VAXBI

DESIGNER'S NOTEBOOK

EK-VBIDS-RM-001

VAXBI DESIGNER'S NOTEBOOK

EK-VBIDS-RM-001

Digital Equipment Corporation • Maynard, Massachusetts

First Printing, January 1986

The manuscript for this book was created using generic coding and, via a translation program, was automatically typeset. Book production was done by Educational Services Development and Publishing in Bedford, MA.

DEC	MASSBUS	UNIBUS
DECmate	PDP	ULTRIX
DECnet	P/OS	VAX
DECUS	Professional	VAXBI
DECwriter	Rainbow	VAXELN
DIBOL	RSTS	VMS
digital™	RSX	VT
	RT	Work Processor

Contents

Preface

The purpose of this notebook is to provide a focus for those who are involved with designing options for the VAXBI bus. This book attempts to put the *VAXBI System Reference Manual* into perspective, explaining which portions of the VAXBI specification option designers need to be most concerned about. Some portions deal with requirements that are implemented by DIGITAL-supplied hardware. The option designer need only use the specified hardware to comply with many of the requirements that are detailed in the comprehensive specification.

With the proper perspective, designers can understand more quickly what they need to do. To help them understand *how* to go about their task, we will provide design examples. In addition, as design tools become available we will include commentary on their use.

Intended Audience

This notebook is primarily for engineers who design options for VAXBI systems. System architects and others who want to understand DIGITAL's design philosophy for the bus will also find this information useful. Chapter 3, which is a guide to module layout, should be read and studied by module layout designers; this chapter also includes information pertinent to manufacturing.

Structure of This Manual

Chapter 1, Introduction to VAXBI Option Design, provides an overview of the VAXBI bus and of the documentation from an option designer's point of view. The chapter also poses design issues and gives hints for designing options.

Chapter 2, The Instrument Control Adapter, is an example of a VAXBI option design. The design requirements and the resulting design decisions are described. The option is a master-port-only design.

Chapter 3, VAXBI Module Layout Guide, serves as a guide for engineers and module layout designers as they design options and build VAXBI modules. It points out problem areas and reports some of DIGITAL's experiences. References are made to the appropriate module control drawings.

Appendix A, VAXBI BIIC Simulation: Physical Chip Modeling, gives requirements to permit physical chip modeling of the BIIC for simulation of a VAXBI option.

Appendix B, VAXBI Base Layout Package, gives information on the documentation and databases in the VAXBI Base Layout Package.

Related Documentation

- *VAXBI System Reference Manual* – The specification for the VAXBI bus and the VAXBI primary interface (the BIIC)
- T1999 – Module Control Drawings for the standard VAXBI module
- T1996 – Module Control Drawings for the VAXBI expansion module
- ELEN 626 – Mechanical outline drawing for the standard VAXBI module
- ELEN 633 – VAXBI module layup specification

The following document provides information about VAX architecture.

- *VAX-11 Architecture Reference Manual*

Chapter 1

Introduction to VAXBI Option Design

by VAXBI Development Group

This chapter provides an overview of the VAXBI bus and of the documentation from an option designer's point of view. The chapter also poses design issues and lists "tips and topics" useful to designers.

Contents

Overview of VAXBI Option Design

Now that you are the holder of a VAXBI™ license, you have before you a number of documents, a database, and a job to do. It's your job to design and build a VAXBI option. Where do you start? You've read the technical summary of the VAXBI bus, and you've paged through that hefty document, the *VAXBI System Reference Manual.* Perhaps you are suffering from "spec shock," an affliction that tends to paralyze.

In this chapter we want to explain why the VAXBI specification is such a lengthy document. We also want to make clear what you, as an option designer, need to be concerned about. Because DIGITAL provides the VAXBI primary interface, your job becomes simpler. Your primary task is to implement the user interface portion of the option—the logic for the functions required of your particular option. The secondary task is to interface to DIGITAL's "VAXBI Corner." Depending on design choices, this is usually easier than interfacing to many of today's industry buses.

We begin by describing DIGITAL's design philosophy for the new bus. If you know the goals set for the bus, you can see more clearly the reasons for the requirements.

DIGITAL's Bus Design Philosophy

After defining its interconnect strategy, DIGITAL set up the VAXBI Development Group to carry out one part of the strategy. Decisions made about the new bus to be designed and built by this group were guided by the following:

- The bus should offer high bandwidth.
- The bus should support multiprocessing and caching.
- The bus should be fully specified.
- The hardware should be built and tested to the full extent of the specified requirements.
- As far as possible, the bus protocol should preserve data integrity in single-bit failure conditions, and when bus errors occur, they should be easy to diagnose.
- It should be a general purpose bus designed from a system perspective.
- The board area required for the standard bus interface should be minimized.
- The bus should be reliable and easy to maintain.

