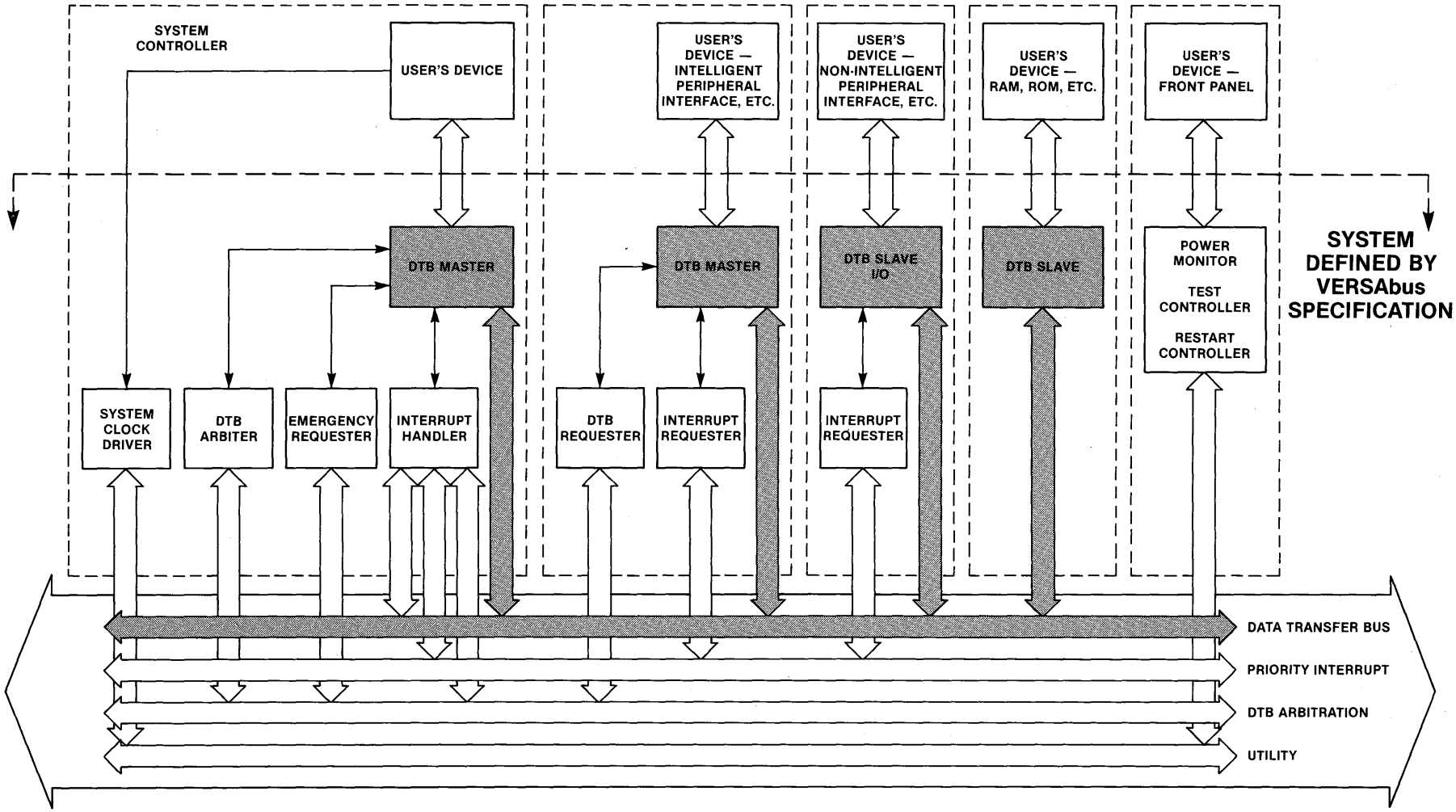


FIGURE 2-1. VERSAbus Data Transfer Functional Block Diagram

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.

If parity is provided, the parity may be verified by the receiving module. (In a write cycle, both address and data parity are verified by the SLAVE. In a read cycle, the SLAVE verifies the address parity and the MASTER verifies the data parity.) Whenever the SLAVE detects a parity error, it responds to the MASTER with an error signal instead of the usual acknowledge signal.

2.2 DATA TRANSFER BUS LINE STRUCTURES

2.2.1 Address Lines

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

23 or 31	address lines	(A01* through A23* if option A24) (A01* through A31* if option A32)
0, 1, or 2	parity lines	(none if option NAP) (APARITY0* if options AP and A24) (APARITY0* and APARITY1* if options AP and A32)
1	address parity valid line (APVAL*)	
8	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 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.

Address parity lines are generated as an even parity adjunct to the address modifier, address, and LWORD* logic lines. Each represents the logical function "exclusive OR" of its associated lines as follows:

Logical Result	Exclusive OR Input
APARITY0*	AM0*-AM7*, A01*-A23*, and LWORD*
APARITY1*	A24*-A31*

Whenever an address parity line is driven by an option AP MASTER, the APVAL* (address parity valid) line must be driven low. This is an indication to an AP SLAVE that address parity should be verified. (NAP SLAVES will not verify the parity regardless of the level of APVAL*).

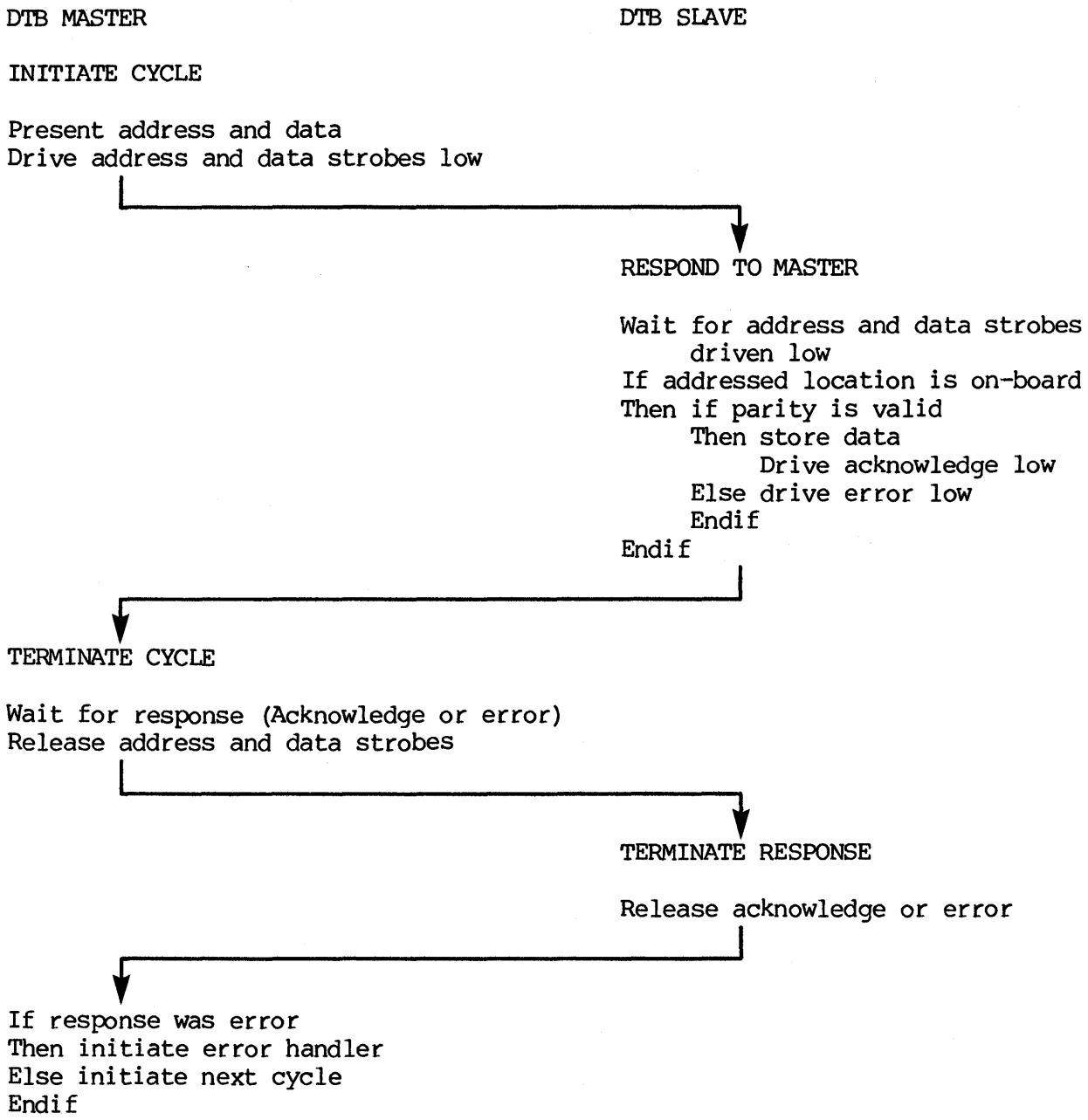


FIGURE 2-2. Typical Write

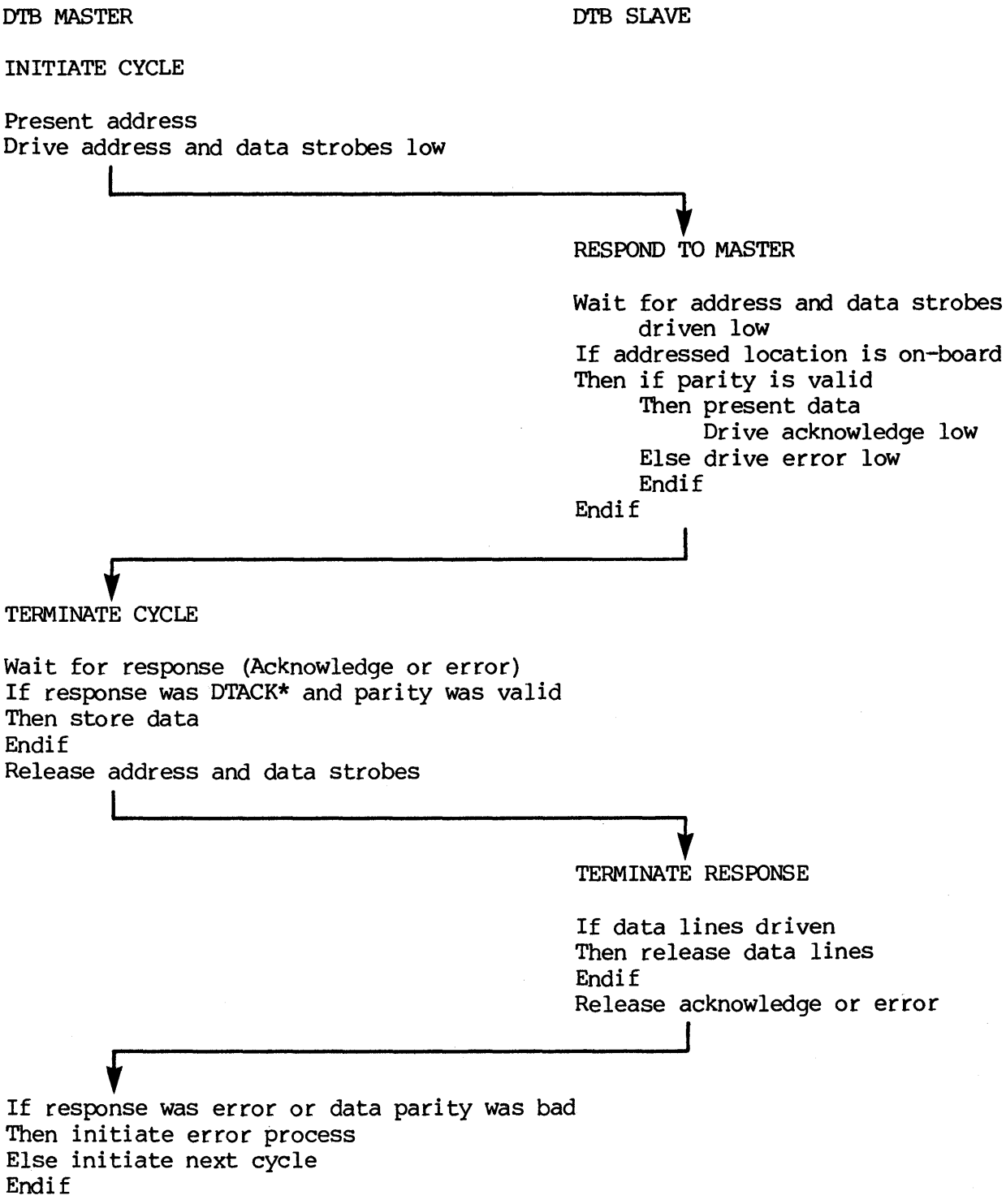


FIGURE 2-3. Typical Read

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

Each SLAVE in the system may be configured (either dynamically or statically) to respond to a single address modifier code. If there are several MASTERS on VERSAbus, 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.

CYCLE TYPE

The AM codes can be used to specify a special type of transfer cycle. VERSAbus specifies two special cycle types. The user could use additional codes to specify others.

The first special cycle type is an interrupt acknowledge cycle. When an interrupt acknowledge AM code is placed on the bus, it causes all SLAVES to ignore the cycle and allows only INTERRUPTERS to respond.

The second special cycle type is a sequential access cycle. There are 16 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 increment the counter after each odd-byte 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 arranged as a contiguous block, and 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 register 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

VERSAbus 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, VERSAbus 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 (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 256 possible address modifier codes and classifies each into one of three categories:

DEFINED BY:
- VERSAbus Spec.
- USER
- RESERVED

Those defined by the VERSAbus Spec are intended for a specific purpose and may not be used for any other. These include codes for sequentially accessing memory, a code for acknowledging interrupts, 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 VERSAboard 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 an 82S129 bipolar PROM. The PROM inputs require no buffering and will not overload the bus lines. 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 01, 02, 05, and 06 if they decode 23 address lines, and AM codes F1, F2, F5, and F6 if they decode 31 address lines. Short addressed I/O SLAVES should be configured to respond to AM codes 11 and 15.

TABLE 2-1. Address Modifier Codes

HEXA-DECIMAL CODE	ADDESS MODIFER 7*	6*	5*	4*	3*	2*	1*	0*	FUNCTION	DEFINED BY
F8-FF	L	L	L	L	L	X	X	X	Undefined	Reserved
F7	L	L	L	L	H	L	L	L	Undefined	Reserved
F6	L	L	L	L	H	L	L	H	Extended Supervisory Program Access	VERSAbus Spec.
F5	L	L	L	L	H	L	H	L	Extended Supervisory Data Access	VERSAbus Spec.
F4	L	L	L	L	H	L	H	H	Undefined	Reserved
F3	L	L	L	L	H	H	L	L	Undefined	Reserved
F2	L	L	L	L	H	H	L	H	Extended Non-Privileged Program Access	VERSAbus Spec.
F1	L	L	L	L	H	H	H	L	Extended Non-Privileged Data Access	VERSAbus Spec.
F0	L	L	L	L	H	H	H	H	Undefined	Reserved
EX	L	L	L	H	X	X	X	X	Undefined	Reserved
DX	L	L	H	L	X	X	X	X	Undefined	Reserved
CX	L	L	H	H	X	X	X	X	Undefined	Reserved
BX	L	H	L	L	X	X	X	X	Undefined	Reserved
AX	L	H	L	H	X	X	X	X	Undefined	Reserved
9X	L	H	H	L	X	X	X	X	Undefined	Reserved
8X	L	H	H	H	X	X	X	X	Standard Ascending Sequential Access	VERSAbus Spec.
7X	H	L	L	L	X	X	X	X	Undefined	User
6X	H	L	L	H	X	X	X	X	Undefined	User
5X	H	L	H	L	X	X	X	X	Undefined	User
4X	H	L	H	H	X	X	X	X	Undefined	User
3X	H	H	L	L	X	X	X	X	Undefined	User
28-2F	H	H	L	H	L	X	X	X	Undefined	User
27	H	H	L	H	H	L	L	L	Interrupt acknowledge	VERSAbus Spec.
20-26	H	H	L	H	H	X	X	X	Undefined	Reserved
1F	H	H	H	L	L	L	L	L	Undefined	User
1E	H	H	H	L	L	L	L	H	Undefined	User
1D	H	H	H	L	L	L	H	L	Short I/O address	User
1C	H	H	H	L	L	L	H	H	Undefined	User
1B	H	H	H	L	L	H	L	L	Undefined	User
1A	H	H	H	L	L	H	L	H	Undefined	User

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

HEXA-DECIMAL CODE	ADDESS MODIFER 7* 6* 5* 4* 3* 2* 1* 0*	FUNCTION	DEFINED BY
19	H H H L L H H L	Short I/O address	User
18	H H H L L H H H	Undefined	Reserved
17	H H H L H L L L	Undefined	Reserved
16	H H H L H L L H	Undefined	Reserved
15	H H H L H L H L	Short Supervisory I/O Access	VERSAbus Spec.
14	H H H L H L H H	Undefined	Reserved
13	H H H L H H L L	Undefined	Reserved
12	H H H L H H L H	Undefined	Reserved
11	H H H L H H H L	Short Non-Prvileged I/O Access	VERSAbus Spec.
10	H H H L H H H H	Undefined	Reserved
08-0F	H H H H L X X X	Undefined	Reserved
07	H H H H H L L L	Undefined	Reserved
06	H H H H H L L H	Standard Supervisory Program access	VERSAbus Spec.
05	H H H H H L H L	Standard Supervisory Data Access	VERSAbus Spec.
04	H H H H H L H H	Undefined	Reserved
03	H H H H H H L L	Undefined	Reserved
02	H H H H H H L H	Standard Non-Privileged Program Access	VERSAbus Spec.
01	H H H H H H H L	Standard Non-Privileged Data Access	VERSAbus Spec.
00	H H H H H H H H	Undefined	Reserved

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 "VERSAbus Spec." should not be used for purposes other than those specified.

Codes defined by "User" may be used for any purpose which the VERSAbus 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.

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 15).

Master boards must, of course, drive the AM lines. If the master board has a real-time executive, that executive should be able to control the AM code placed on the VERSAbus 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 content 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:

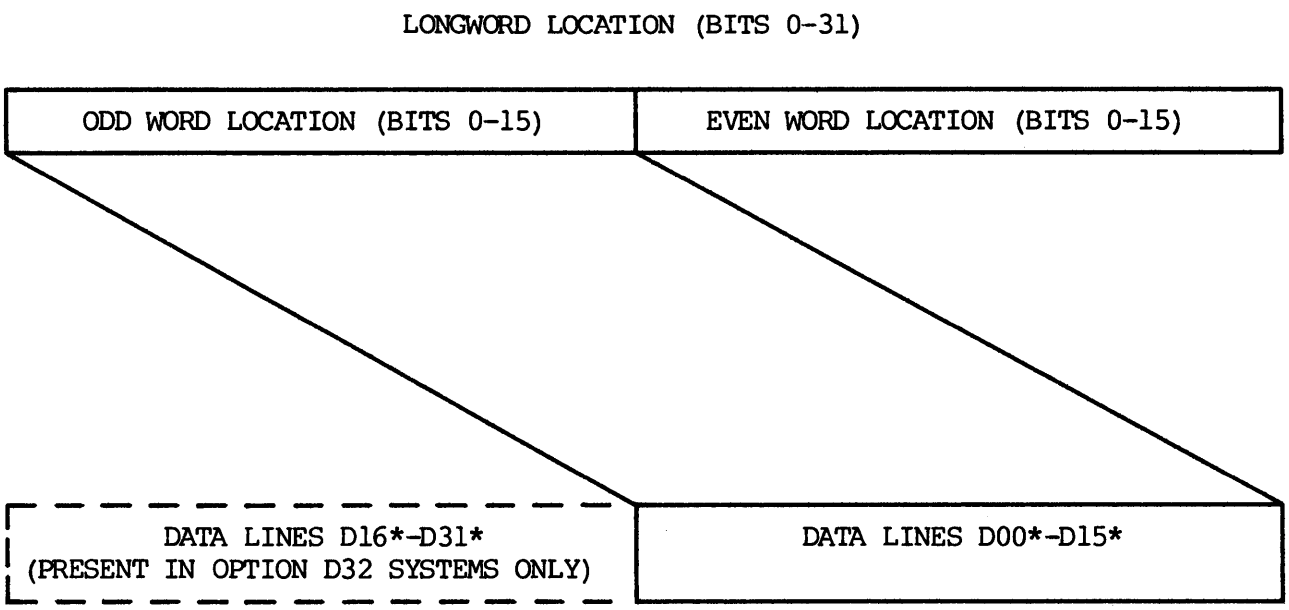
16 or 32	data lines	(D00* through D15* if option D16) (D00* through D31* if option D32)
0, 2, or 4	parity lines	(none if option NDP) (DPARITY0* and DPARITY1* if options DP and D16) (DPARITY0* through DPARITY3* if options DPY and D32)
1	data parity valid line (DPVAL*)	

Not all of the parity lines need to be driven if the currently selected transfer size is less than the maximum allowed under the data size option selected for the system. There is one DPARITY* line for each eight bits. Therefore, a byte transfer requires only the driving of data lines D00* through D07* and their corresponding parity line DPARITY0*, or data lines D08* through D15* and their corresponding parity line DPARITY1*.

A word transfer requires the driving of data lines D00* through D15* and their corresponding parity lines DPARITY0* and DPARITY1*. Only on a LONGWORD transfer do all of the data lines D00* through D31* and all parity lines DPARITY0* through DPARITY3* require driving. Note that word transfers always use the same 16 lines, while byte transfers use different lines for odd and even bytes.

Whenever an option DP MASTER or SLAVE drives the data bus, it should also drive DPVAL* (data parity valid) low. This is an indication that the appropriate data parity lines carry a valid parity bit and may be verified.

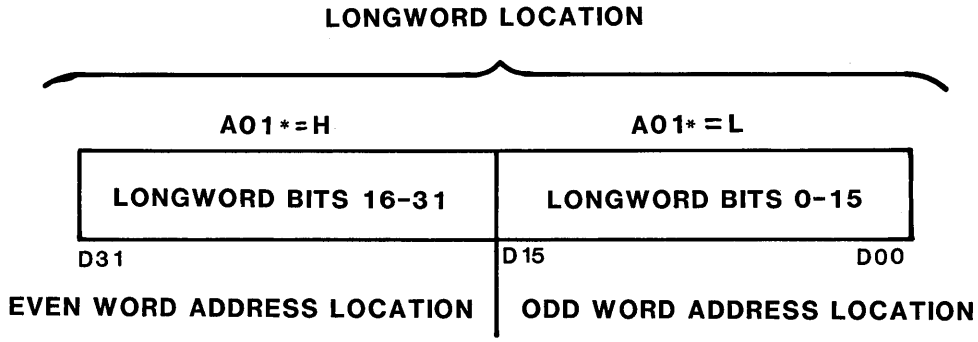
Only LONGWORD transfers use lines D16* through D31*. When LONGWORD transfers take place, the address presented is the even word address in the LONGWORD location (A01* is high). There are two word addresses for each LONGWORD. When accessing the data stored in a LONGWORD location on a word basis, the even word address (A01* high) corresponds to the LONGWORD data bits D16 through D31. Likewise, the structure of the bytes within a word have BYTE1 (selected by DS1*) as the most significant eight bits of the word (see Figures 2-4, 2-5, and 2-6).



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

FIGURE 2-4. Odd Word Location Accesses

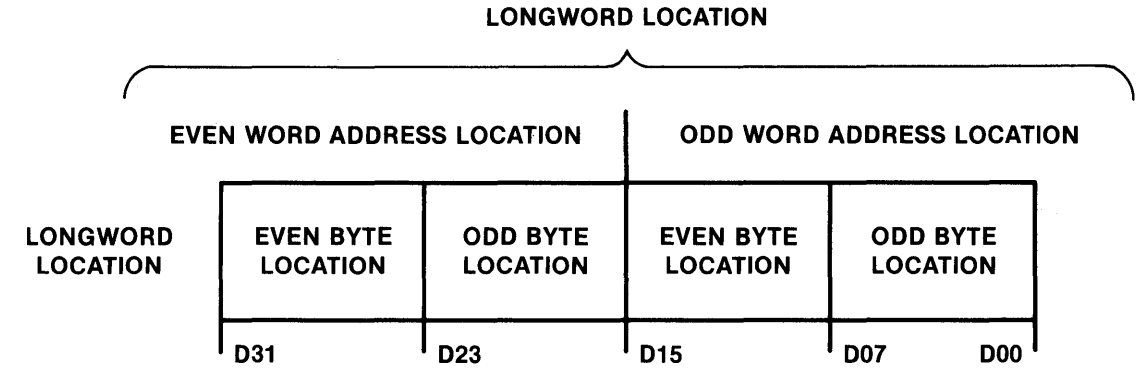
When LONGWORD data is written into memory, the 32 data bits are stored at a LONGWORD location. There are two word locations for each LONGWORD location. When reading the LONGWORD location on a word basis, the even word address (A01=0) corresponds to the LONGWORD data bits D16-D31. See Figure 2-5.



THE ODD WORD LOCATION (A01 = 1) CORRESPONDS TO THE LONGWORD DATA BITS D00*-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..XX10	XX..X11
Byte Access				
LWORD*	high	high	high	high
A01*	high	high	low	low
DS1*	low	high	low	high
DS0*	high	low	high	low
Word Access				
LWORD*	high	Note	high	Note
A01*	high	1	low	1
DS1*	low		low	
DS0*	low		low	
LONGWORD Access				
LWORD*	low	Note	Note	Notes
A01*	high	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 write)
	Data Strobe to MASTER	(If read)

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 bits 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 low is to be found on data lines D00* through D07*. Likewise, DS1* low means that the byte which would be addressed with A00* set high 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 normally perform. The sender is not prohibited from driving the data lines which are not being strobed. The receiver is required to ignore any and all levels and/or transitions which occur on non-strobed data lines.

LWORD* has meaning only to those modules which allow 32-bit transfers. It 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.

TABLE 2-2. Data Transfer Control Table

LWORD*	A01*	DS0*	DS1*	CONSTRAINT/ACTION
L	L	L	L	Illegal (not on proper boundary)
L	L	L	H	Illegal (not on proper boundary)
L	L	H	L	Illegal (not on proper boundary)
L	L	H	H	Illegal (not on proper boundary)
L	H	L	L	Long word transfer
L	H	L	H	Illegal (only one data strobe)
L	H	H	L	Illegal (only one data strobe)
L	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)
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)

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 indicates that some illegality has been detected in the request as processed at the SLAVE. This illegality could be the result of an address parity error, data parity error, 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,
- if the data from the MASTER on a write is invalid, or
- if the address provided to the SLAVE is invalid.

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 VERSAbus access would require a MASTER module. If the same board also contained memory accessible from the VERSAbus, it would also require 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.

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 highest level 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. (For simplicity, the generation and checking of parity has not been shown.)

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. 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 and DS1* 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.

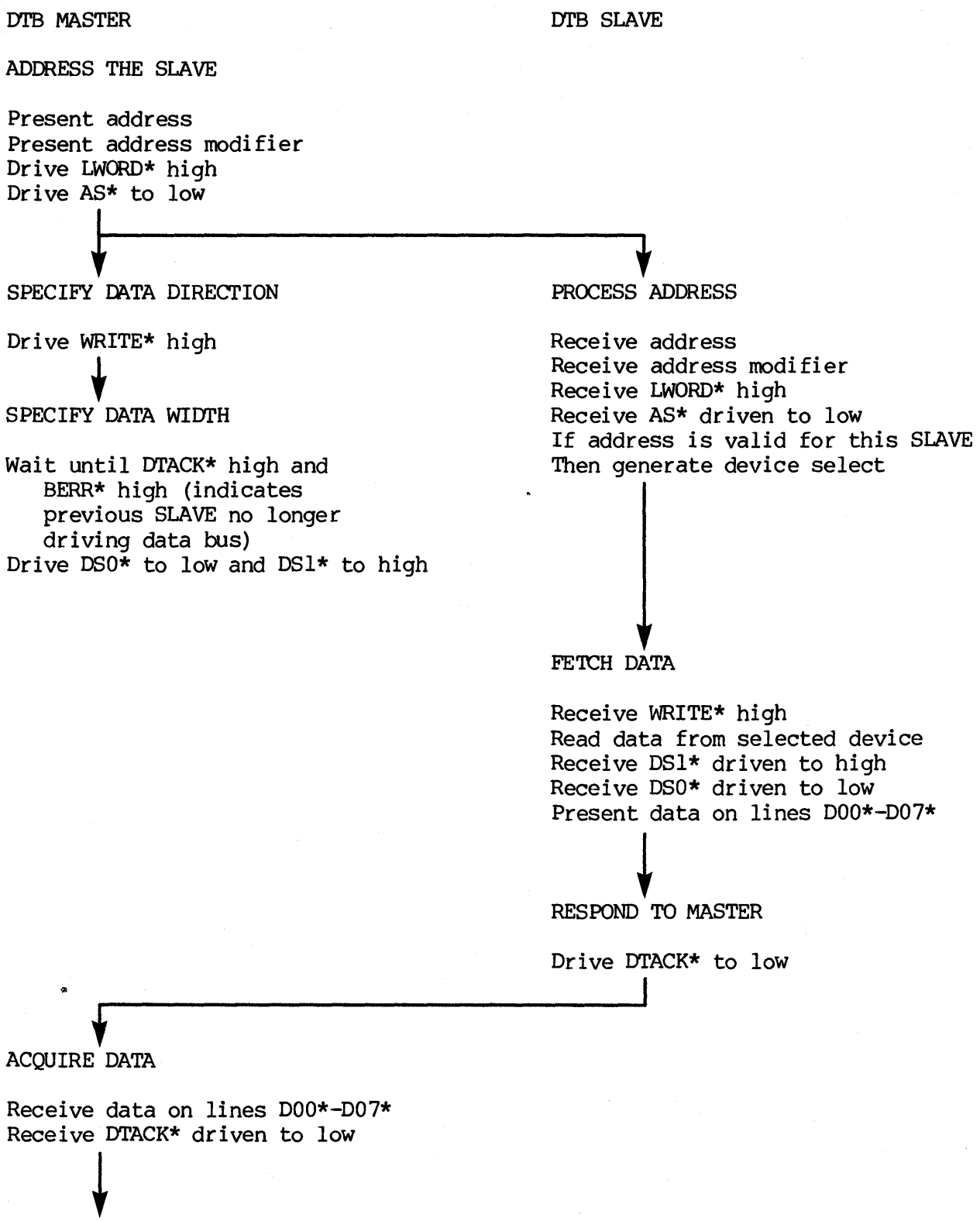
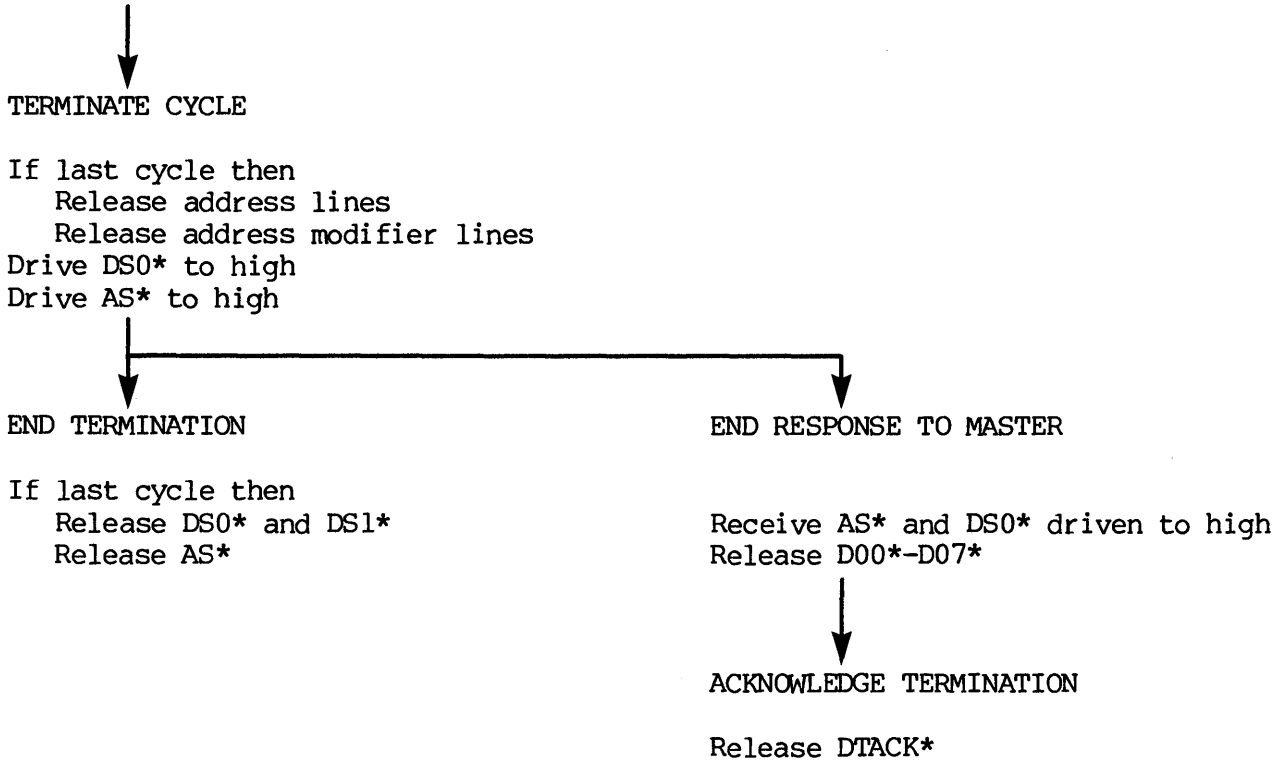


FIGURE 2-7. Data Transfer Bus, Byte Read Cycle (Sheet 1 of 2)



NOTE

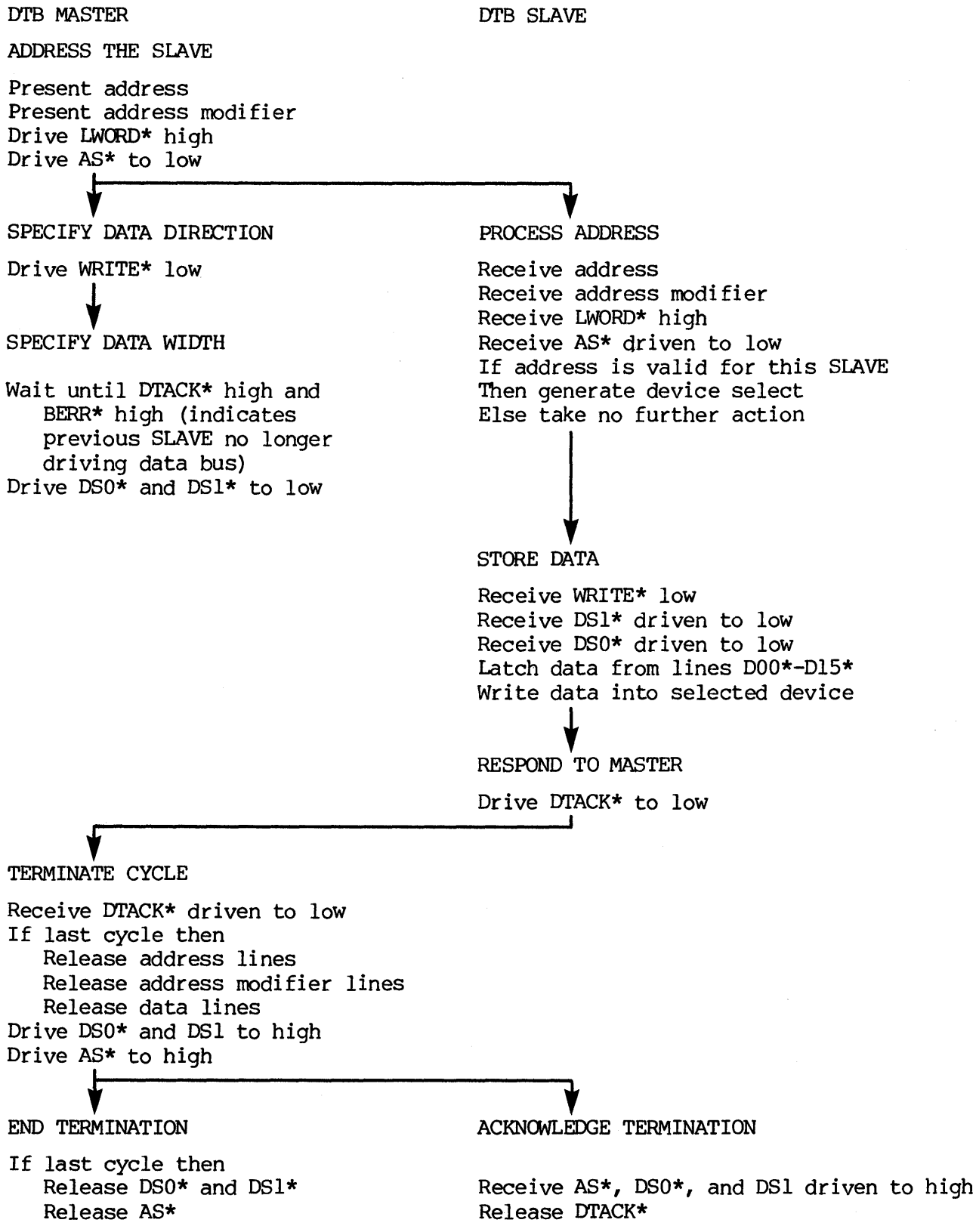
For simplicity, the generation and checking of address and data parity have been ignored, and the assumption has been made that no transfer causes a bus error.

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 completed, the MASTER terminates the cycle by releasing the address lines, driving DS0* and AS* to high, and then releasing DS0* and AS*.

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

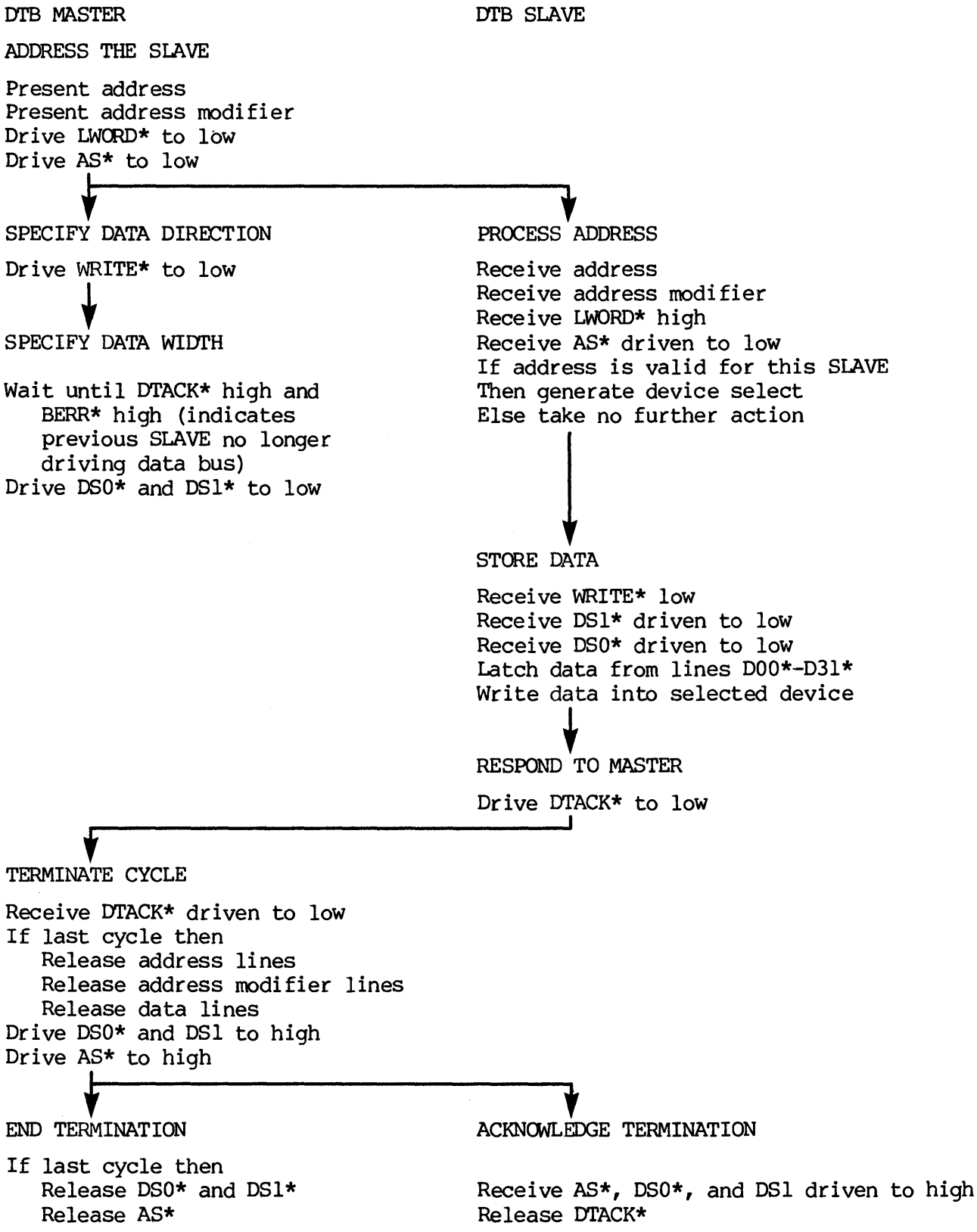
The cycle flow for word and LONGWORD data transfers is very similar to the byte cycle. Flow charts for these cycles are shown in Figures 2-8 and 2-9.



NOTE

For simplicity, the generation and checking of address and data parity have been ignored, and the assumption has been made that no transfer causes a bus error.

FIGURE 2-8. Data Transfer Bus, Word Write Cycle



NOTE

For simplicity, the generation and checking of address and data parity have been ignored, and the assumption has been made that no transfer causes a bus error.

FIGURE 2-9. Data Transfer Bus, LONGWORD Write Cycle

2.5 FORMAL SPECIFICATIONS

The following text defines specifications for data movements on the VERSAbus.

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 BBSY* to be released 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., as 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 the 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.

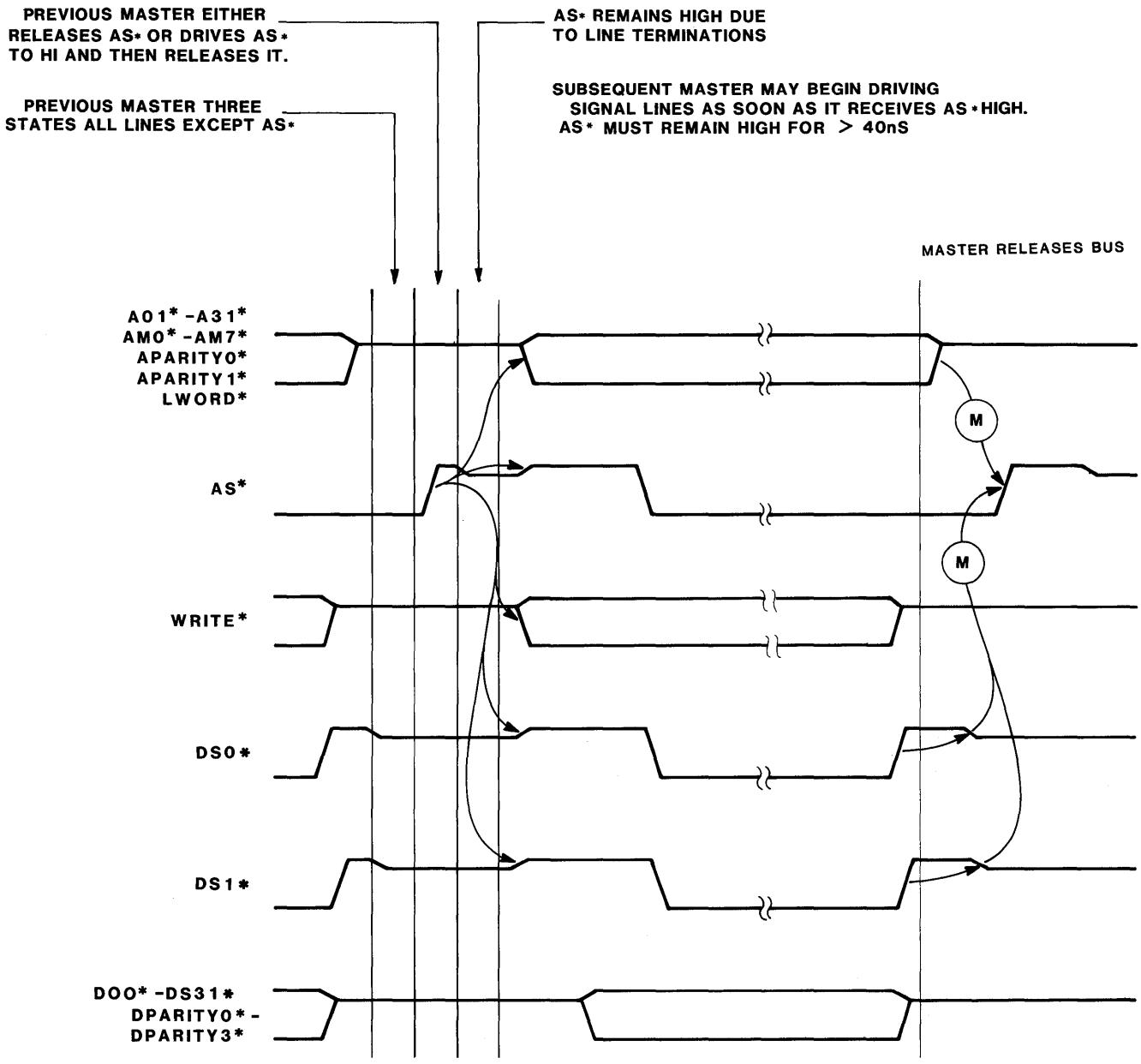


FIGURE 2-10. Data Transfer Bus MASTER Exchange Sequence

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 A01*-A23*, and the address modifier code is presented on AM0*-AM7*. The transfer is identified as a non-32-bit transfer by driving LWORD* high. A parity generator on the MASTER board may also present the address parity on APARITY0*.

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.

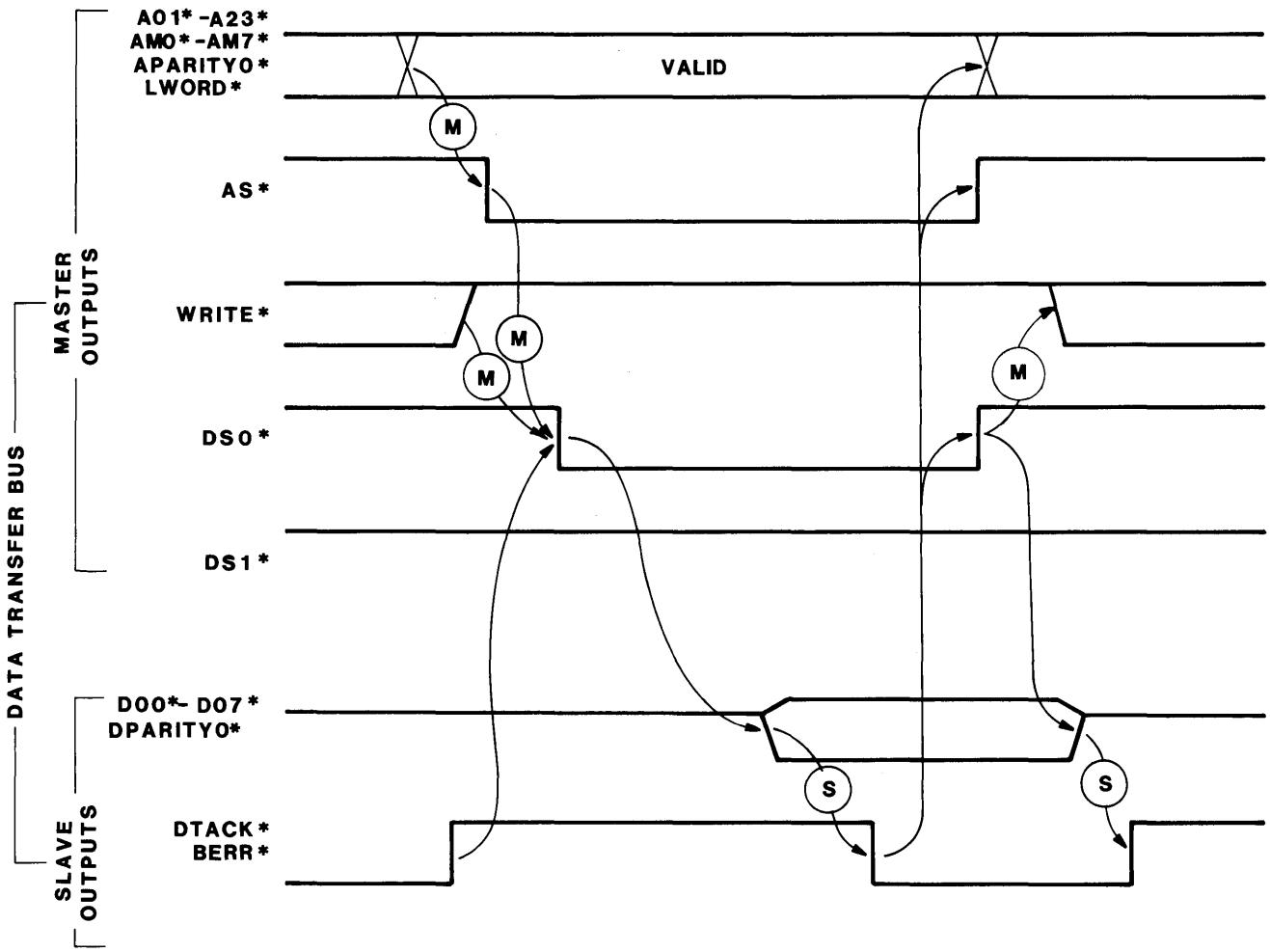
When the SLAVE receives AS* driven to low, its address decoder will compare the incoming address with the pre-assigned SLAVE address. In addition, the SLAVE may verify the received address and address modifier by recalculating the address parity and comparing it with the address parity line (APARITY0*). If the address is valid for this device but the parity is wrong, the SLAVE will respond by driving BERR* to low if he is an option DP SLAVE. If the address is valid for this device and no address parity error is detected by the SLAVE, a data transfer select is generated internal to the SLAVE.