By definition, a bus provides a standard method of interconnecting devices within a computer system. A device that functions properly in a system and does not disrupt the operation of other devices within the system is said to be *compatible*. Furthermore, devices should not depend upon a particular system configuration for compatibility. Any device for a particular type of system should function properly with any combination of devices for that system.

DIGITAL has been successful in maintaining software compatibility among the various members of the VAX computer family. The goal of the VAXBI Development Group was to attain this same standard of compatibility at the bus level. Just as the compatibility of VAX systems is supported by a very extensive and detailed *VAX-11 Architecture Reference Manual*, the compatibility of VAXBI options is supported by the *VAXBI System Reference Manual.*

"Compatibility," when used in the context of bus operation or of devices, requires a discussion of the various *levels* of compatibility. The VAXBI SRM includes information on all these levels:

- Mechanical, power, environmental
- Electrical
- Logical
- Software/architectural

We have provided rules to help ensure compatibility among all VAXBI options. In practical terms, it is not possible to achieve 100 percent compatibility, for specifications would have to define every aspect of the bus from the electrical signals to software behavior. However, we believe that we have moved far beyond previous efforts at providing a bus that offers compatibility.

The following pages explain what efforts we took to ensure that compatibility characterizes VAXBI systems.

* VAXBI is a trademark of Digital Equipment Corporation.

- We first discuss **What DIGITAL has defined for you**. The specification for the VAXBI bus addresses various levels of compatibility. We explain the intent of the specification and give some examples of what has been defined.
- We then explain **What DIGITAL has done for you**. A large portion of compatibility is assured by DIGITAL-supplied hardware and artwork. The bus hardware was tested to verify that the VAXBI bus operates as described in the specification.
- We then present **What remains for you to do**. The areas of compatibility described here are your responsibility as an option designer.

In short, we defined and designed the bus, we built it, we tested it. But designers need more than the VAXBI SRM and the standard hardware. They need to be aware of the system operation so that they can uphold the compatibility standard.

What DIGITAL Has Defined for You

The *VAXBI System Reference Manual* defines aspects of the VAXBI bus that have been specified to ensure that it operates as a bus should; that is, that all aspects provide for the successful communication among devices that may be connected to the bus. As shown in Figure 1-1, these aspects include software/architectural requirements, logical bus protocol, electrical characteristics, mechanical components, and power and environmental constraints.

Other specifications for bus systems, including those defined by DIGITAL, are considerably less precise and less inclusive than the VAXBI specification. To understand why the VAXBI SRM is such a lengthy document, one must realize the breadth of information that a bus specification ought to cover. A bus is more than the electrical conductors etched on the backplane. That is, a bus is not only the medium of communication, but it also encompasses the format of communication. A description of a bus needs to include much more than the physical components that traditionally define a bus.

Based on much experience with buses, DIGITAL has attempted to fully specify a set of requirements that, if followed, will provide complete compatibility at the mechanical, electrical, and logical levels. Areas that have been troublesome to bus users in the past because of lack of definition have been specified for the VAXBI. For example, on some buses device types are associated with specific address locations. However, on the VAXBI bus, control and status registers for a particular device are located in a predefined section of I/O space. Sixteen sections are defined; one section is associated with each node ID. To see what devices are on a particular VAXBI, the software can read each node's device type, which is in a BIIC CSR. Similarly, device interrupt vectors are determined by node ID and are independent of device type.

For reasons of compatibility, we have defined three "classes" of nodes (processors, memories, and adapters), and for each class we have defined required sets of transactions. The intent of these requirements is to ensure that no node will require of another node capabilities that it does not have. For instance, a processor that issues an octaword-length interlock read to memory will not find that the memory supports only longword-length interlock reads. Similarly, an adapter whose status can be found only by reading a particular register with a longword read won't find it can't be used with a processor because the processor cannot issue longword reads. A VAXBI node can belong to more than one of the three classes. To be compatible, such a multifunction node must meet the requirements set for each class to which it belongs.

Some other major areas were defined that support the design goals for the bus:

- The kinds of transactions that provide for caching
- The VAXBI Corner of a module

The VAXBI SRM attempts to precisely define the bus—from its physical components to its operation. The major categories of requirements are described below; included with the requirements are references to the appropriate sections of the VAXBI SRM.

As you read the requirements, keep in mind that you as an option designer do not need to do much to comply with the bulk of these requirements. Compliance with the requirements at the first three levels is assured by using the standard VAXBI hardware, which is described on page 1-7. Your primary responsibility in ensuring compatibility is to comply with the requirements at the software/architectural level. This topic is discussed on page 1-9.

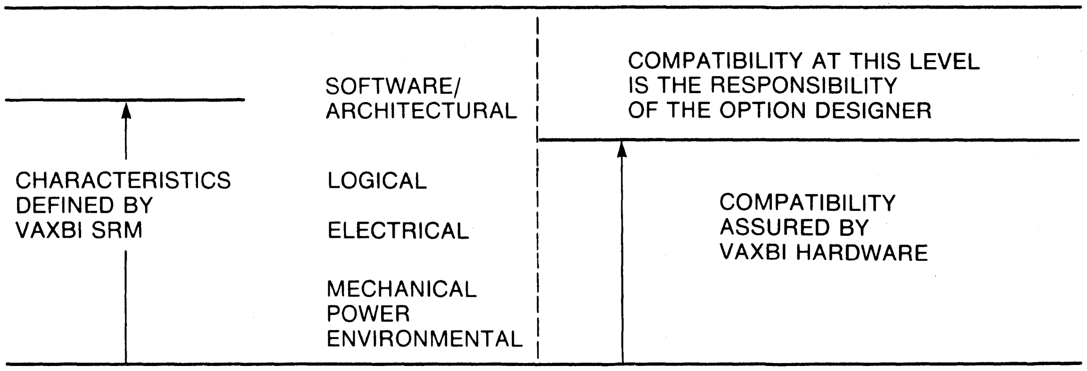

SOFTWARE/
ARCHITECTURAL
COMPATIBILITY AT THIS LEVEL IS THE RESPONSIBILITY OF THE OPTION DESIGNER
CHARACTERISTICS DEFINED BY VAXBI SRM
LOGICAL
ELECTRICAL
COMPATIBILITY ASSURED BY VAXBI HARDWARE
MECHANICAL
POWER
ENVIRONMENTAL

FIGURE 1-1 *Achieving Compatibility in a Computer System*
Typically, bus protocol logic and electrical requirements are specified for buses. The VAXBI SRM defines *the bus from the most basic mechanical level on through the electrical and logical levels and even into the architectural level. We* assure *compatibility up through the logical level.*

Mechanical, Power, and Environmental Requirements (Chapter 13 and the Module Control Drawings)

Mechanical requirements include all characteristics of a node that assure physical compatibility with VAXBI systems. Such characteristics include module size (length, width, and thickness), module layup, and maximum component height. The mechanical requirements are basic bus requirements, since if a module does not physically fit into the system, any other compatibility requirements are of little concern. The VAXBI SRM also specifies the location of certain components that are common to all nodes (components in the VAXBI Corner and the self-test status LEDs).

Power compatibility requires that a node use only the supported voltages (Section 13.1.6.1) and abide by all current (Section 13.1.6.2) and power dissipation (Section 13.1.9.1) limits.

Environmental compatibility requires that nodes be designed to operate within the environmental range specified for the VAXBI bus (Section 13.7).

Electrical Requirements (Chapter 12)

Electrical requirements assure that bus signals are driven and received in a compatible manner. Specified characteristics include signal voltage levels, receiver input characteristics, backplane etch characteristics, and signal timing.

With the exception of a few lines driven by user logic, all electrical compatibility is provided by the BIIC, the clock driver, the clock receiver, the VAXBI Corner layout and layup, and the specified mechanical components.

Logical Requirements (Chapters 3, 4, 5)

Logical standards define the basic bus protocol including at the lowest level the individual signal definitions. At slightly higher protocol levels, the arbitration and transaction formats are defined.

The BIIC assures compliance with VAXBI logical requirements.

Software/Architectural Requirements (Chapters 2, 3, 5–11, Ap. Notes)

At the top level of compatibility are those specifications that define operations—not physical or electrical entities. The VAXBI SRM gives some architectural requirements that, if followed, allow various bus components to communicate and to allow a system—whatever its configuration—to operate.

The primary goal of architectural rules is to assure that nodes are sufficiently compatible at the transaction level to communicate with each other. For example, the VAXBI SRM (Section 8.2) gives the minimum set of transactions that a node must be able to generate and respond to (MIS, MRS sets). To improve performance, you may choose to implement more than the minimum requirements, but defining a minimum assures that option designers can depend on a basic set of functions common to all nodes.