2.5.2.2 Data Bus Sequencing

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 before 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 address parity error, it will abort the cycle by driving BERR* to low after receiving a data strobe driven to low. The SLAVE must wait 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*. The parity is placed on DPARITY0*. After valid data and parity have been presented to the bus, the SLAVE drives DTACK* to low. A delay time is provided between valid data and driving DTACK* to low. This provides for bus skew and latch setup time for the MASTER'S input latches. 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 and WRITE* line, 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

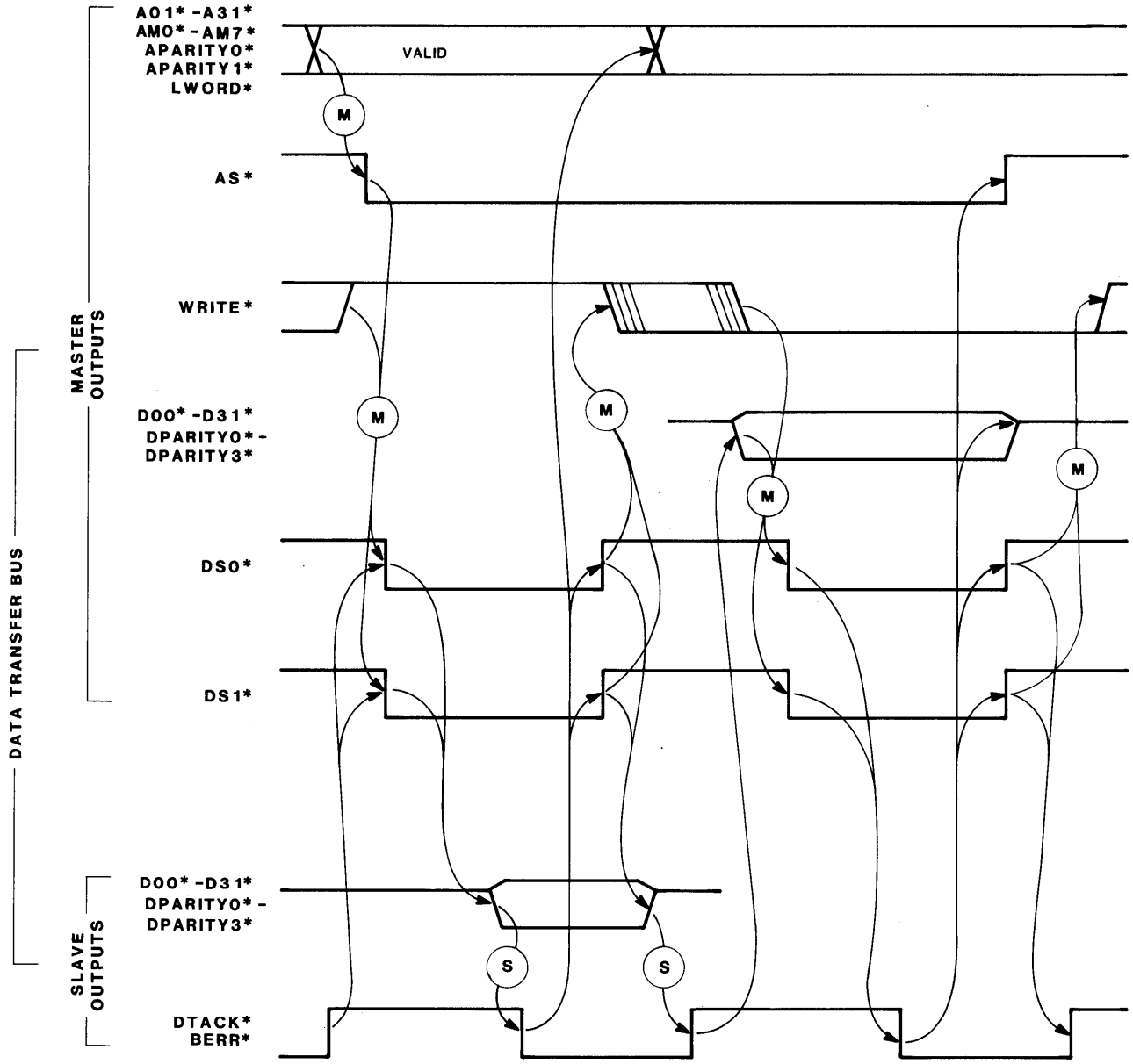
In multiprocessor systems which share resources such as memory and I/O, a method of allocating the resources must be established. The difficulty of any method of allocation is synchronizing the asynchronous requests for that resource. This is best described by the following example.

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 with 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 not allow AS* to go high. Instead, it repeatedly drives the data strobes low 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 all 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 location 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 (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 are 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 30 nanoseconds of skew, while each SLAVE is guaranteed only 20 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 VERSAbus 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 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. 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 the SLAVE 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 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 setup time guarantees the MASTER that the data bus is valid and stable the minimum setup time before 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	MIN.	MAX.	NOTES
1	Address valid to AS* low	30		B
2	DTACK* low to invalid address	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	30		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		20	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	Data valid to DTACK* low	20		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.

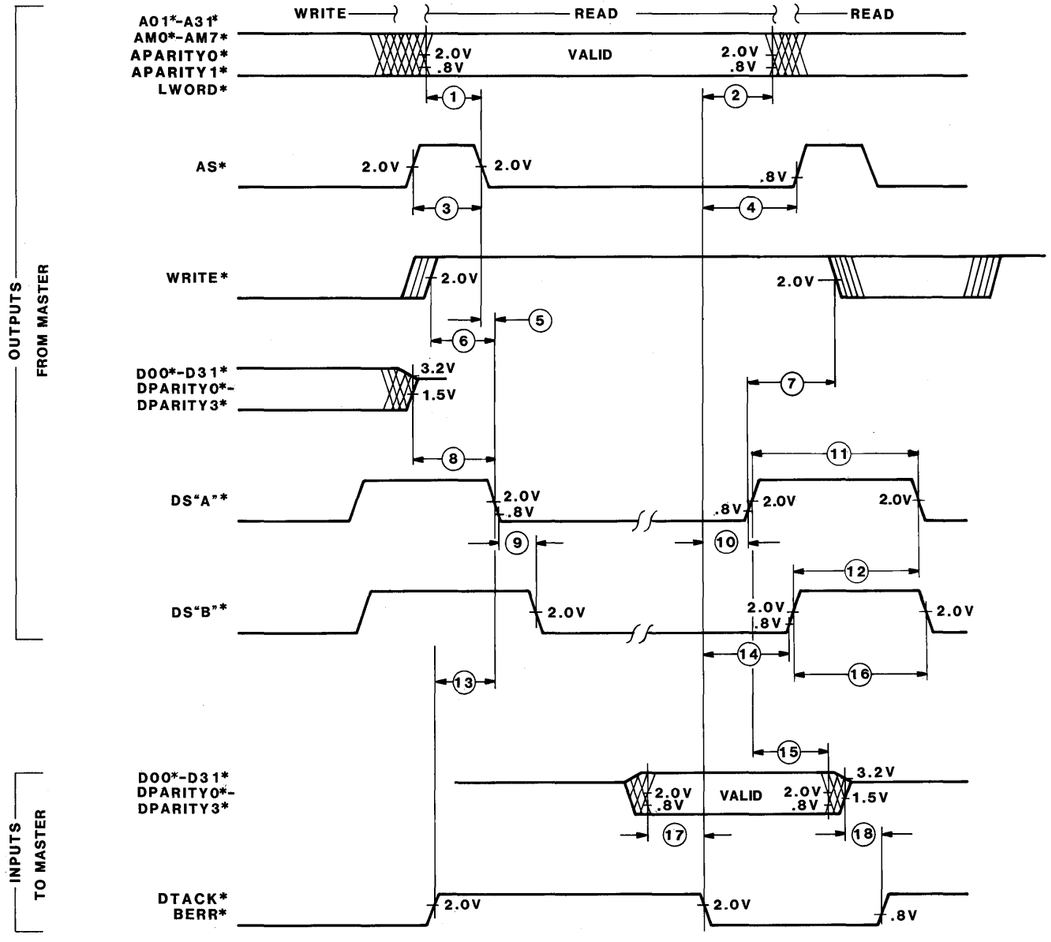


FIGURE 2-13. DTB MASTER Timing: Write Cycle Followed by Read Cycle

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 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. 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 the SLAVE detects 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	MIN.	MAX.	NOTES
1	Address valid to AS* low	30		B
2	DTACK* low to invalid address	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	30		B
7	DS"A"* high to invalid WRITE*	10		B
8	DTACK* high to active data bus	0		C
9	Data valid to DS"A"* low	30		B
10	DTACK* low to invalid data	0		C
11	DS"A"* to DS"B"* skew	0	20	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.

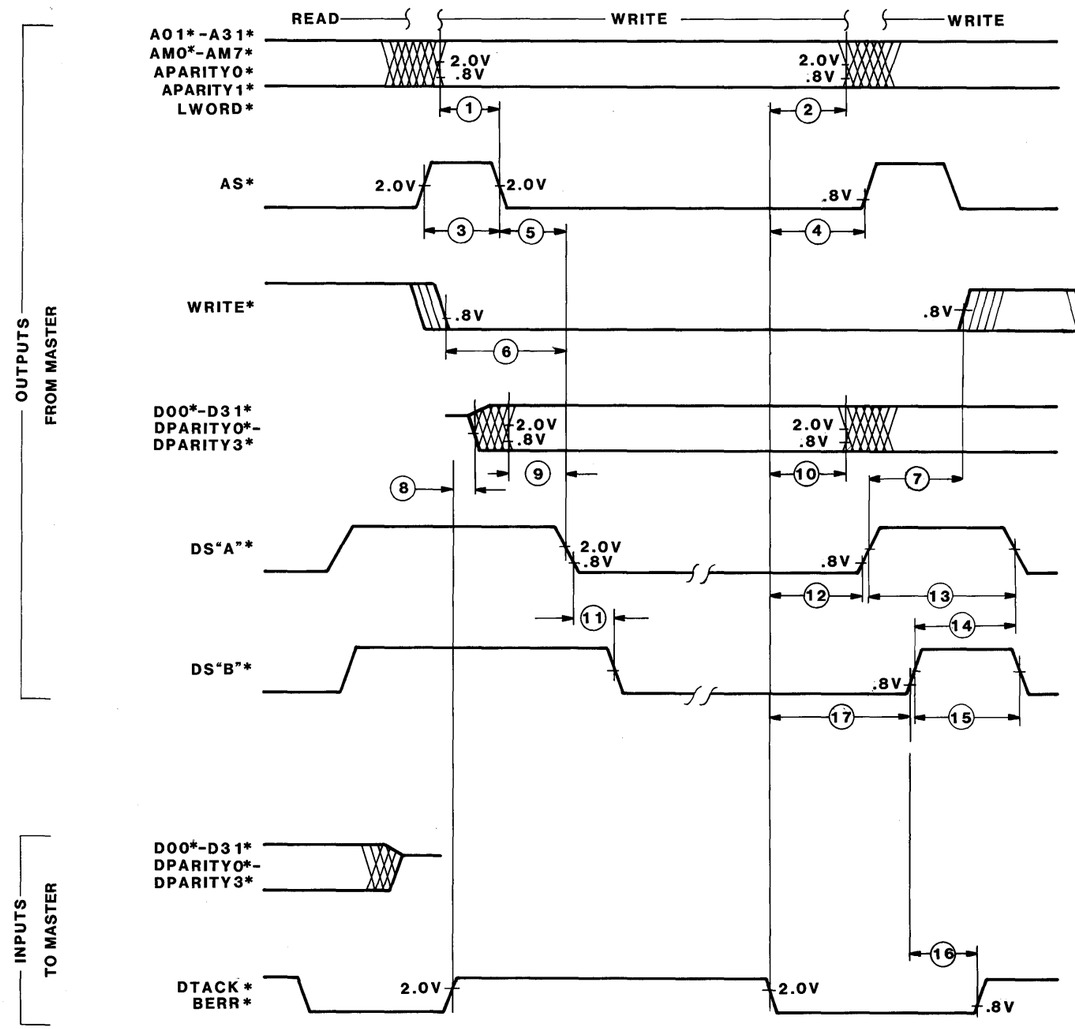


FIGURE 2-14. DTB MASTER Timing: Read Cycle Followed by Write Cycle

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.

NOTE

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 20 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*).

MASTER B then drives the DTB in accordance with the timing given in paragraph 2.7.2. (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).

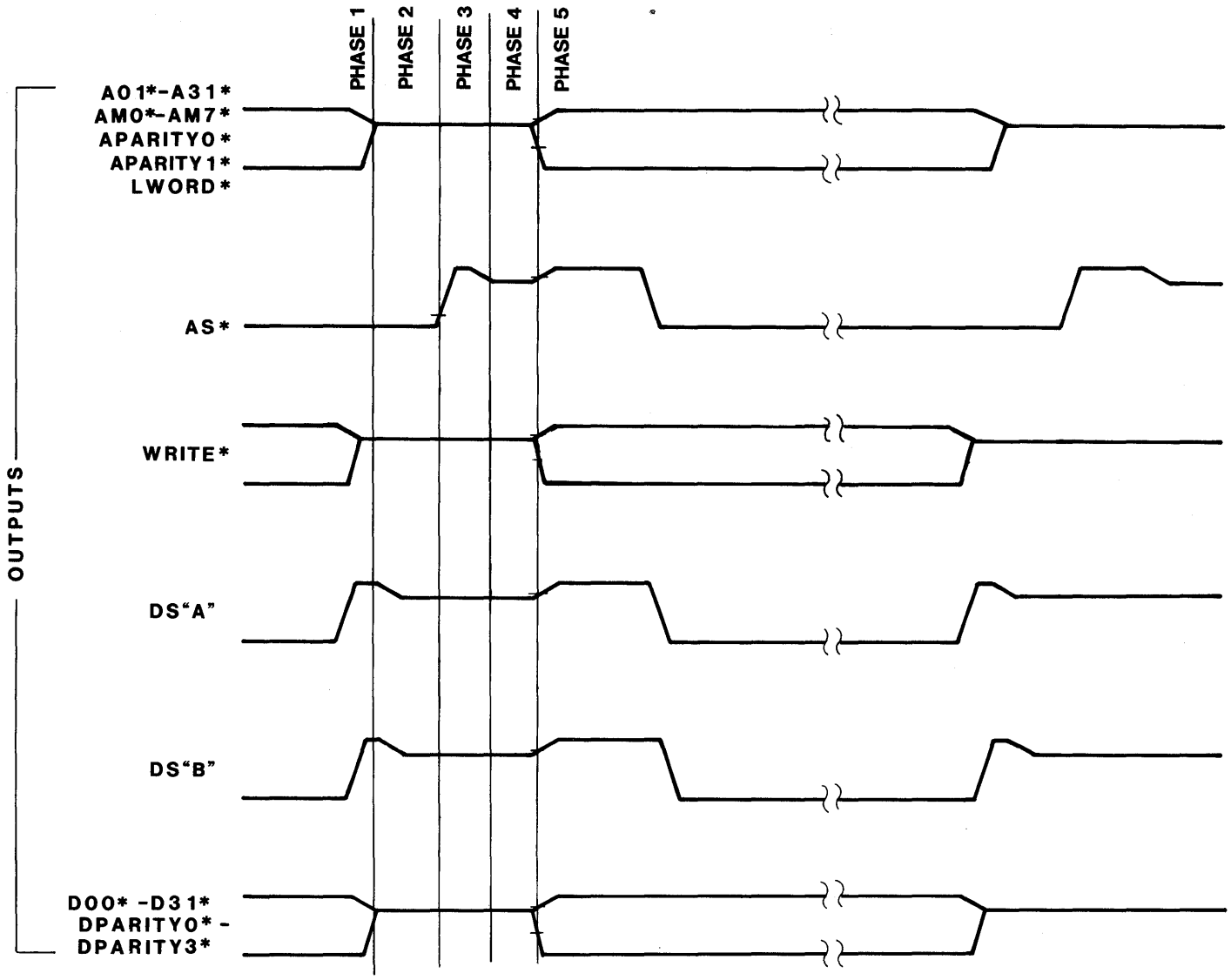


FIGURE 2-15. MASTER Timing: Control Transfers of DTB

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	MIN.	MAX.	NOTES
1	Address valid to AS* low	20		E
2	DTACK* low to invalid address	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	20		E
7	DS"A"* high to invalid WRITE*	0		E
8	DS"A"* to DS"B"* skew		30	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	30		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.

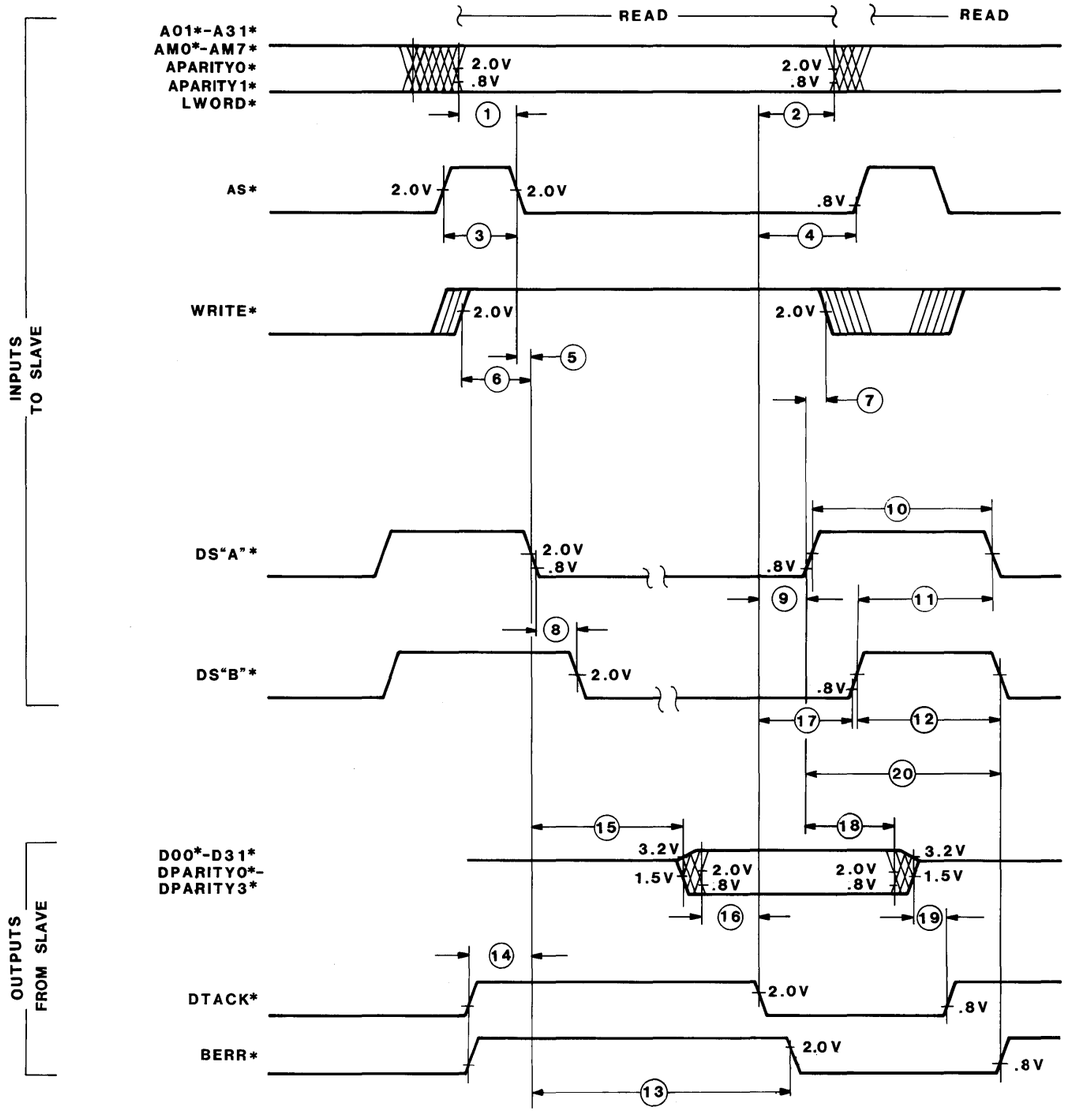


FIGURE 2-16. Data Transfer Bus SLAVE Read Cycle

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 14 These times guarantee that both data strobes will be concurrently driven 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	MIN.	MAX.	NOTES
1	Address valid to AS* low	20		D
2	DTACK* low to invalid address	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	20		D
7	DS"A"* high to invalid WRITE*	0		D
8	Data valid to DS"A"* low	20		D
9	DTACK* low to invalid data	0		C
10	DS"A"* to DS"B"* skew		30	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.

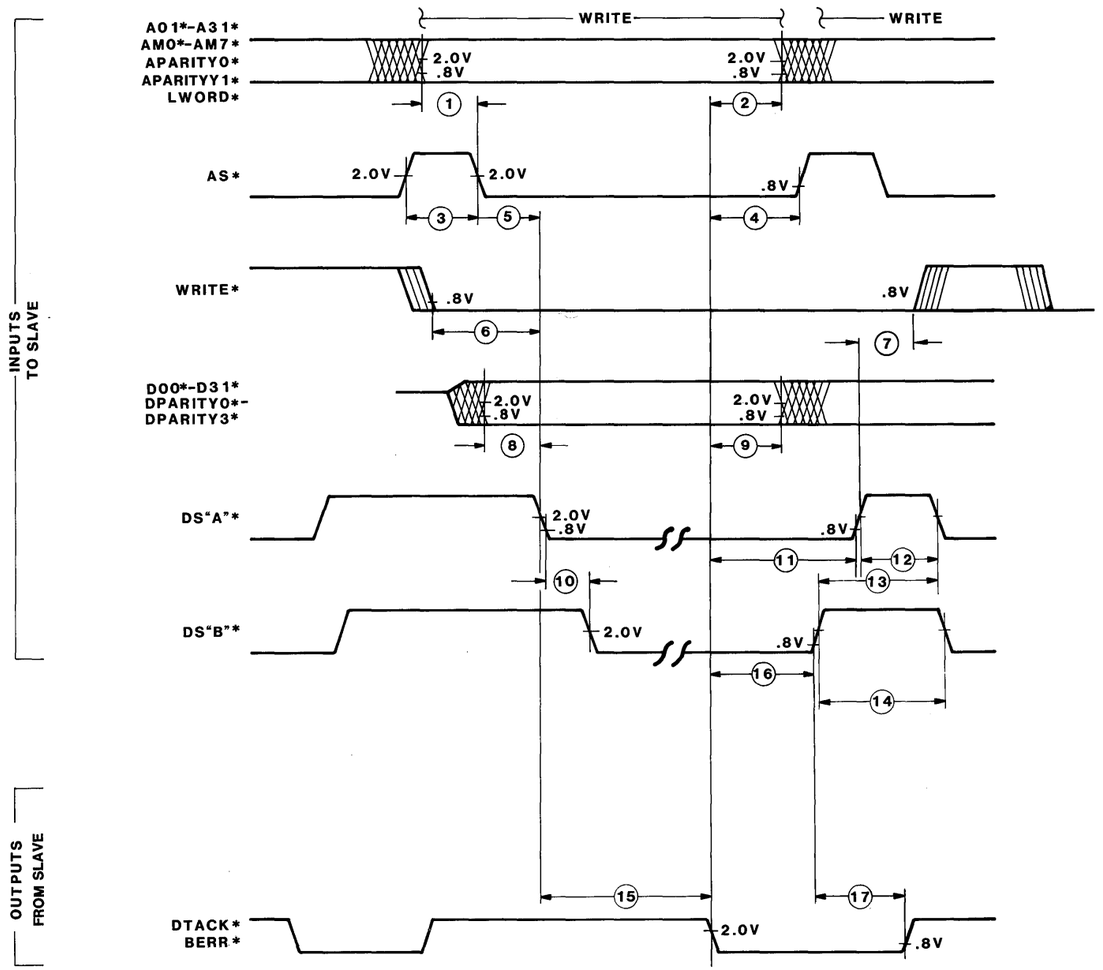


FIGURE 2-17. Data Transfer Bus SLAVE Write Cycle

CHAPTER 3

VERSAbus DATA TRANSFER BUS ARBITRATION

CHAPTER 3

VERSAbus DATA TRANSFER BUS ARBITRATION

3.1 BUS ARBITRATION PHILOSOPHY

Past microprocessor system designs have generally consisted of a single central processor accessing its system resources by reading or writing into registers. In multitasking versions of such systems, the system resources were allocated by a real-time executive which:

- Prevented simultaneous use of a resource by two tasks
- Allocated system resources on a priority basis

As microprocessor costs decrease, many 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 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 VERSAbus meets this need with its Bus Arbitration subsystem. (See Figure 3-1).

The VERSAbus arbitration subsystem is designed to:

- Prevent simultaneous access of the bus by two MASTERS
- Prioritize 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 two versions. The Option NPF (No Power Fail) ARBITER accepts normal bus requests and arbitrates them, but it has no provision for emergencies. The Option PF (Power Fail) ARBITER has additional circuitry to allow emergency demands for the data bus in case of power failure or hardware malfunctions.

3.1.2 ARBITER Operation

The bus arbitration subsystem accepts requests for the bus on five request lines, which are assigned a sequential priority from highest (BR4*) to lowest (BR0*). These lines are driven by open collector drivers so that several MASTERS can share a common request line. Each request line has a corresponding

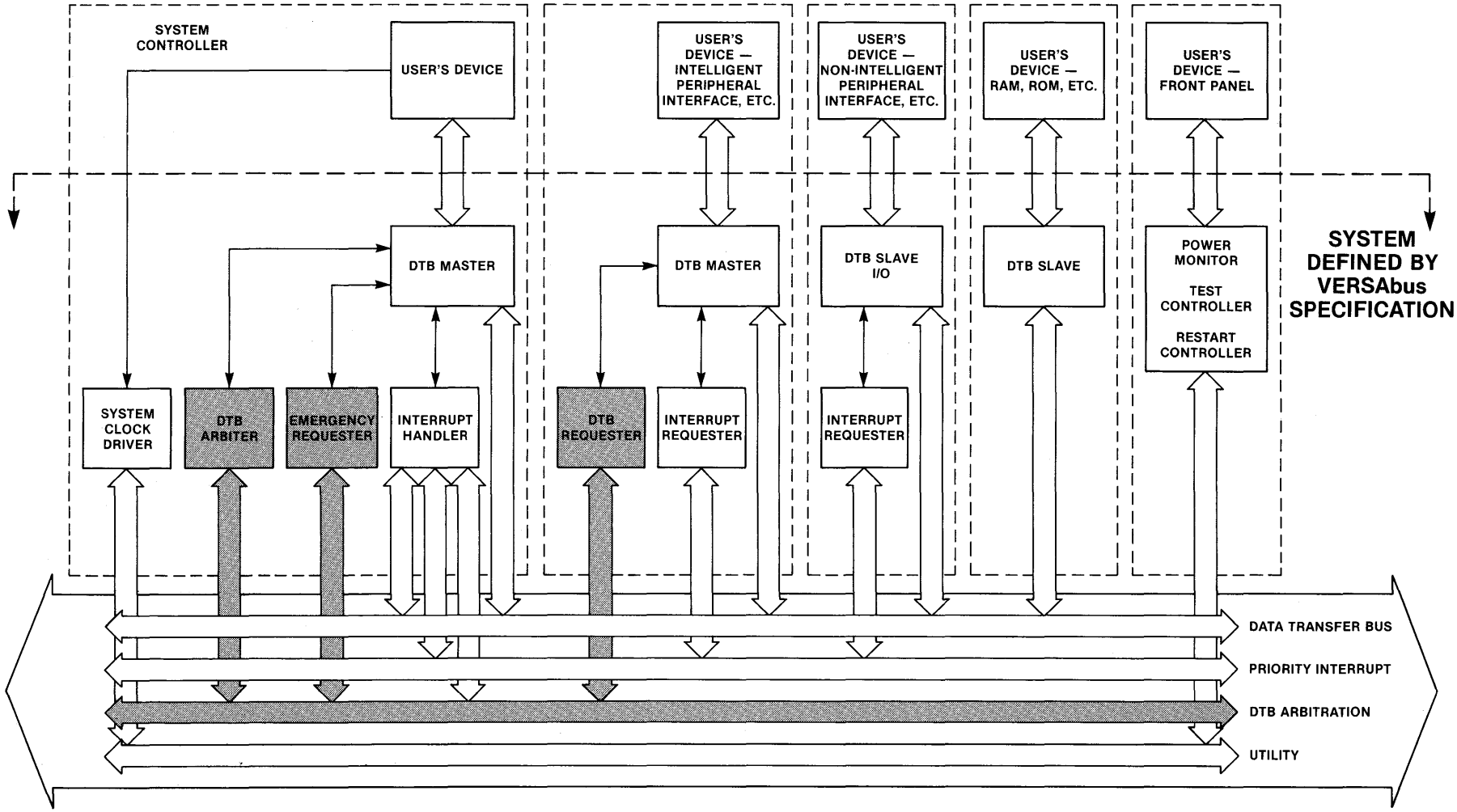


FIGURE 3-1. VERSAbus Arbitration Functional Block Diagram

grant daisy-chain line (BG4IN* through BG0IN* and BG4OUT* through BG0OUT*). If the bus is idle when a request is received, the ARBITER will immediately respond on the grant line corresponding to the highest level of request pending. If the bus is busy, and the request level of the MASTER using the bus is lower than the current highest request level, the ARBITER will request that the current MASTER relinquish control to the new request. If the requests currently pending are lower than or equal to the current MASTER'S request level, they will not result in any action until the current MASTER relinquishes the bus. At that point, 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 a priority 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.

3.2 ARBITRATION BUS LINE STRUCTURES

The arbitration bus consists of eight bussed VERSAbus lines and five 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.

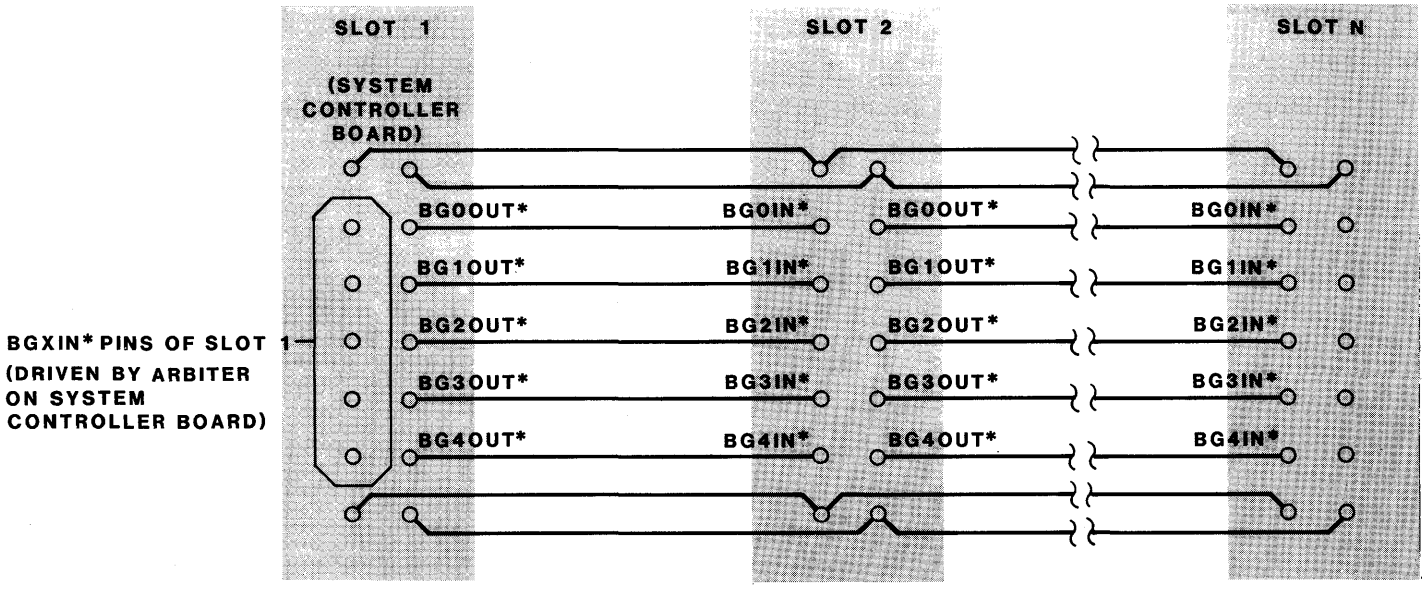


FIGURE 3-2. Illustration of the Daisy Chained Bus Grant Lines

NOTE

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 four.

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

1	bus request line ... (one of BR0* through BR4*)
5	bus grant lines out. (BG0OUT* through BG4OUT*)
1	bus busy line ..... (BBSY*)

The ARBITER will drive the following:

1	bus clear line ..... (BCLR*)
5	bus grant lines in . (BG0IN* through BG4IN*)
1	emergency bus grant line . (PF ARBITER only; on-board signal; not found on VERSAbus)

The EMERGENCY REQUESTER (used exclusively with an Option PF ARBITER) will drive:

1	bus release line ... (BREL*)

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 the slot being examined 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 slot 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 the 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 VERSAbus access. The five bus request lines are prioritized so that the ARBITER will issue a grant to the highest level of BR signal (BR4*).

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, and any REQUESTERS on the board in slot 1 follow the same process as they would on any other board. Such a design creates uniformity in the structure of the VERSAbus 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 the 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*.

3.2.4 Bus Release Line (BREL*)

BREL* is used to clear the DTB in the event of a system emergency, such as loss of power. It is driven by the EMERGENCY REQUESTER on the system controller board. The ARBITER treats it as a "BR5*" level of request. At the same time, this signal instructs the REQUESTER/MASTER set currently in control of the DTB to release the bus within 200 microseconds. When the DTB is released, the system MASTER will be given control of the DTB to execute an orderly shutdown.

3.3 FUNCTIONAL MODULES

The arbitration subsystem is composed of several modules:

- One data transfer bus ARBITER (Option PF or NPF)
- One or more data transfer bus REQUESTERS
- One or more data transfer MASTERS

NOTE

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 driven by the power monitor module.

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 highest level REQUESTER by generating the matching BGxIN* signal for that level, and to inform any MASTER currently in control of the bus that a higher level request is pending. If the ARBITER is an Option PF, it also performs the task of responding to the BREL* command as issued by the EMERGENCY REQUESTER and presenting it a special bus grant.

A block diagram of an NPF ARBITER is given in Figure 3-3. It uses as its prioritization inputs the five bus request lines BR0* through BR4*, and responds with BG0IN* through BG4IN*, as appropriate.

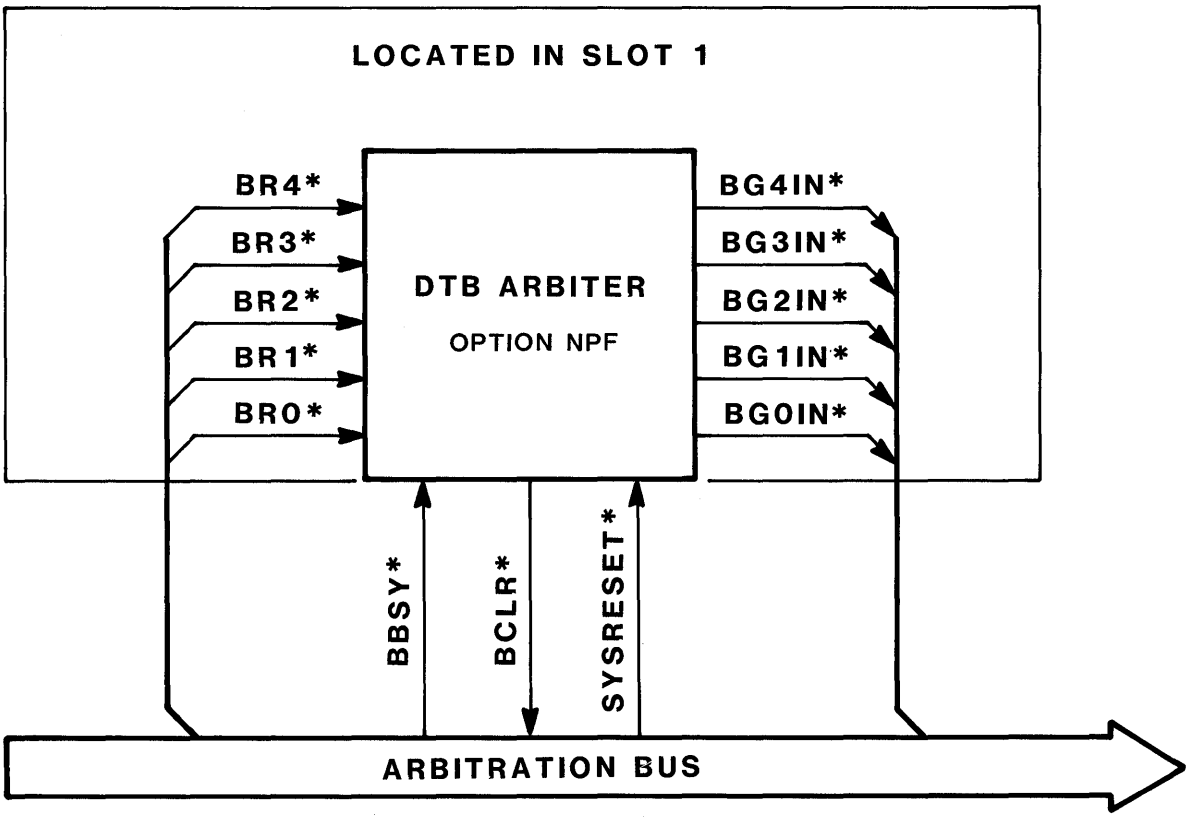


FIGURE 3-3. Block Diagram: Option NPF DTB ARBITER

When we examine a PF version ARBITER as shown in Figure 3-4, an additional complexity obviously has been added. The ARBITER now has one additional input (BREL*) and two additional outputs. One of these is a special grant to the on-board EMERGENCY REQUESTER. Since no pins are designated in the bus for this signal, a PF option ARBITER must be on the same board with the EMERGENCY REQUESTER and the system MASTER. The ARBITER will treat BREL* as equivalent to a "BR5*" signal, and a special on-board grant is presented to the EMERGENCY REQUESTER. Upon receiving this grant, the EMERGENCY REQUESTER will drive BBSY* low to indicate that a MASTER has the bus.

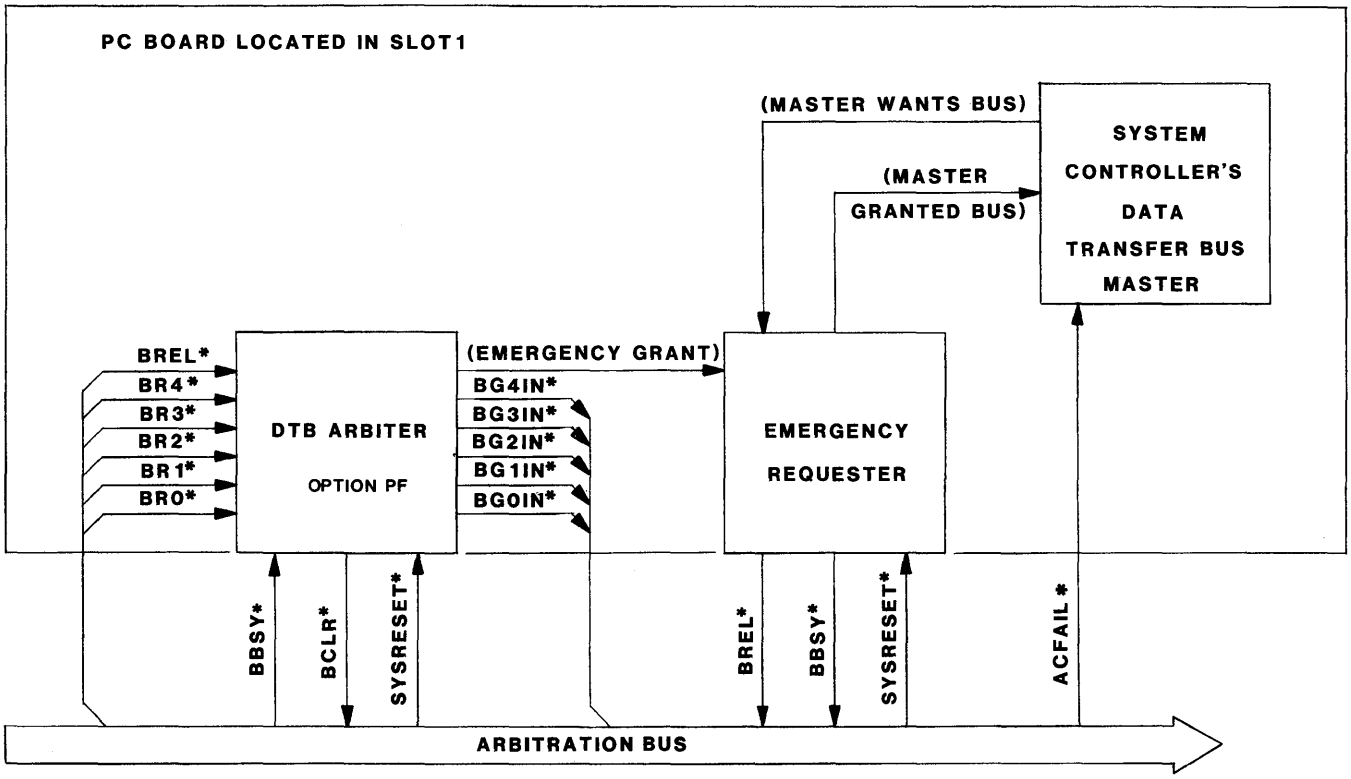


FIGURE 3-4. Block Diagram: Option PF DTB ARBITER

3.3.2 Data Transfer Bus REQUESTER

Each REQUESTER in the system is required to:

- translate the MASTER WANTS BUS on-board signal 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 signals MASTER GRANTED BUS and Bus Busy (BBSY*) as long as the MASTER is providing MASTER WANTS BUS.

In the simplest REQUESTER, (Option RWD (Release When Done)), MASTER GRANTED BUS and BBSY* will both be released upon loss of MASTER WANTS 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 MASTER GRANTED BUS and BBSY* upon loss of MASTER WANTS BUS. This REQUESTER would monitor the five bus request lines and bus release, and drop these signals 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.

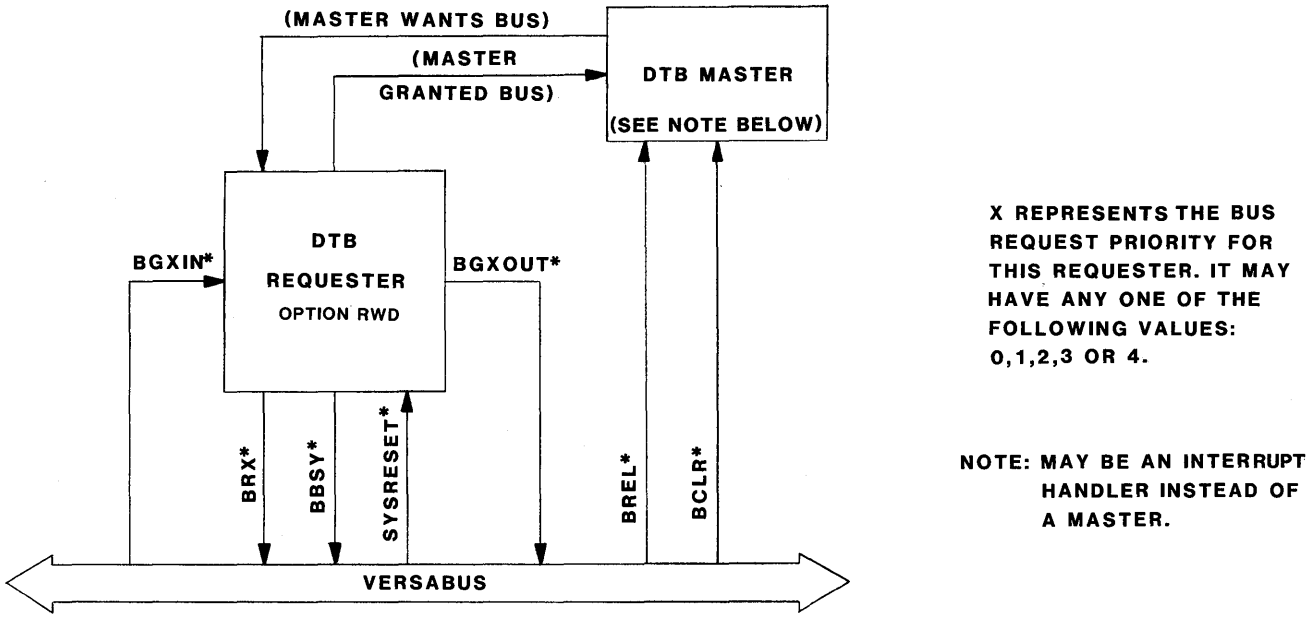


FIGURE 3-5. Block Diagram: Option RWD REQUESTER

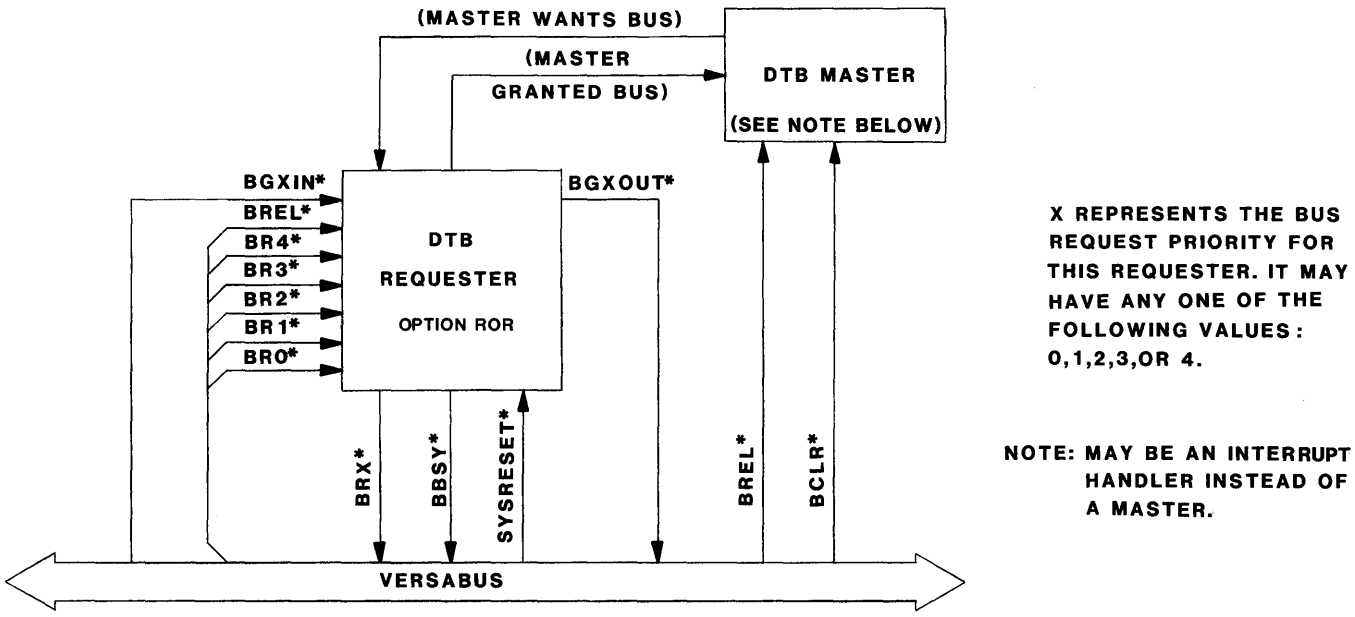


FIGURE 3-6. Block Diagram: Option ROR REQUESTER

3.3.3 Data Transfer Bus MASTER

While the actual data transfer characteristics of a MASTER have been covered in a previous chapter, we now need to look at additional controls and responses which the MASTER must deal with to use the arbitration subsystem. For an ordinary MASTER, there are two on-board signals and two VERSAbus signals. The two VERSAbus signals are BREL* 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 design determines how long the MASTER 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. BREL* differs from BCLR* in that it informs the MASTER that an emergency exists, 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.