Other examples of software/architectural requirements include:

- The interpretation of address lines (Section 5.3.1)
- Self-test operation (Chapter 11)
- The initialization process and sequence of events (Chapter 6)

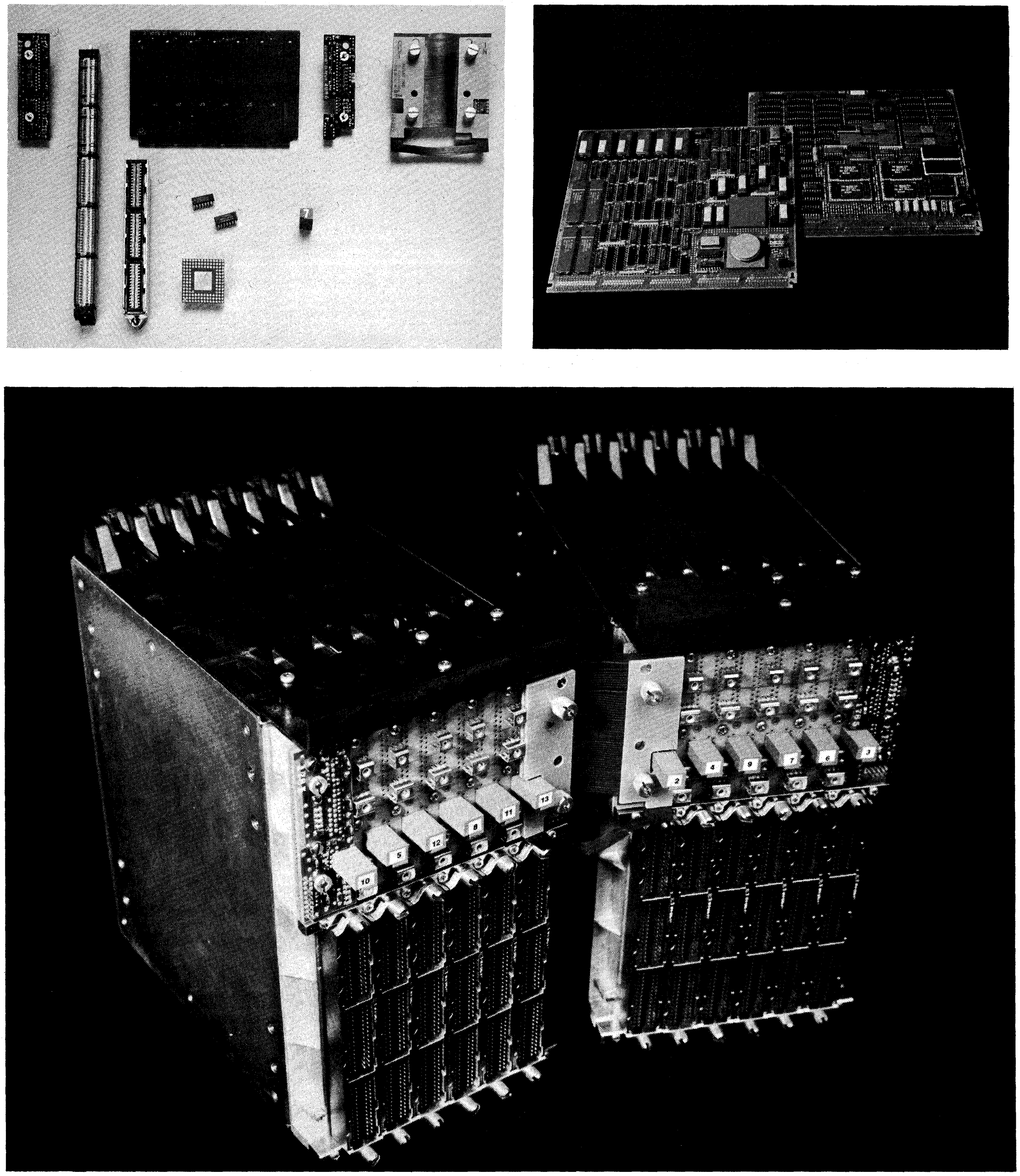


FIGURE 1-2 *Standard VAXBI Hardware*

The first photo shows some VAXBI components that guarantee the correct operation of the VAXBI bus: (left to right, top) terminator, connector, backplane, terminator, backplane extension cable (bottom) I/O header, clock driver, clock receiver, node ID plug, and BIIC. The second photo shows a two-module option: a standard VAXBI module with the required components in the Corner and an expansion module. The third photo shows two VAXBI cages joined by the flexible backplane extender.

The VAXBI SRM cannot completely specify all node operations, because system operation depends upon the software implementation. For example, the VAXBI SRM does not specify architectural guidelines for software error handling.

What DIGITAL Has Done for You

Table 1-1 lists the standard VAXBI hardware and indicates the aspects of compatibility that each piece helps to assure. As can be seen from this list, the VAXBI standardization has gone beyond the typical physical components to include the silicon implementation of the bus control logic.

Built Standard VAXBI Hardware

The majority of the compatibility requirements for a node are satisfied through the use of standard VAXBI hardware (see Figure 1-2). This hardware was specifically designed for the VAXBI bus, and its proper operation in all system configurations and applications has been verified through extensive worst-case testing.