A special case exists for the DTB MASTER which shares a board with the PF option ARBITER. This MASTER must also have an associated EMERGENCY REQUESTER, and the MASTER is designated as the "system MASTER". It has the added responsibility of monitoring the ACFAIL* line and generating a MASTER WANTS BUS to its on-board EMERGENCY REQUESTER upon detecting ACFAIL* low. The EMERGENCY REQUESTER uses BREL* as its "BR5*" to ask the ARBITER to grant it the DTB, and to request the

current DTB MASTER to relinquish the DTB. When the ARBITER has detected the release of the bus by the MASTER currently in possession, it will inform the system MASTER that it has the bus. It is now the responsibility of the system MASTER to take those actions most essential to safe power-down of the system. As examples, it may save the contents of a limited amount of highly volatile memory on some storage device which does not lose data on loss of power. It may instruct hard disks to retract their heads to eliminate potential erasures of sectors. If this sytem is part of a distributed network, the system MASTER may inform remote sites of its pending departure from the network. These activities are all designed to minimize the damage done by unscheduled power disruptions. The system MASTER is allowed to be a system problem monitor, and it may use similar techniques to minimize the damage if other types of problems are detected.

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.

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.

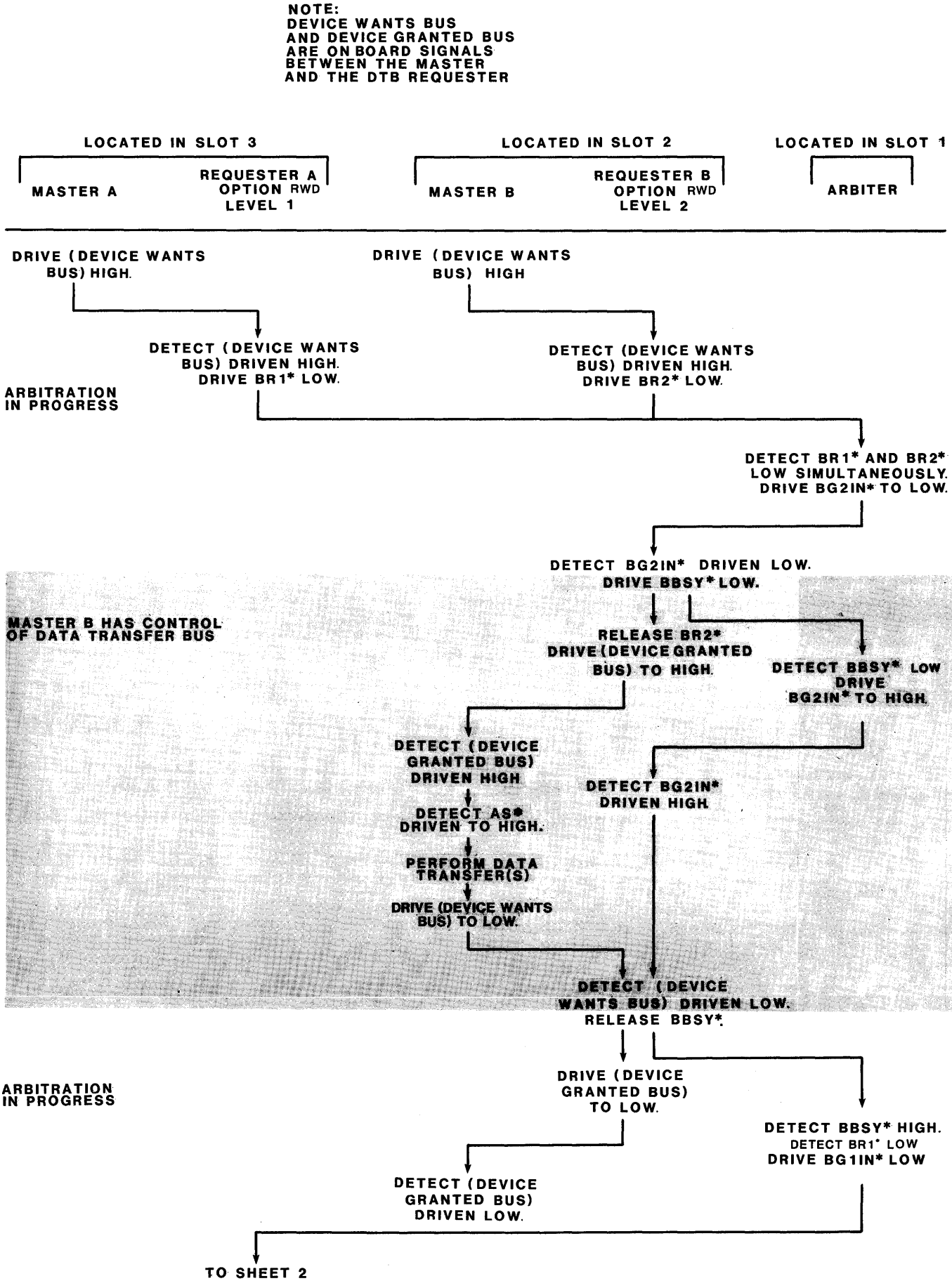
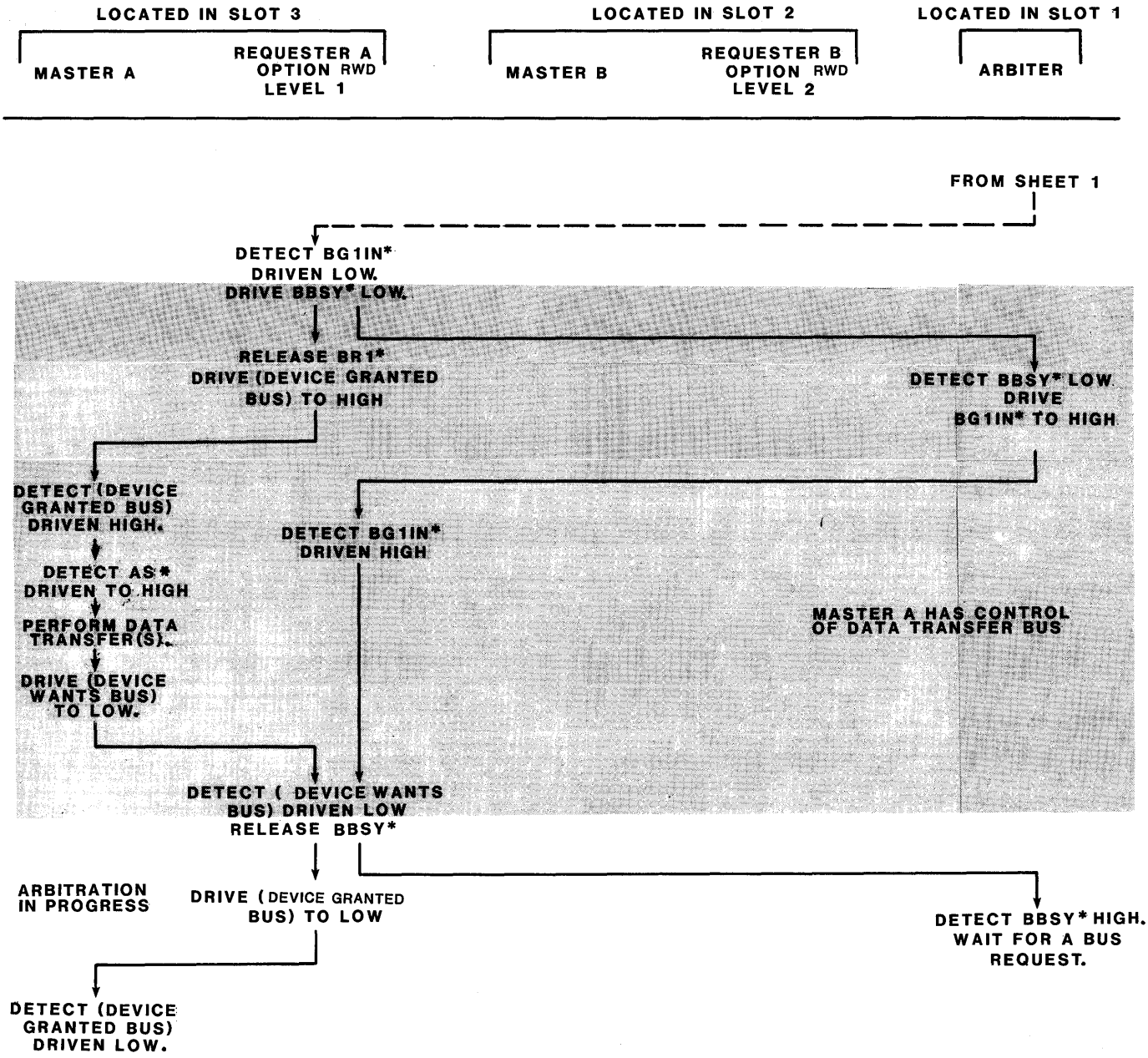


FIGURE 3-7. Arbitration Flow Diagram: Two REQUESTERS, Two Request Levels (Sheet 1 of 2)



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)

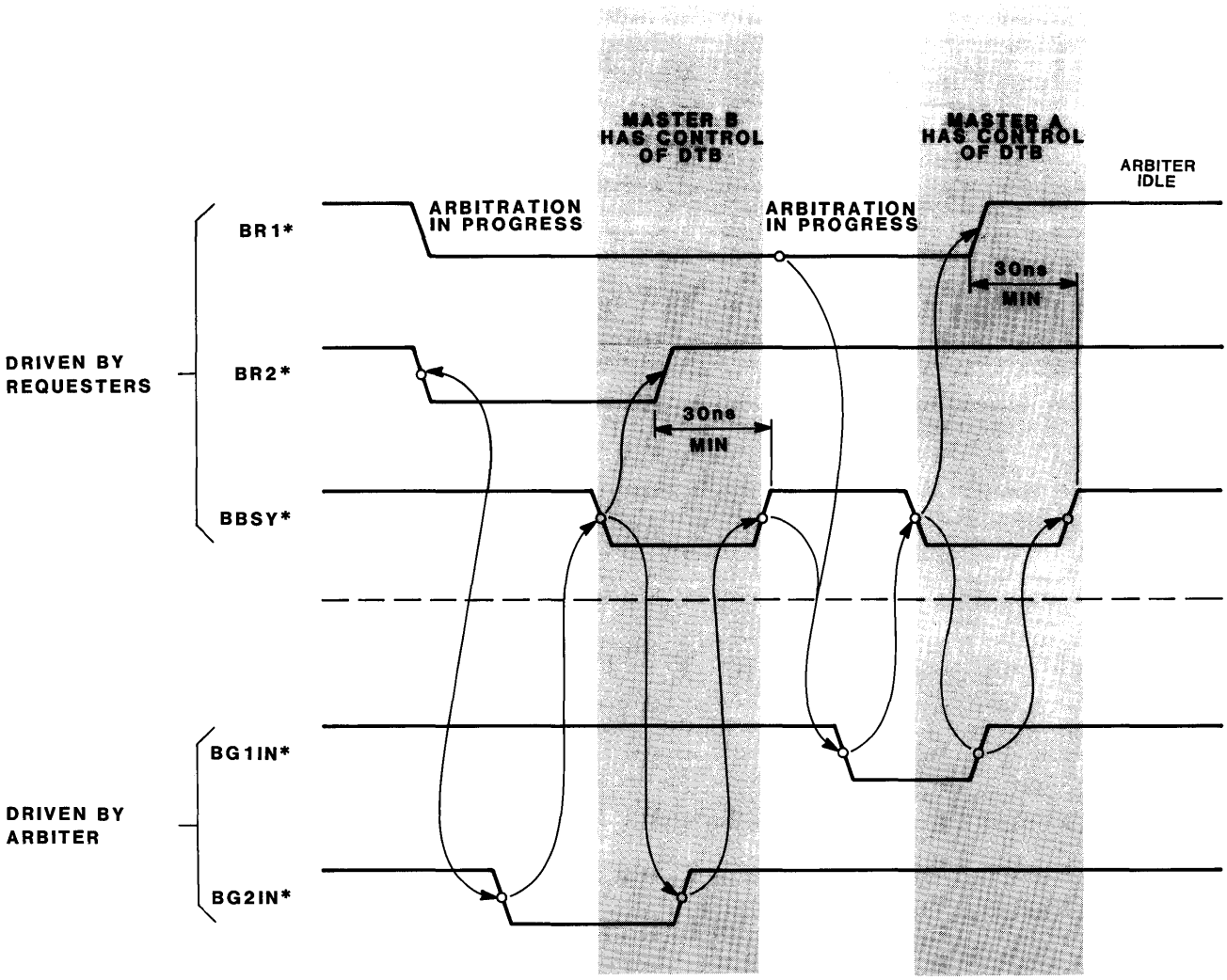


FIGURE 3-8. Arbitration Sequence Diagram: Two REQUESTERS, Two Request Levels

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.

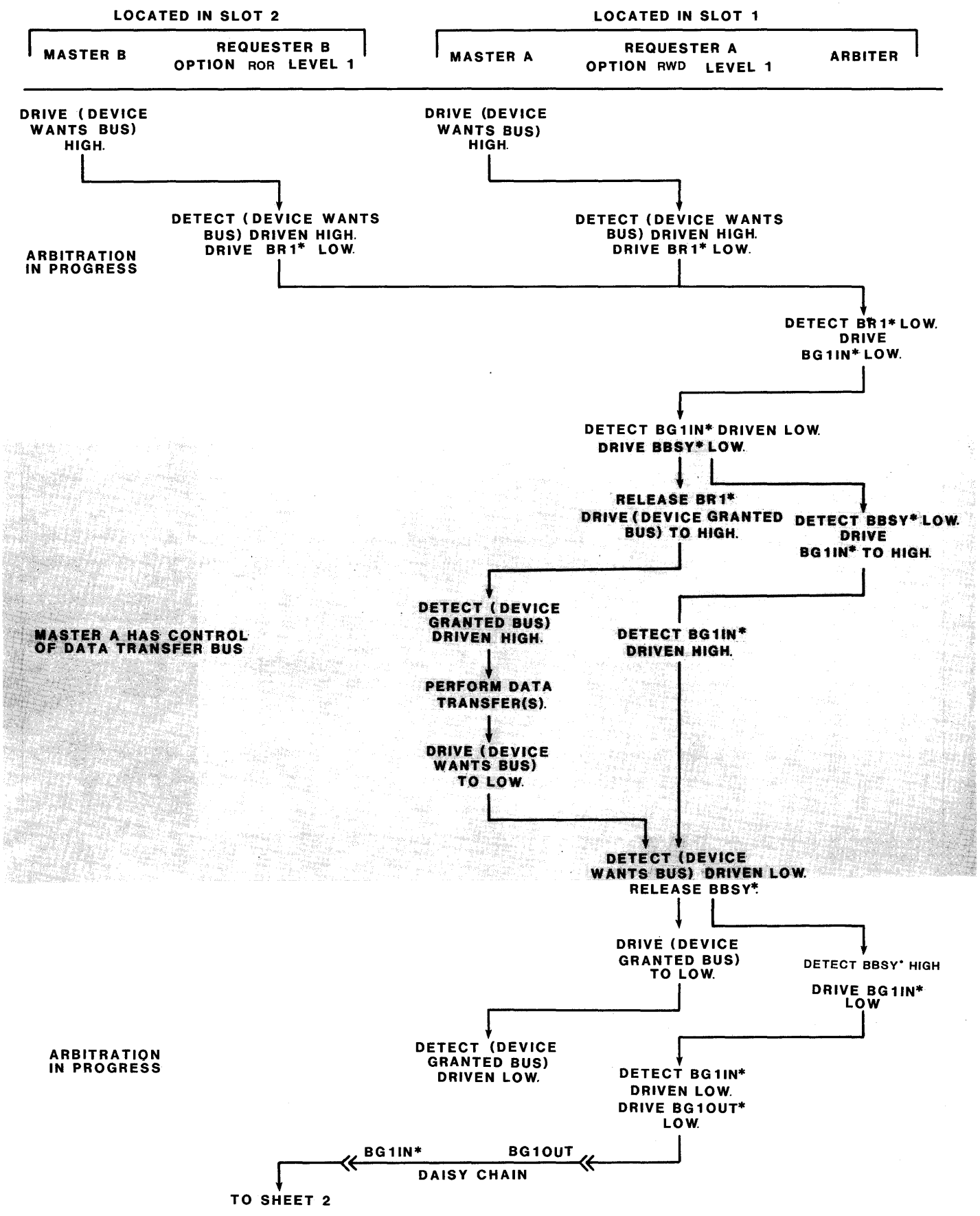


FIGURE 3-9. Arbitration Flow Diagram - Two REQUESTERS/Same Request Level (Sheet 1 of 2)

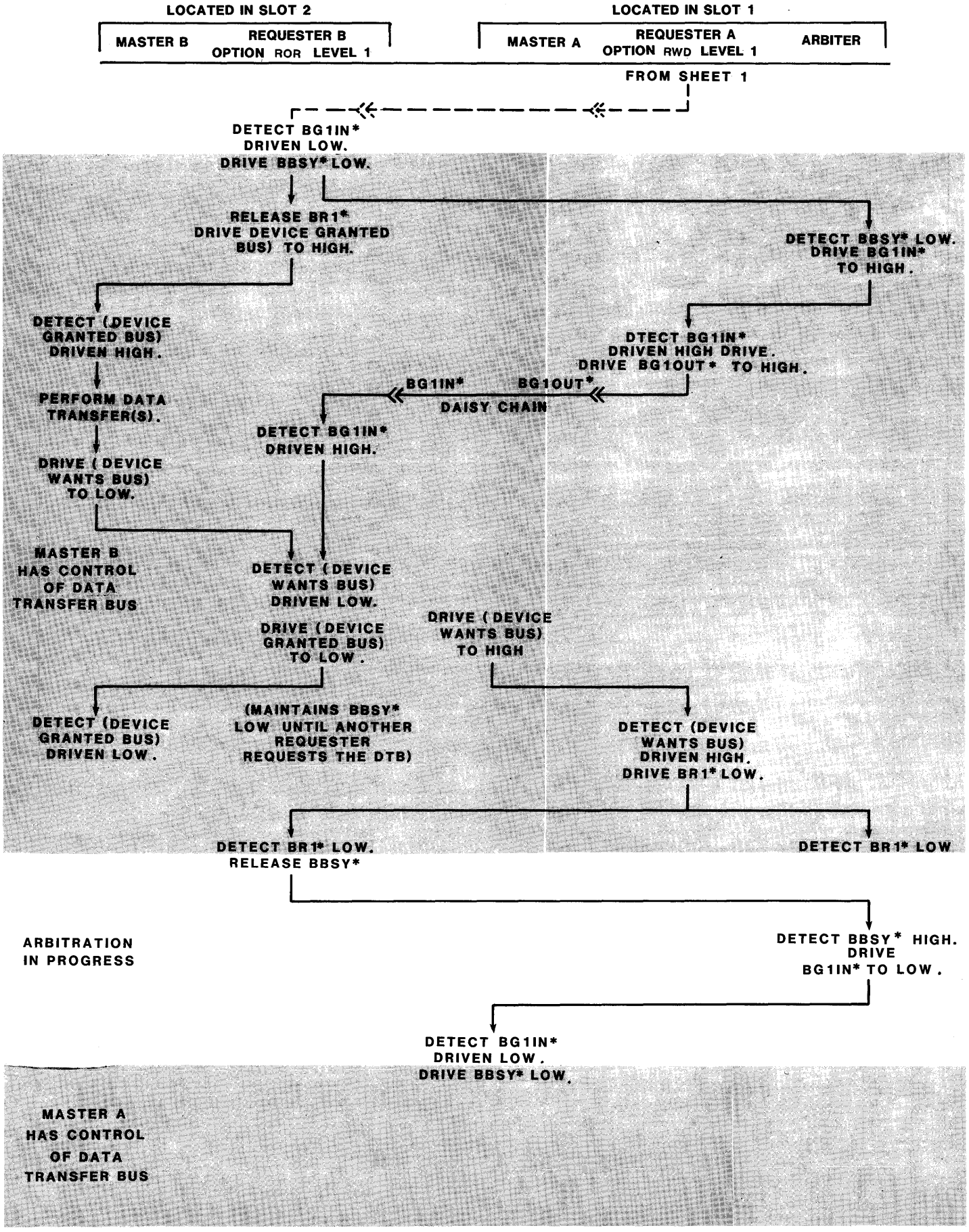
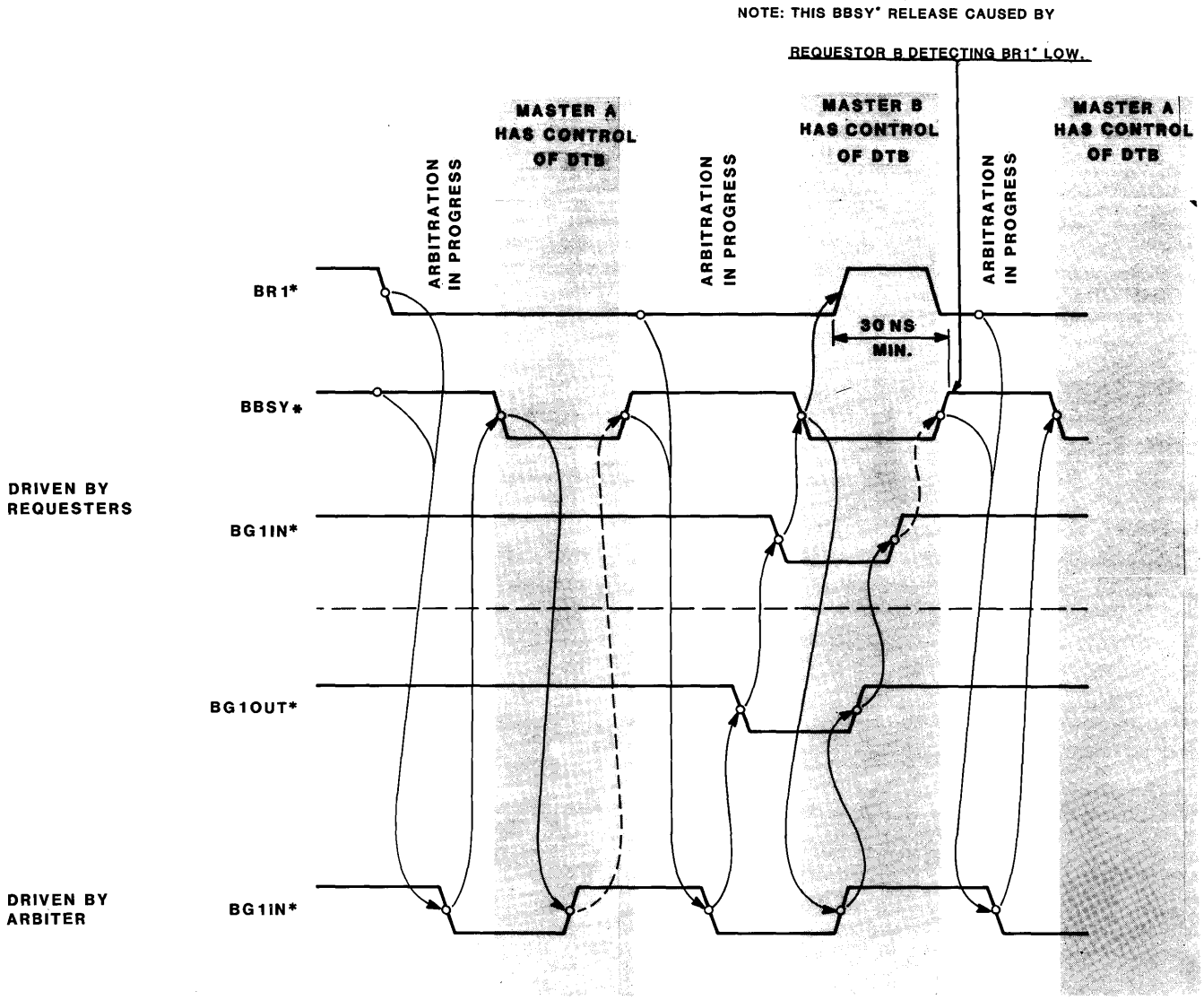


FIGURE 3-9. Arbitration Flow Diagram - Two REQUESTERS/Same Request Level (Sheet 2 of 2)



NOTE

The time indicated by the dotted arrows may be many data transfer cycles.

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

3.4.3 Arbitration During Power-Down Sequence

Upon system power-down, the EMERGENCY REQUESTER is used to accomplish two tasks:

a. It demands and acquires control of the data transfer bus so that its on-board MASTER can save the contents of volatile memory in the system.

b. It maintains control of the data transfer bus until system shutdown.

NOTE

In all cases, the EMERGENCY REQUESTER requests the bus by driving BREL* low. This is treated by the DTB ARBITER as the highest priority bus request. A low BREL* also causes all MASTERS within the system to relinquish control of the DTB within 200 microseconds.

Figures 3-11 and 3-12 show a typical power-down sequence. To provide power-down capability, the system controller board located in slot 1 must be equipped with an option PF ARBITER (with EMERGENCY REQUESTER) and a system DTB MASTER.

When the sequence begins, REQUESTER A has been granted control of the DTB. While this REQUESTER is maintaining control of the DTB by driving BBSY* low, the POWER MONITOR detects an AC failure and drives the ACFAIL* line low. When the system controller DTB MASTER in card slot 1 detects the ACFAIL* line low, it indicates to the EMERGENCY REQUESTER that it needs the DTB. The EMERGENCY REQUESTER drives BREL* low, which accomplishes two things:

a. It tells the active DTB MASTER to relinquish the DTB within 200 microseconds.

b. It tells the option PF ARBITER to grant it the bus. (BREL* is recognized by the option PF ARBITER as the highest request level -i.e., higher than BR4*).

When MASTER A detects BREL* low, within 200 microseconds it indicates to its on-board REQUESTER A that it is finished using the DTB. REQUESTER A then releases BBSY*. Upon detecting BBSY* high, the option PF ARBITER drives EMERGENCY GRANT high to its on-board EMERGENCY REQUESTER. The EMERGENCY REQUESTER then drives BBSY* low and informs the system controller DTB MASTER that the DTB is available. The system MASTER may then use the DTB to move crucial data to non-volatile memory before the system's DC power fails. Meanwhile, the ARBITER, upon detecting BBSY* low, drives EMERGENCY GRANT low. The system MASTER, after storing the crucial data, does not indicate that it is through using the DTB. Therefore, the EMERGENCY REQUESTER continues to drive BBSY* low until system shutdown. This prevents other MASTERS in the system from modifying the contents of the non-volatile memory prior to system shutdown.

NOTE

Bus interrupts cannot be handled as long as the EMERGENCY REQUESTER maintains control of the bus (e.g., during system shutdown).

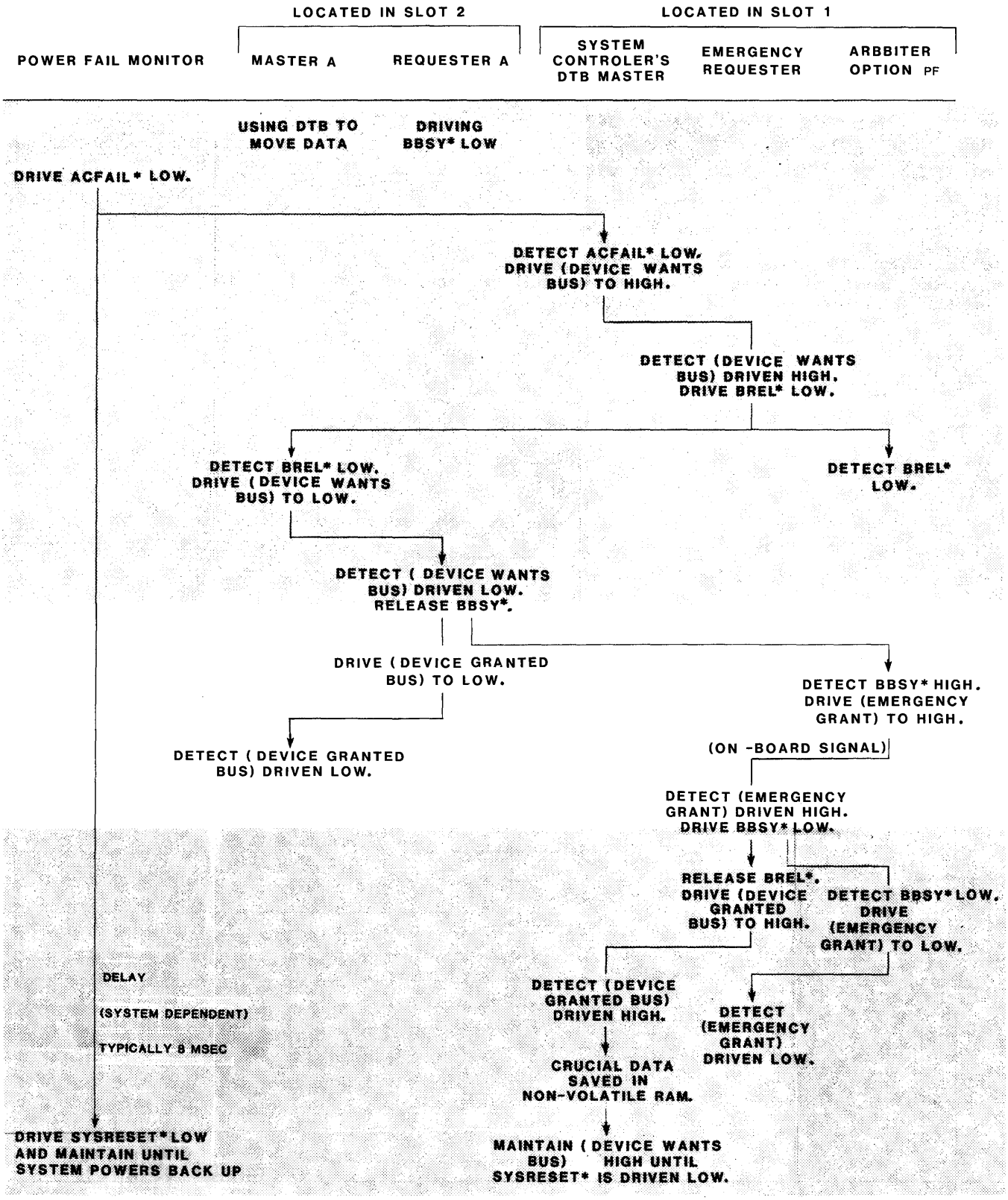


FIGURE 3-11. Power-Down Flow Diagram

Figure 3-12 is a sequence diagram showing the signal line levels on the arbitration bus and the way in which the transitions of these levels are interlocked with the on-board signals.

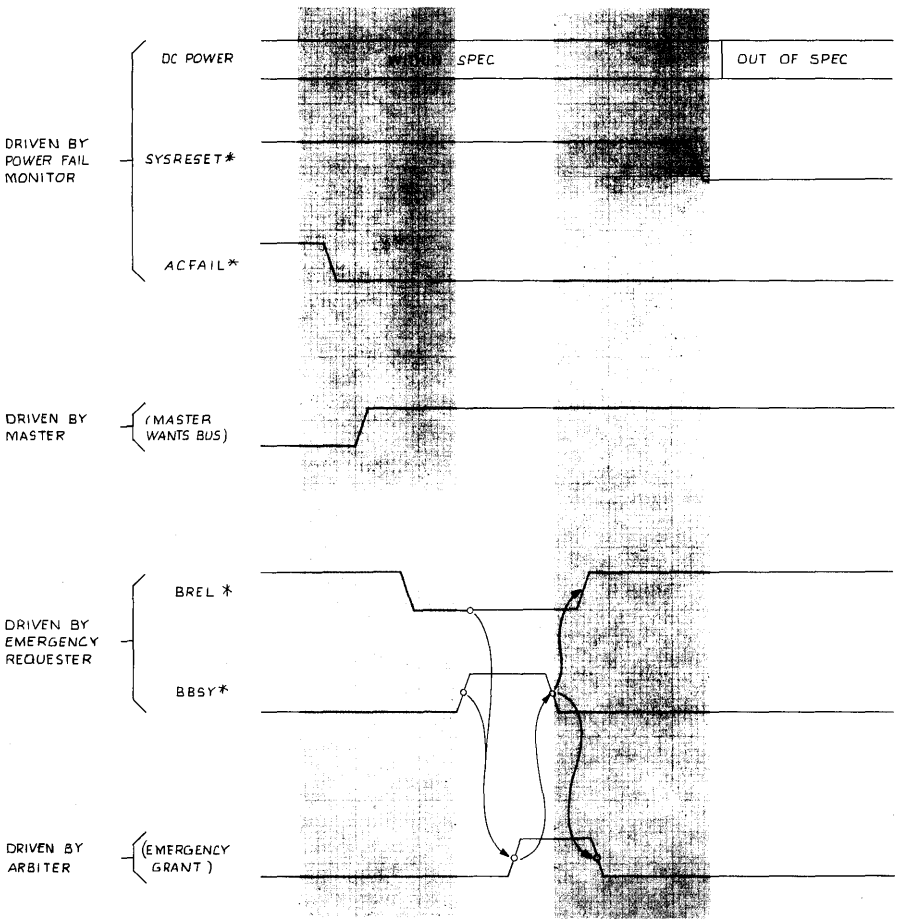


FIGURE 3-12. Power-Down Sequence Diagram

3.4.4 Arbitration During Power-Up Sequence

Upon system power-up, the EMERGENCY REQUESTER is used to accomplish two tasks:

a. It acquires control of the data transfer bus before any other MASTER in the system via the BREL* line. (This allows its on-board MASTER to run tests and to restore volatile data before system operation begins.)

b. It relinquishes the data transfer bus when its on-board MASTER is finished with its power-up sequence.

To accomplish this, BREL* is held low by the EMERGENCY REQUESTER when the system is powered back up. This causes control of the bus to be granted to the EMERGENCY REQUESTER. The DTB is then used to restore the system RAM, based upon the data stored in the non-volatile memory during the previous system shutdown. When this restoration process is complete, the EMERGENCY REQUESTER relinquishes the DTB, allowing other REQUESTERS to be granted the bus when they make a bus request.

Figures 3-13 and 3-14 show a typical power-up sequence. To provide power-up capability, the system controller board located in slot 1 must be equipped with an option PF ARBITER (with EMERGENCY REQUESTER) and a DTB MASTER.

When AC power is turned on to the system, the POWER MONITOR holds SYSRESET* low. The System Controller's EMERGENCY REQUESTER drives BREL* low as soon as the DC power within the system stabilizes (the time required for stabilization will vary from system to system). The power failure monitor continues to maintain SYSRESET* low at least 200 milliseconds beyond the time when the system DC power stabilizes. When it then releases SYSRESET*, the ARBITER samples the levels of the bus request lines (including BREL*) and arbitrates the requests. Since BREL* is the highest priority bus request, the ARBITER will drive EMERGENCY GRANT high to the on-board EMERGENCY REQUESTER.

The EMERGENCY REQUESTER, upon detecting EMERGENCY GRANT high, drives BBSY* low and drives the on-board MASTER GRANTED BUS high to indicate to the system MASTER that the DTB is available to transfer data. The system MASTER may then use the DTB to do two things:

a. It may run a power-up test on boards in the system which are not capable of testing themselves.

b. It restores crucial data from non-volatile RAM.

This latter action assures that the important data will be available in RAM when normal system operation is resumed (i.e., when the system MASTER relinquishes the DTB).

While the two activities above are taking place, the ARBITER detects the BBSY* low and drives the on-board EMERGENCY GRANT low. Only when the EMERGENCY REQUESTER detects EMERGENCY GRANT low may it release BBSY*. Therefore, the low EMERGENCY GRANT is an acknowledgement by the ARBITER that BBSY* has been detected driven low.

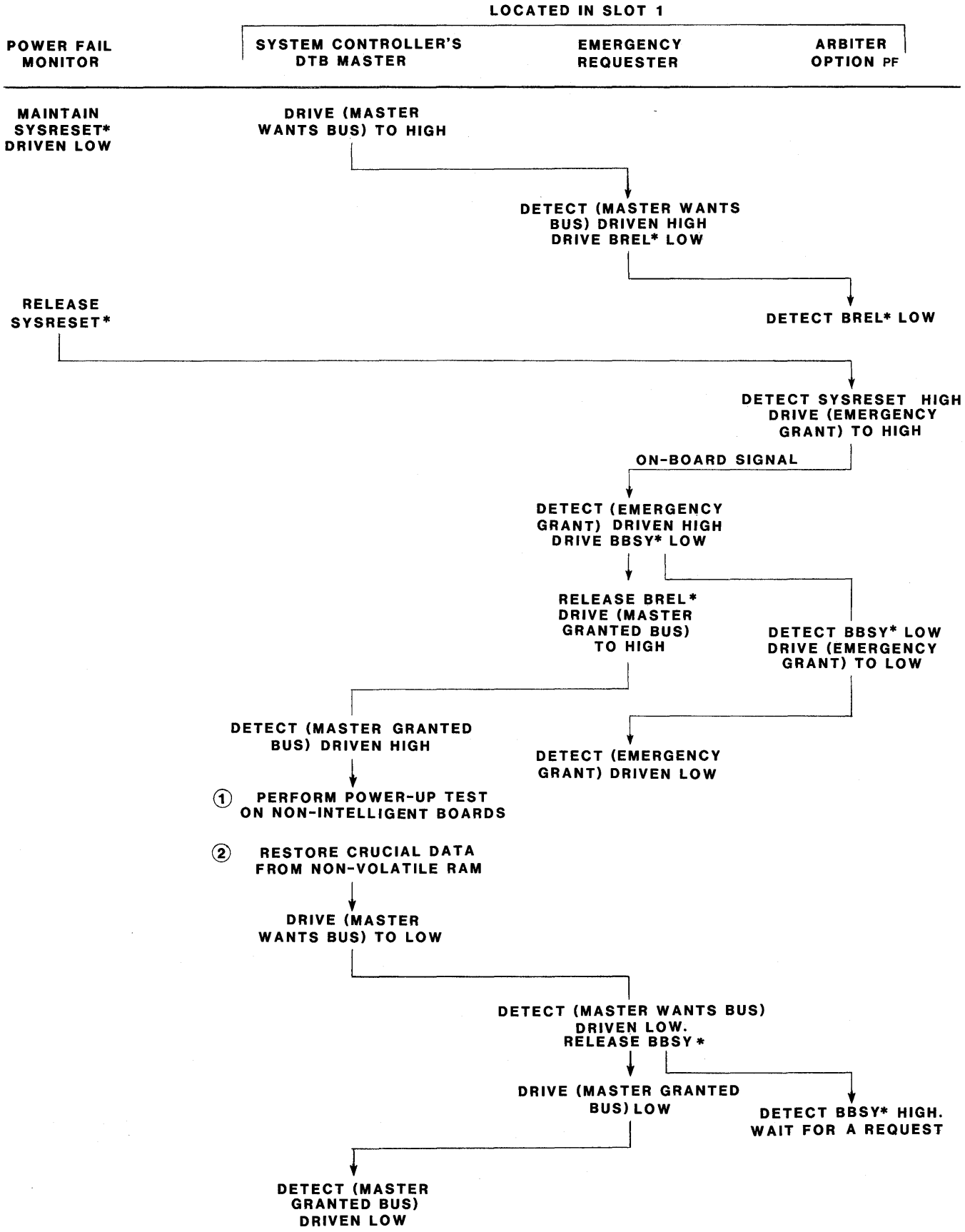


FIGURE 3-13. Power-Up Flow Diagram

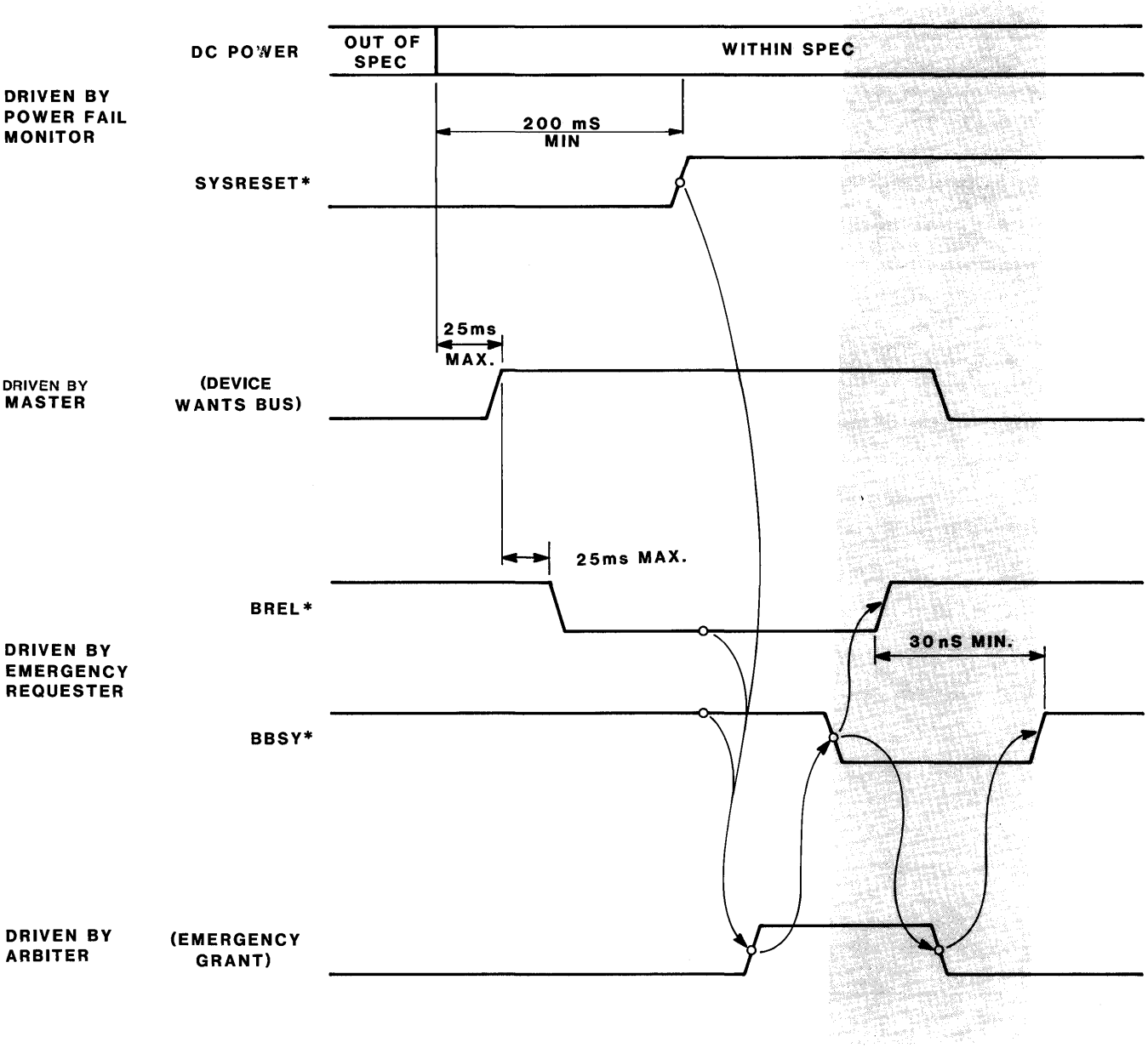


FIGURE 3-14. Power-Up Sequence Diagram

When the EMERGENCY REQUESTER detects EMERGENCY GRANT low, it may release BBSY*, provided the system MASTER drives MASTER WANTS BUS low and that 30 nanoseconds have elapsed since the release of BREL*. This 30-ns delay is required to assure that BREL* is high when the ARBITER detects BBSY* high and samples the bus request lines for a new DTB arbitration. (If BREL* were not high, it would be interpreted by the ARBITER as another bus request.) In this case, there are no other requests on the bus, so the ARBITER will continue to periodically sample the bus request lines until a request is detected.

Figure 3-14 is a sequence diagram showing the signal line levels on the arbitration bus and the way in which the transitions of these levels are interlocked with the on-board signals.

3.5 STATE DIAGRAMS

The following sections provide state diagrams for the DTB REQUESTER, DTB ARBITER, and EMERGENCY REQUESTER modules. The information provided in these diagrams is rather concentrated and will require study before it will be completely understood. If the reader is unfamiliar with the use of state diagrams, he may find it instructive to study the material in Appendix B before continuing.

3.5.1 Data Transfer Bus REQUESTER.

Figure 3-15 shows the state diagram for a data transfer bus REQUESTER.

The state diagram shows all allowed transitions which a REQUESTER may make. REQUESTERS may make only those transitions shown on the diagram and may enter only those states shown on the diagram.

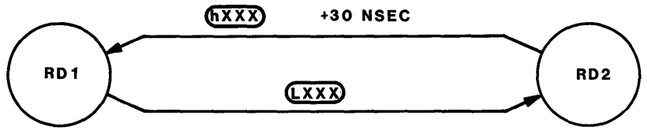
Two on-board signals are used to communicate between the DTB REQUESTER and the on-board MASTER or INTERRUPT HANDLER that wishes to obtain control of the DTB:

a. DEVICE WANTS BUS - signal from the on-board DTB MASTER or INTERRUPT HANDLER which causes the REQUESTER to request the DTB when it is high and causes the REQUESTER to relinquish the DTB when it is low.

b. DEVICE GRANTED BUS - signal to the on-board DTB MASTER or INTERRUPT HANDLER indicating, when it is high, that the DTB may be used to transfer data and indicating, when it is low, that the DTB has been relinquished.

Figure 3-16 is a sequence diagram showing a typical sequence for obtaining the bus. Refer to Figures 3-15 and 3-16 for the following discussion.

Assume the REQUESTER is in state hhHL (i.e., a system reset has just occurred). When the on-board device indicates a need for the data transfer bus, the REQUESTER moves into state LhHL and drives BRx* low. The delay state diagram then makes a transition to state RD2. The REQUESTER must then wait until it detects its BGxIN* being driven low by the ARBITER. In this example, when BGxIN* is detected low, the REQUESTER makes a transition to state LLHL, driving the BBSY* low. Since the delay diagram is in state RD2, the REQUESTER then makes a transition to state hLHL, releasing the bus request line. Assuming that



THE STATES SHOWN IN THE DIAGRAM ABOVE DON'T RESULT IN ANY OUTPUT CHANGES. (THEY REPRESENT THE STATE OF AN INTERNAL MONOSTABLE FLIP-FLOP.) THEY ARE LABELLED SIMPLY "REQUESTER DELAY STATES 1 AND 2."

THE STATES ON THE DIAGRAM BELOW ARE LABELLED ACCORDING TO THE LEVELS OF THE REQUESTER'S OUTPUT LINES:

BRX*	BBSY*	BGXOUT*	DEVICE GRANTED BUS

NOTES:

1 TRANSITION REQUIREMENT IN BRACKETS [ ] APPLIES TO OPTION ROR REQUESTER ONLY.
2 ∧ = LOGICAL "AND"
3 ∨ = LOGICAL "OR"
4 IN "BGXIN*" X=0,1,2,3, OR 4: (BUS ARBITRATION PRIORITY LEVEL)
5 A REQUEST IS PENDING IF ANY OF THE BUS REQUEST LINES OR IF THE BUS RELEASE LINE IS LOW
6 "h" IS USED TO INDICATE THAT THIS SIGNAL LINE MAY BE DRIVEN LOW BY OTHER OPEN COLLECTOR DRIVERS ON VERSAbus

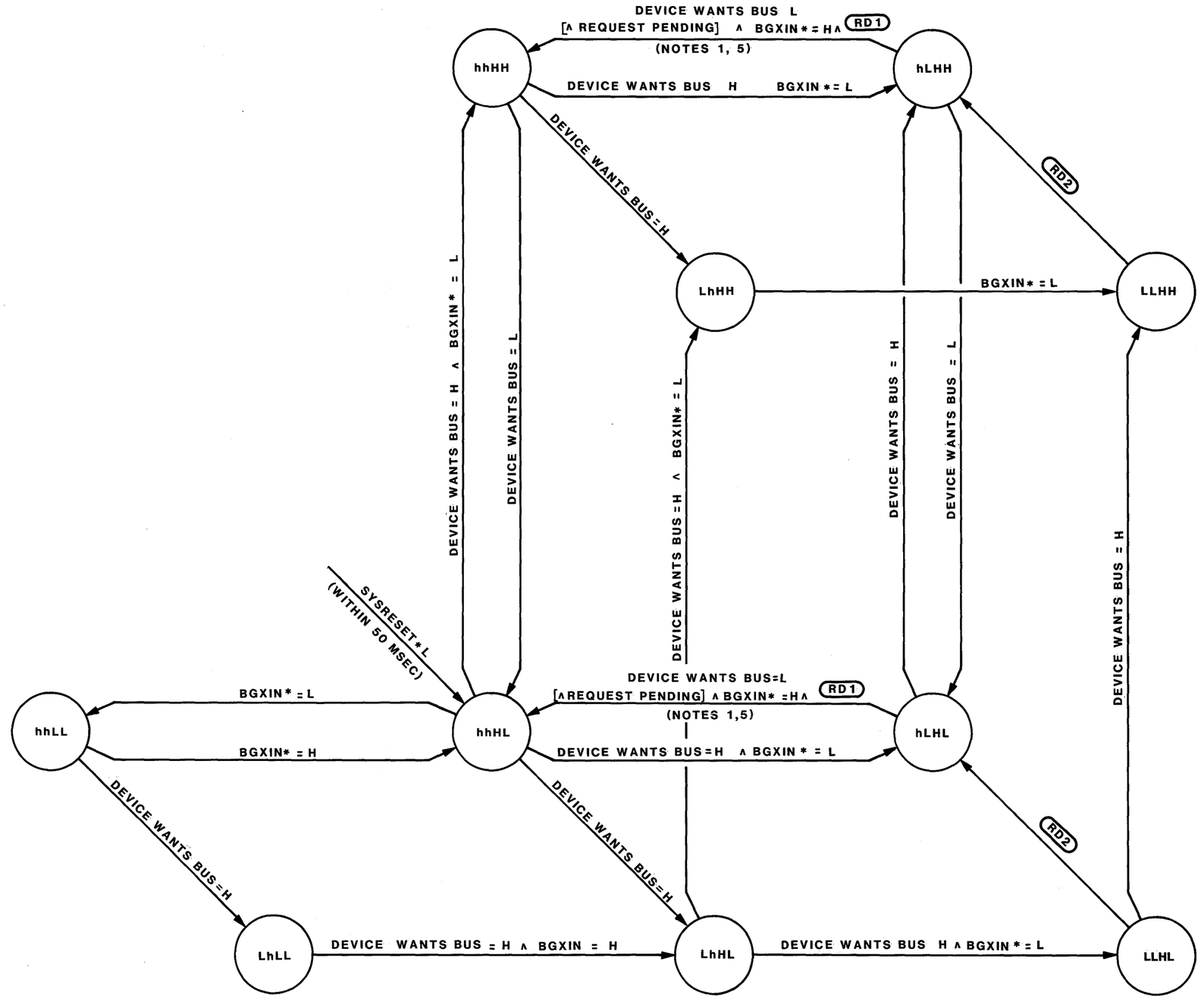
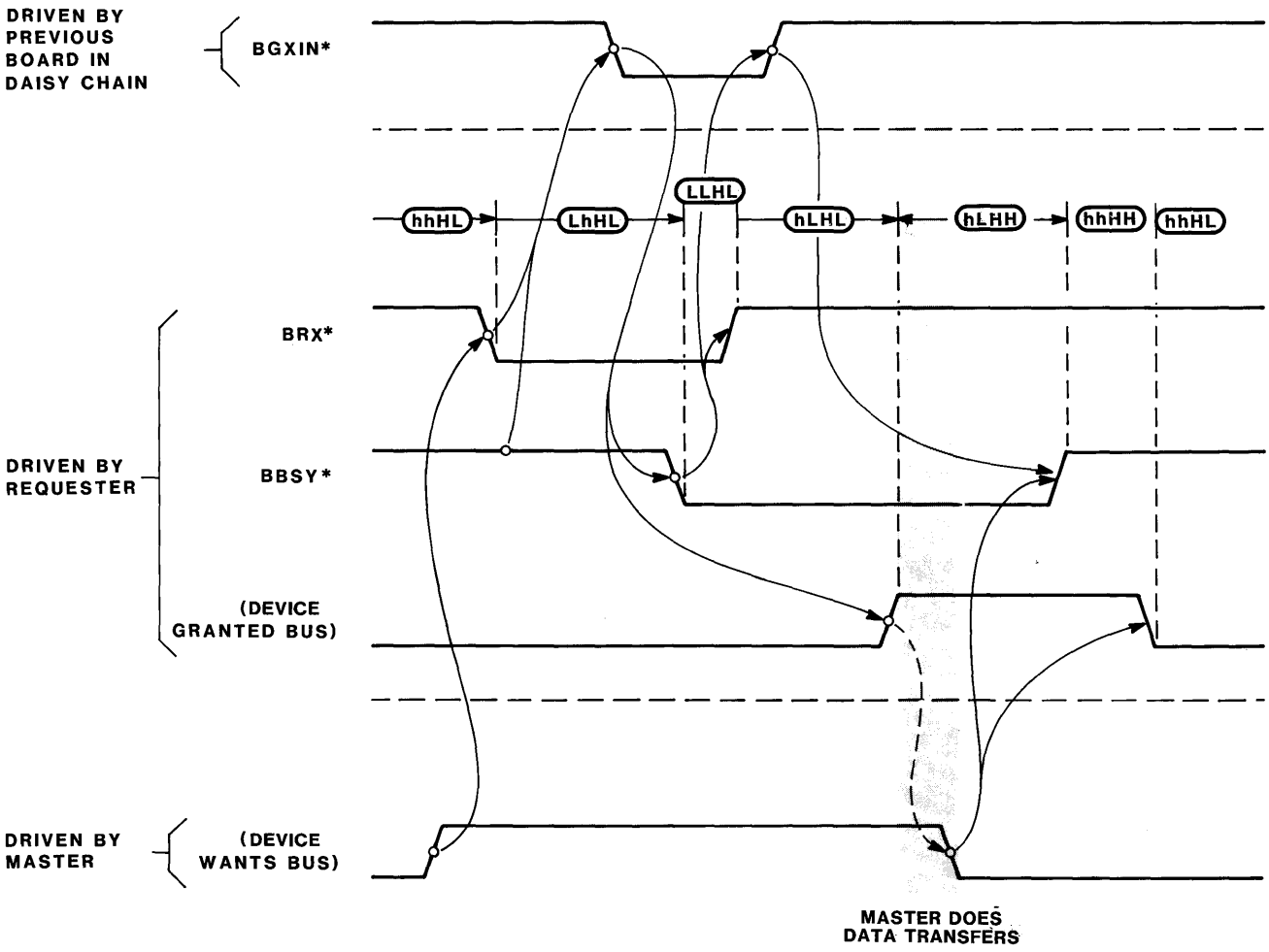


FIGURE 3-15. DTB REQUESTER State Diagram



NOTE

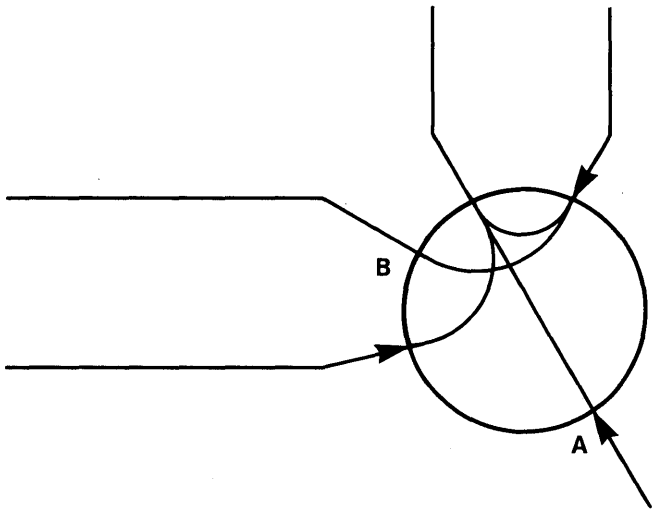
The time indicated by the dotted arrows may be many data transfer cycles.

FIGURE 3-16. Sequence Diagram: Typical Sequence for Requesting the DTB

DEVICE WANTS BUS is still high (which it must be because the MASTER is required to maintain it high until a high DEVICE GRANTED BUS is received), a transition is then made to state hLHH, driving the DEVICE GRANTED BUS signal high. This indicates to the MASTER or INTERRUPT HANDLER that the DTB may be used. When the data transfers are complete, the MASTER or INTERRUPT HANDLER drives DEVICE WANTS BUS low. Assuming that the ARBITER has driven the BGxIN* to high and assuming that this high has propagated down the daisy-chain, a transistion is then made to state hhHH, releasing BBSY*. Assuming that DEVICE WANTS BUS remains low (which it must because the MASTER is required to maintain it low until a low DEVICE GRANTED BUS is received), a transition is then made to state hhHL, driving DEVICE GRANTED BUS low.

The reader should note that this is only one example of how the REQUESTER might traverse its state diagram. Other paths might have been followed. For example, instead of going from LLHL to hLHL and hLHH, the REQUESTER might have gone from LLHL to LLHH and hLHH. In the latter case, the DEVICE GRANTED BUS signal would have been driven high before the bus request line was released, instead of the sequence shown in Figure 3-16. While this may appear to complicate the state diagram, it actually represents an addition of design freedom and will make the logic design simpler.

Lines shown within a state circle represent legal logic flows. These constraints are not part of the logic of the module being described, but show the only paths which may be taken due to the interlocked relationship with another module. As an example, consider this expanded portion of state hLHL.



To enter at point A, we have the condition previously met that DEVICE WANTS BUS is high. To exit via point B, we need the condition that DEVICE WANTS BUS is low. An external constraint exists which defines an interlock condition. Once a MASTER has asserted DEVICE WANTS BUS, it may not release the signal until it sees DEVICE GRANTED BUS. Since no logical path which enters at point A has provided DEVICE GRANTED BUS, the logic is constrained from flowing from A to B. State circles containing only one entry or exit do not require these lines. State circles containing multiple entry and exit points, but no flow lines, indicate that any point of exit is possible for any point of entry.

States hhLL and LhLL are associated with driving BGxOUT* low. All REQUESTERS which share a common bus request line are daisy-chained. Each must be capable of passing on a bus grant if it is not requesting the DTB. If the REQUESTER detects a low on BGxIN* while it is in state hhHL, it may move to the state hhLL and drive BGxOUT* low. On the other hand, if the on-board MASTER or INTERRUPT HANDLER indicates it wants the DTB prior to the transition to hhLL, the REQUESTER can make the transition to LhHL and then take control of the DTB. In either case, upon detecting BGxIN* low, a REQUESTER will pass the grant by driving BGxOUT* low or obtain the bus by driving BBSY* low. Figure 3-17 shows a timing sequence for the case where the REQUESTER drives its BGxOUT* low.

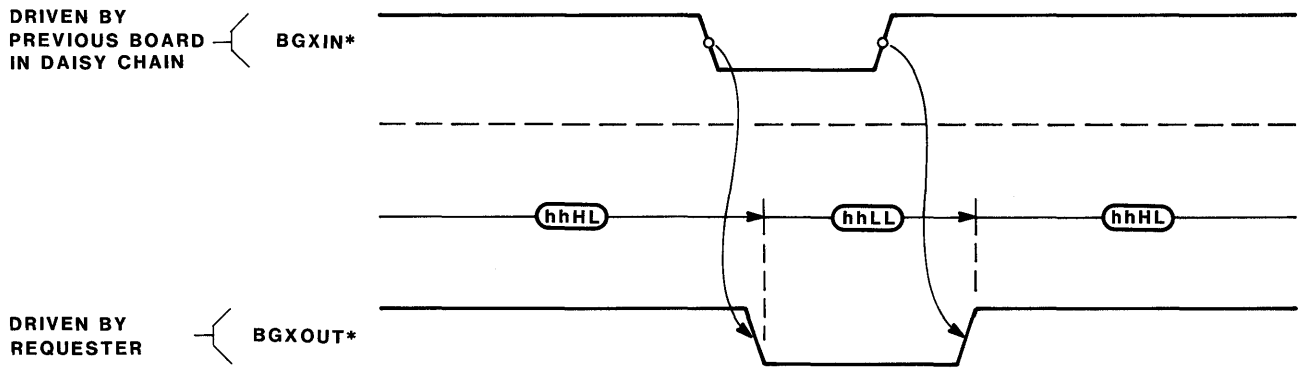


FIGURE 3-17. Sequence Diagram: REQUESTER Drives BGxOUT*

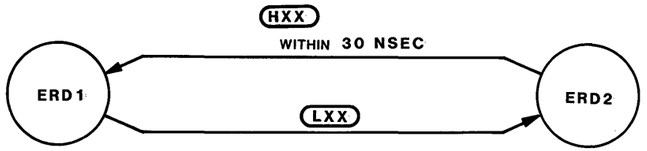
NOTE

The transition shown from hhHL to hLHL allows the REQUESTER to take control of the DTB if its on-board MASTER has indicated a need for the DTB even though the REQUESTER has not yet driven the bus request line low. While this may seem a little startling, it is entirely logical. The bus request line which it would be driving low is already low, since this is the only way it could detect a low on its BGxIN*. In fact, the user observing the arbitration bus lines would not be able to say with any certainty whether this particular REQUESTER drove BRx* low. Since the low BGxIN* is not passed along the daisy-chain to the next board, the REQUESTER driving the BRx* low will continue to hold it low until another arbitration takes place.

One internal timing restriction is placed upon the REQUESTER. This is defined by the small state diagram in the upper left corner of Figure 3-15. The REQUESTER must wait 30 nanoseconds after releasing its bus request line before releasing BBSY*. This assures that when the DTB ARBITER detects BBSY* high and samples the level of the bus request lines, it will not misinterpret the old bus request and grant the data transfer bus to the same REQUESTER again.

The MASTER or INTERRUPT HANDLER may indicate to its on-board REQUESTER that it is finished using the DTB after it has driven AS* to low on its last data transfer cycle. If the REQUESTER then immediately releases BBSY*, the arbitration to determine the next active bus MASTER may occur during the last data transfer cycle. This saves bus time, since the next active DTB MASTER may begin using the data transfer bus as soon as the cycle completes (i.e., as soon as AS* goes high).

The EMERGENCY REQUESTER functions similarly to a normal REQUESTER. However, it does not have a daisy-chained grant line. This results in the simplified state diagram, Figure 3-18.



THE STATES SHOWN IN THE DIAGRAM ABOVE DON'T RESULT IN ANY OUTPUT CHANGES. (THEY REPRESENT THE STATE OF AN INTERNAL MONOSTABLE FLIP-FLOP.) THEY ARE LABELLED SIMPLY "EMERGENCY REQUESTER DELAY STATE 1 AND 2'

THE STATES IN THE DIAGRAM TO THE RIGHT ARE LABELLED ACCORDING TO THE LEVELS OF THE EMERGENCY REQUESTER'S OUTPUT LINES :

BREL*	BBSY*	DEVICE GRANTED BUS

NOTES:

1 ∧ = LOGICAL "AND"

2 ∨ = LOGICAL "OR"

3 DOTTED LINES INDICATE TRANSITIONS WHICH WILL NEVER OCCUR UNDER NORMAL OPERATION, BUT MAY BE IMPLEMENTED TO SIMPLIFY LOGIC DESIGN

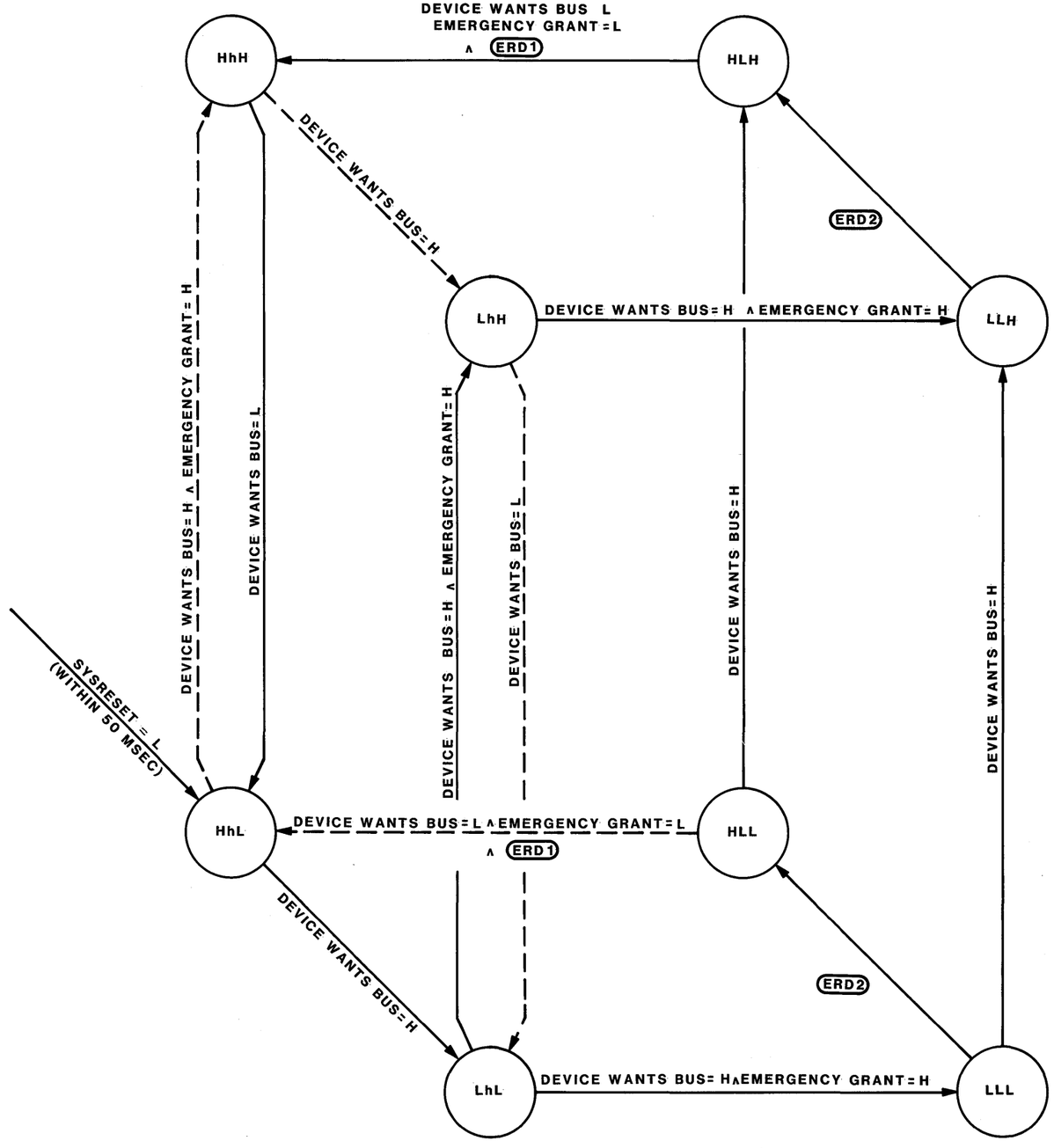


FIGURE 3-18. EMERGENCY REQUESTER State Diagram

3.5.2 Data Transfer Bus ARBITER

As mentioned in paragraph 3.3.1, the ARBITER is used to accomplish two tasks:

a. to prioritize incoming requests for the data transfer bus and issue a grant to the highest priority REQUESTER, and

b. to inform the active data transfer MASTER whenever a higher level bus request is pending.

Refer to the ARBITER state diagram (Figure 3-19) and sequence diagram (Figure 3-20) for the following discussion.

3.5.2.1 Prioritizing of Incoming Bus Requests

Assume that the ARBITER has come up from a reset condition and is in state HHHHH/LHR with all of its inputs (BR0*-BR4* and BBSY*) high. From this state, and upon receiving a bus request, the option NPF ARBITER may enter any one of five states: LHHHH/LHR, HLHHH/LHR, HHLHH/LHR, HHHLH/LHR, HHHHL/LHR. Entering any of these states from HHHHH/LHR causes a bus grant line to be driven low. Note that the conditions which allow these transitions effectively prioritize the incoming requests. For example, if BR3* is being driven low and the higher priority bus requests are not being driven low, the transition from state HHHHH/LHR to HHHLH/LHR will be made. In state HHHLH/LHR, BG3IN* is driven low. The REQUESTER on level 3 will acknowledge receipt of the BG3IN* signal by driving BBSY* low, thus causing the ARBITER to leave state HHHLH/LHR, enter HHHHH/LHS, and release BG3IN*. When the REQUESTER driving BBSY* no longer needs the DTB, it will release BBSY*. The ARBITER will then leave state HHHHH/LHS and go back to state HHHHH/LHR. When the ARBITER enters state HHHHH/LHR, it will once again grant the DTB to the highest level REQUESTER; however, if no requests are pending, the ARBITER will remain in state HHHHH/LHR until a bus request line is driven low.

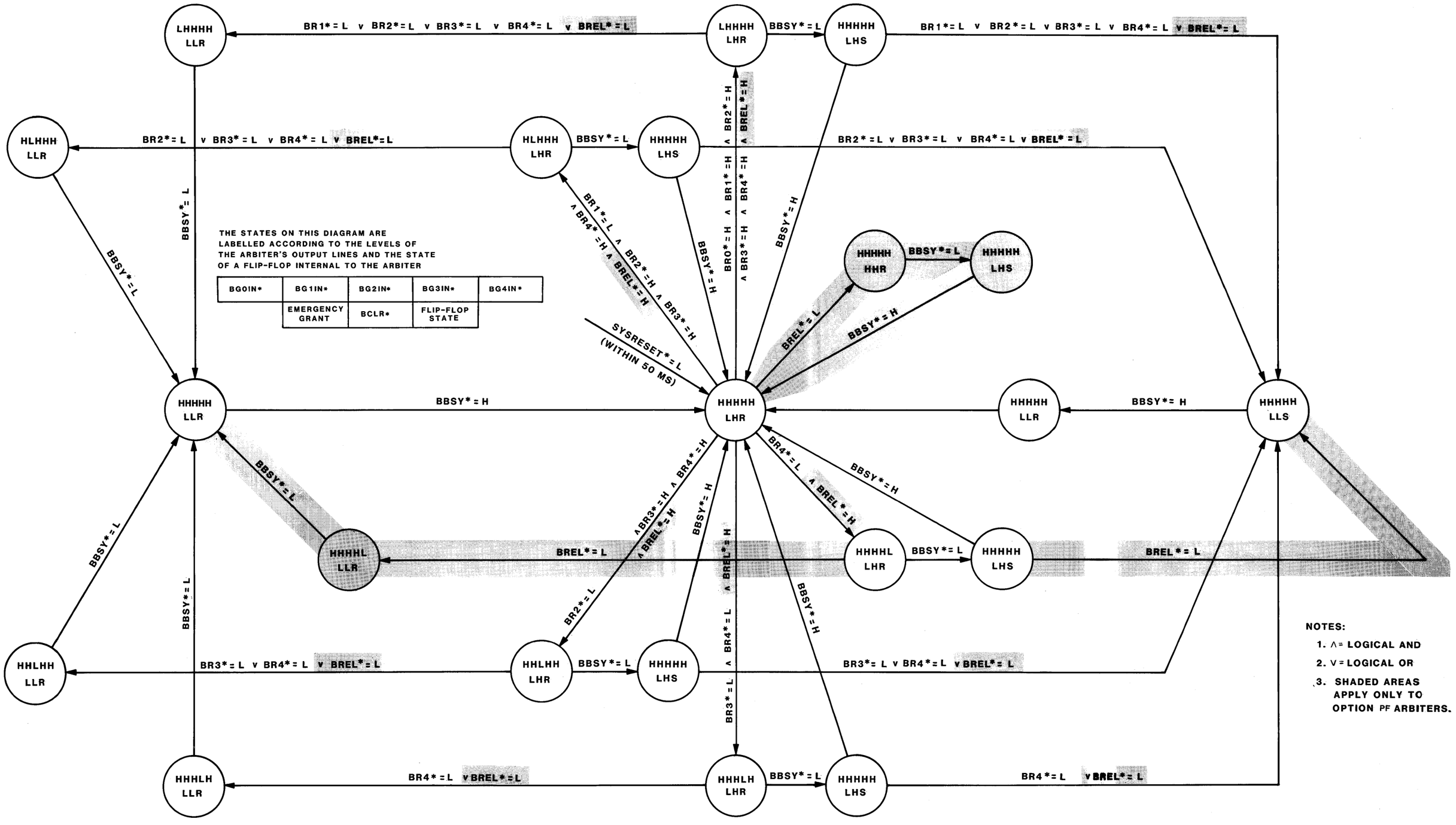


FIGURE 3-19. ARBITER State Diagram

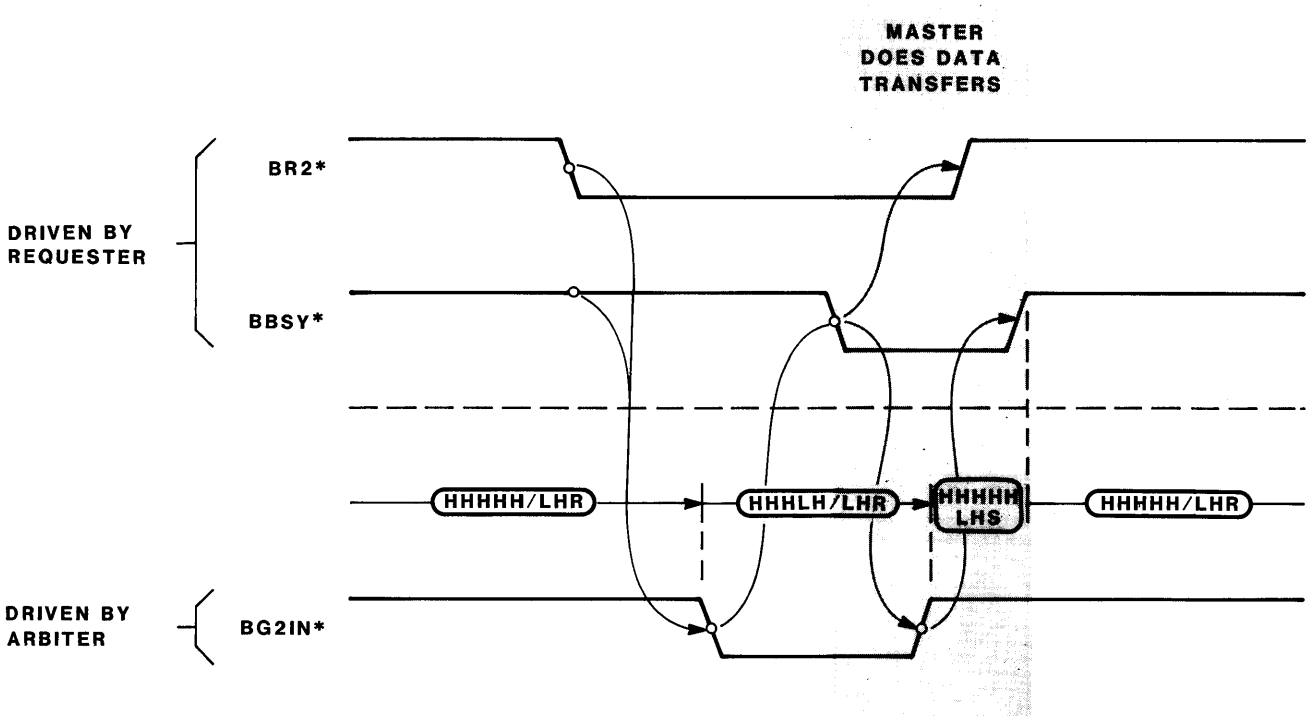


FIGURE 3-20. Sequence Diagram: Arbitration

3.5.2.2 Clearing the DTB Upon a Higher Priority Bus Request

The reader should refer to Figure 3-21 for an example showing how the ARBITER drives BCLR* low when it detects that a higher priority bus request is pending.

If the higher priority bus request line is driven low while the ARBITER is still holding a BGxIN* line low, a transition will be made to state LHHHH/LLR, HLHHH/LLR, HHLHH/LLR, HHHLH/LLR, or HHHHL/LLR, according to the respective bus grant line being driven. This causes the ARBITER to drive BCLR* low. (State HHHHL/LLR will be entered only for an option PF ARBITER.) When BBSY* is driven low by the REQUESTER in response to the bus grant being low, state HHHHH/LLR will be entered. When the REQUESTER releases BBSY*, the ARBITER will return to state HHHHH/LHR and release BCLR*.

If a higher priority bus request line is driven low after the bus grant line has been released, state HHHHH/LLS will be entered. This again causes the ARBITER to drive BCLR* low. Again, the ARBITER will return to state HHHHH/LHR (releasing BCLR*) when the REQUESTER releases BBSY*.

The option PF ARBITER monitors BREL* and recognizes this signal as the highest priority request. The option PF ARBITER will drive the on-board EMERGENCY GRANT line high in response to a low BREL* level. HHHHH/HHR, HHHHH/LHS, and HHHHL/LLR are the three additional states for this option.

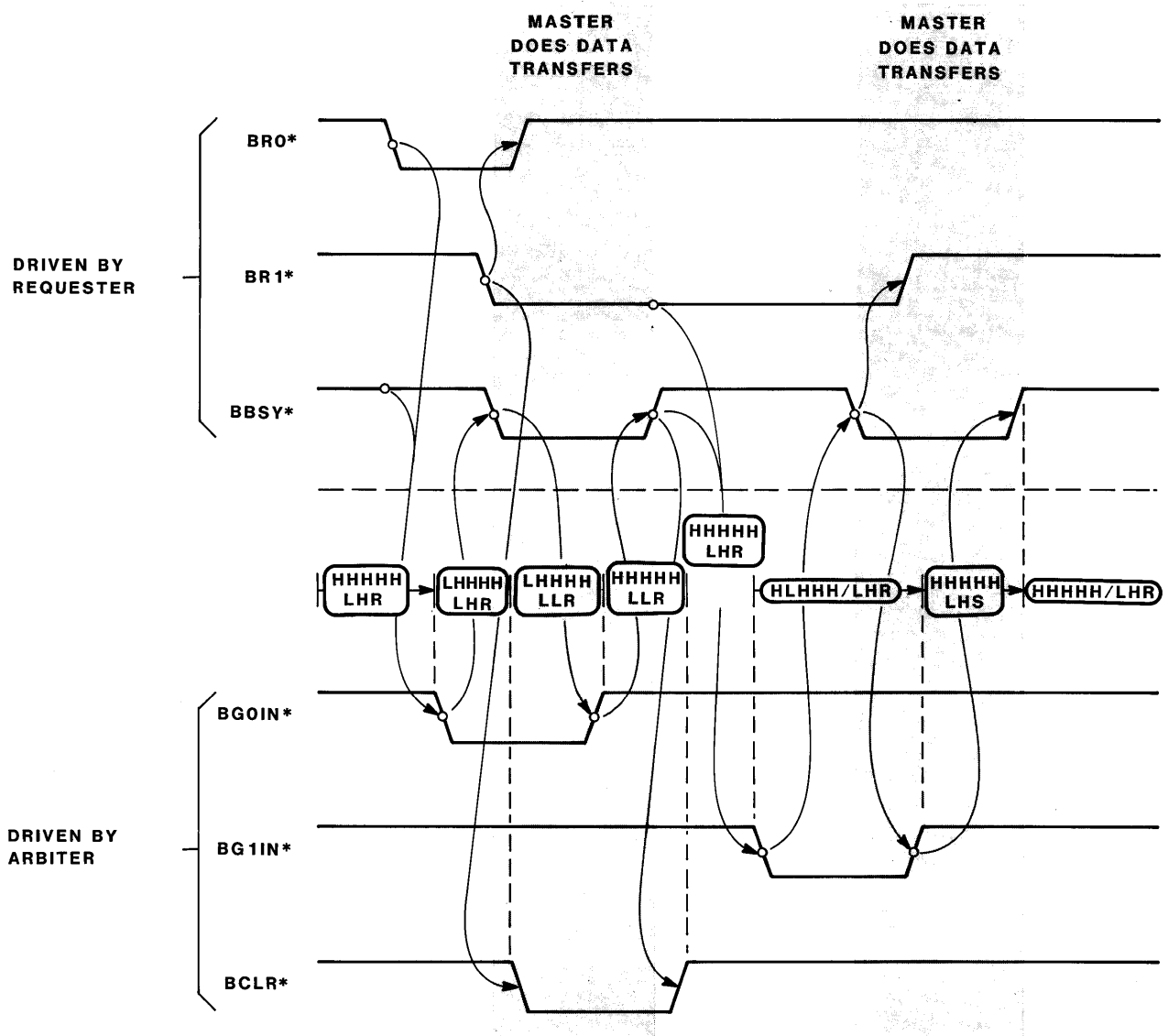


FIGURE 3-21. Sequence Diagram: Clearing the DTB Upon a Higher Priority Request

CHAPTER 4

PRIORITY INTERRUPT

CHAPTER 4

PRIORITY INTERRUPT

4.1 INTERRUPT PHILOSOPHY

Multiple processor systems require a much more sophisticated interrupt handling structure than single processor systems. A bus designed to support such systems must have a very flexible interrupt subsystem protocol.

Multiple processor 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.

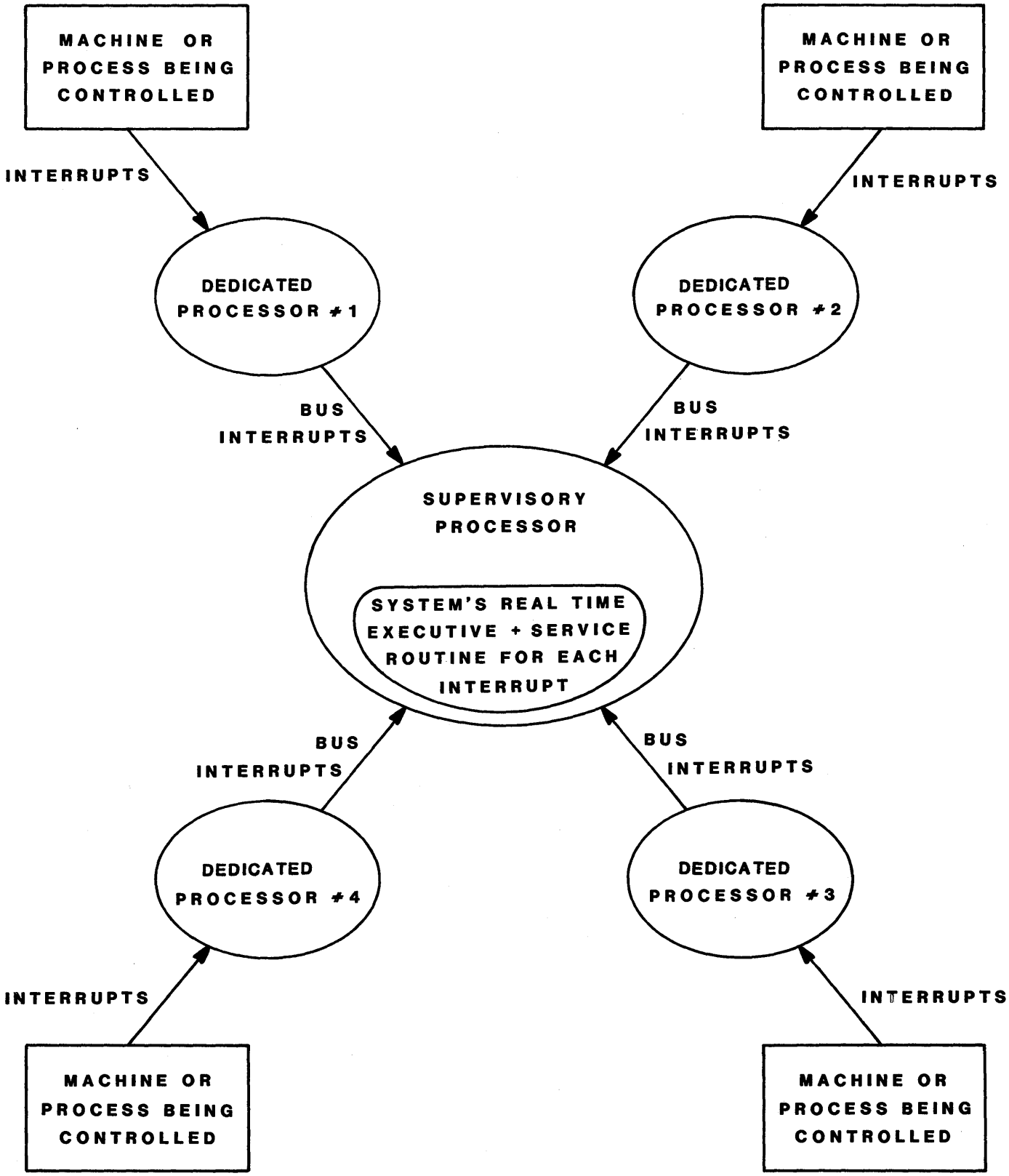


FIGURE 4-1. Interrupt Subsystem Structure: Single Handler System

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 and one daisy-chain signal line:

IRQ1*
IRQ2*
IRQ3*
IRQ4*
IRQ5*
IRQ6*
IRQ7*
ACKIN*/ACKOUT*

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).

4.2.2 Acknowledge Daisy Chain - ACKIN*/ACKOUT* (Data Transfer Bus)

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 ACKNOWLEDGE DAISY CHAIN. This daisy-chain line passes through each board on the VERSAbus. When an interrupt is acknowledged, ACKIN* 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.

ACKIN* will be low for 128 of the 256 possible address modifier codes. Therefore, ACKIN* low alone is insufficient to indicate interrupt acknowledge. The interrupt acknowledge code must also be present on the address modifier lines (hex 27). Failure to decode the address modifier lines will lead to improper bus operation.

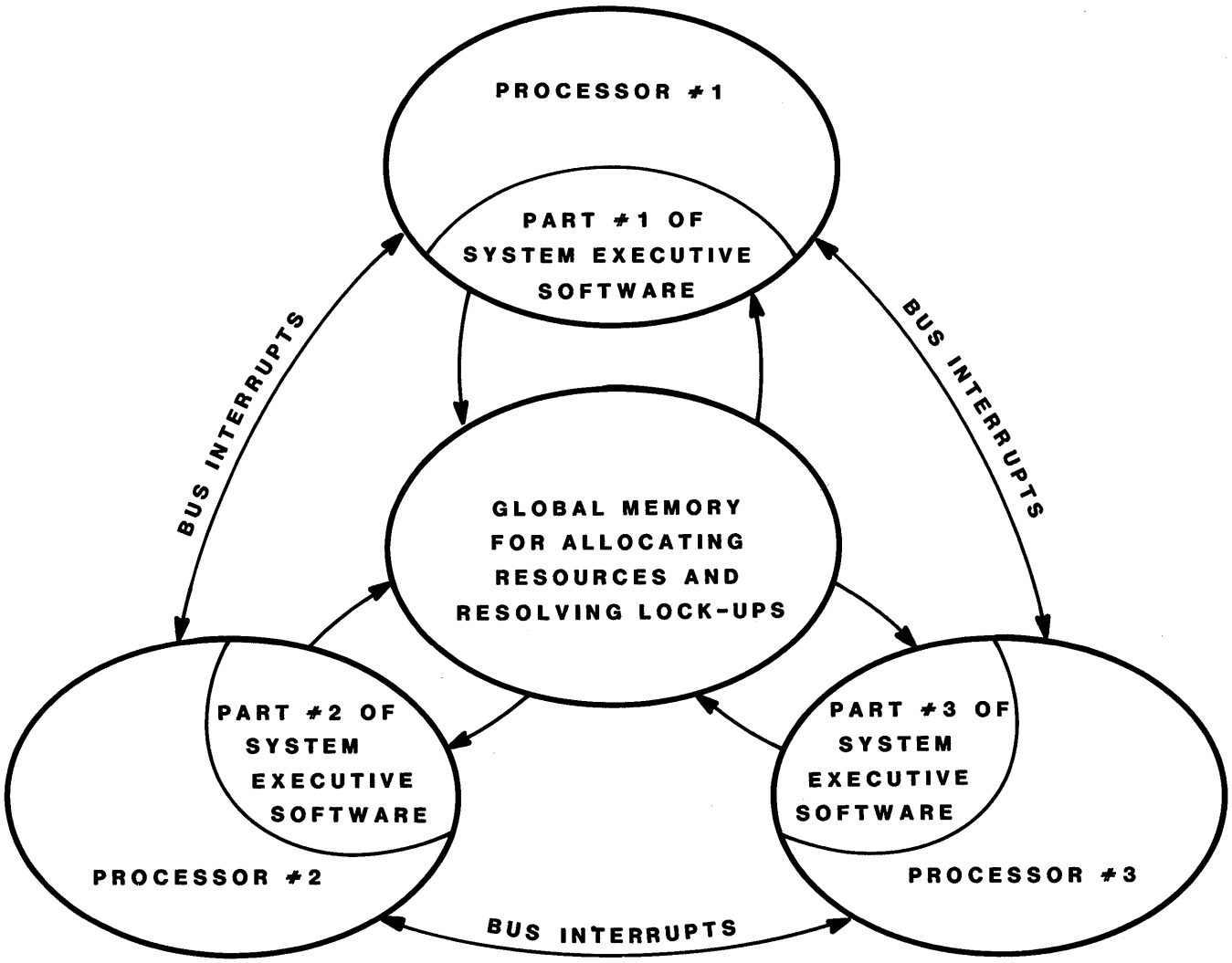


FIGURE 4-2. Interrupt Subsystem Structure: Distributed System

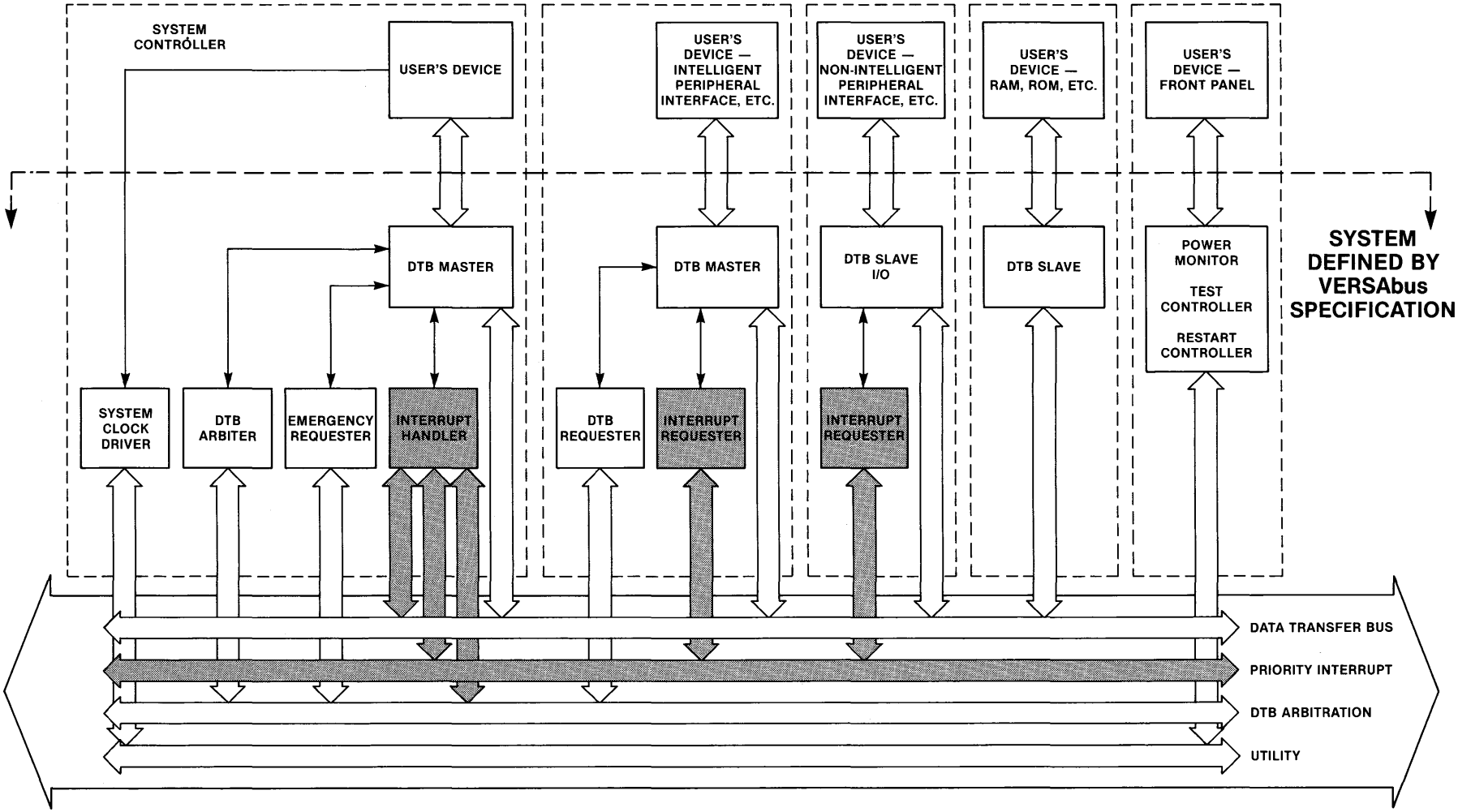


FIGURE 4-3. VERSAbus Priority Interrupt Functional Block Diagram

4.3 FUNCTIONAL MODULES

Section 4.2 discussed the additional lines on VERSAbus 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 VERSAbus 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 VERSAbus.

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 VERSAbus 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 VERSAbus.

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.

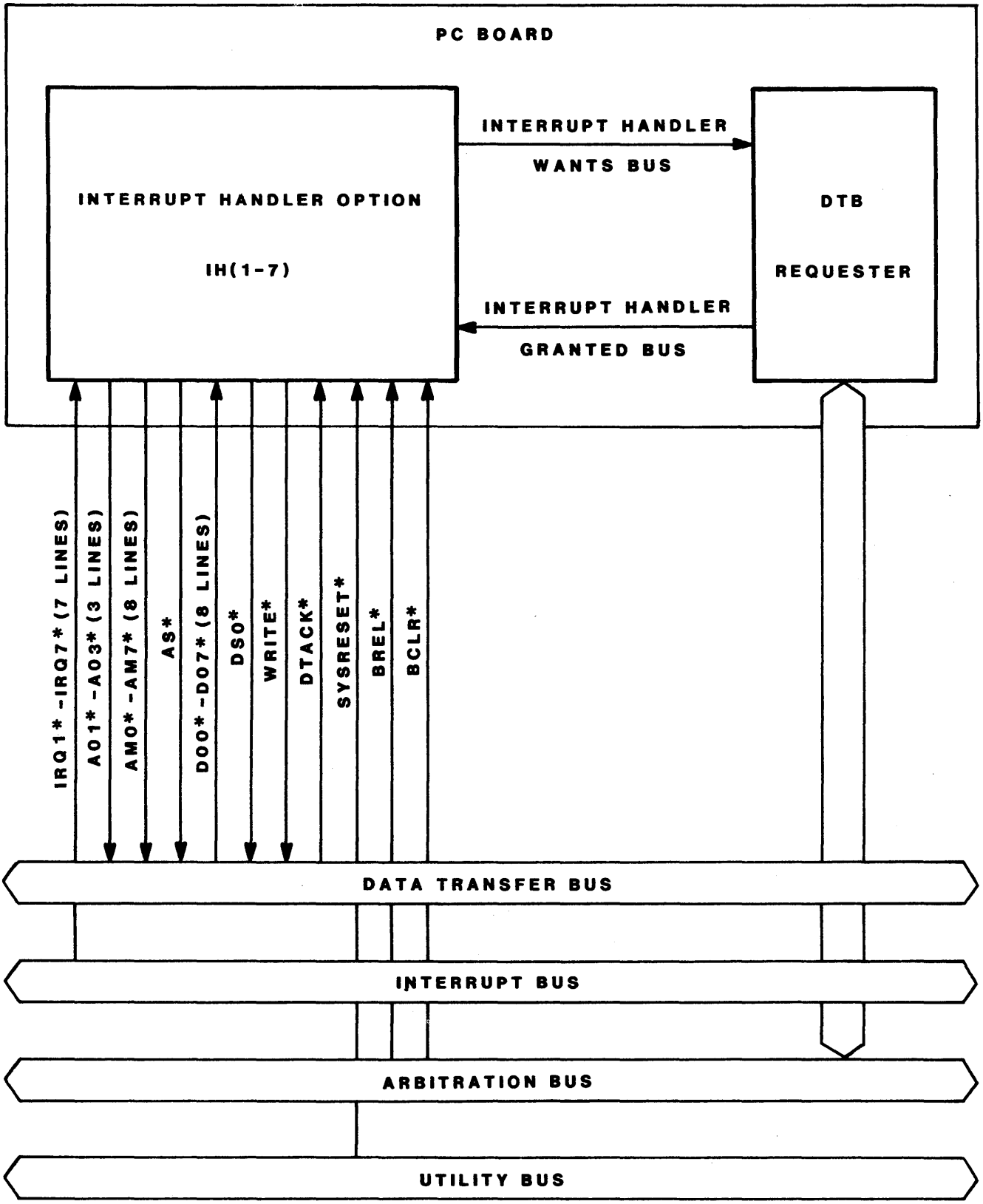


FIGURE 4-4. Signal Lines Used by an IH(1-7) INTERRUPT HANDLER

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 has no interrupt request signal on the bus.