To assure compatibility at the mechanical level, we specified, built, and tested modules, the backplane, and all pieces of the card cage. The area of a VAXBI module that interfaces to the bus, known as the VAXBI Corner, has been fully specified, including the module layup, the components in the Corner, and their location.

Table 1-1 Function of VAXBI Standard Hardware

	Mechanical	Electrical	Logical
Card Cage	X		
Connector	X	X	
Module	X	X	
Backplane	X	X	
Backplane Extension Cables	X	X	
Terminators	X	X	
VAXBI Corner		X	
Clock Driver		X	
Clock Receiver		X	
BIIC		X	X

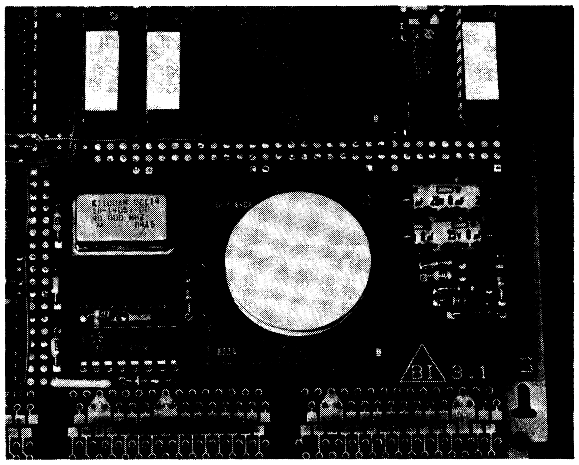


FIGURE 1-3 *VAXBI Corner Layout*

The VAXBI Corner on this module includes a BIIC (under the round heat sink), a clock driver, clock receiver, and oscillator. Only one module in a system requires a clock driver.

Compatibility at the electrical level is assured by the standard bus interface on the module side and on the backplane side by the backplane subassembly which consists of the backplane and the signal terminators. The placement for custom-designed VLSI devices in the VAXBI Corner has been defined for optimal signal integrity. The VAXBI module construction (module layup) has been specified to guarantee impedance characteristics. The module layup includes signal layer spacings, line widths, and copper thicknesses.

Each Corner has a BIIC, which is the primary interface to the VAXBI bus, and a clock receiver. One node in a system also has a clock driver and oscillator in the VAXBI Corner (see Figure 1-3). The electrical characteristics of the BIIC's transceivers are optimized to drive and receive signals on the VAXBI bus. Since all nodes interface to the VAXBI with the BIIC, electrical compatibility is assured.

In addition to its contribution to providing electrical compatibility, the BIIC assures compatibility at the logical level. All aspects of transaction protocol are performed by the BIIC. These operations include arbitration, the format for transactions, and error checking and logging.

FIGURE 1-4 *A VAXBI System*

The VAXBI bus is more than a backplane and related protocol. The bus has been designed and specified for system operation. One VAXBI system can have from one to six VAXBI card cages, each of which can hold six modules. A system can have up to 16 nodes; that is, 16 modules with a VAXBI Corner which interfaces directly to the bus. When a node requires more than one module, the node consists of a VAXBI module with a VAXBI Corner and one or more expansion modules, which do not have VAXBI Corners.

Tested the VAXBI Bus

The VAXBI Development Group used sophisticated tools to design and test the VAXBI bus. During the design process, software simulation tools (mechanical modeling tools and circuit and logic simulators) were used to verify the proper operation of the bus under worst-case conditions. Tests of the standard hardware verified the accuracy of the software results.

The VAXBI group built a special module that could automatically generate and test transactions, data patterns, and node IDs under all system configurations. This special module was known as TRAGEN (for transaction generator). The goal of the TRAGEN design was to verify the operation of the VAXBI bus and the BIIC in a simulated system environment. The testing was done with 16 TRAGEN nodes, all of which could perform master port transactions, generate interrupts, and respond to transactions as a slave. Over 40 trillion randomly generated transactions were run on a fully configured 6-cage, 36-module system. The testing validated the specifications (see Figure 1-4).

The bus was tested to determine the margins within which the bus would operate. Data was recorded from the following types of margin tests:

- **Clock Margin**. The BI TIME and PHASE frequencies were increased until an error occurred.
- **Voltage Test**. Transactions were run at lower and lower voltages until an error occurred.
- **Capacitive Load Margin**. The capacitance was increased until an error was detected.

All margin tests were run using a TRAGEN setup that generated transactions between nodes at opposite ends of the bus, so that timing was also worst case. The margin tests confirmed that the standard hardware had margin on all bus requirements.

Other tests verified that the bus would operate under the following conditions:

- Operating adjacent cages at the specified power supply extremes.
- At the specified temperature extremes.
- Varying the node IDs and the physical slots.
- Varying the BIICs and the physical slots.

Signal integrity of the bus signals was tested at the component, module, and system level. The areas of concern were noise (power and signal), crosstalk, and signal distortion due to reflections and loading.

* * *

A high-performance bus needs to be fully specified if nodes are to be compatible in any system configuration. We've given examples to show that the VAXBI bus specification addresses more areas than the typical bus specification covers. For the reasons presented in the preceding pages, the VAXBI bus can provide a greater degree of module and system compatibility than any existing bus can offer.

The next section attempts to orient you to the VAXBI SRM and to demonstrate what you need to be concerned about in designing and building an option for the VAXBI bus.

What Remains for You to Do

When designing to the VAXBI bus, the option designer's challenge is in the implementation of an application, not in doing the bus interface (see Figure 1-5). The fully specified VAXBI Corner has been standardized so that any VAXBI module can be plugged into a VAXBI system and the bus will operate as specified. Everything that comprises the VAXBI Corner contributes to the successful operation of the bus: the ten board layers that separate the power and ground planes from the signal layers, the layout for the Corner, and, of course, the BIIC. Many of the VAXBI requirements that are in Part One of the VAXBI SRM are implemented in the BIIC. You do not need to be concerned about signal integrity, bus capacitance loads, and crosstalk. If you use the BIIC, these requirements are met.

Instead of doing a complicated bus interface then, you interface to the BCI signals in the

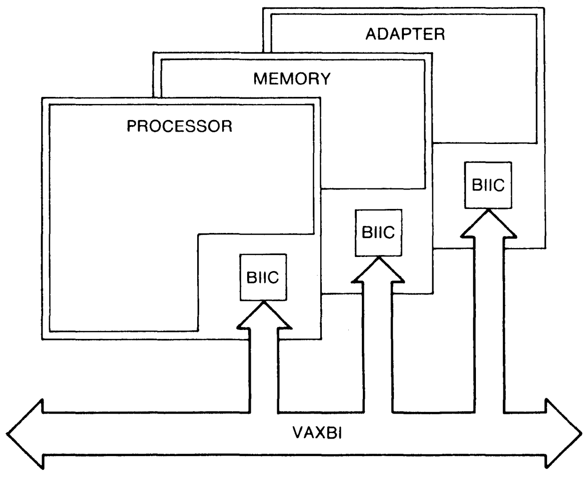


FIGURE 1-5 *VAXBI Primary Interface*
The BIIC serves as the primary interface for all VAXBI nodes regardless of function.

VAXBI Corner boundary. Chapter 15 in the VAXBI SRM describes the BIIC lines that connect to the user interface logic. In addition to these lines, 34 other lines connect the VAXBI Corner boundary and the user interface logic. These lines are mentioned in Section 13.2.2, and the designer must drive or receive some of these signal lines.

Your responsibility for compatibility is primarily at the software/architectural level. You must design your option so that it meets the VAXBI architectural requirements. However, even if your option meets all requirements in the VAXBI SRM, you can introduce incompatibilities at the system level. For example, restrictions can be placed on the software if in your adapter you require that all data buffers be on longword-aligned boundaries. VAX system architecture does not require that data buffers align on longword boundaries. From the perspective of system operation you should take care that your option does not cause such problems.

Understand VAXBI Option Design Philosophy

Before beginning a design, you need to plan the relationship of the hardware, firmware, and software. In evaluating how an option is to perform within a system, you function as a system architect. In this role, you must consider how the option is to perform in a system environment. By understanding how partners in a transaction interact, you can optimize system performance.

For example, it may seem reasonable for an adapter to issue a RETRY rather than STALL in response to a command to avoid long bus latencies for other nodes. In doing so, however, an adapter may cause a given node to receive repeated RETRY responses for a very long time, and this could degrade overall system performance more severely than if STALL were used instead.

Optimizing the Use of the BIIC

Working with the VAXBI bus requires a rethinking of how to do an adapter design. The register transfer model (programmed I/O) was acceptable for other buses, but in a VAXBI system a DMA type of adapter is more efficient. With the programmed I/O approach, a VAXBI adapter would need a slave port, which would add complexity. The BIIC performs a minimum set of slave functions, which makes possible a master-port-only interface to the BIIC. An adapter can use the BIIC General Purpose Registers (GPRs) to communicate with other nodes. A processor can deposit the address of a GPR in memory and can then communicate with the adapter through this area, so that a slave port interface may not be needed in an adapter design. Before deciding that a particular adapter requires both a master and slave port interface, you should assess the complexity that a slave port interface adds to the design. Figure 1-6 shows a block diagram of a VAXBI node.