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 deals with 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 VERSAbus.

The INTERRUPTER uses an IRQx* line (in this case, IRQ4*) to request an interrupt. It then monitors the DTB address, address modifier, and ACKIN*/ACKOUT* 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 three primary differences in the use of the DTB by the INTERRUPT HANDLER and the MASTER:

a. placement of codes on the address modifier bus,
b. number of lines driven on the address bus,
c. use of lines on the data bus.

The INTERRUPT HANDLER is required to place the interrupt acknowledge code on the address modifier bus whenever it makes use of the DTB. MASTERS are never allowed to place this code on the address modifier bus.

The INTERRUPT HANDLER is required to drive only the lowest three address lines (A01*-A03*). 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.

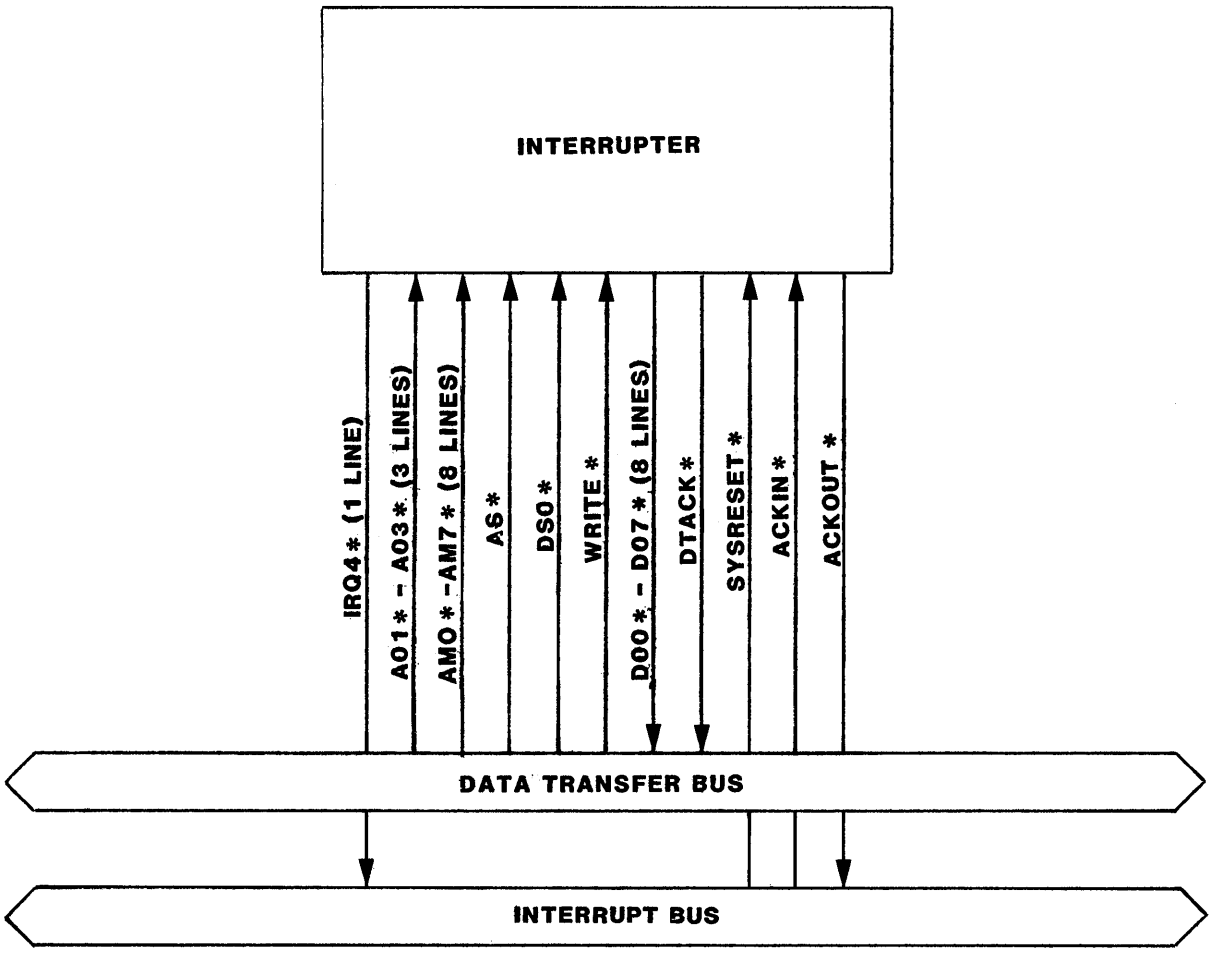


FIGURE 4-5. Signal Lines Used by an I(4) INTERRUPTER

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.

4.3.3.2 INTERRUPTER vs SLAVE: Differences

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

a. decoding the address modifier bus,
b. interpreting the address bus,
c. using the data bus.

The INTERRUPTER, upon decoding the address modifier code, will respond only if it is an interrupt acknowledge code. A SLAVE never responds to an interrupt acknowledge code, but may respond to many others.

The SLAVE decodes the appropriate number of address lines (15, 23, or 31) and determines solely from the levels of these lines and the current address modifier code whether 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 ACKIN* daisy-chain line.

If any of these three conditions is not met, it does not respond to the acknowledge. If only conditions a and c are met, or if only condition c is met, then the INTERRUPTER passes the low level of ACKIN* to the next module in the daisy-chain via ACKOUT*.

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*.

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 in a centralized system.

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 VERSAbus.

The protocol for the priority interrupt subsystem describes the module interaction required during phase 1 and phase 2. Phase 3 may not require anv 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.

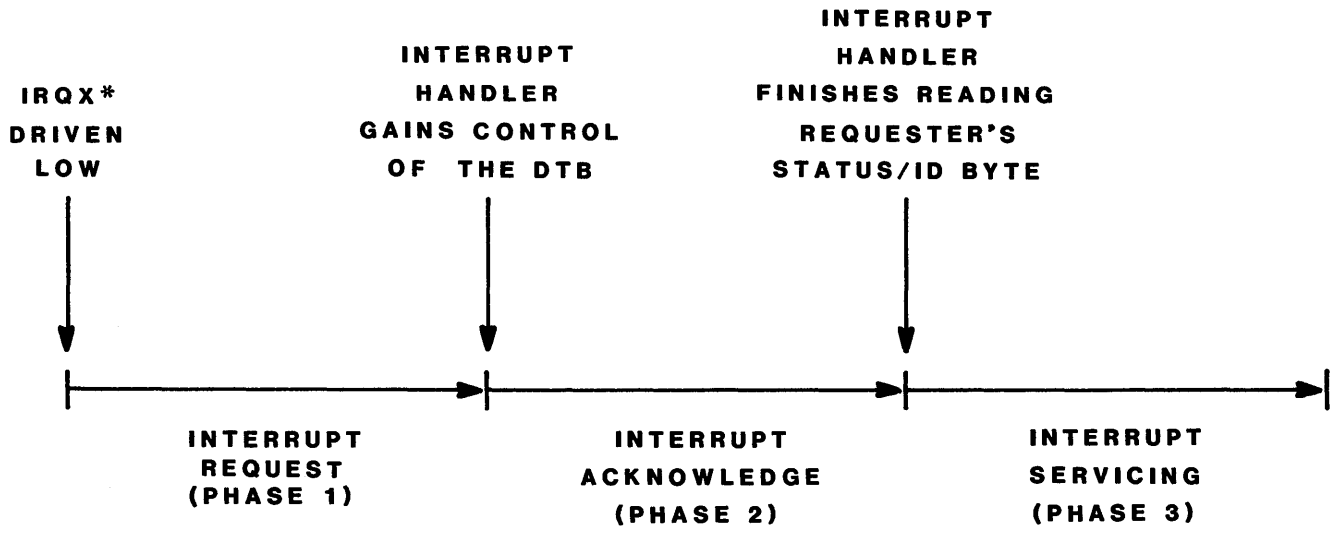


FIGURE 4-6. The Three Phases of an Interrupt Sequence

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 made 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 status/ID bytes from an INTERRUPTER driving its own interrupt request line. When two interrupt request lines on the bus are driven low simultaneously, the INTERRUPT HANDLER with the highest priority on-board DTB REQUESTER will be granted control of the DTB first.

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 they are driven low simultaneously, 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.

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.

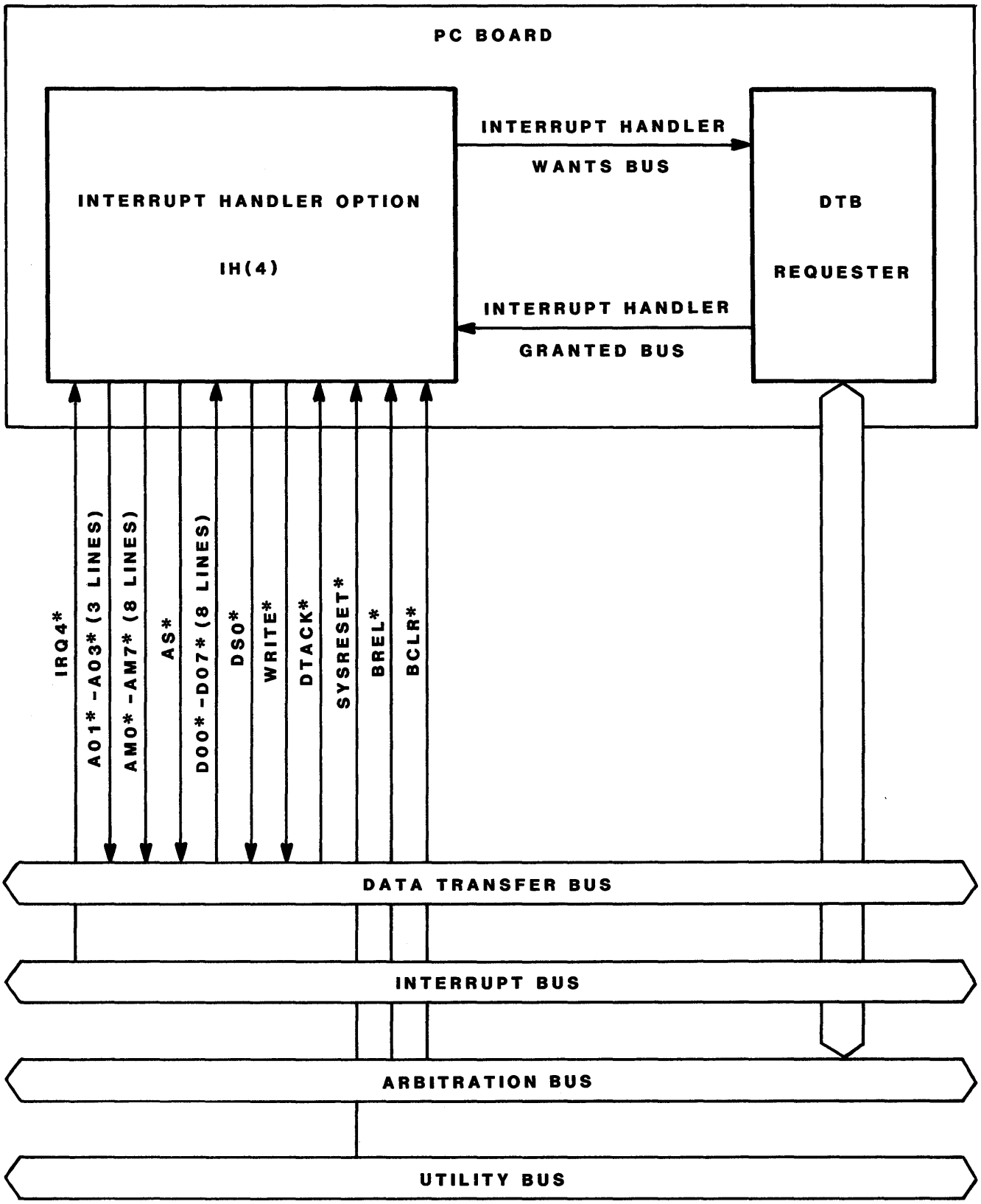


FIGURE 4-7. INTERRUPT HANDLER Monitoring Only IRQ4*

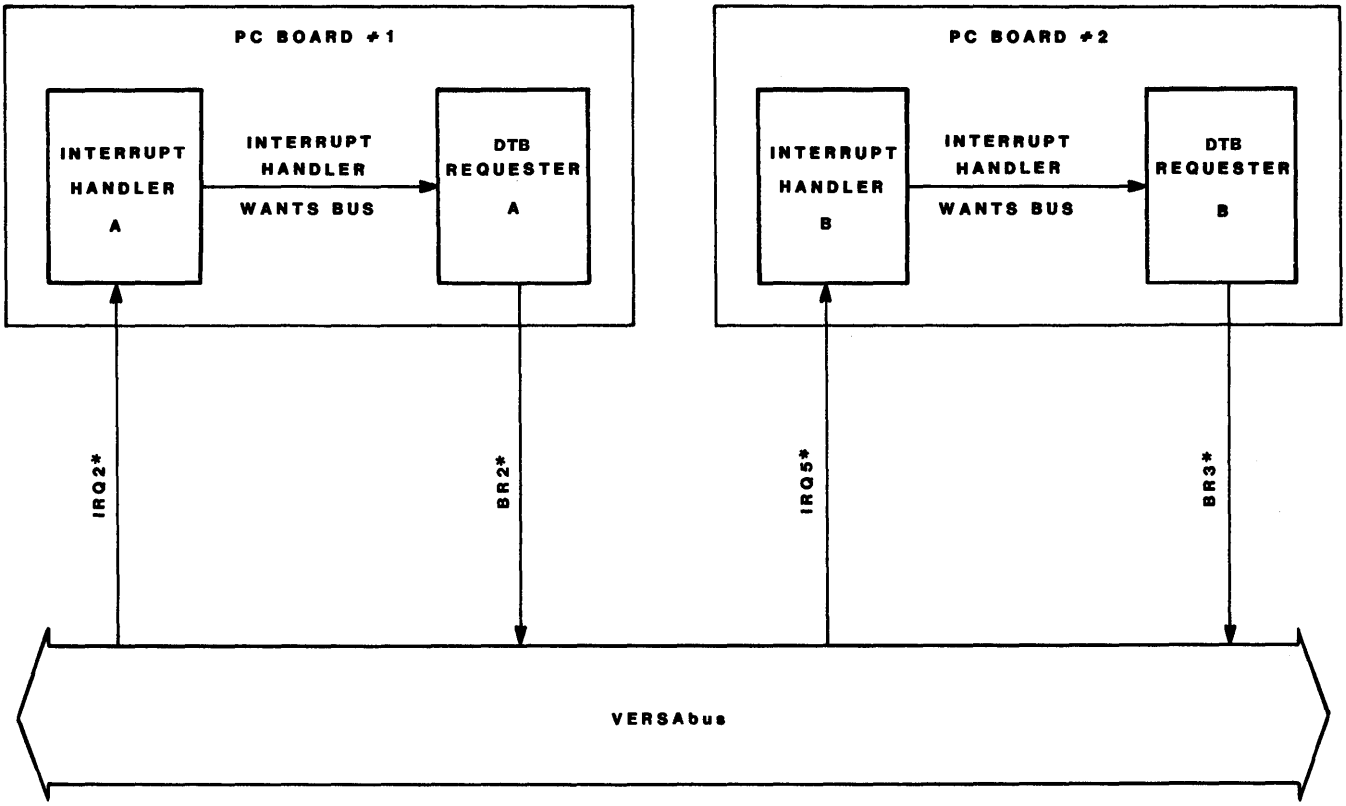


FIGURE 4-8. Two INTERRUPT HANDLERS, Each Monitoring One Interrupt Request Line

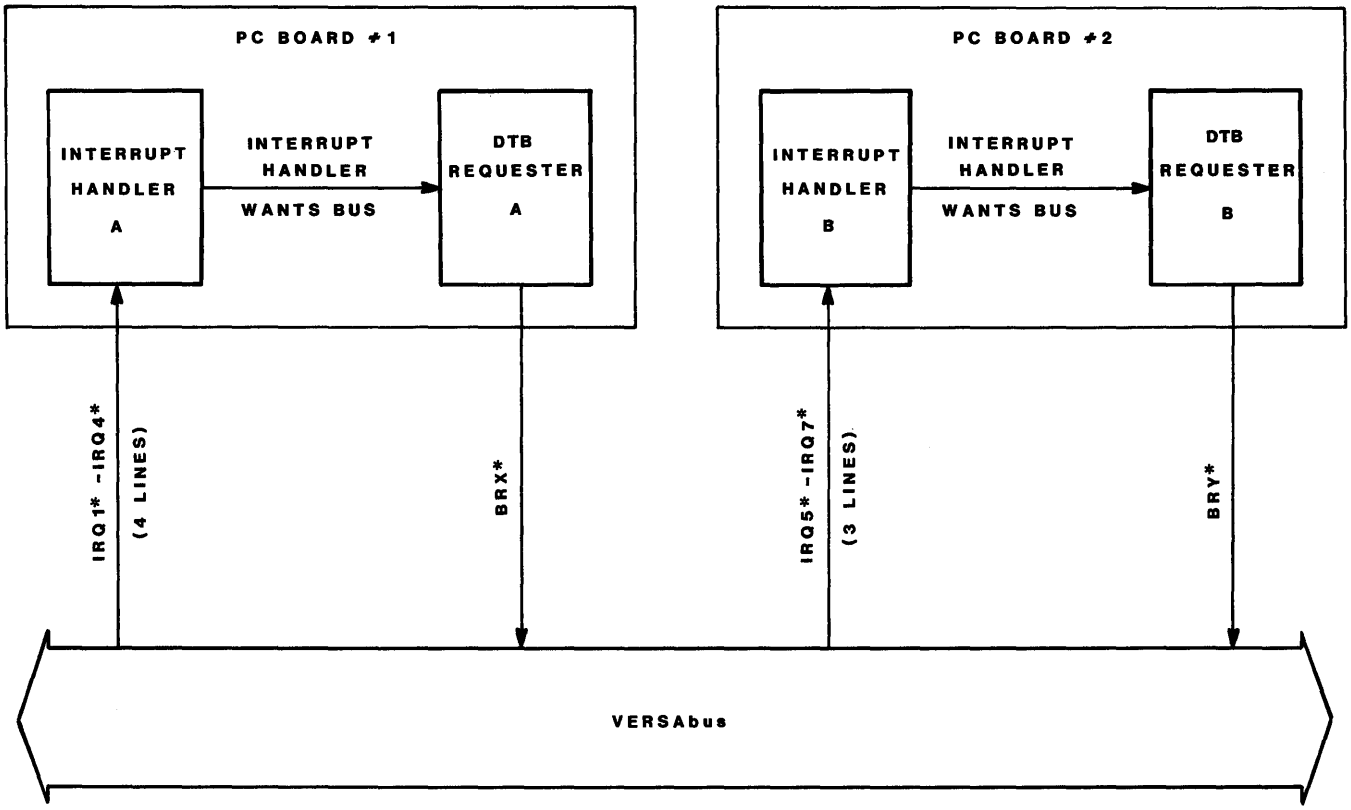


FIGURE 4-9. Two INTERRUPT HANDLERS, Each Monitoring Several Interrupt Request Lines

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 BR4* low. Upon detecting the bus request, the ARBITER drives BCLR* low, indicating that a higher priority REQUESTER is waiting for the DTB. 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

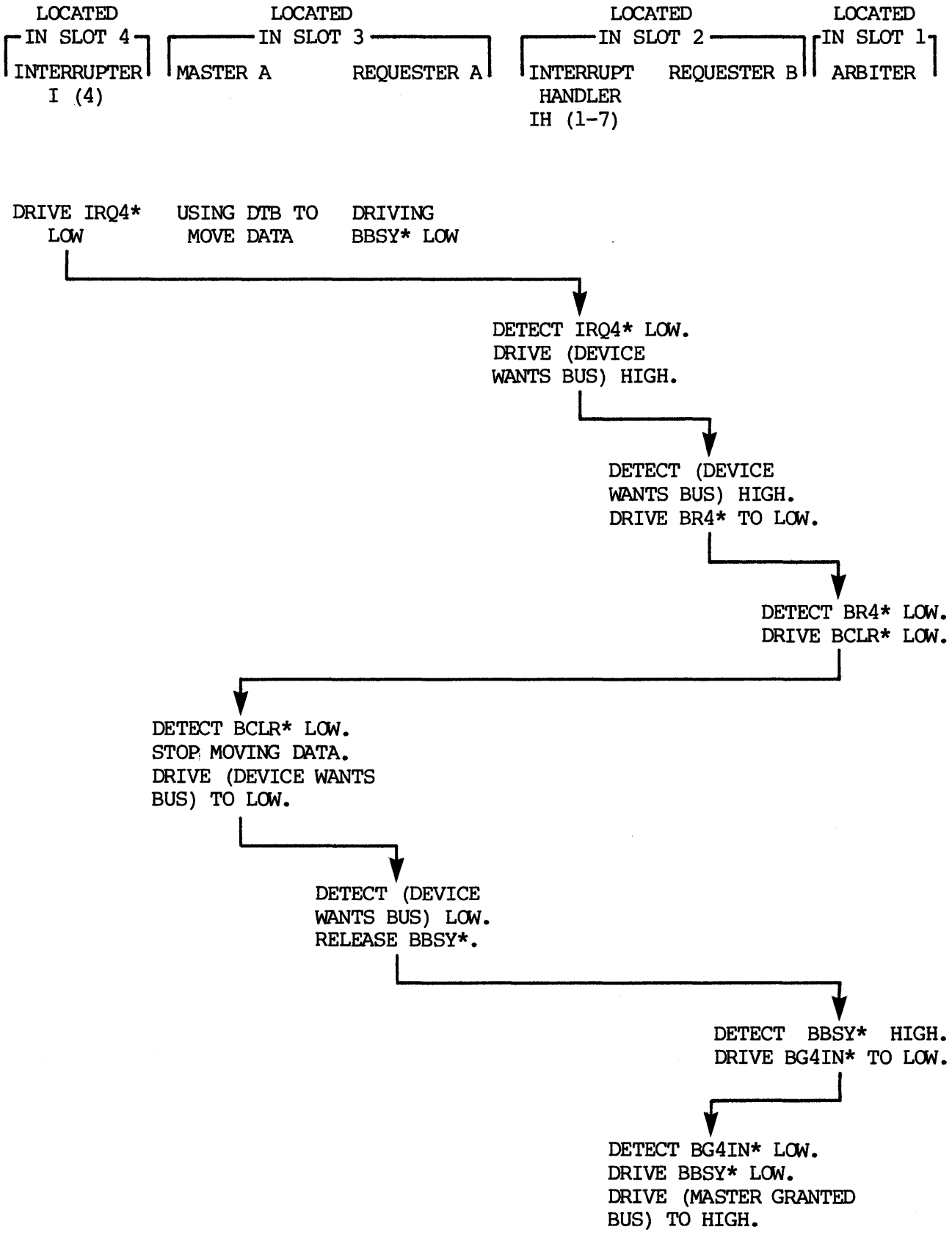
The active DTB MASTER is not required to relinquish the DTB within a specified time period, but a prompt response to the BCLR* line allows a system to run more efficiently.

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 places an 8-bit interrupt acknowledge code on the address modifier bus (see Table 4-2), indicating that it is acknowledging an interrupt, and drives AS* low. The resulting low level on the Address Modifier 5 line (AM5*) causes a low level to be propagated down the acknowledge daisy-chain on the bus (ACKIN*/ACKOUT*).

When the INTERRUPTER detects a low level on its incoming daisy-chain line, it waits for address strobe, and then (1) checks the lower three address bits to see if they match the interrupt request line which it is driving low, and (2) verifies that the interrupt acknowledge code is on the AM bus. 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 VERSAbus specification to define what must take place during an interrupt service sequence. This may or may not involve use of the data transfer bus.



TO SHEET 2

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

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	H	H	L
IRQ2*	X - X	H	L	H
IRQ3*	X - X	H	L	L
IRQ4*	X - X	L	H	H
IRQ5*	X - X	L	H	L
IRQ6*	X - X	L	L	H
IRQ7*	X - X	L	L	L

TABLE 4-2. 8-Bit Interrupt Acknowledge Code

ADDRESS MODIFIER BUS INTERRUPT ACKNOWLEDGE CODE							
AM7*	AM6*	AM5*	AM4*	AM3*	AM2*	AM1*	AM0*
H	H	L	H	H	L	L	L

(27 HEX)

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 BR4* low. Upon detecting BR4* low, the DTB ARBITER drives BG4IN* to low on slot 1. This low signal is passed down the BG4IN*/BG4OUT* 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 places an 8-bit interrupt acknowledge code (see Table 4-2) on the address modifier lines to indicate that it is acknowledging an interrupt, and a 3-bit code (see Table 4-1) is placed on the address bus.

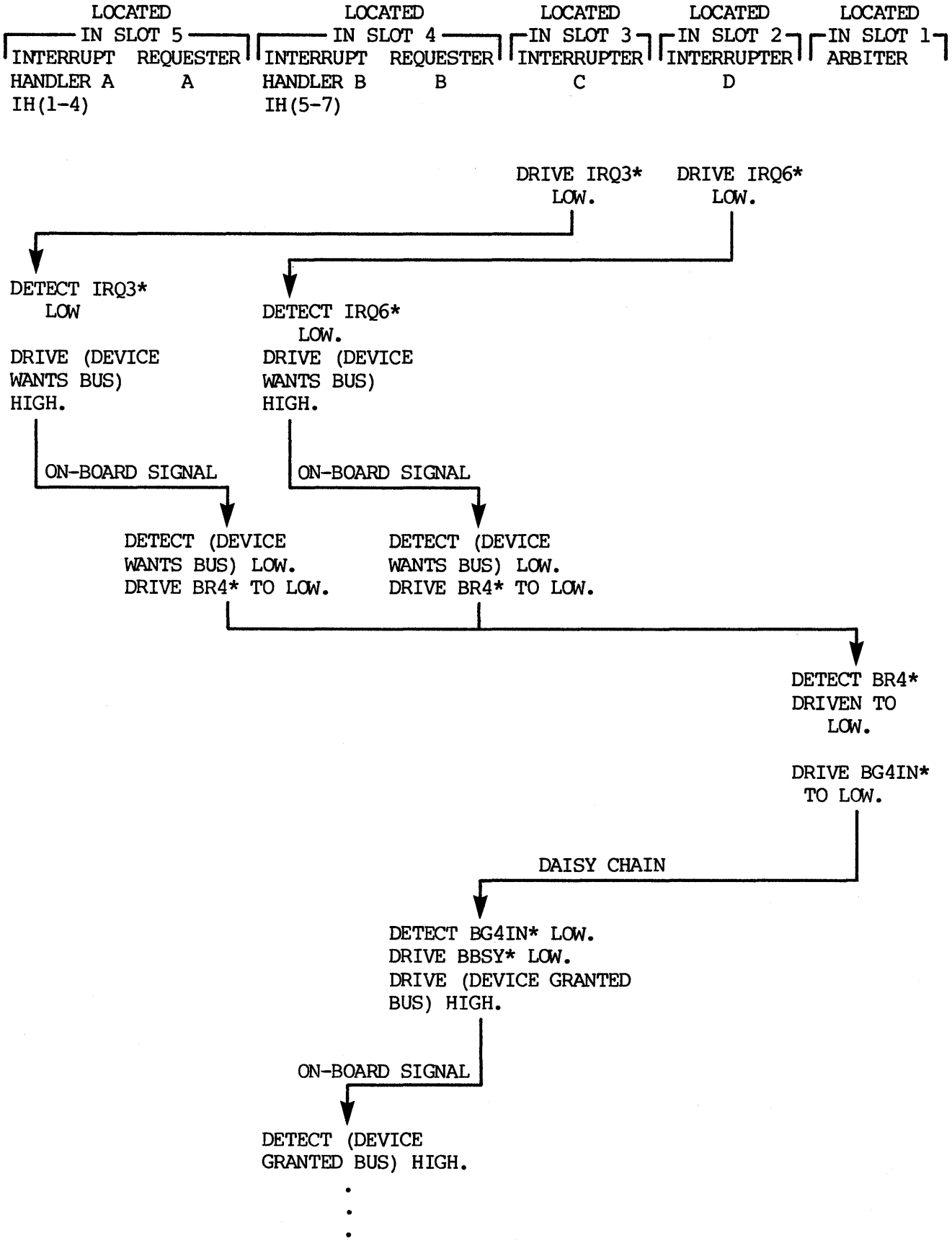


FIGURE 4-11. Distributed Interrupt System with Two INTERRUPT HANDLERS

4.5 STATE DIAGRAMS

The following sections contain state diagrams for the INTERRUPTER and INTERRUPT HANDLER modules. The information provided is rather "concentrated", and may require study before it will be completely understood. The reader who is unfamiliar with the use of state diagrams may refer to the material in Appendix B before continuing.

NOTE

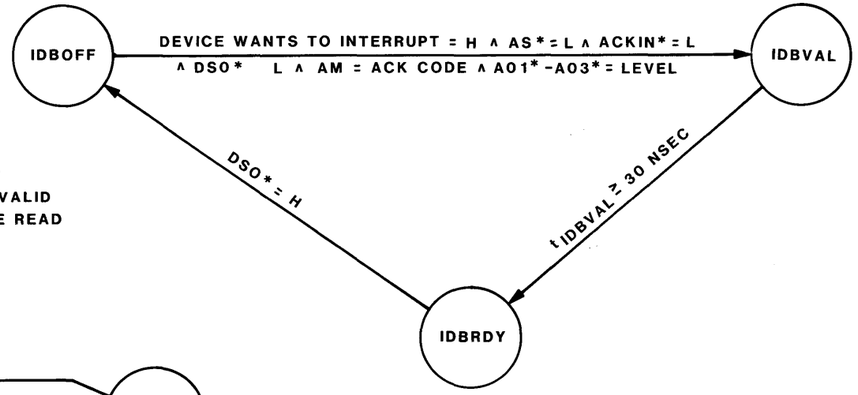
In the following discussion, several on-board signals are defined to allow discussion of the interaction between the INTERRUPTER/INTERRUPT HANDLER module and other on-on-board logic. These signals are not intended to place restrictions on the board designer, but do illustrate the information which must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

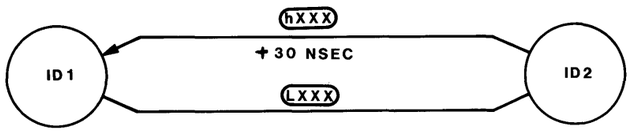
4.5.1 INTERRUPTER

Figure 4-12 shows the state diagram for an INTERRUPTER. Although this diagram does not reflect any particular implementation of an INTERRUPTER, it shows all allowed transitions that an implementation may make. Therefore, a given implementation may not enter all the states shown, but will make no transition not shown in the diagram. The block diagram for an INTERRUPTER is shown in Figure 4-13.

THE THREE STATES ON THE DIAGRAM TO THE RIGHT REPRESENT THE STATE OF THE DATA BUS DRIVERS IN THE INTERRUPTER:

IDBOFF = DATA BUS DRIVERS OFF
IDBVAL = DATA BUS DRIVEN AND VALID
IDBRDY = DATA BUS READY TO BE READ





THE STATES SHOWN IN THE DIAGRAM ABOVE DON'T RESULT IN ANY OUTPUT CHANGES. (THEY REPRESENT THE STATE OF AN INTERNAL MONOSTABLE FLIP-FLOP.) THEY ARE LABELLED SIMPLY "INTERRUPTER DELAY STATES 1 AND 2"

THE STATES IN THE DIAGRAM TO THE RIGHT ARE LABELLED ACCORDING TO THE LEVELS OF THE INTERRUPTERS OUTPUT LINES:

IRQX *	DTACK *	ACKOUT*	VERSAbus INTERRUPT ACKNOWLEDGED

NOTES:

1 ∧ = LOGICAL "AND"
2 v = LOGICAL "OR"
3 IN "IRQX*" X = 0,1,2,3,4,5,6,7
4 ACK CODE = (AM0* = L) ∧ (AM1* = L) ∧ (AM2* = L) ∧ (AM3* = H) ∧ (AM4* = H) ∧ (AM5* = L) ∧ (AM6* = H) ∧ (AM7* = H)

FIGURE 4-12. INTERRUPTER State Diagram

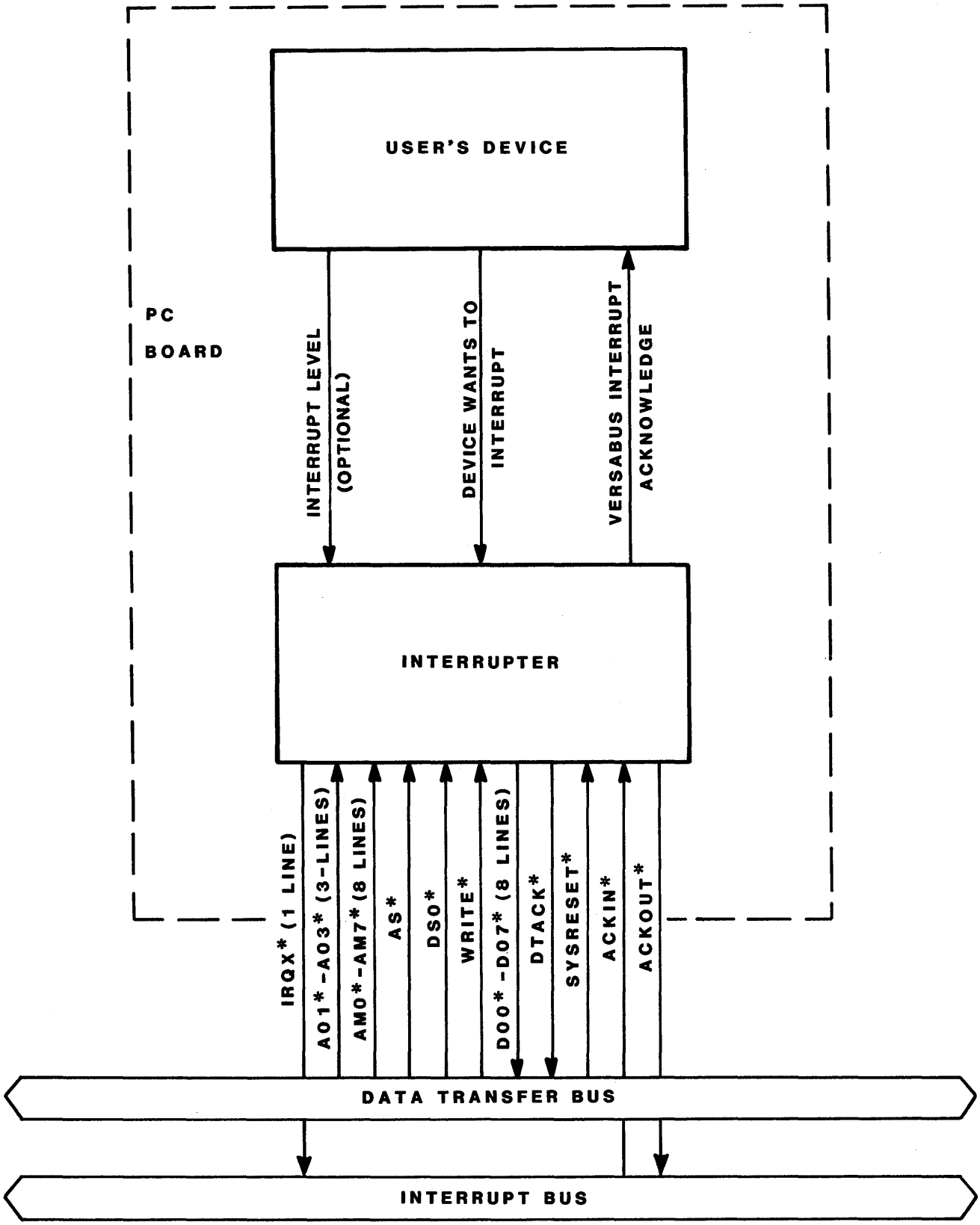


FIGURE 4-13. INTERRUPTER Block Diagram

The on-board device (see Figure 4-13) must have a means of signaling to the on-board INTERRUPTER its desire to interrupt the VERSAbus. This procedure may be as simple as a single line signaling DEVICE WANTS TO INTERRUPT, with the INTERRUPTER being configured to generate an interrupt on a single interrupt level. It might be as complicated as providing three encoded lines to the INTERRUPTER to select the level of interrupt and a DEVICE WANTS TO INTERRUPT control to tell the INTERRUPTER the lines are valid. For the following discussion, we will assume that a DEVICE WANTS TO INTERRUPT line exists, and that a high level on this line indicates that the on-board device wants to generate an interrupt to the VERSAbus.

Regardless of the levels and protocol for informing the INTERRUPTER, a means must be provided to allow the INTERRUPTER to signal the on-board device when the VERSAbus interrupt has been acknowledged. This signal, VERSABUS INTERRUPT ACKNOWLEDGED, is a signal from the INTERRUPTER to the on-board device. This line is driven high by the INTERRUPTER to indicate to its on-board device that the INTERRUPT HANDLER on the VERSAbus has acknowledged the interrupt. It is driven low to indicate that the INTERRUPT HANDLER has finished reading the status/ID byte and that the device may drive the DEVICE WANTS TO INTERRUPT on-board line high again.

Refer to Figure 4-12 for the following discussion.

Assume that the INTERRUPTER is in state hHHL. When the on-board device drives the DEVICE WANTS TO INTERRUPT line high, the INTERRUPTER moves into state LHHL and drives IRQx* low. The delay diagram then makes a transition to state ID2. The INTERRUPTER must then wait until:

a. it receives AS* driven to low,

b. it reads the address modifier lines and verifies that they match interrupt acknowledge code,

c. it reads the address lines and verifies that they represent the interrupt request line which the INTERRUPTER is driving low,

d. it receives the ACKIN* daisy chain line driven to low.

When all of the conditions have been met and when DS0* is received driven low, the INTERRUPTER places its status/ID byte on the data bus (state IDBVAL). After the byte has been valid for 30 ns, a transition is made to state IDBRDY. This, in turn, allows a transition to state LLHL, driving DTACK* low. Since the delay diagram state is still ID2, a transition is made to hLHL, releasing the interrupt request line. Assuming that DEVICE WANTS TO INTERRUPT is still high (which it must be because the device is required to maintain it high until a high VERSABUS INTERRUPT ACKNOWLEDGED is received), a transition is then made to state hLHH, driving VERSABUS INTERRUPT ACKNOWLEDGED high. This indicates to the device that the status/ID byte is being read. The device may then drive DEVICE WANTS TO INTERRUPT low. When the INTERRUPT HANDLER finishes reading the status/ID byte, it drives AS* high. If 30 ns have passed since the interrupt request line was released and if the data bus has been three-stated, a transition is then made to state hHHH, causing DTACK* to be released. Since DEVICE WANTS TO INTERRUPT remains low (the device is required to maintain it low until a low VERSABUS INTERRUPT ACKNOWLEDGED is received), a transition is then made to state hHHL, driving VERSABUS INTERRUPT ACKNOWLEDGED low.

The reader should note that this is only one example of how the INTERRUPTER might traverse the diagram. Other paths might have been followed. For example, instead of going from LLHL to hLHL and hLHH, the INTERRUPTER might have gone from LLHL to LLHH and hLHH. In the latter case, the VERSABUS INTERRUPT ACKNOWLEDGED signal would have been driven high before the bus request line was released. While this may appear to complicate the state diagram, it actuallv represents an addition of design freedom and will make the logic design simpler.

States hHLL and LHLL are associated with driving ACKOUT*. All INTERRUPTERS in the system are daisy-chained by the ACKNOWLEDGE DAISY CHAIN. Each must be capable of passing on a low level on this daisy chain if it does not have an interrupt request on the interrupt request line being acknowledged. If the INTERRUPTER detects a low on ACKIN* while it is in state hHHL, it may move to state hHLL and drive ACKOUT* low. On the other hand, if the on-board device drives DEVICE WANTS TO INTERRUPT high prior to the transition to HHLL, the INTERRUPTER can make the transition to LHHL if the interrupt line being acknowledged is the one that it would have used to generate the interrupt, anyway. In doing this, the INTERRUPTER would be "stealing" the interrupt acknowledge of an INTERRUPTER lower on the daisy chain. In either case, upon detecting ACKIN* low, a REQUESTER will either pass the low level by driving ACKOUT* low, or respond to the interrupt acknowledge.

The transition from hHHL to hLHL just mentioned allows the INTERRUPTER to respond to the interrupt acknowledge if its on-board device drives DEVICE WANTS TO INTERRUPT high, even though the INTERRUPTER has not yet driven its interrupt request line low. While this may seem a little startling, it is entirely logical. The interrupt request line which it would be driving low is already low by an INTERRUPTER further down the daisy chain, since this is the only way it could receive an interrupt acknowledge for that line. In fact, the user, observing the interrupt bus lines, would not be able to say with any certainty whether or not this INTERRUPTER drove its interrupt request line low. Since the low level on ACKIN* is not passed along the daisy-chain to the next board, the INTERRUPTER driving the IRQX* line being acknowledged will continue to hold it low until another interrupt acknowledge cycle takes place.

Two internal timing restrictions are placed upon the INTERRUPTER:

- The INTERRUPTER must wait 30 ns after releasing its interrupt request line before releasing DTACK*. This ensures that the INTERRUPT HANDLER will not detect the old interrupt request and acknowledge it again after reading the status/ID byte.

- When the INTERRUPTER has passed a low ACKIN* signal on to the next board in the daisy chain, it is required to drive this line high within 50 ns after receiving AS* high. Since all INTERRUPTERS see AS* go high at essentially the same time, this eliminates the need for the high level generated by the INTERRUPT HANDLER at the end of the interrupt acknowledge cycle to propagate through the daisy chain.

4.5.2 INTERRUPT HANDLER

Figure 4-14 shows the block diagram of an INTERRUPT HANDLER. The INTERRUPT HANDLER consists of three sub-elements:

a. Interrupt prioritizer

b. Address bus driver

c. Data bus controller

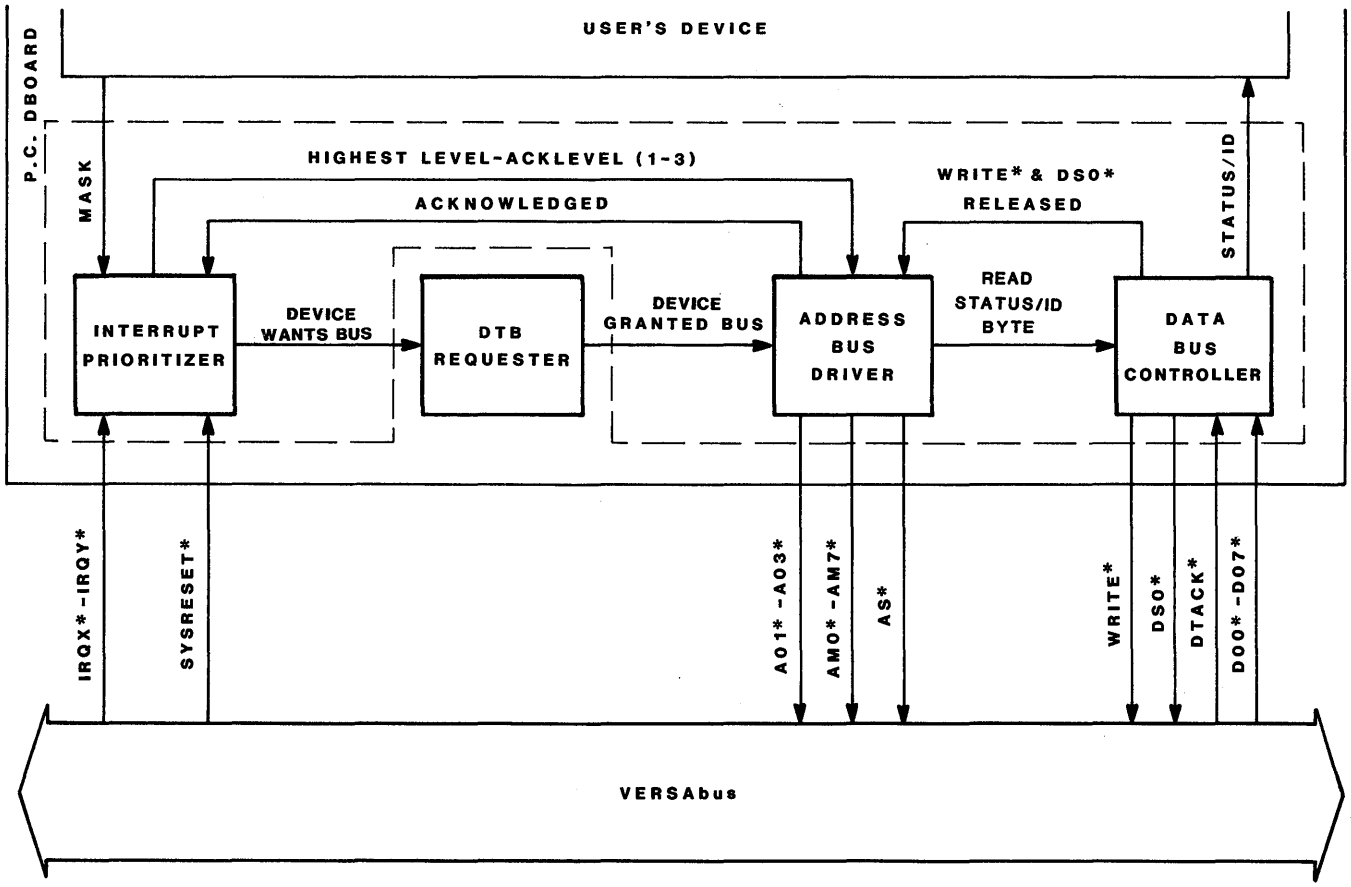


FIGURE 4-14. Block Diagram: INTERRUPT HANDLER

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 state diagrams for single-level and the seven-level INTERRUPT HANDLERS differ only in interrupt prioritizer sub-elements. 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 address bus driver to indicate what 3-bit code should be placed on the address lines during the interrupt acknowledge. When the address bus driver receives a signal from the DTB REQUESTER indicating that the bus is available, it places the interrupt acknowledge code on the address modifier bus, places the interrupt acknowledge level on the lower three address lines, and drives address strobe (AS*) to low. It then sends a signal to the data bus controller, indicating that the status/ID byte may be read.

The data bus controller 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. Both the data bus controller and the address bus driver respond simultaneously to the low DTACK*. The data bus controller reads the byte, releases WRITE* and DS0*, and then signals the address bus driver that it is no longer driving any bus lines. The address bus driver releases the address lines and the address modifier lines, 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.)

Figures 4-15 and 4-17 show the state diagrams for a seven-level INTERRUPT HANDLER. Figure 4-16 shows the state diagrams for a single-level INTERRUPT HANDLER which handles interrupt requests received on IRQ4*. A detailed discussion of each is included in the following sections.

While it would also be possible to show state diagrams for INTERRUPT HANDLERS which handle between two and six interrupt request levels, because of the large number of possible combinations, this will not be done. For a discussion of legal combinations and for the option notations representing each combination, see Chapter 6.

Although the state diagrams shown do not reflect any particular implementation of an INTERRUPT HANDLER, they show all allowed transitions that any implementation may make. Therefore, a given implementation may not enter all the states shown, but will make no transitions not shown on the diagram.

4.5.2.1 Interrupt Prioritizers

4.5.2.1.1 Seven-Level Interrupt Prioritizer

Figure 4-15 illustrates the function of a seven-level interrupt prioritizer. When the VERSAbus system is powered up, this interrupt prioritizer enters the L/XXX state, where it will remain until an unmasked interrupt line on the interrupt bus is driven low by an INTERRUPTER. When this condition is met, the prioritizer moves to one of the seven H/XXX states. Because of the conditions placed upon each transition, the highest priority interrupt will govern the state transition in the event that two interrupt lines are driven low simultaneously.

When any one of these H/XXX states is entered, the level to be acknowledged is provided to the address bus driver via three lines named ACKLEVEL1, ACKLEVEL2, and ACKLEVEL3. At the same time, a DEVICE WANTS BUS signal is driven high to the on-board DTB REQUESTER, causing it to request the DTB.

4.5.2.1.2 Single-Level Interrupt Prioritizer

Figure 4-16 illustrates the function of a single-level interrupt prioritizer. It does not prioritize interrupts in the strictest sense, but it does provide the capability to mask incoming interrupts. VERSAbus allows up to seven of these single-level INTERRUPT HANDLERS to be present within an interrupt system. While a casual study of the two prioritizer state diagrams might lead the reader to conclude that all single-level INTERRUPTERS are proper subsets of seven-level interrupt prioritizer, one important distinction must be recognized. In a system consisting of seven single-level INTERRUPT HANDLERS, there would be no prioritizing of the interrupt levels because each INTERRUPT HANDLER would operate completely independent from all others (e.g., an IRQ7* interrupt will not necessarily be handled prior to an IRQ1* interrupt).

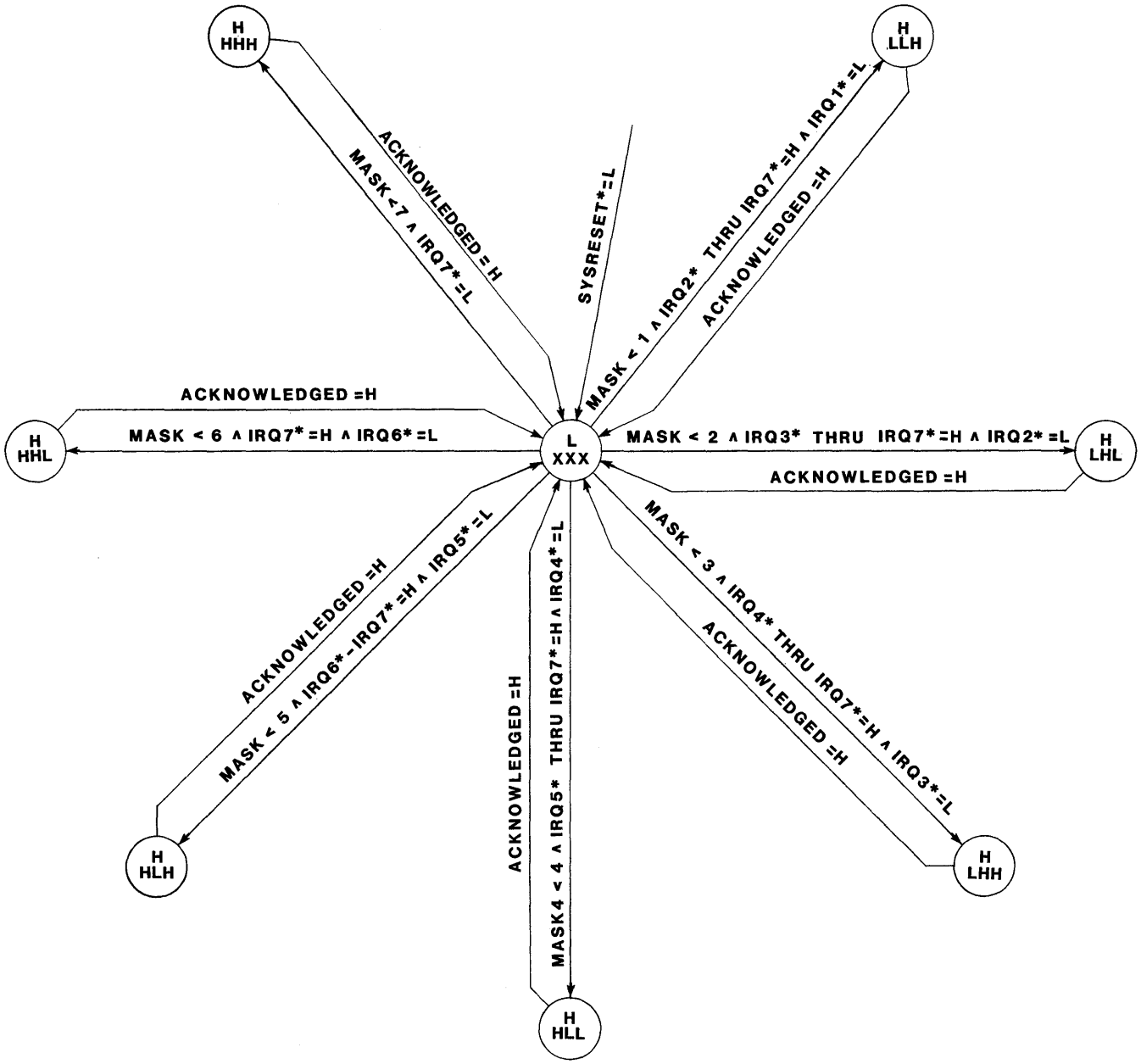


FIGURE 4-15. State Diagram for the Interrupt Prioritizer of a Seven Level INTERRUPT HANDLER

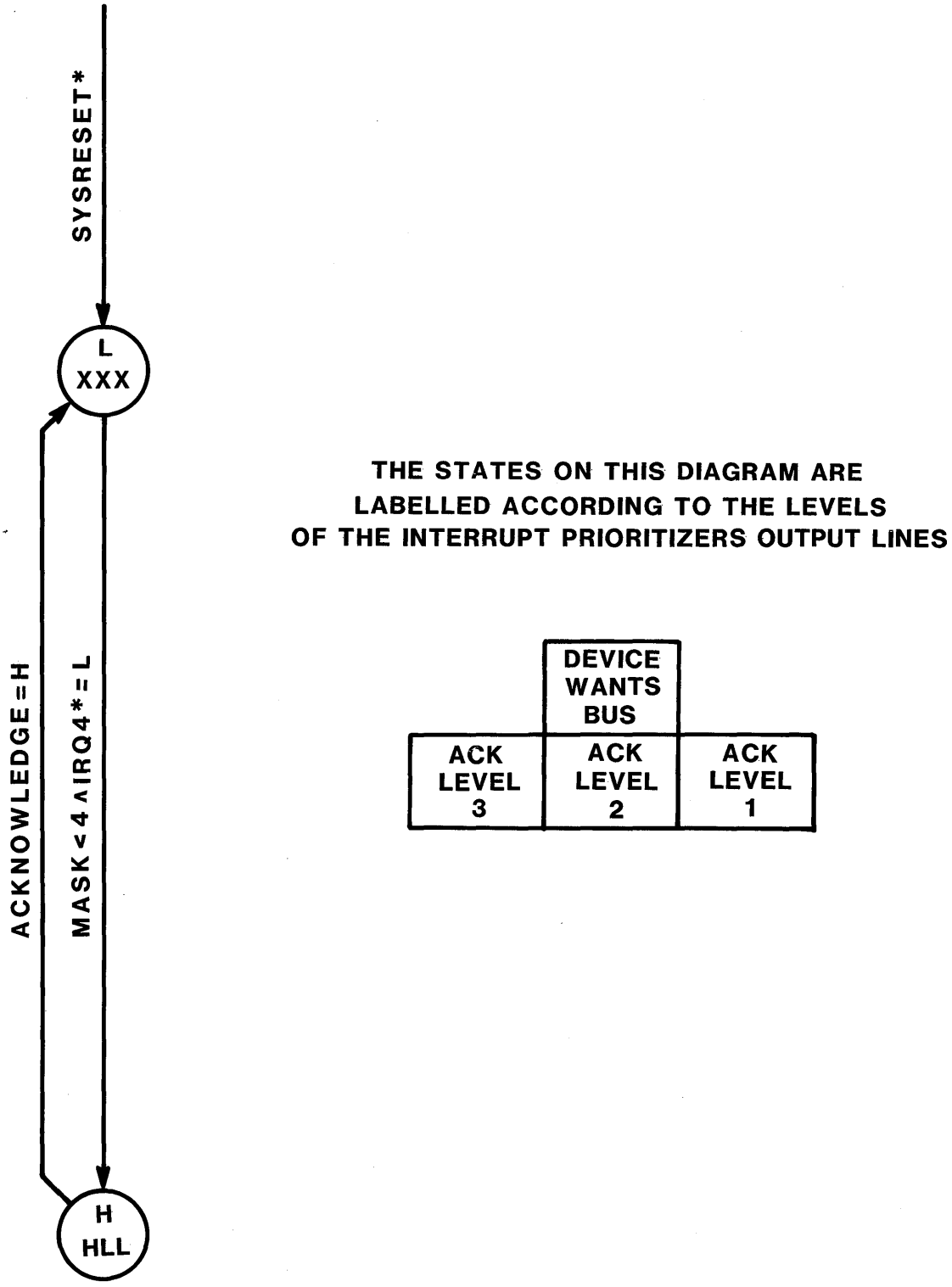


FIGURE 4-16. State Diagram for the Interrupt Prioritizer of a Single Level INTERRUPT HANDLER (Level 4)

4.5.2.1.3 Interrupt Masking

The purpose of the interrupt prioritizer state diagrams is to define the order in which the various interrupt request lines on VERSAbus will be acknowledged. No limit has been placed on the time allowed for state transitions from L/XXX. Therefore, an interrupt prioritizer may take a considerable amount of time after the conditions are met before leaving state L/XXX without violating the VERSAbus specification. It is not possible to determine just by observing the VERSAbus lines whether a delay in making this transition is the result of a slow prioritizer or the result of of its mask being set to a high level. However, the delay causes no incompatibility problems between modules in the interrupt subsystem. For this reason, the question of whether a particular interrupt prioritizer implementation is able to mask interrupts is not a matter which needs to be resolved in the VERSAbus specification. The MASK <X condition on the state diagrams is shown only to make clear how an interrupt prioritizer might be implemented with masking capability.

NOTE

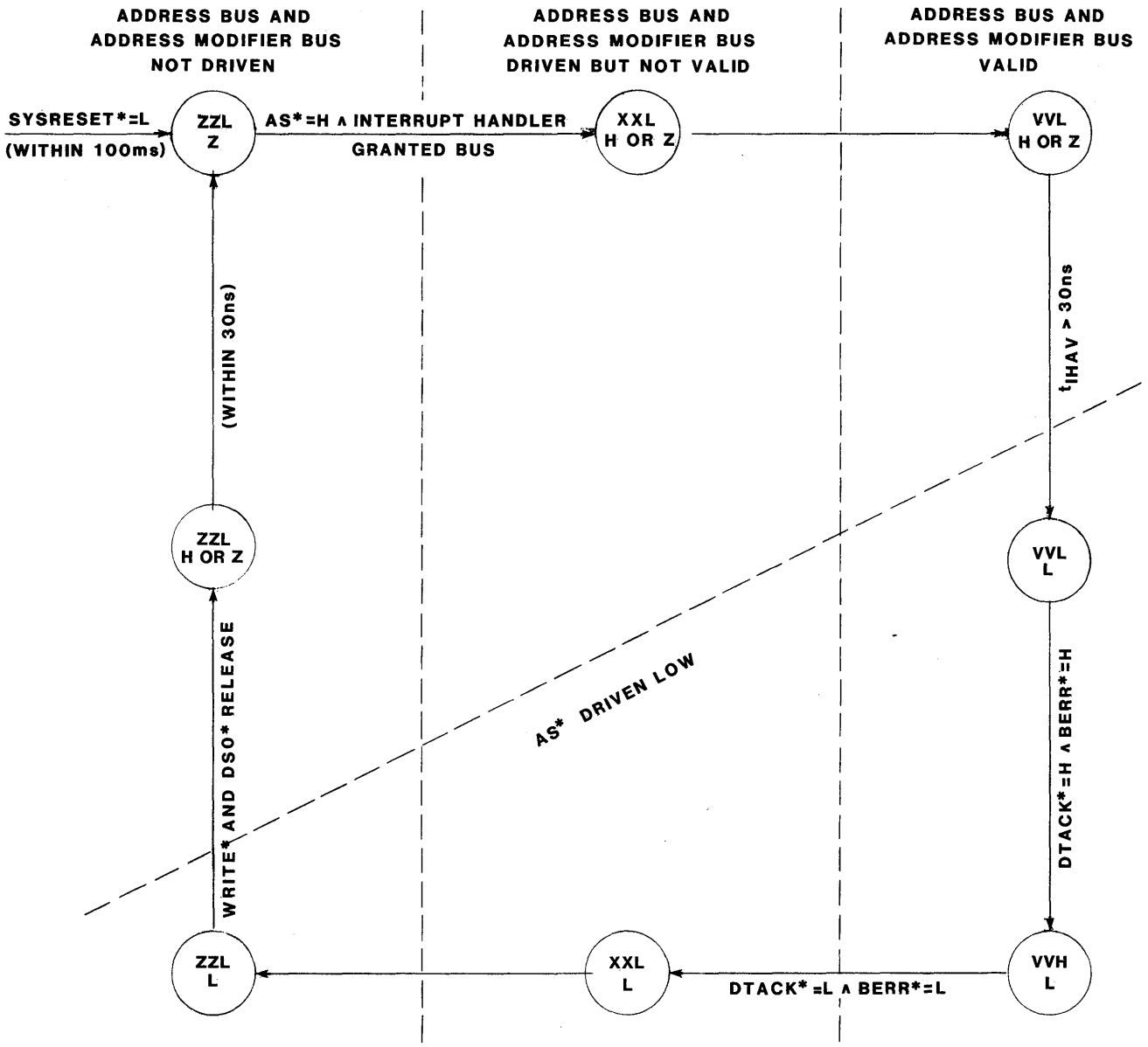
The MASK <X condition shown on state diagrams Figures 4-15 and 4-16 should not be regarded as a requirement for VERSAbus compatibility.

4.5.2.2 Address Bus Driver

Figure 4-17 illustrates the function of the address bus driver sub-element. When SYSRESET* is driven low, the address bus driver enters the ZZL/Z state, where it remains until the on-board DTB requester obtains control of the DTB. (It does this in response to the indication bv the interrupt prioritizer that an interrupt has been detected.) When the requester indicates that the bus has been granted the address driver waits until the previous MASTER (or interrupt handler) quits driving the address bus (AS* = H). It then moves into state XXL/HorZ and turns on its address and address modifier bus drivers. (AS* may or may not be driven at this time. If it is driven, it must be driven high.)

Sometime later, all of the address and address modifier lines become stable and valid (state VVL/H or Z). A 30 ns set-up time is then required before moving into state VVL/L and driving AS* low. Once state VVL/L has been entered, the address driver must wait until both DTACK* and BERR* are high. In most cases, they will already be high; but if the SLAVE from the previous cycle has not yet released its handshake line, the address driver will not enter VVH/L. This is important because as soon as the VVH/L state is entered, the data bus controller (see next paragraph) will command the INTERRUPTER to drive the data bus. If the SLAVE from the previous cycle has not yet released the data bus, contention may result. The release of DTACK* and BERR* indicates that the data bus is no longer driven.

When the address bus driver enters its VVH/L state, the data bus controller commands the INTERRUPTER to place its status/ID byte on the data bus lines D00*-D07* (see section 4.5.2.3). The INTERRUPTER does this and then drives DTACK* low when the byte is valid.



NOTES:

Z = THREE STATE
V = VALID
X = H OR L

THE STATES ON THIS DIAGRAM ARE LABELLED ACCORDING TO THE LEVELS OF THE INTERRUPT HANDLERS ADDRESS BUS DRIVER OUTPUT LINES.

A01*–A03*	AM0*–AM7*	READ STATUS/ID BYTE
	AS*	

FIGURE 4-17. State Diagram: Interrupt Handler's Address Bus Driver

When the address bus driver detects the low DTACK*, it removes the 3-bit interrupt level code from the address bus and the interrupt acknowledge from the address modifier bus (state XXL/L). It then stops driving the address and address modifier buses (i.e., enters state ZZL/L). However, it is not allowed to enter state ZZL/HorZ until the data bus controller stops driving DS0* and WRITE* (i.e., until the data bus controller enters its idle state). When this condition is met, and the address driver moves into the ZZL/HorZ state (i.e., it either releases address strobe or drives it high), it must move into the ZZL/Z state within 30 ns. (This means that once AS* has been driven high, it must be released within 30 ns to avoid bus contention.)

4.5.2.3 Data Bus Controller

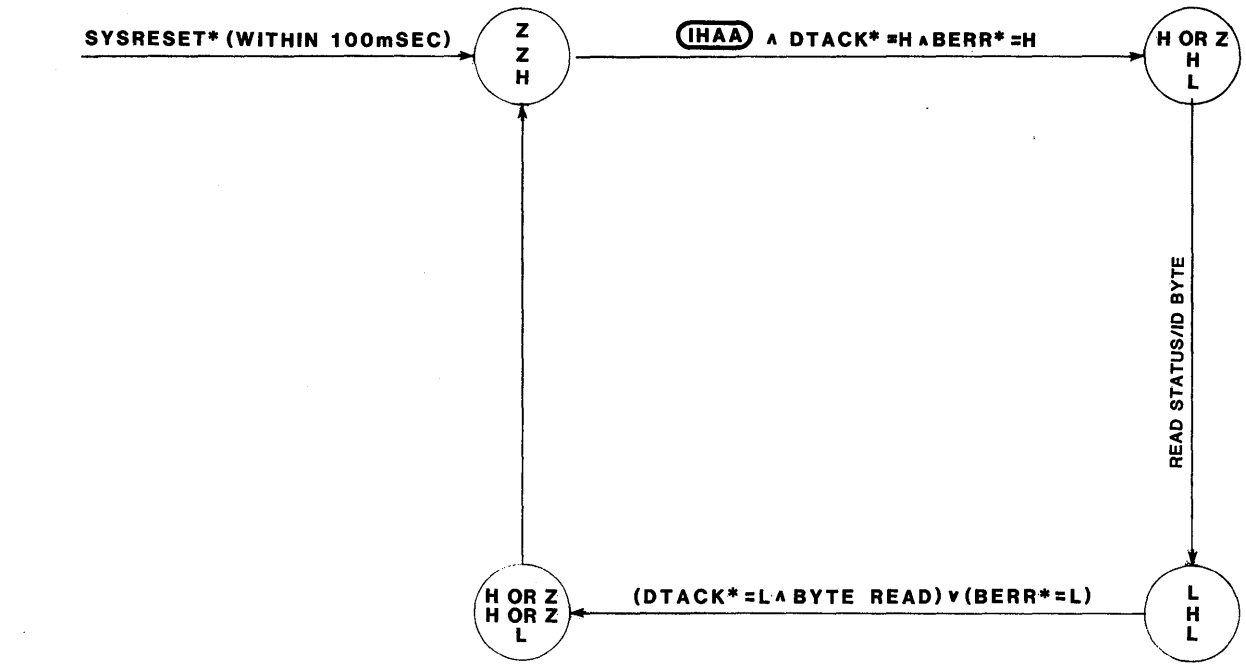
Figure 4-18 illustrates the function of the data bus controller.

When SYSRESET* is driven low, the data bus controller enters the Z/Z/H state, where it remains until:

a. the address bus driver starts driving its bus (i.e., it enters state XXL/H or Z), and

b. DTACK* and BERR* have both been released by the SLAVE from the previous cycle. (This indicates that the previous SLAVE is no longer driving the data bus.)

It then enters state HorZ/H/L. In the HorZ/H/L state, the WRITE* line is driven high. The DS0* line may or may not be driven, but if it is, it will be driven high. After the address bus driver has driven AS* low and detected DTACK* and BERR* high, it will drive the on-board READ BYTE signal high. The data bus controller, upon detecting a high level on this line, drives DS0* low (enters state L/H/L) and awaits a low level on DTACK* or BERR*. When it receives a low DTACK*, it reads the status/ID byte from D00*-D07* and enters state H or Z/H or Z/L. When it then makes a transition to Z/Z/H (releasing both WRITE* and DS0*), it signals that fact to the address bus driver in the form of an on-board signal (WRITE* AND DS0* RELEASED).



THE STATES IN THIS DIAGRAM ARE LABELLED ACCORDING TO THE LEVELS OF THE INTERRUPT HANDLERS DATA BUS CONTROLLER OUTPUT LINES.

DSO *
WRITE *
WRITE * AND DSO * RELEASED

FIGURE 4-18. State Diagram: Interrupt Handler's Data Bus Controller

CHAPTER 5

VERSAbus UTILITIES

CHAPTER 5

VERSAbus UTILITIES

5.1 INTRODUCTION

This chapter identifies and defines the signal lines and modules which serve utility-type functions for the VERSAbus. Input/output line characteristics and J2/P2 option compatibility are also discussed.

Utility lines provide periodic timing signals, support time-of-day function, allow AC zero crossing indication, and provide start-up and testing capability for the VERSAbus. Utility lines include:

System Clock	(SYSCLK)
AC Clock	(ACCLK)
AC Fail	(ACFAIL*)
System Reset	(SYSRESET*)
System Test	(TEST0*, TEST1*, SYSFAIL*)

Many of these lines are driven by a POWER MONITOR module. Its purpose is to detect power failures, generate the AC clock signal, reset the system upon power up, and initiate a power-up self-test.

5.2 UTILITY SIGNAL LINES

5.2.1 Bus Clocks

Two clock sources are available on the VERSAbus backplane: the 16 megahertz system clock and the line frequency AC clock. Neither clock has any fixed phase relationships with other VERSAbus timing. The system clock is a free-running clock signal, and the AC clock is tied to the system power source.

5.2.1.1 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. SYSCLK has no fixed phase relationships with other VERSAbus timing. 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.

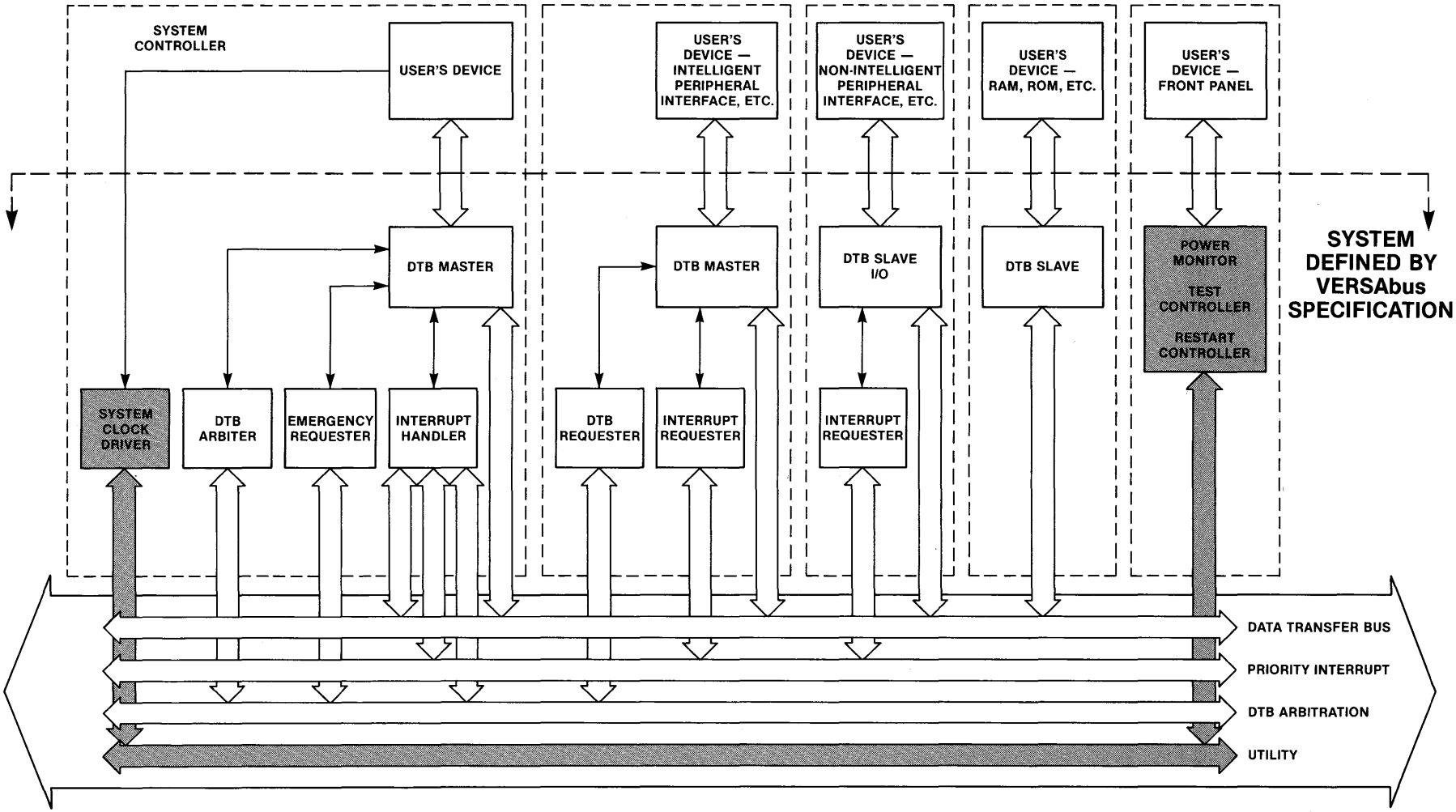


FIGURE 5-1. VERSAbus Utility Block Diagram

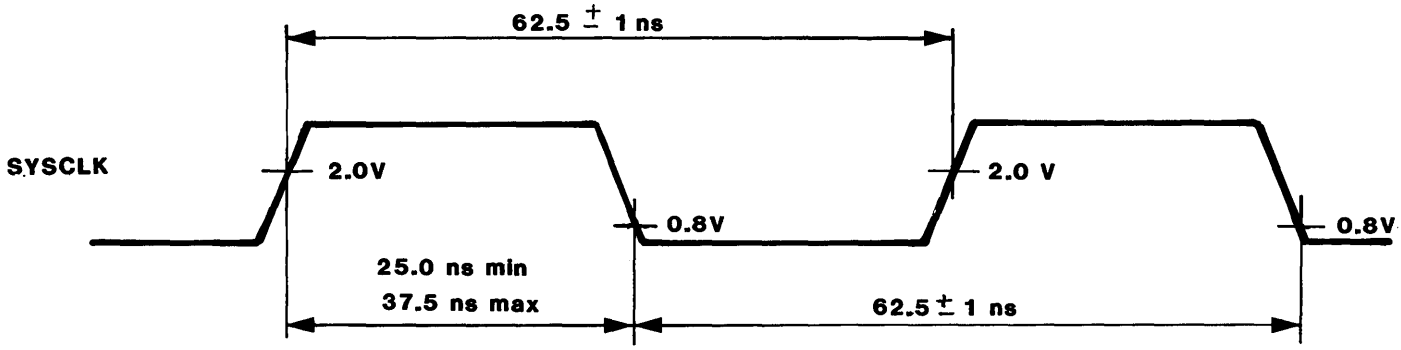


FIGURE 5-2. System Clock Timing Diagram

5.2.1.2 AC Clock (ACCLK) Specification

The AC clock is a 50 or 60 hertz (nominal) signal derived from the power supply line frequency. It can be used as a clock source to generate time-of-day, or to detect line frequency zero-point crossings. Figure 5-3 illustrates the timing relationship between the edges of ACCLK and the zero crossings of the AC line. ACCLK edges may lead or lag the AC zero crossing by a maximum of 115 usecs on a 50 cycle system, or 95 usecs on a 60 cycle system.

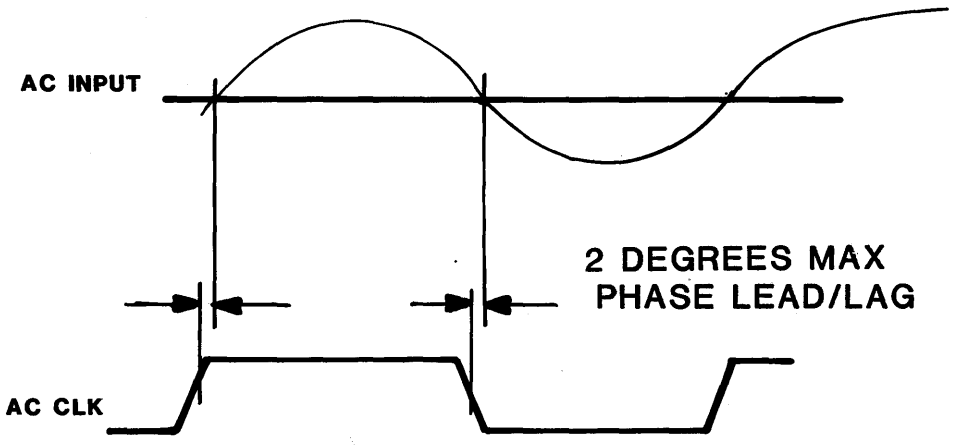


FIGURE 5-3. AC Clock Timing Diagram

5.2.2 System Initialization and Diagnostics

VERSAbus allows several options for system initialization and diagnostics through the System Reset (SYSRESET*) and test lines (TEST0* and TEST1*). These open collector lines are 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*). All boards within the VERSAbus system must drive this system fail line low upon power up, and must maintain it low until they have passed their respective self tests. Non-intelligent boards which are incapable of self test must maintain this line low until a MASTER in the system writes to their on-board test register.

5.2.2.1 System Reset (SYSRESET*)

The SYSRESET* is monitored by all functional modules, and is driven from an operator's panel or POWER MONITOR module. Its rising edge acts as a strobe for the test lines. Whenever SYSRESET* is driven to low, it must be held there for a minimum period of 200 milliseconds. Figure 5-4 shows the required timing relationship which must be maintained between the system reset and the test lines.

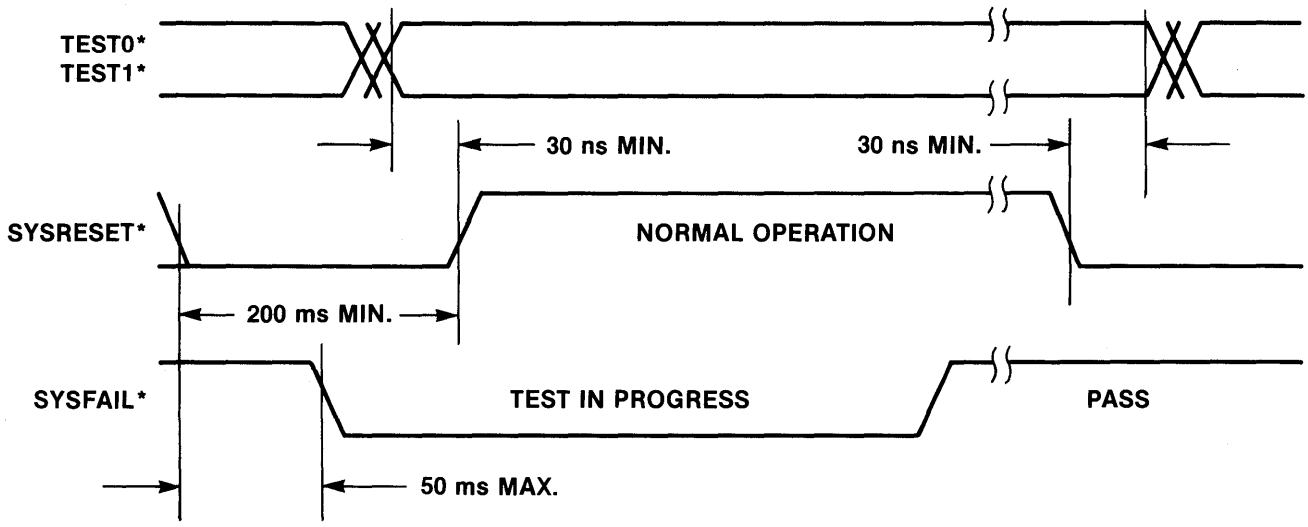


FIGURE 5-4. System Reset and Test Timing Diagram

5.2.2.2 System Test (TEST0*, TEST1*, SYSFAIL*)

The test lines allow four different test modes. The test modes are shown in Table 5-1.

TABLE 5-1. Test Modes

TEST1*	TEST0*	MODE
H	H	Enter EXEC immediate (no test required)
H	L	Enter Debug Mode
L	H	Long test (no time limit) then enter EXEC
L	L	Short test ($\leq$ 2 seconds) then enter EXEC

As shown in Figure 5-4, the test line states are established while SYSRESET* is low. After SYSRESET* is released, the system enters the selected test mode. Upon completion of testing, the system can go into 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, a local processor may periodically perform a self-test and activate SYSFAIL* if a failure occurs.

5.3 POWER MONITOR MODULE

This module is usually an external PC board (Figure 5-5) upon which the ACFAIL*, ACCLK, and (optionally) TEST0*, TEST1*, and SYSRESET* drivers are located. Logic to interface an operator's panel and to detect power fail may also be placed on this circuit card.

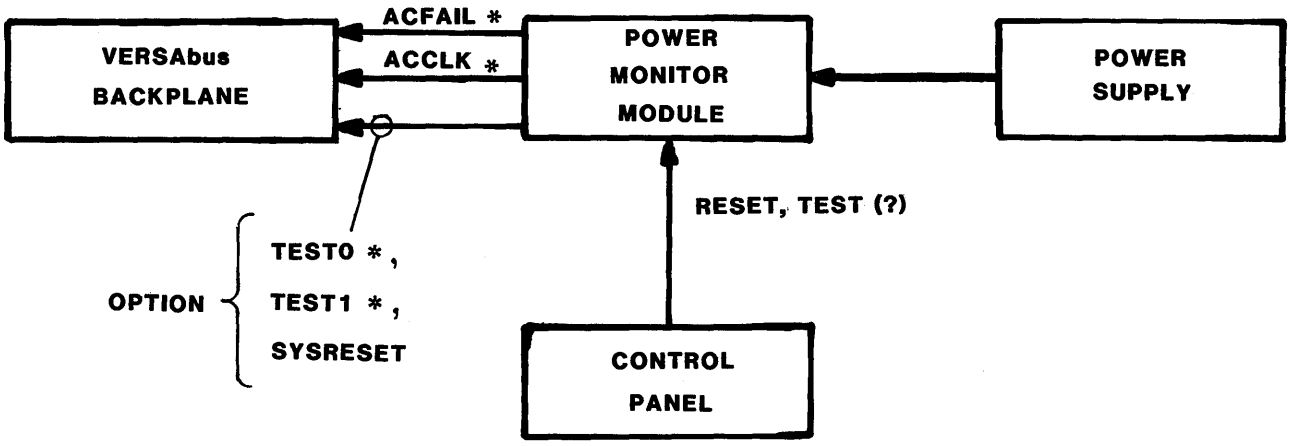


FIGURE 5-5. Block Diagram of POWER MONITOR Module

The ACFAIL* and SYSRESET* signals and the point at which the system DC voltages violate specification have certain timing constraints. These constraints are spelled out in Figures 5-6 and 5-7.

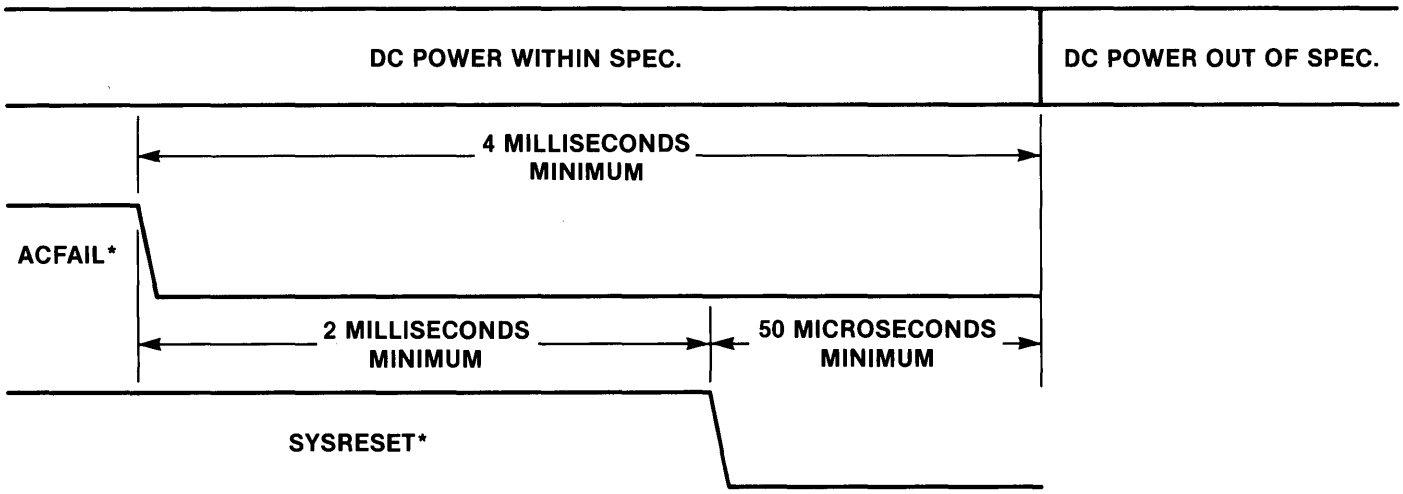


FIGURE 5-6. System Power Fail Timing

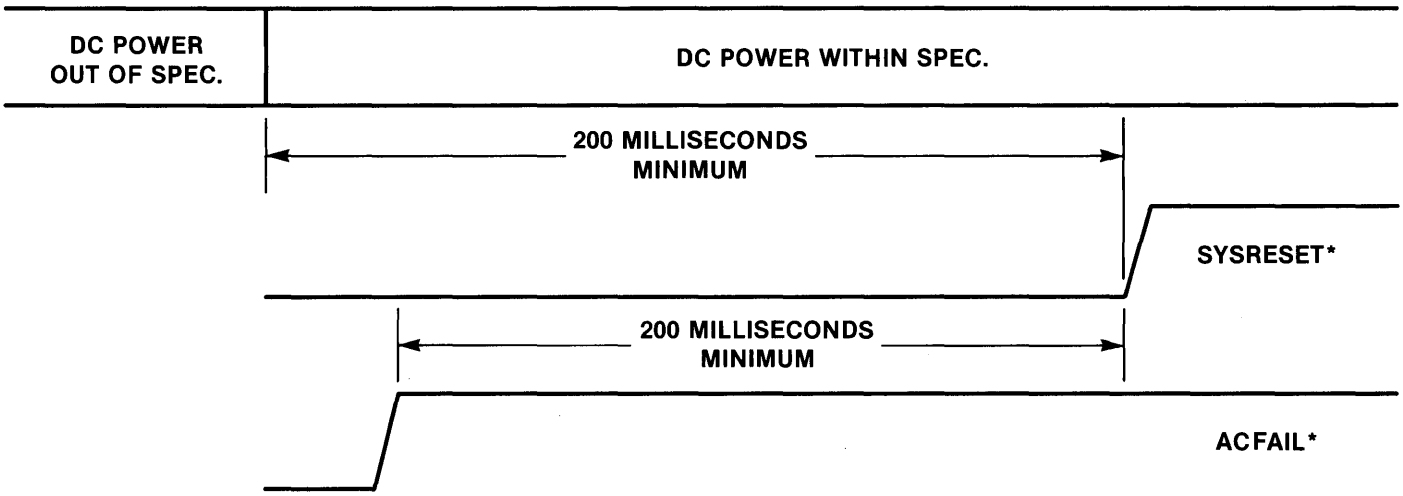


FIGURE 5-7. System Power Restart Timing

5.4 INPUT/OUTPUT LINES

The secondary 120-pin J2/P2 connector on VERSAbus has two options: expanded backplane and non-expanded backplane. The expanded bus backplane supports VERSAbus address and data bus expansion to 32 bits. It contains eight address lines plus parity, 16 data lines plus parity, 50 I/O lines, and miscellaneous reserved lines and power. The non-expanded backplane eliminates the expanded address and data line support, and uses these pins to provide 50 additional I/O pins for a total of 100 pins (see Appendix E).

5.4.1 I/O Cabling

The I/O pins on the backplane are grouped in quantities of 25 lines. Connections to these I/O lines can be made via 50 conductor flat ribbon cables, as described in paragraph 8.2.5, with two cables on the expanded backplane and four cables on the non-expanded 100 I/O pin backplane. The cable arrangement provides alternating ground/signal pairs to minimize cable signal-to-signal crosstalk.

The I/O connections are defined by the user. However, high voltage AC or DC should not be brought onto a VERSAbus card through these pins. (A 24-volt maximum for any I/O signal is recommended.)

5.4.2 Power Pins

Although individual pins on the VERSAbus cards are specified as having 2.5 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 2.5 amperes times the number of pins. Following is a table of current limits for each VERSAboard.

+5 Volts	8 A
+5 Standby	3 A
+12 Volts	6 A
-12 Volts	3 A
+15 Volts	3 A
-15 Volts	3 A

5.4.3 Reserved Lines

The VERSAbus specification labels several signal lines as RESERVED. These lines are set aside for future expansion of VERSAbus functional capabilities, and should not be used in any VERSAboard designs.

CHAPTER 6

VERSAbus OPTIONS

CHAPTER 6

VERSAbus OPTIONS

6.1 INTRODUCTION

The VERSAbus 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 VERSAboards 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 VERSAboards 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 VERSAboards will incorporate dynamic option selectivity. These boards 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 VERSAbus 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
- Use of parity lines
- Number of microseconds before MASTER bus timeout

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 Options

There are three data bus 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 Parity Options

VERSAbus provides a means for transmitting address and data parity between MASTERS and SLAVES. Since some users will find parity necessary and others will find it a needless expense, several parity options are allowed.

The user may select from four parity options:

Address Parity (AP)
No Address Parity (NAP)
Data Parity Generator and Verifier (DP)
No Data Parity (NDP)

The parity used is even parity. Even parity is defined as the Boolean result of the exclusive-OR of all values/bits equals zero. For the case of data byte 0, this may be expressed as:

```
(((D00.XOR.D01).XOR.(D02.XOR.D03)).XOR.((D04.XOR.D05).XOR.(D06.XOR.D07)))
.XOR.DPARITY0=0
```

In effect, if the number created was expressed as a 9-bit binary number, and then all of the "1" bits were counted, the resulting number should be even. For example, if the number of 1 bits in a particular byte is 5, then a "1" parity bit would be generated to make it 6, an even number. If the number of 1 bits is 4, a "0" parity bit would be generated.

NOTE

Data, address, address modifier, and longword lines are assumed to be "1" when low.

Address Parity Options

There are two address parity lines defined by VERSAbus. APARITY0* provides parity for address modifier lines AM0* through AM7*, address lines A01* through A23*, as well as LWORD* and AM0* through AM7*. APARITY1* provides parity for address lines A24* through A32*.

In addition, an Address Parity Valid (APVAL*) line is provided to indicate whether the address parity lines are being driven with valid parity.

The AP MASTER generates a parity bit on one or both address parity lines, depending upon whether it is placing a short, standard, or extended address on the bus. An AP MASTER generates a parity bit on line APARITY0* when placing short or standard addresses on the bus. It generates a parity bit on lines APARITY0* and APARITY1* when placing an extended address on the bus. Since the APARITY0* line provides parity for address lines A01* through A23*, the upper eight lines (A16*-A23*) must be in a known stable state during short address transfers. If the AP MASTER drives these lines, it must take the levels of these lines into account when generating the APARITY0* bit. If the AP MASTER does not drive these lines, they may be assumed high (logic "0"). An AP MASTER also drives APVAL* low whenever it places an address on the bus to indicate that the address parity lines are valid.

An option AP SLAVE must verify even address parity on lines for which parity bits have been generated by the AP MASTER. It determines whether address parity has been generated by monitoring the level of APVAL*. When APVAL* is low, an AP SLAVE must verify APARITY0* when decoding short or standard addresses. An extended address AP SLAVE must verify APARITY0* and APARITY1* when decoding 32-bit addresses, and must verify only APARITY0* when decoding 16-or 24-bit addresses.

A NAP (No Address Parity) MASTER will not generate parity bits on lines APARITY0* or APARITY1* or drive APVAL* low. NAP SLAVES will not monitor APVAL* and will not verify address parity.

Data Parity Options

There are four data parity lines defined by VERSAbus. DPARITY0* provides a parity bit for data lines D00* through D07*. DPARITY1* provides parity for D08* through D15*. These two parity lines are on the J1 bus connectors, and are available to all MASTER and SLAVE VERSAboards. The remaining two data parity lines, DPARITY2* and DPARITY3*, provide parity for D16* through D23* and D24* through D31*, respectively. These latter two lines are found on the J2 signal lines of expanded backplanes only.

DP (Data Parity) MASTERS and SLAVES generate even parity bits when transmitting data, and will verify even parity when receiving data from SLAVES. The parity lines driven are dependent upon which data lines carry data. A parity bit is generated for DPARITY0* or DPARITY1* for byte transfers, both DPARITY0* and DPARITY1* for word transfers, and DPARITY0* through DPARITY3* for longword transfers. DP MASTERS and SLAVES also drive DPVAL* low whenever they place data on the bus to indicate that the data parity lines are valid. They determine whether parity has been generated by monitoring the level of DPVAL*. When DPVAL* is low, they must verify the data parity.

NDP (No Data Parity) MASTERS and SLAVES do not generate data parity and do not drive DPVAL* low. In addition, they do not monitor DPVAL* and do not verify the parity on received data.

Mixing Parity Options

Because of the "parity valid" signal lines, it is possible to configure systems with a mix of parity options. While no system incompatibilities will result, the user should recognize that parity is only verified on some data transfers.

6.2.1.4 Time-out Options

Each MASTER must 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 as if it 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.). The time-out option 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.

6.2.2 Arbitration Options

The arbitration options define:

- whether the ARBITER supports power-up/down sequencing,
- what basis the REQUESTER uses to release the DTB.

6.2.2.1 There are two ARBITER options:

- PF
- NPF

A PF ARBITER provides the bus arbitration required to support orderly power-up/power-down sequencing. A PF ARBITER must always be paired with an EMERGENCY REQUESTER. Both are located on a common PC board in slot 1.

An NPF ARBITER does not support the clearing of the DTB and storage of data prior to power-down. It is not paired with an EMERGENCY REQUESTER.

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 whether 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 (or EMERGENCY 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(2-6).

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(5) 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 VERSAboard 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)
NEXP (NON-EXPANDED)
HALF (HALF SIZE)

6.2.6.1 Expanded Configuration

An expanded MASTER or SLAVE is an option A32 and/or D32 standard width (see Chapter 8) VERSAboard which is capable of driving/monitoring signal pins 89-120 of the J2 connector on the backplane.

An expanded backplane provides bused interconnects for signal pins 71-120 of the J2 edge connectors. This allows expanded MASTERS and SLAVES to make use of the extended address and data bus.

See Appendix E, Table 1.

6.2.6.2 Non-Expanded Configuration

A non-expanded MASTER or SLAVE is an option (A24 or A16)/(D16 or D8) standard width (see Chapter 8) VERSAboard which uses signal pins 71-120 of the J2 connectors as I/O lines.

A non-expanded backplane does not bus signal pins 71-120 of the J2 edge connectors. This allows these pins to be used for I/O.

See Appendix E, Table 2.

6.2.6.3 Half-Size Configuration

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

A half-size backplane has no J2 connector. The busing provided for the signal lines on the J1 connector is identical to that of the non-extended backplane.

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

Non-expanded VERSAboards should not be installed in an expanded backplane because the backplane bused signal lines on the J2 connectors will short I/O ports together from board to board. If high voltage levels (e.g., RS-232 ±12 V) are driven by some board, it may cause damage to others on the backplane.

Expanded VERSAboards may be installed in a non-expanded backplane, 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 VERSAbus and since they are able to do word transfers, they will work satisfactorily with the non-expanded boards which share the backplane.

Half-size VERSAboards may be installed in either expanded or non-expanded backplanes. However, the card cage above the backplane will provide no support for one edge of the board (this may be satisfactory for development purposes).

Expanded and non-expanded VERSAboards cannot be installed in a half-size backplane because of the obvious mechanical incompatibility.

It would be possible to construct a backplane/cardcage which would support all three VERSAboard 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 VERSAbus compatible products may use the option notation described in this chapter to concisely describe product features.