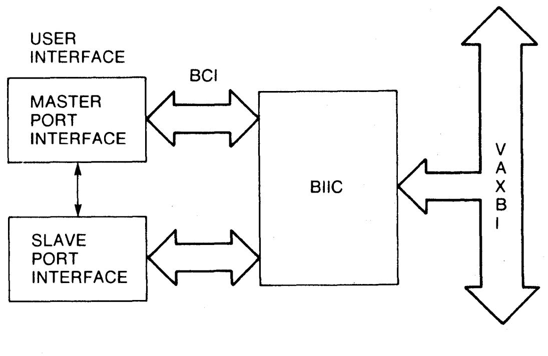


FIGURE 1-6 *Block Diagram of a VAXBI Node*
A VAXBI node can function either as slave or master in a VAXBI transaction. Because some slave functions are provided by the BIIC, the user interface may not require a slave port interface.

In a VAXBI system it is often advantageous for every node to be a master. In terms of system performance, it is better for an adapter to initiate a transaction when it is ready than to always be ready to accept CPU commands If an adapter cannot immediately execute a command, the bus could be tied up until the adapter can respond. The master port approach of VAXBI design is supported by logic provided in the BIIC.

Maximizing System Performance

There are two aspects to maximizing system performance: minimizing the use of VAXBI bus bandwidth and minimizing the time required to perform the node operation. If performance is important, it is advisable to use a double-buffer scheme in the design. When the actions of hardware, firmware, and software proceed concurrently, the time required for some operations can be dramatically reduced.

Even though an adapter conforms to the VAXBI specs, its design can adversely affect system performance. For example, Figure 1-7 shows a simple master port adapter design, in which the CPU puts command packets in a queue in main memory for the adapter to process. The adapter fetches a command packet and performs the required processing and data transfers to or from main memory. When the adapter finishes with a command, the adapter returns status in the command packet and interrupts the CPU to inform it that the command has been completed. The adapter then looks for the next job on the queue and processes that packet. If there are no more commands on the queue, the adapter must wait until the CPU issues a new command by filling a command packet.

A designer has two choices at this point: either the adapter continually polls the queue in memory, or the adapter "sleeps" until the CPU wakens it. The latter option is easily implemented using features of the BIIC; for example, you can use the event (EV) codes to wake up the adapter. Continually polling the queue could tie up a large amount of bus bandwidth and memory bandwidth and still be less responsive to a command from the CPU. Note that this design choice has an impact on the hardware, software, and firmware design of the adapter.