VERSAbus Card Rack
SPECIFICATION

ENVIRONMENTAL OPTIONS:

OPERATING TEMPERATURE: 0° C to 70° C

STORAGE TEMPERATURE: -65° C to +150° C

MAXIMUM OPERATING HUMIDITY: 90%

PHYSICAL CONFIGURATION OPTIONS

EXP (4 SLOTS)

NEXP (4 SLOTS)

VERSAboard
SPECIFICATION

MASTER DATA TRANSFER OPTIONS

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

ARBITER OPTIONS

ANY ONE OF PF, or NONE (STAT)

REQUESTER OPTIONS

ANY ONE OF R(0), R(1), R(2), R(3), or R(4) (STAT)
ROR or RWD

INTERRUPT HANDLER OPTIONS

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

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

VERSAbus ELECTRICAL SPECIFICATIONS

CHAPTER 7

VERSAbus ELECTRICAL CONSIDERATIONS

7.1 INTRODUCTION

This chapter defines the non-timing electrical specifications for proper VERSAbus interfacing. VERSAbus 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 VERSAbus 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.

$\pm$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.

$\pm$15 Vdc voltages are intended for analog specifications. These voltages may be used directly by VERSAboards, or further regulated on-board where required. These voltage sources should be very carefully regulated with respect to AC ripple and noise filtering. Care should be taken in their use to avoid superimposing voltage variations on these lines due to varying loads or noise.

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 edge connector of any card plugged into the backplane.

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

VARIATION = DISTRIBUTION + LINE-REGULATION + LOAD-REGULATION

TABLE 7-1. Bus Voltage Specifications

MNEMONIC	DESCRIPTION	VARIATION (see NOTE)	RIPPLE & NOISE BELOW 10 MHz (PK-PK)	CONNECTOR P1 PIN NUMBER	CONNECTOR P2 PIN NUMBER	MAXIMUM CURRENT DRAW PER SLOT
+5V	+5 Vdc power	+5.0%/-2.5%	50 mV	1,2,129-132	7 - 10	8 amps
+12V	+12 Vdc power	+5.0%/-3.0%	50 mV	125 - 128	11 - 12	6 amps
-12V	-12 Vdc power	+3.0%/-5.0%	50 mV	121 - 122	15 - 16	3 amps
+5V STDBY	+5 Vdc standby	+5.0%/-2.5%	50 mV	133 - 134	-	3 amps
+15V	analog power	+5.0%/-3.0%	25 mV	-	69 - 70	3 amps
-15V	analog power	+3.0%/-5.0%	25 mV	-	67 - 68	3 amps
GND	ground	Ref.	Ref.	3,4,23-24, 27-28,31-32, 61-62,67-68, 71-72,119-120, 123-124,139-140	1-2,3-4,5-6, 97-98,101-102	
15V GND	$\pm$15V ground return	Ref.	Ref.	-	13 - 14	

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

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.

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 Ground Distribution

The main ground distribution (mnemonic "GND") is the system return for all +5 Vdc and ±12 Vdc current. A large number of connector pins have been provided along both connectors, and these allow an extensive, low inductance ground network. These pins should be tied to a common ground plane in the backplane pc board.

The ±15 Vdc ground return should be kept separate from the main system ground in the backplane. The analog power must have very low noise and must have little DC variation from the system ground at the analog conversion devices in the system. Therefore, the ±15 Vdc ground return should be tied to the system ground only on the pc board with analog conversion logic devices and, further, a very low resistive loss interconnect (large metal areas) should be used to avoid DC voltage variation in the ground potential.

7.2.3 Card Edge Connector Electrical Ratings

The card edge 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: 2.5 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 VERSAbus 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 $\leq$ Low level $\leq$ 0.8 V
b. 2.0 V $\leq$ High level $\leq$ 5.0 V

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

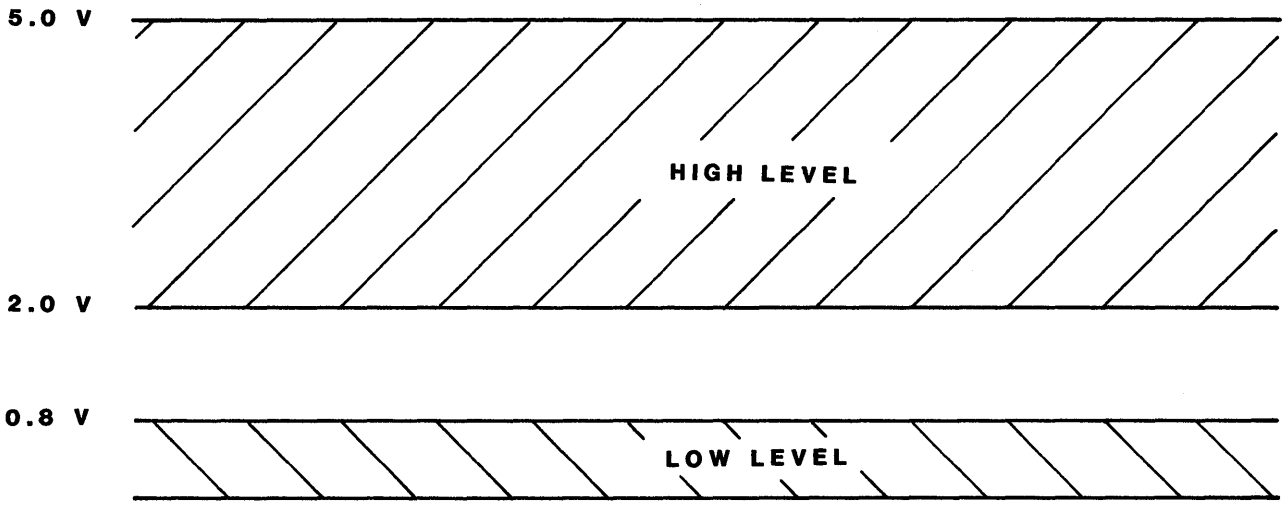


FIGURE 7-1. VERSAbus Signal Levels

Depending on the function required, VERSAbus 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. The 74S240 can be used for totem-pole or three-state drivers requiring 64 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})		0.55	V	Sink 64 mA
(High current)	High state (V_{OH})	2.4		V	Source 3 mA
Totem-pole	Low state (V_{OL})		0.5	V	Sink 8 mA
(Low current)	High state (V_{OH})	2.7		V	Source 400 uA
Three-state	Low state (V_{OL})		0.55	V	Sink 64 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

VERSAbus is an asynchronous, high speed bus intended for high performance systems. Generally, the signal line distance on the backplane will be short (18" maximum) 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 VERSAbus 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.

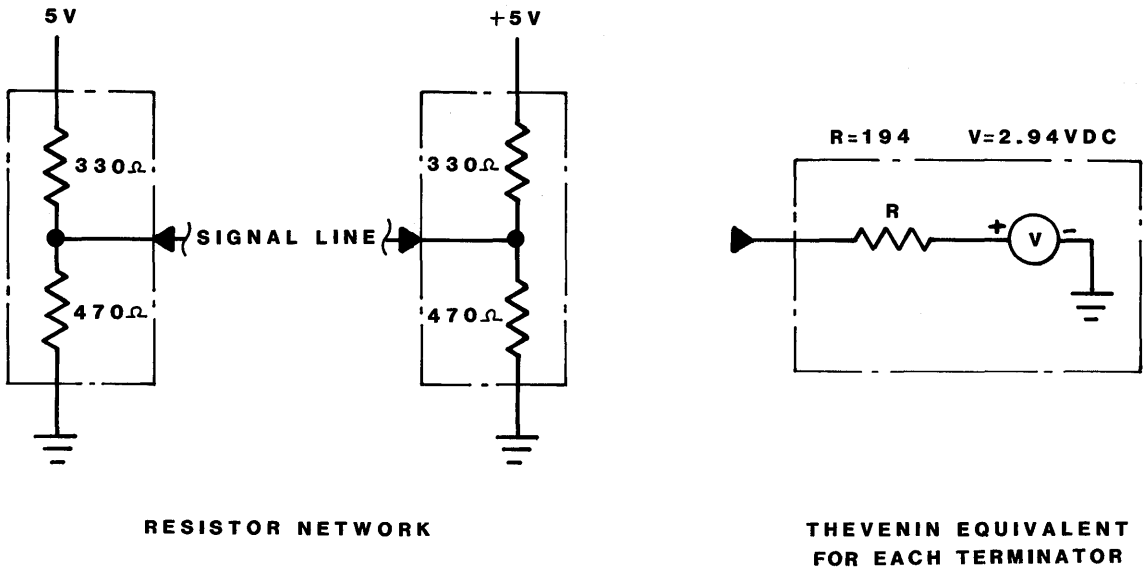


FIGURE 7-2. Termination Network

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 no 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.

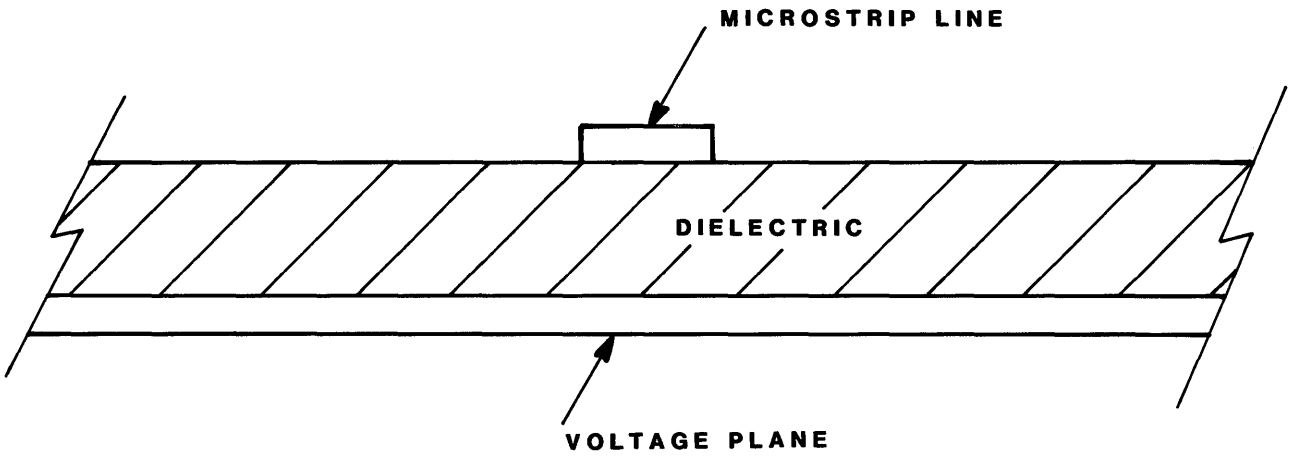


FIGURE 7-3. Backplane Microstrip Signal Line Cross Section

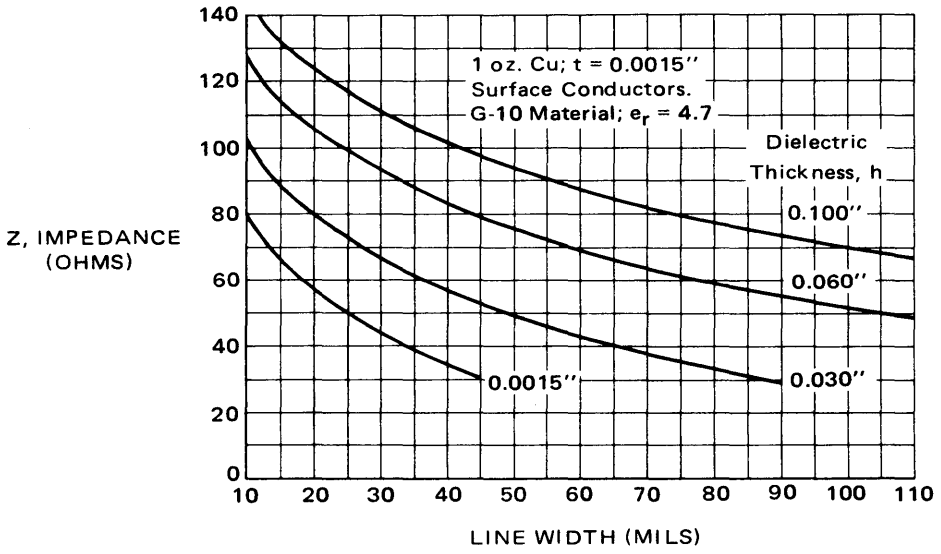


FIGURE 7-4. Impedance versus Line Width and Dielectric Thickness for Microstrip Lines

As stated in section 7.6.1, the terminations on VERSAbus 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 unloaded Z_o (considering no 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 effective characteristic impedance 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 7-5.

The resulting Z_o' should not go below 60 ohms.

When PC boards are in the svstem, 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 VERSAboard, the following rules apply:

a. Total capacitive load on any VERSAbus 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 VERSAbus can be calculated using the capacitance per foot shown in Figure 7-5.

b. Any signal line input to a VERSAboard 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 8 pf.

7.6.4 I/O Pin Voltage/Current/Frequency Constraints

To limit noise effects, the following limits are <u>recommended</u> for each I/O pin:

- Maximum voltage = $\pm$ 24 volts
- Maximum Rate of Change = 2.5 volts/nanosecond
- Maximum Current = 1 amp

CHAPTER 8

MECHANICAL SPECIFICATIONS

CHAPTER 8

MECHANICAL SPECIFICATIONS

8.1 INTRODUCTION

Information in this chapter is provided to assure that VERSAbus backplanes, card racks, and PCB's are mechanically compatible. Throughout this chapter, specific vendor part numbers are given which may be used to meet the requirements of this specification. As long as the specifications given in this chapter are complied with, any compatible vendor part may be substituted.

Throughout this chapter, the following terminology is used:

Backplane - A multilayer PC board into which 140-pin and, in most cases, 120-pin edge connectors are installed. This PC board interconnects some of the pins of these connectors to provide a bus.

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

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

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

The "bottom edge" of a VERSAboard is the edge which provides the PC fingers for insertion into the edge connectors of the backplane.

8.2 VERSAbus BACKPLANE

This portion of text provides the following VERSAbus 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 description of a multiple layer design composed of two discrete, two-sided, glass epoxy boards separated by a mylar insulator with a mylar clear top layer. Figure 8-1 illustrates this backplane construction. The two 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. Refer to Figure 8-2. The first and fourth layers of the backplane contain the signal conductors. The second layer provides the voltage plane, and the third layer is the ground plane. Contact pins are press fitted into the two boards and the mylar insulators, forming the four-layer backplane. This process relies upon the backplane to provide the structural rigidity of the edge connector. Plastic insulators are then inserted over the contacts to form edge connectors.

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

NOTE

Due to the large number of contacts on the VERSAboards that plug into the backplane, the board insertion forces are quite high. Therefore, stiffness of the backplane is a major design consideration.

8.2.2 Reference Designations and Pin Numbering Standards

The following standards are recommended for backplane identification purposes (see Figure 8-3):

a. Card slot locations are designated A1, A2, A3, etc. The numbering technique is illustrated in Figure 8-3. Sequence must be consistent with the direction of flow of the daisy-chains, with A1 being first in the chain.

b. Edge connectors on the backplane will be designated J1 and J2. J1 is the 140-pin socket, and J2 is the 120-pin socket.

c. Numbering of both the 140- and 120-pin sockets is depicted in Figure 8-3. Odd numbered pins are associated with the left side of the edge connector, and the even pins with the right side.

d. Additional connectors (e.g., for power) located on the front side of the backplane will be designated P1, P2, P3, etc. The numbering technique is illustrated in Figure 8-3.

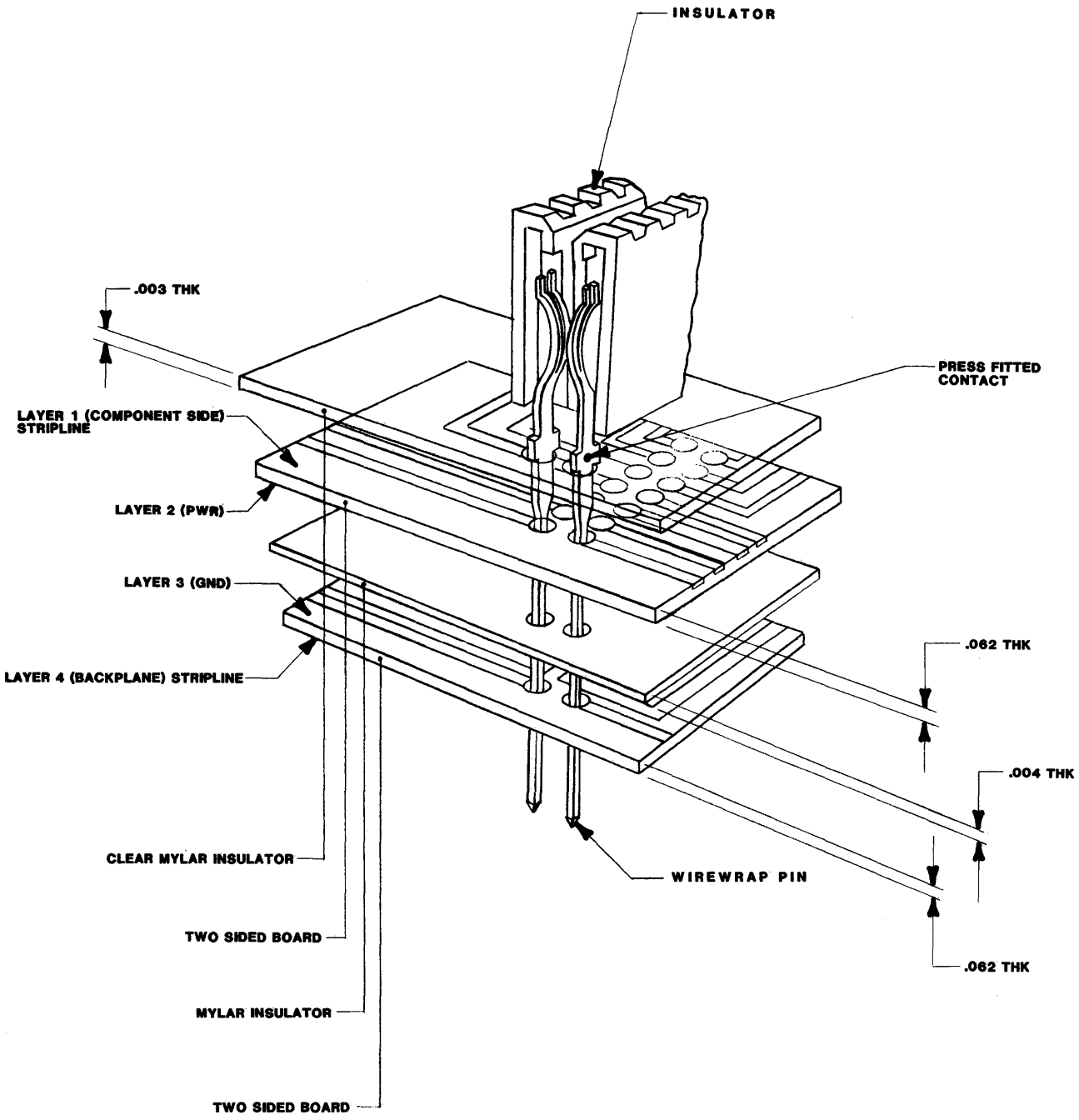


FIGURE 8-1. Typical Multilayer Backplane/PCB Construction Technique

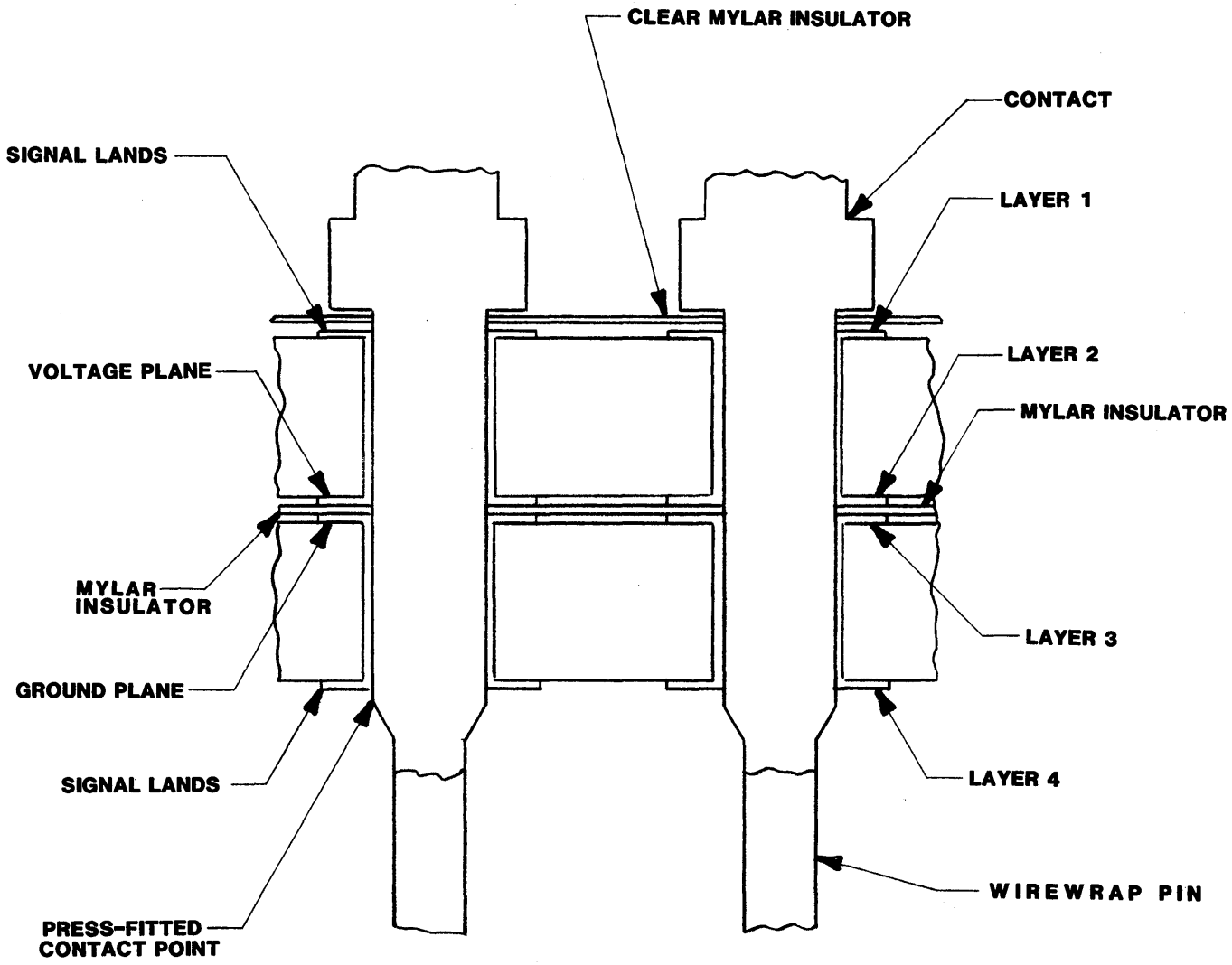


FIGURE 8-2. Typical Multilayer Backplane/PCB Cross-Sectional Area View

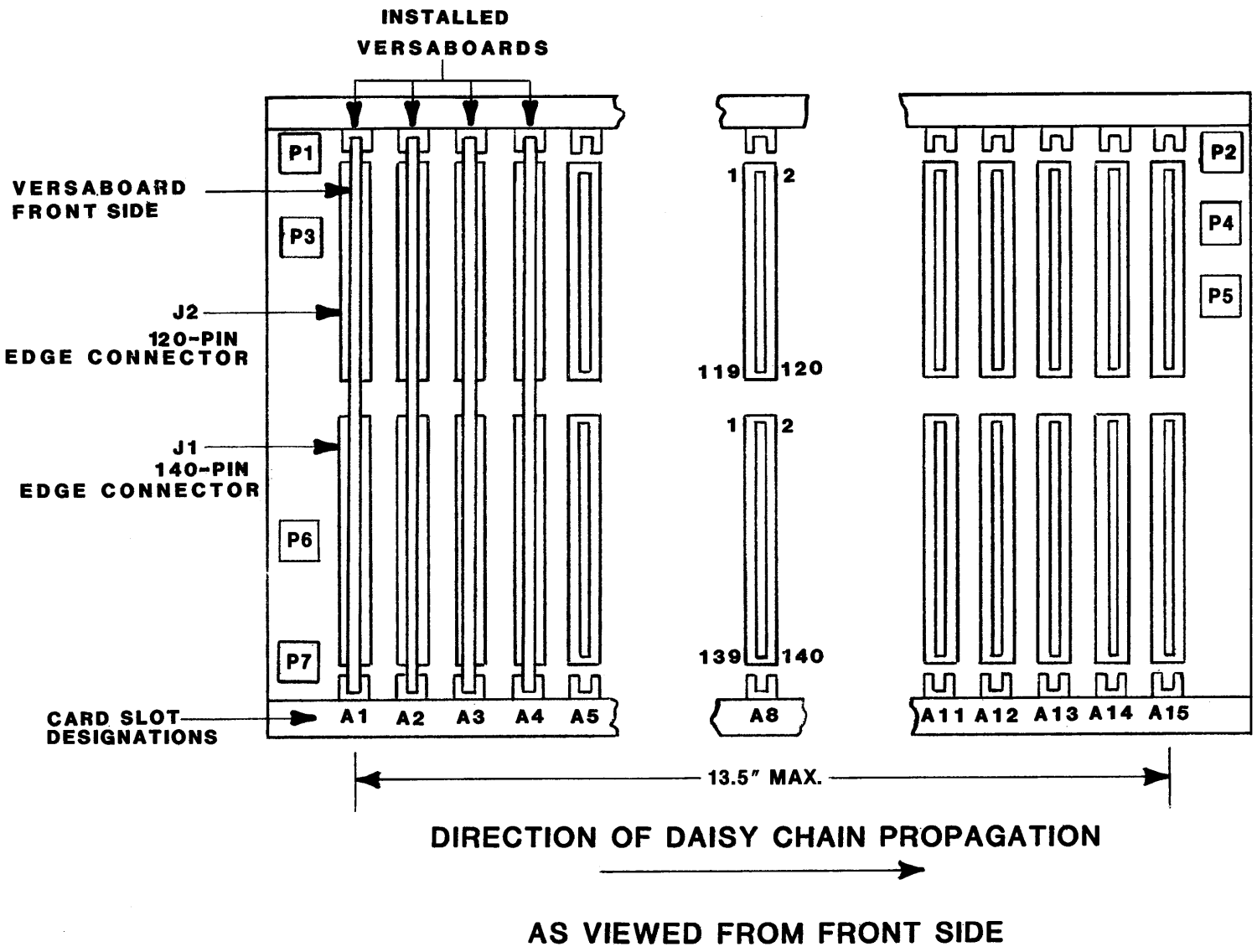


FIGURE 8-3. Backplane Reference Designations and Pin Numbering Standard

8.2.3 Backplane/VERSAboard Dimensional Requirements

Certain specifications must be adhered to when designing a VERSAbus backplane and compatible VERSAboards. Figure 8-4 illustrates the following relationships:

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

    1. Printed Circuit Board spacing - .900 minimum
    2. Wire Wrap Board spacing - 1.4 minimum

b. Board Thickness (BT)- board thickness is 0.062 $\pm$ .005 inch.

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

d. Component Height (CH) - height of the components on the front of each VERSAboard must not exceed 0.50 inch.

e. Board Warpage (BW) - maximum allowable VERSAboard warpage is .125 inch.

f. Wire Wrap Pin Height (WW) - length of the wire wrap pins protruding through the back of the VERSAboards must not exceed .650 inch.

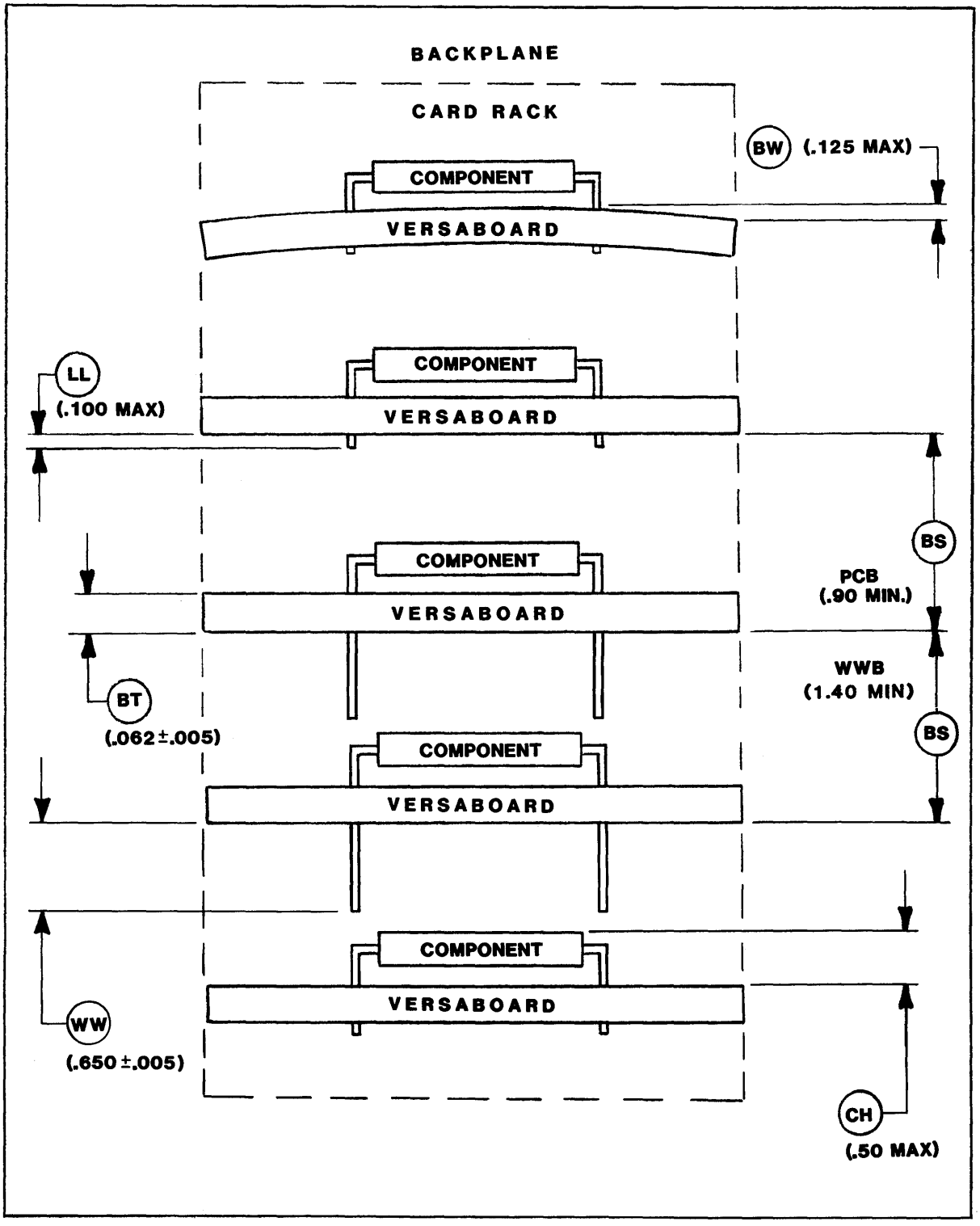


FIGURE 8-4. Backplane/VERSAboard Dimensional Requirements

8.2.4 Edge Connectors

The P1 and P2 edge fingers on the VERSAboard are inserted into two edge connectors mounted to the backplane. There are basically two types of edge connectors:

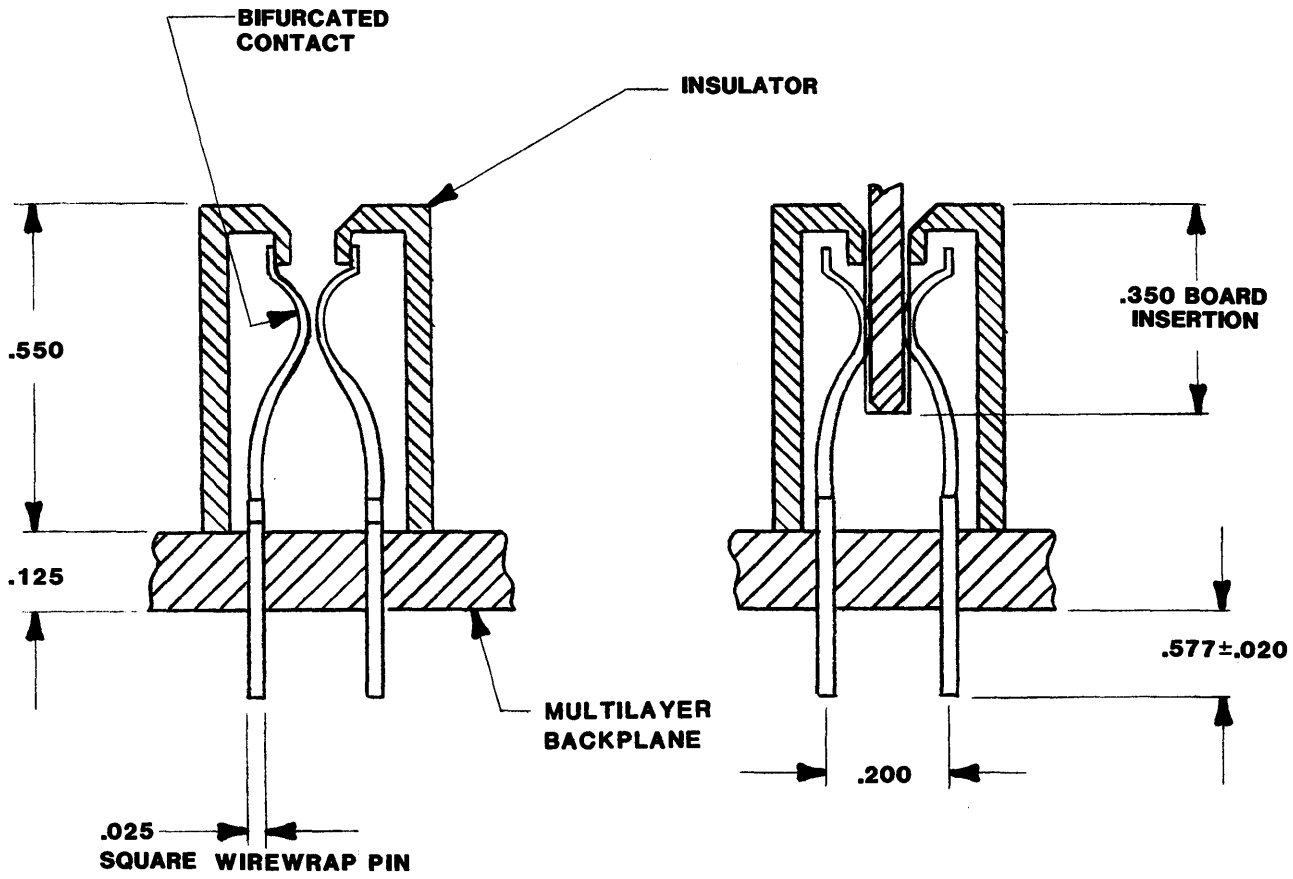
a. modular
b. prefabricated

Modular edge connectors are installed on the backplane in two steps. First, contacts are press fitted into the backplane. Plastic insulator strips are then inserted over the socket contacts to form the edge connector. Refer to Figure 8-5. Recommended vendors are SYMTRON and ELFAB.

Prefabricated edge connectors are inserted into the backplane in a one-step operation. The socket contacts and socket insulator construction is of a one-piece design. Recommended vendors are SAE and AMP.

CAUTION

These edge connnnectors are not necessarily interchangeable dimensionally, and may affect the design of the card cage (see Figure 8-18).



CAUTION

THE ABOVE DIMENSIONS CAN VARY,
DEPENDING ON MANUFACTURER.

FIGURE 8-5. Typical Backplane Edge Connector

8.2.5 Auxiliary Pins

It is a VERSAbus requirement that auxiliary pins be provided on the backplane. These pins must all be connected to ground and mounted per Figure 8-6. There must be one such pin for each unbussed pin in connector P2.

8.2.6 I/O Connections

Ribbon cable I/O connections are made via ribbon connectors which are pushed onto the wirewrap pins protruding from the back of the backplane. These wirewrap pins are part of the contacts of connector J2. The ribbon connectors push onto pins 17 through 66 and 71 through 120. For proper placement on the row of wirewrap pins, keying headers are installed over the pins, and then the ribbon cable connector is inserted into the keying header. See Figures 8-6 through 8-9.

The I/O connector is a 50-conductor ribbon cable female connector that is inserted into the keying header. See Figure 8-6. As illustrated, the ribbon connector is molded plastic. A convenient strain relief handle or plastic loop handle is recommended to minimize stress on the connector and cable assembly during removal operations. See Figure 8-6. Recommended vendors are 3M and AMP.

The keying headers may be installed two different ways on the backplane.

Figures 8-8 and 8-9 illustrate single and dual I/O connector header configurations.

See Figure 8-8. The first method (called the single cable method) allows 50 I/O signals to be taken from the backplane on one 50-conductor ribbon cable. While this minimizes the amount of cabling required, it may introduce unacceptable levels of cross-talk between signal lines.

See Figure 8-9. The second method (called the dual cable method) takes 25 I/O signals from the backplane in each of two 50-conductor cables. The additional 25 lines in each ribbon cable are grounded by the auxiliary ground pins which parallel the two rows of signal pins.

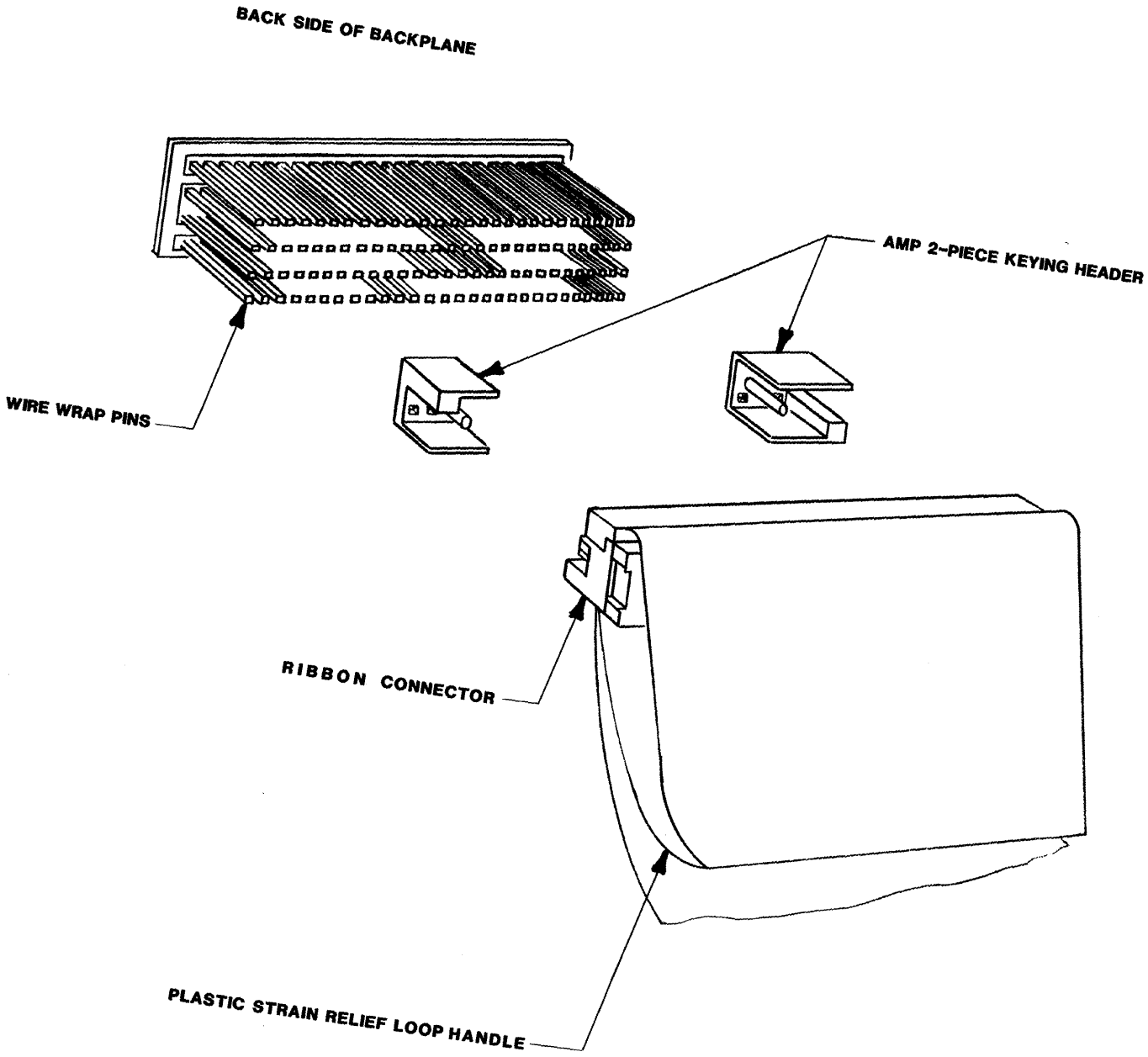


FIGURE 8-6. I/O Cable Connection

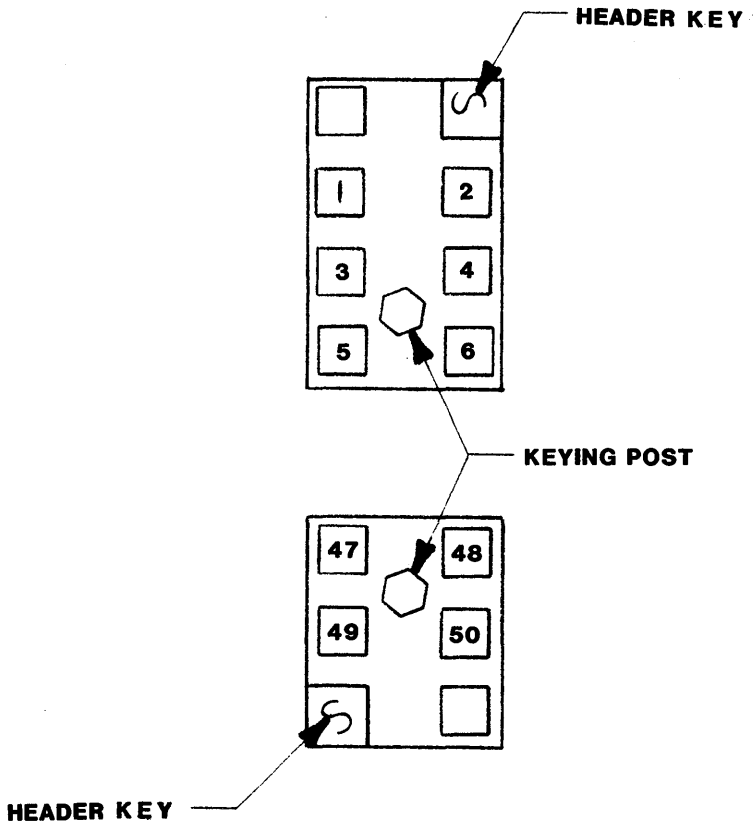
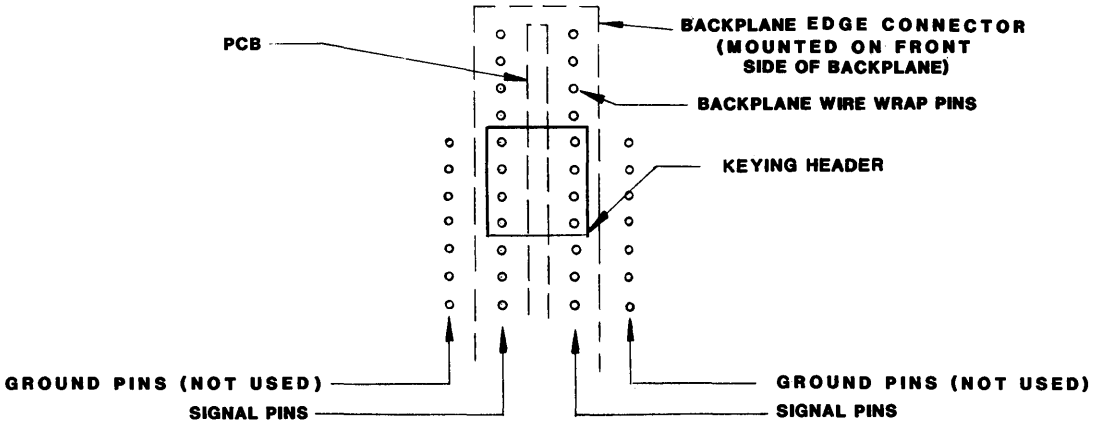


FIGURE 8-7. AMP Two-Piece Keying Header



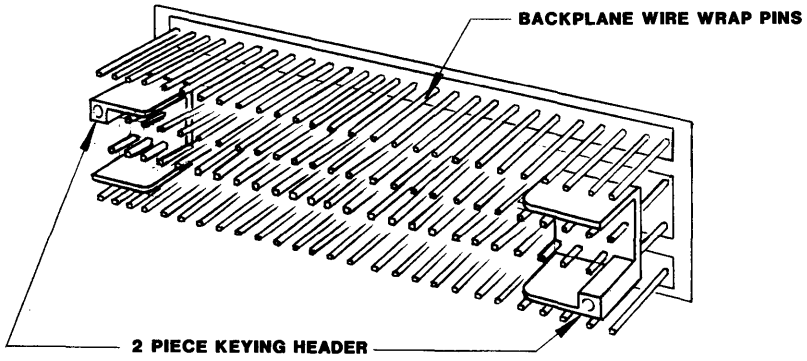


FIGURE 8-8. Single Cable Method Keying Header Configuration

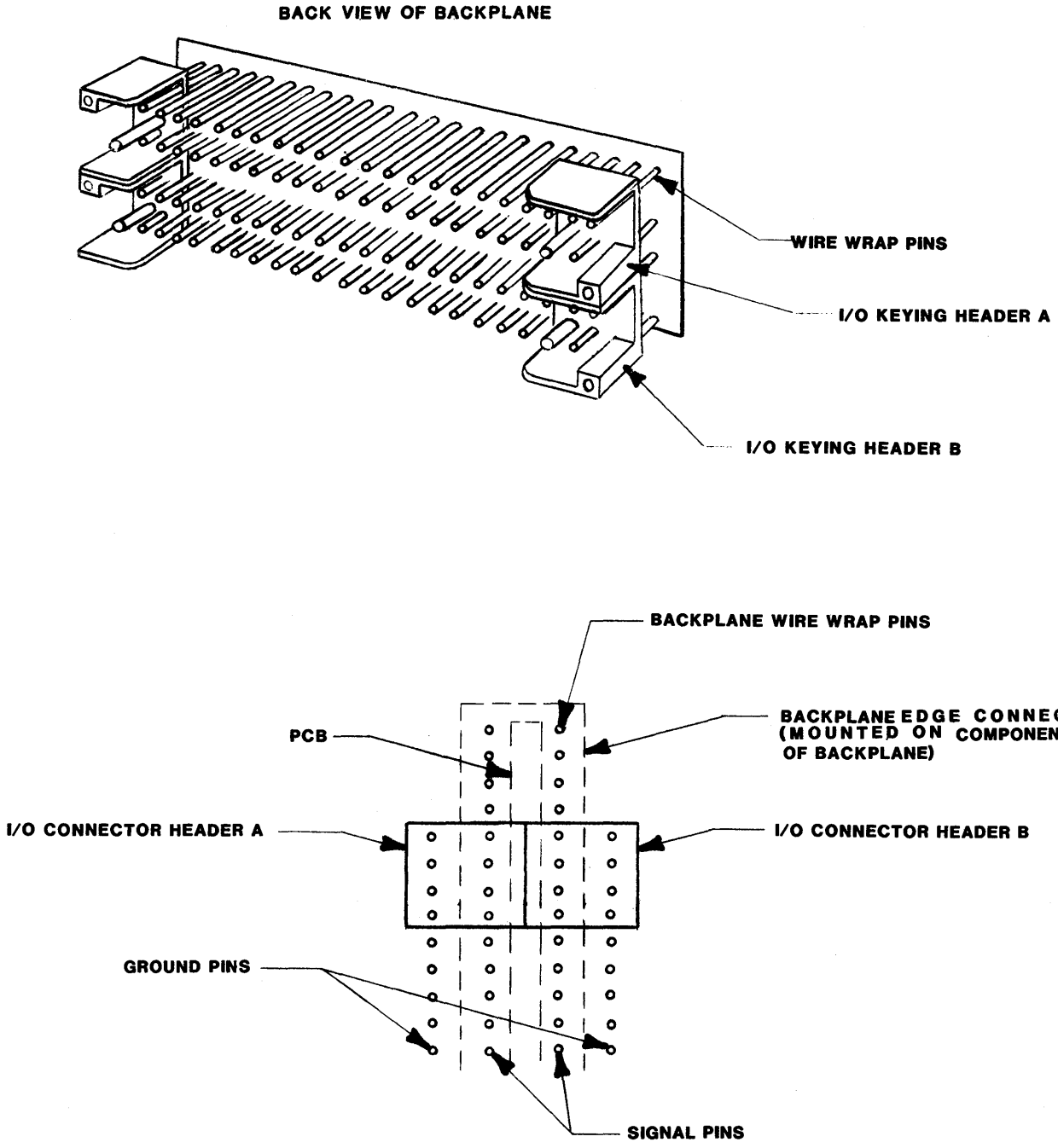


FIGURE 8-9. Dual Cable Method Keying Header Configuration

8.3 VERSAboards

This portion of text provides the following VERSAboard information:

a. construction techniques,
b. reference designations and pin numbering standards,
c. overall dimensions,
d. edge connector information,
e. ejector information.

8.3.1 VERSAboard Construction Techniques

Many PCB construction techniques are available to the designer. Because of the large physical size of the board, a multilayer PC design is recommended; however, this is not a requirement for VERSAbus 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 8-10. The following standards are recommended for PCB identification purposes:

a. Edge connectors on the bottom edge of the VERSAboard are designated P1 and P2. P1 is the 140-pin edge connector, and P2 is the 120-pin edge connector.

b. Numbering of both the 140- and 120-pin edge connectors is depicted in Figure 8-10. Odd numbered pins make contact with the PCB component side, and the even pins with the solder side.

c. Edge connectors on the top edge of the PCB are designated J1, J2, J3, etc.

d. Numbering of J1, J2, and J3 edge connectors is depicted in Figure 8-10. Odd numbered pins are associated with the PCB component side, and the even pins with the solder side.

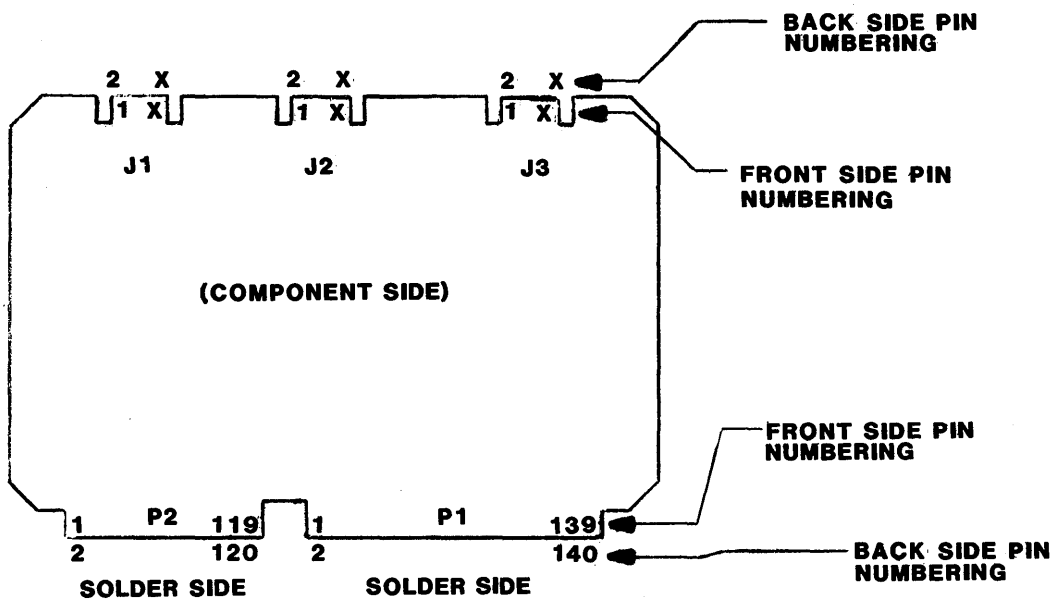


FIGURE 8-10. VERSAboard Reference Designations and Pin Numbering Standards

8.3.3 VERSAboard Dimensions

Figure 8-11 illustrates the standard dimensions applicable to VERSAboards. In addition to the standard dimensions, component mounting zones and "top edge" cabling clearances for card rack enclosures are given. The remainder of the PCB, including edge connectors P1 and P2, must be within the dimensions shown in Figure 8-11.

Figure 8-12 illustrates the alternate (half size or reduced) VERSAboard dimensions. These dimensions enable a designer to manufacture a half size board that is compatible only with the 140-pin VERSAbus backplane socket J1. Unless the free edge is supported, it is not recommended that this size board be used in a card cage with full size boards.

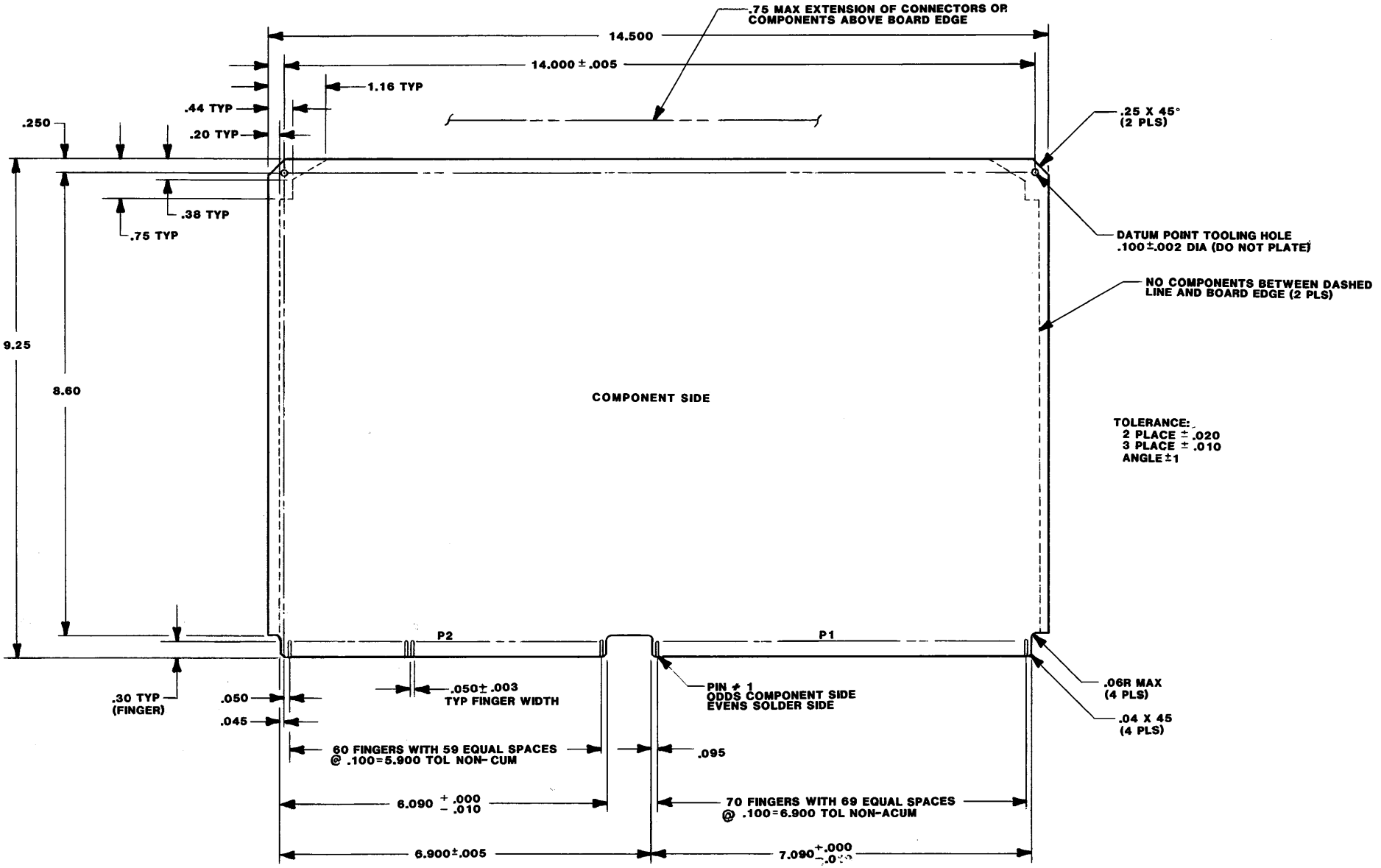


FIGURE 8-11. Standard Size PCB

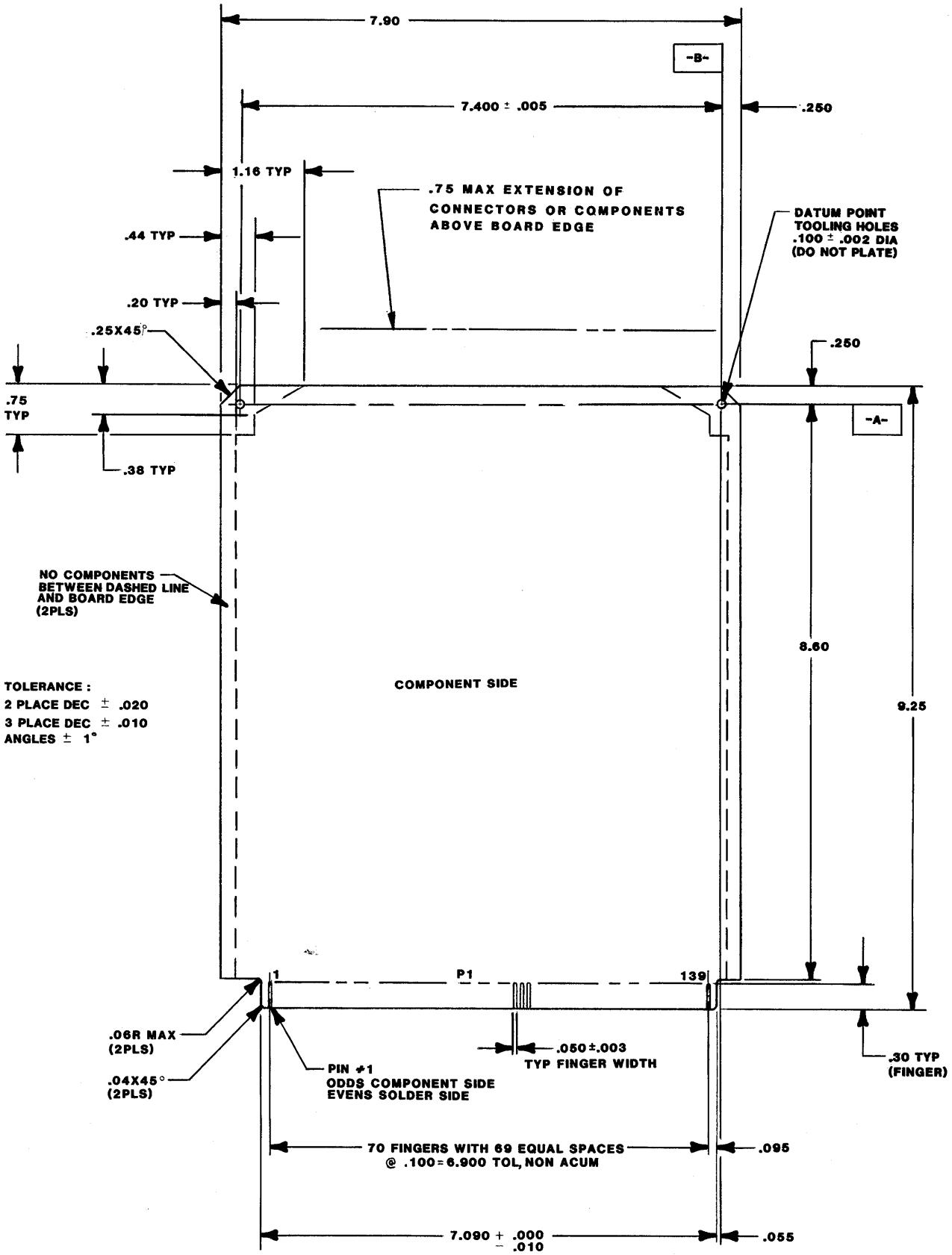


FIGURE 8-12. Half Size PCB

8.3.4 VERSAboard Bus Edge Connectors

Figures 8-11 and 8-12 define the VERSAboard bus edge connector requirements. All information contained in Figure 8-13 must be adhered to for VERSAbus compatibility.

8.3.5 VERSAboard Non-bus Edge Connectors

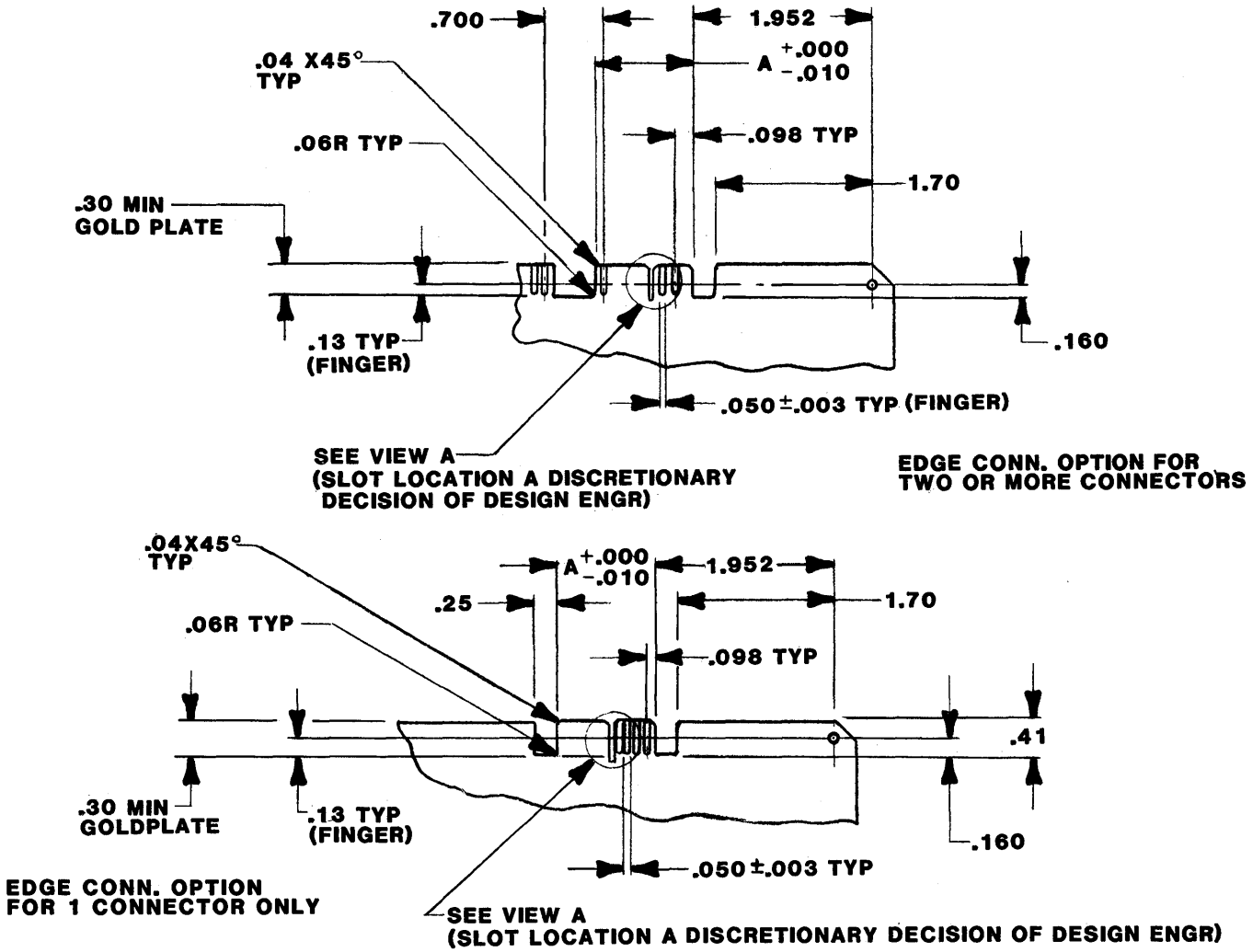
Figure 8-13 illustrates the PCB non-bus edge connector type and spacing information. There is no prescribed limit to the number or type of I/O connections which may be made off the top edge of the board as long as they don't exceed the allowed dimensional restrictions given in Figure 8-13.

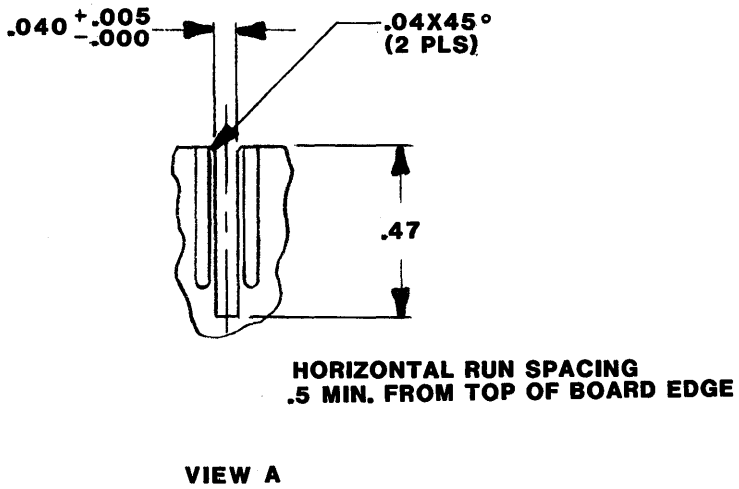
NOTE

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

8.3.6 VERSAboard Ejectors

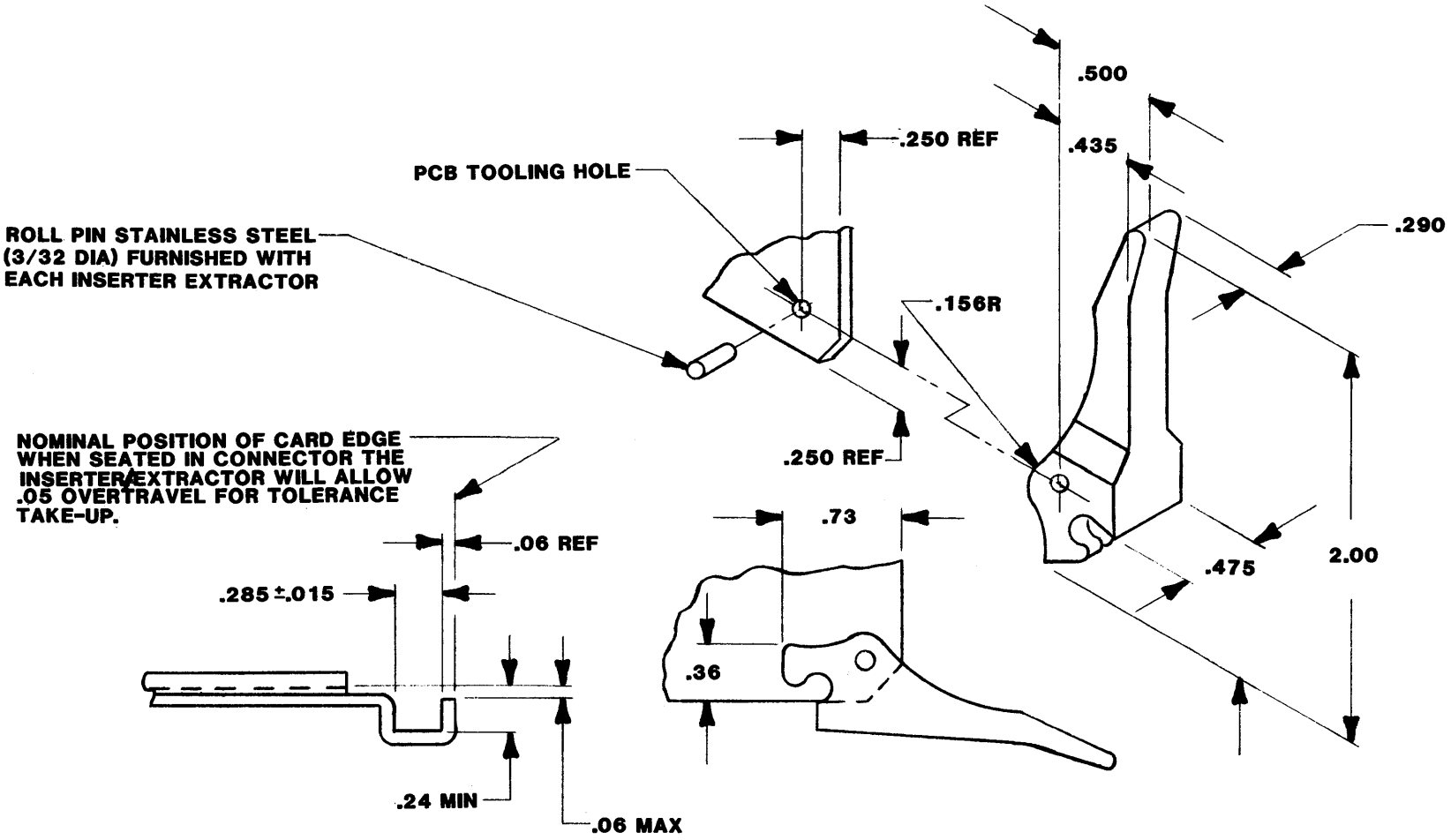
VERSAboard ejectors facilitate the insertion and removal of VERSAboards within a card rack environment. Figure 8-14 defines the ejector and associated card rack requirements. A recommended vendor is CALMARK.





CONTACT QTY	DIM A
20	1.095
26	1.395
34	1.795
40	2.095
50	2.595

FIGURE 8-13. Typical VERSAboard Non-bus Edge Connectors



INSERTER EXTRACTOR CALMARK P/N 107-1013

APPLICATION DATA : TWO INSERTER/EXTRACTORS ARE REQUIRED PER CARD. PROVIDES INSERTION AND EXTRACTION TRAVEL OF .35 MINIMUM WITH A MECHANICAL ADVANTAGE OF APPROX 4:1 .

FIGURE 8-14. VERSAboard Ejectors

APPENDIX A

GLOSSARY OF VERSAbus TERMS

Axx* - the symbolic notation for a particular address line on the VERSAbus, where 'xx' may have the values 01 through 31. These lines are driven by the module currently designated as the VERSAbus 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 VERSAbus 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 VERSAbus 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.)

APARITY0* - the parity line associated with the signal lines A01* through A23*, LWORD*, and AM0* through AM7*. This line represents the 'exclusive OR' of the 32 lines above. Therefore, the 'exclusive OR' of this line with all of those lines should always be zero. This condition is often referred to as even parity. If the MASTER currently in control of the VERSAbus and the SLAVE being addressed are both option AP (meaning: address parity), then the MASTER will generate this parity signal and the SLAVE will verify it. (See the section on SUBSET compatibility for further information on other combinations of MASTER and SLAVE options.)

APARITY1* - the parity line associated with the signal lines A24* through A31*. This line represents the exclusive OR of these eight additional address lines used in extended addressing on the VERSAbus, Option A32. (See APARITY0* for additional details and the section on SUBSET compatibility for information on combinations of MASTER and SLAVE options.)

AP - the option designated when a module will properly generate (if MASTER) or test (if SLAVE) address parity.

APVAL* - a signal line driven low by AP MASTERS whenever they place an address on VERSAbus. This low level indicates to the addressed SLAVE that the APARITY lines are valid. If the SLAVE is option AP, it will verify this parity.

ARBITER - the term used to reference the logic circuitry connected to VERSAbus 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 VERSAbus. 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 VERSAbus, where 'x' may have the values 0 through 4. 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-chains. 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 VERSAbus, where 'x' may have the values 0 through 4. 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 VERSAbus, 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 VERSAbus lines involved in a data transfer, consisting of the address lines, the address parity lines, the data lines, the data parity lines, and the lines LWORD*, WRITE*, AS*, DS0*, DS1*, DTACK*, and BERR*.

DPARITY0* - the parity line associated with the signal lines D00* through D07*. This line represents the 'exclusive OR' of those eight data lines. Therefore, the 'exclusive OR' of this line with all of those lines should always be zero. This condition is often referred to as even parity. If the module currently providing data to the VERSAbus is defined as having option DP, it will generate this signal. If the module currently receiving the data is defined as having option DP, it will verify this parity and flag errors. (See the section on SUBSET compatibility for further information on various combinations of options.)

DPARITY1* - the parity line associated with the signal lines D08* through D15*. This line represents the 'exclusive OR' of eight of the data lines referenced by DS1*. (See DPARITY0* for more details.)

DPARITY2* - the parity line associated with the signal lines D16* through D23*. This line represents the 'exclusive OR' of eight of the data lines used during longword transfers, and is only driven by modules having option D32. (See DPARITY0* for more details on parity, and the section on SUBSET compatibility for further information on various combinations of options.)

DPARITY3* - the parity line associated with the signal lines D24* through D31*. This line represents the 'exclusive OR' of eight of the data lines used during longword transfers, and can only exist on modules having option D32. (See DPARITY0* for more details on parity, and the section on SUBSET compatibility for further information on various combinations of options.)

DPVAL* - a signal line driven low by DP MASTERS and DP SLAVES whenever they place data on VERSAbus. This low level indicates to the MASTER/SLAVE receiving the data that the DPARITY lines are valid and may be used to verify data.

DP - the option designated when a module will generate data parity when presenting data, and test data parity when receiving data.

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.)

LWORD* - the signal on the VERSAbus used to invoke 32-bit data transfers (See also LONGWORD.)

MASTER - a module capable of requesting control of the VERSAbus data transfer bus via its associated REQUESTER, and upon being signaled by its REQUESTER that the data transfer bus has been granted, is capable of addressing another module by driving the address lines and sending data to or receiving data from the module so addressed.

NAP - the option designated when a module will not generate (if MASTER) or test (if SLAVE) address parity.

NDP - the option designated when a module will not generate or test data parity.

NPF - the option designated for a VERSAbus ARBITER which does not have the additional logic on board to allow response to a system MASTER under emergency conditions such as power down. (See chapter on bus arbitration and definition of PF for further details.)

PF - the option designated for a VERSAbus ARBITER capable of responding to the BREL* signal as an emergency bus request. (See chapter on bus arbitration for further details.)

READ - a data transfer initiated by a MASTER, with the data flow from MASTER to SLAVE.

SLAVE - a module capable of decoding the address lines of the VERSAbus, 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. Through the use of a PF ARBITER, the system MASTER can gain quick control of the data transfer bus for emergency purposes.

WRITE - a data transfer initiated by a MASTER, with the data flow from MASTER to SLAVE.

APPENDIX B

STATE DIAGRAM NOTATION

State diagrams are used to describe in a complete and unambiguous manner how a given functional module will respond to incoming signals. They are most useful when the possible combinations of input signal levels are very large or when the timing of these signals is unpredictable (e.g., arbitration logic). In these cases, many different timing diagrams would be required to describe how a device would act in all situations.

Another useful feature of state diagrams is their ability to completely define the state which a functional module will enter upon system initialization. Failure to define this can lead to system lock-ups or desynchronization when the system is initialized.

In some cases, the output of a functional module may change levels as the result of an incoming signal level change. In other cases, the incoming signal may only cause the state of an internal flip-flop to change. Each time an output changes or a flip-flop changes state, the functional module is said to enter a new "state".

States are represented on state diagrams as circles. Each circle is labeled with a string of letters which shows the level of each output line and the state of each internal flip-flop. When a device's outputs change levels or when its internal flip-flops change state, it is said to make a "transition" from one state to another. The only transitions allowed on a state diagram are those where only one level or one flip-flop changes state. Each allowed transition is represented on the state diagram by an arrow between two states. Each arrow is labeled with a set of conditions which must be met before that transition may take place. The functional module must remain in its current state until all of the conditions required for making a transition to another state become true.

There are several types of conditions which may have to be met.

1. A VERSAbus signal line may have to be at a prescribed level.

2. A signal level may have to be present from an on-board device.

3. Another state diagram may have to be in a prescribed state (this is used to guarantee minimum delays between two output signal level changes).

When more than one condition must be met for a transition to occur, the "AND" symbol is inserted between the conditions listed. When any of several conditions is sufficient to allow the transition to take place, the "OR" symbol is inserted.

When another state diagram must be in a prescribed state for the transition to take place, the name of that state is shown as a label with an oval around it.

A functional module is not required to make a transition immediately just because all conditions for that transition have been met. In most cases, a slow module will reduce the performance of a system but will not cause any system failure. There are a few cases, however, where the device is required to respond within a prescribed time limit. Where this is true, the transition is labeled "within t", where 't' is the maximum time allowed.

In some cases, the reader may find that the same set of conditions allows transitions to two possible states. In this case, the module may be designed to make either state transition.

The VERSAbus specification allows the user to implement different subsets of some modules. Where this is true, the state diagram may contain conditions and transitions which are required by one option but not another. In these cases, the optional transitions are clearly labeled.

A special case exists for the sytem reset line of the VERSAbus. When this line is driven low, all modules on the VERSAbus are required to enter a prescribed state. The initial state on the state diagram is pointed to by an arrow labeled "SYSRESET*=L". Whenever this reset line is low, the module should enter this state within the prescribed time limit and remain in it until the system reset line goes high.

APPENDIX C

VERSAbus CONNECTOR/PIN DESCRIPTION

INTRODUCTION

This appendix describes the VERSAbus pin connections. The following table identifies the VERSAbus signals by signal mnemonic, connector and pin number, and signal characteristic. Unless otherwise specified, all signal lines are driven by the master.

VERSAbus Signal Identification

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
ACCLK	J1: 69	AC CLOCK - Open collector driven clock signal generated by the power monitor that indicates the power line frequency and zero voltage transition points.
ACFAIL*	J1: 78	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.
ACKIN*	J1: 95	ACKNOWLEDGE IN - Totem-pole driven signal. ACKIN* and ACKOUT* signals form a daisy-chained acknowledge. The ACKIN* signal indicates to VERSAboard that an acknowledge cycle is in progress.
ACKOUT*	J1: 96	ACKNOWLEDGE OUT - Totem-pole driven signal. ACKIN* and ACKOUT* signals form a daisy-chained acknowledge. The ACKOUT* signal indicates to the next board that an acknowledge cycle is in progress.
AM0*-AM7*	J1: 59,60, 63,83-86, 94	ADDRESS MODIFIER (bits 0-7) - Three-state driven lines that provide additional information about the address bus - such as size, cycle type, and/or DTB master identification.
APARITY0*	J1: 33	ADDRESS PARITY 0 - Three-state driven signal which provides an even parity bit for address bits A01*-A23*, LWORD*, and AM0*-AM7*.
APARITY1*	J2: 88	ADDRESS PARITY 1 - Three-state driven signal which provides an even parity bit for address bits A24*-A31* for use in 32-bit expansion.

VERSAbus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
APVAL*	J1: 117	ADDRESS PARITY VALID - Three-state driven signal that indicates that valid parity has been placed on the appropriate APARITY lines.
AS*	J1: 30	ADDRESS STROBE - Three-state driven signal that indicates a valid address is on the address bus.
A01*-A23*	J1: 36-58	ADDRESS bus (bits 1-23) - Three-state driven address lines that specify a memory address.
A24*-A31*	J2: 89-96	ADDRESS bus (bits 24-31) - Three-state driven optional address lines that specify an extended memory address.
BBSY*	J1: 112	BUS BUSY - Open-collector driven signal generated by the current DTB master to indicate that it is using the bus.
BCLR*	J1: 113	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.
BERR*	J1: 81	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*-BG4IN*	J1: 97,99, 101,103, 105	BUS GRANT (0-4) 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 signal indicates to this board that it may become the next bus master.
BG0OUT*-BG4OUT*	J1: 98,100, 102,104, 106	BUS GRANT (0-4) 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*-BR4*	J1: 107-111	BUS REQUEST (0-4) - Open-collector driven signals generated by Requesters. These signals indicate that a DTB master in the daisy-chain requires access to the bus.
BREL*	J1: 114	BUS RELEASE - Open-collector driven signal generated by an emergency requester to indicate to the current DTB master that the master must clear the bus within 16 data transfer cycles. It also informs the arbiter that a highest priority bus request exists.

VERSAbus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
DPARITY0*	J1: 21	DATA PARITY 0 - Three-state driven bidirectional signal, generated by either the master or the slave, which provides an even parity bit for data bits D00*-D07*.
DPARITY1*	J1: 22	DATA PARITY 1 - Three-state driven bidirectional signal, generated by either the master or the slave, which provides an even parity bit for data bits D08*-D15*.
DPARITY2*	J2: 103	DATA PARITY 2 - Three-state driven bidirectional signal, generated by either the master or the slave, which provides an even parity bit for data bus D16*-D23*.
DPARITY3*	J2: 104	DATA PARITY 3 -Three-state driven bidirectional signal, generated by either the master or the slave, which which provides an even parity bit for data bits D24*-D31*.
DPVAL*	J1: 118	DATA PARITY VALID - Three-state driven bidirectional signal that indicates during a data transfer that valid parity has been placed on the appropriate DPARITY lines.
DS0*	J1: 25	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*	J1: 26	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*	J1: 29	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*	J1: 5-20	DATA BUS (bits 0-15) - Three-state driven bidirectional data lines that provide a data path between the DTB master and slave.
D16*-D31*	J2: 105-120	DATA BUS (bits 16-31) - Three-state driven bidirectional data lines that provide an expanded data path between the DTB master and slave for the optional expanded data bus configuration (option EADB).

VERSAbus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
GND	J1: 3,4,23, 24,27,28, 31,32,61, 62,67,68, 71,72, 119,120, 123,124, 135-140	GROUND
GND	J2: 1-6	GROUND
GND	J2: 97,98, 101,102	GROUND (EXPANDED BUS OPTION ONLY)
GND (±15V)	J2: 13,14	ANALOG GROUND
[I/O PIN]	J2: 17-66	INPUT/OUTPUT PIN - I/O signal lines set aside for user peripheral interfacing applications on expanded bus backplanes.
	J2: 17-66, 71-120	INPUT/OUTPUT PIN - I/O signal lines set aside for user peripheral interfacing applications on non-expanded bus backplanes.
IRQ1*-IRQ7*	J1: 87-93	INTERRUPT REQUEST (1-7) - Open-collector driven signals, generated by an interrupter, which carry prioritized interrupt requests. Level seven is the highest priority.
LWORD*	J1: 35	LONGWORD - Three-state driven signal specifying that the cycle is a byte/word transfer (when high) or a longword transfer (when low). LONGWORD transfers are only possible between an option D32 MASTER and an option D32 SLAVE using an expanded backplane.
[RESERVED]	J1: 64,66, 73,75,76, 77,82, 115-116	RESERVED - Signal lines reserved for future VERSAbus enhancements These lines must not be used by users.
	J2: 71-87, 99,100	RESERVED - (EXPANDED BUS backplanes ONLY.)
SYSCLK	J1: 70	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.

VERSAbus Signal Identification (cont'd)

SIGNAL MNEMONIC	CONNECTOR AND PIN NUMBER	SIGNAL NAME AND DESCRIPTION
SYSFAIL*	J1: 80	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 VERSAbus.
SYSRESET*	J1: 74	SYSTEM RESET - Open-collector driven signal which, when low, will cause the system to be reset.
TEST0*- TEST1*	J1: 65 79	SYSTEM TEST - Open collector driven signals that specify the mode to be entered when the SYSRESET* line is released.
WRITE*	J1: 34	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.
+5V STDBY	J1: 133-134	+5 Vdc STANDBY - This line supplies +5 Vdc to devices requiring battery backup.
+5V	J1: 1,2, 129-132 J2: 7-10	+5 Vdc Power - Used by system logic circuits.
+12V	J1: 125-128 J2: 11,12	+12 Vdc Power - Used by system logic circuits.
+15V	J2: 69,70	+15 Vdc Power - Used by system analog circuits.
-12V	J1: 121,122 J2: 15,16	-12 Vdc Power - Used by system logic circuits.
-15V	J2: 67,68	-15 Vdc Power - Used by system analog circuits.

APPENDIX D

VERSAbus BACKPLANE EDGE CONNECTOR J1

AND

VERSAboard EDGE CONNECTOR P1

IDENTIFICATION

INTRODUCTION

This appendix identifies the VERSAbus backplane edge connector J1/P1 pin assignments. The following table lists the pin assignments by pin number order.

J1/P1 Pin Assignments

ODD PIN NUMBER (P1 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P1 SOLDER SIDE)	SIGNAL MNEMONIC
1	+5V	2	+5V
3	GND	4	GND
5	D00*	6	D01*
7	D02*	8	D03*
9	D04*	10	D05*
11	D06*	12	D07*
13	D08*	14	D09*
15	D10*	16	D11*
17	D12*	18	D13*
19	D14*	20	D15*
21	DPARITY0*	22	DPARITY1*
23	GND	24	GND
25	DS0*	26	DS1*
27	GND	28	GND
29	DTACK*	30	AS*
31	GND	32	GND
33	APARITY0*	34	WRITE*
35	LWORD*	36	A01*
37	A02*	38	A03*
39	A04*	40	A05*
41	A06*	42	A07*
43	A08*	44	A09*
45	A10*	46	A11*
47	A12*	48	A13*
49	A14*	50	A15*
51	A16*	52	A17*
53	A18*	54	A19*
55	A20*	56	A21*
57	A22*	58	A23*
59	AM4*	60	AM7*
61	GND	62	GND
63	AM3*	64	[RESERVED]
65	TEST0*	66	[RESERVED]

J1/P1 Pin Assignments (cont'd)

ODD PIN NUMBER (P1 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P1 SOLDER SIDE)	SIGNAL MNEMONIC
67	GND	68	GND
69	ACCLK	70	SYSCLK
71	GND	72	GND
73	[RESERVED]	74	SYSRESET*
75	[RESERVED]	76	[RESERVED]
77	[RESERVED]	78	ACFAIL*
79	TEST1*	80	SYSFAIL*
81	BERR*	82	[RESERVED]
83	AM0*	84	AM1*
85	AM2*	86	AM6*
87	IRQ1*	88	IRQ2*
89	IRQ3*	90	IRQ4*
91	IRQ5*	92	IRQ6*
93	IRQ7*	94	AM5*
95	ACKIN*	96	ACKOUT*
97	BGOIN*	98	BGOOUT*
99	BG1IN*	100	BG1OUT*
101	BG2IN*	102	BG2OUT*
103	BG3IN*	104	BG3OUT*
105	BG4IN*	106	BG4OUT*
107	BRO*	108	BR1*
109	BR2*	110	BR3*
111	BR4*	112	BBSY*
113	BCLR*	114	BREL*
115	[RESERVED]	116	[RESERVED]
117	APVAL*	118	DPVAL*
119	GND	120	GND
121	-12V	122	-12V
123	GND	124	GND
125	+12V	126	+12V
127	+12V	128	+12V
129	+5V	130	+5V
131	+5V	132	+5V
133	+5V STDBY	134	+5V STDBY
135	GND	136	GND
137	GND	138	GND
139	GND	140	GND

APPENDIX E

VERSAbus BACKPLANE EDGE CONNECTOR J2 AND VERSAboard EDGE CONNECTOR P2 IDENTIFICATION

INTRODUCTION

This appendix identifies the VERSAbus backplane edge connector J2 pin assignments. Table 1 lists the J2/P2 pin assignments by pin number order for the expanded bus option. Table 2 lists the J2/P2 pin assignments for the non-expanded bus option.

TABLE 1. J2/P2 Pin Assignments for the Expanded Bus Option

ODD PIN NUMBER (P2 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P2 SOLDER SIDE)	SIGNAL MNEMONIC
1	GND	2	GND
3	GND	4	GND
5	GND	6	GND
7	+5V	8	+5V
9	+5V	10	+5V
11	+12V	12	+12V
13	GND (± 15V)	14	GND (± 15V)
15	-12V	16	-12V
17	[I/O PIN]	18	[I/O PIN]
19	[I/O PIN]	20	[I/O PIN]
21	[I/O PIN]	22	[I/O PIN]
23	[I/O PIN]	24	[I/O PIN]
25	[I/O PIN]	26	[I/O PIN]
27	[I/O PIN]	28	[I/O PIN]
29	[I/O PIN]	30	[I/O PIN]
31	[I/O PIN]	32	[I/O PIN]
33	[I/O PIN]	34	[I/O PIN]
35	[I/O PIN]	36	[I/O PIN]
37	[I/O PIN]	38	[I/O PIN]
39	[I/O PIN]	40	[I/O PIN]
41	[I/O PIN]	42	[I/O PIN]
43	[I/O PIN]	44	[I/O PIN]
45	[I/O PIN]	46	[I/O PIN]
47	[I/O PIN]	48	[I/O PIN]
49	[I/O PIN]	50	[I/O PIN]
51	[I/O PIN]	52	[I/O PIN]
53	[I/O PIN]	54	[I/O PIN]
55	[I/O PIN]	56	[I/O PIN]
57	[I/O PIN]	58	[I/O PIN]
59	[I/O PIN]	60	[I/O PIN]

TABLE 1. J2/P2 Pin Assignments for the Expanded Bus Option (cont'd)

ODD PIN NUMBER (P2 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P2 SOLDER SIDE)	SIGNAL MNEMONIC
61	[I/O PIN]	62	[I/O PIN]
63	[I/O PIN]	64	[I/O PIN]
65	[I/O PIN]	66	[I/O PIN]
67	-15V	68	-15V
69	+15V	70	+15V
71	[RESERVED]	72	[RESERVED]
73	[RESERVED]	74	[RESERVED]
75	[RESERVED]	76	[RESERVED]
77	[RESERVED]	78	[RESERVED]
79	[RESERVED]	80	[RESERVED]
81	[RESERVED]	82	[RESERVED]
83	[RESERVED]	84	[RESERVED]
85	[RESERVED]	86	[RESERVED]
87	[RESERVED]	88	APARITY1*
89	A24*	90	A25*
91	A26*	92	A27*
93	A28*	94	A29*
95	A30*	96	A31*
97	GND	98	GND
99	[RESERVED]	100	[RESERVED]
101	GND	102	GND
103	DPARITY2*	104	DPARITY3*
105	D16*	106	D17*
107	D18*	108	D19*
109	D20*	110	D21*
111	D22*	112	D23*
113	D24*	114	D25*
115	D26*	116	D27*
117	D28*	118	D29*
119	D30*	120	D31*

NOTE: Pins 17 through 66 are not bussed together by the backplane.

TABLE 2. J2/P2 Pin Assignments for the Non-Expanded Bus Option

ODD PIN NUMBER (P2 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P2 SOLDER SIDE)	SIGNAL MNEMONIC
1	GND	2	GND
3	GND	4	GND
5	GND	6	GND
7	+5V	8	+5V
9	+5V	10	+5V
11	+12V	12	+12V
13	GND ($\pm$ 15V)	14	GND ($\pm$ 15V)
15	-12V	16	-12V
17	[I/O PIN]	18	[I/O PIN]
19	[I/O PIN]	20	[I/O PIN]
21	[I/O PIN]	22	[I/O PIN]
23	[I/O PIN]	24	[I/O PIN]
25	[I/O PIN]	26	[I/O PIN]
27	[I/O PIN]	28	[I/O PIN]
29	[I/O PIN]	30	[I/O PIN]
31	[I/O PIN]	32	[I/O PIN]
33	[I/O PIN]	34	[I/O PIN]
35	[I/O PIN]	36	[I/O PIN]
37	[I/O PIN]	38	[I/O PIN]
39	[I/O PIN]	40	[I/O PIN]
41	[I/O PIN]	42	[I/O PIN]
43	[I/O PIN]	44	[I/O PIN]
45	[I/O PIN]	46	[I/O PIN]
47	[I/O PIN]	48	[I/O PIN]
49	[I/O PIN]	50	[I/O PIN]
51	[I/O PIN]	52	[I/O PIN]
53	[I/O PIN]	54	[I/O PIN]
55	[I/O PIN]	56	[I/O PIN]
57	[I/O PIN]	58	[I/O PIN]
59	[I/O PIN]	60	[I/O PIN]
61	[I/O PIN]	62	[I/O PIN]
63	[I/O PIN]	64	[I/O PIN]
65	[I/O PIN]	66	[I/O PIN]
67	-15V	68	-15V
69	+15V	70	+15V
71	[I/O PIN]	72	[I/O PIN]
73	[I/O PIN]	74	[I/O PIN]
75	[I/O PIN]	76	[I/O PIN]
77	[I/O PIN]	78	[I/O PIN]
79	[I/O PIN]	80	[I/O PIN]
81	[I/O PIN]	82	[I/O PIN]
83	[I/O PIN]	84	[I/O PIN]
85	[I/O PIN]	86	[I/O PIN]
87	[I/O PIN]	88	[I/O PIN]
89	[I/O PIN]	90	[I/O PIN]
91	[I/O PIN]	92	[I/O PIN]
93	[I/O PIN]	94	[I/O PIN]
95	[I/O PIN]	96	[I/O PIN]

TABLE 2. J2/P2 Pin Assignments for the Non-Expanded Bus Option (cont'd)

ODD PIN NUMBER (P2 COMPONENT SIDE)	SIGNAL MNEMONIC	EVEN PIN NUMBER (P2 SOLDER SIDE)	SIGNAL MNEMONIC
97	[I/O PIN]	98	[I/O PIN]
99	[I/O PIN]	100	[I/O PIN]
101	[I/O PIN]	102	[I/O PIN]
103	[I/O PIN]	104	[I/O PIN]
105	[I/O PIN]	106	[I/O PIN]
107	[I/O PIN]	108	[I/O PIN]
109	[I/O PIN]	110	[I/O PIN]
111	[I/O PIN]	112	[I/O PIN]
113	[I/O PIN]	114	[I/O PIN]
115	[I/O PIN]	116	[I/O PIN]
117	[I/O PIN]	118	[I/O PIN]
119	[I/O PIN]	120	[I/O PIN]

NOTE: Pins 17 through 66 and pins 71 through 120 are not bussed together by the backplane.

APPENDIX F

DC SIGNAL SPECIFICATION

This appendix provides a summary showing which signal lines on VERSAbus 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 TYPE	DRIVER MODULE	RECEIVER MODULE	TERMINATION NETWORK
A01*-A31* (31 lines)	ADDRESS BUS	THREE	MASTERS, INTERRUPT HANDLERS	SLAVES, INTERRUPTERS	YES
AM0*-AM7* (8 lines)	ADDRESS MODIFIER				
AS*	ADDRESS STROBE				
LWORD*	LONGWORD				
WRITE*	WRITE				
DS0*-DS1* (2 lines)	DATA STROBES				
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
DPARITY0*-DPARITY3* (4 lines)	DATA PARITY	THREE	DP MASTERS DP SLAVES	DP SLAVES DP MASTERS	YES
DPVAL*	DATA PARITY VALID	THREE	DP MASTERS DP SLAVES	DP MASTERS DP SLAVES	YES
APARITY0*-APARITY1* (2 lines)	ADDRESS PARITY	THREE	AP MASTERS	AP SLAVES	YES
APVAL*	ADDRESS PARITY VALID	THREE	AP MASTERS	AP SLAVES	YES

BUS DRIVER AND RECEIVER SUMMARY (cont'd)

SIGNAL MNEMONIC	SIGNAL NAME	DRIVER TYPE	DRIVER MODULE	RECEIVER MODULE	TERMINATION NETWORK
BR0*-BR4* (5 lines)	BUS REQUEST	OC	REQUESTERS	ARBITER	YES
BG0IN*-BG4IN* (5 lines)	BUS GRANT	TP LC	ARBITER	REQUESTERS	NO
BG0OUT*-BG4OUT* (5 lines)			REQUESTERS	(NOT APPLICABLE)	NO
BBSY*	BUS BUSY	OC	EMERGENCY REQUESTER	ARBITER	YES
BCLR*	BUS CLEAR	TP HC	ARBITER	MASTERS	YES
BREL*	BUS RELEASE	TP HC	EMERGENCY REQUESTER	MASTERS	YES
IRQ1*-IRQ7* (7 lines)	INTERRUPT REQUEST	OC	INTERRUPTERS	INTERRUPT HANDLERS	YES
ACKIN*/ACKOUT*	ACKNOWLEDGE DAISY CHAIN	TP LC	INTERRUPT HANDLERS, INTERRUPTERS	INTERRUPTERS INTERRUPTERS	NO
SYSRESET*	SYSTEM RESET	OC	POWER MONITOR, MANUAL SWITCH	ANY ANY	YES
ACFAIL*	AC FAILURE	OC	POWER MONITOR	EMERGENCY REQUESTER	YES
ACCLK	AC CLOCK	OC	POWER MONITOR	ANY	YES
SYSCLK	SYSTEM CLOCK	TP HC	CLOCK DRIVER	ANY	YES
SYSFAIL*	SYSTEM FAIL	OC	MASTERS, SLAVES	ANY ANY	YES
TEST0*-TEST1* (2 lines)	SYSTEM TEST	OC	POWER MONITOR MANUAL SWITCH	ANY ANY	YES

MOTOROLA *Semiconductor Products Inc.*

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

13142-7 PRINTED IN USA 8-81 IMPERIAL LITHO 97951 5000