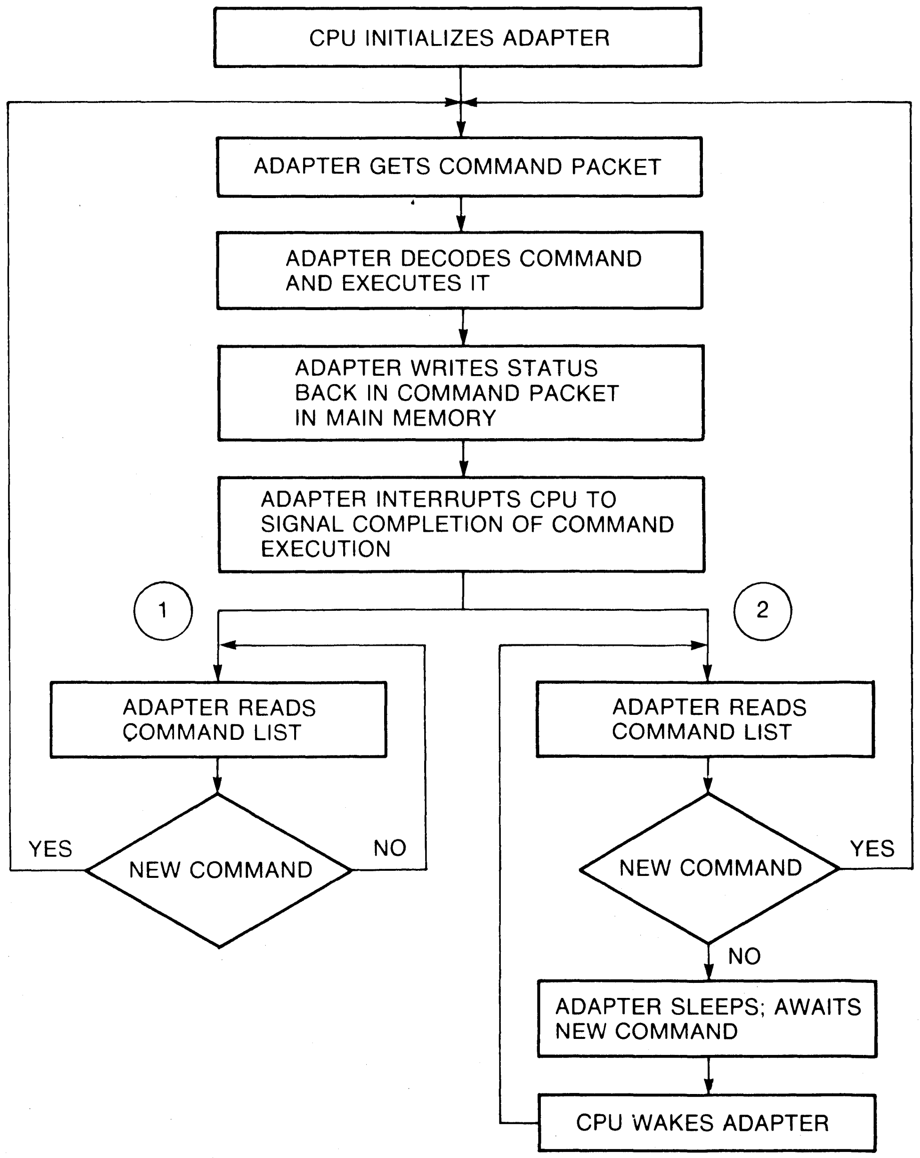

CPU INITIALIZES ADAPTER
ADAPTER GETS COMMAND PACKET
ADAPTER DECODES COMMAND AND EXECUTES IT
ADAPTER WRITES STATUS BACK IN COMMAND PACKET IN MAIN MEMORY
ADAPTER INTERRUPTS CPU TO SIGNAL COMPLETION OF COMMAND EXECUTION
1
2
ADAPTER READS COMMAND LIST
ADAPTER READS COMMAND LIST
YES
NEW COMMAND
NO
NEW COMMAND
YES
NO
ADAPTER SLEEPS; AWAITS NEW COMMAND
CPU WAKES ADAPTER

FIGURE 1-7 *Master Port Design with Two Options*
How a design is implemented can have great impact on system performance. In option 2 after a device finishes a task, it stops and is restarted by the CPU. This approach uses less VAXBI bandwidth than if the device continually polls main memory, which requires use of the bus, to see if its services are required (option 1).

Take a System-Level Approach

The design example above makes it clear that design decisions dramatically affect the hardware, firmware, and software architecture. None of these areas can be considered in isolation. In a system-level approach, you need to design the option architecture (see Figure 1-8).

With such an approach, the hardware specification (derived from the VAXBI SRM), the software specification, and the design process specification are formulated together.

Various design constraints determine how the option will be implemented. If a goal is to keep the design on one board, your design may require the use of custom or semicustom logic. Custom chips and gate arrays usually require the use of simulation tools. You will also need computer-aided design tools for the printed cir-

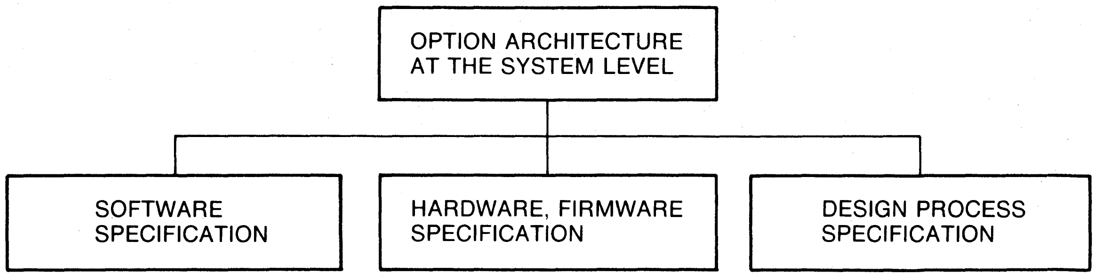

OPTION ARCHITECTURE AT THE SYSTEM LEVEL
SOFTWARE SPECIFICATION
HARDWARE, FIRMWARE SPECIFICATION
DESIGN PROCESS SPECIFICATION

FIGURE 1-8 *System-Level Approach to Option Design*

Decision-making in a VAXBI system requires knowledge of more than hardware. The design constraints determine how an option will be implemented.

cuit boards, the silicon, and perhaps for testing. Figure 1-9 shows steps that may be required in the design process.

Use Design Tools

Tools are being developed to assist in simulating node operation, in doing module layout, and in debugging. As Application Notes on the design tools become available, they will be added to this notebook.

Chapter 3 of the notebook, Module Layout Guide, comments on the Module Control Drawings and gives suggestions for using the associated databases.

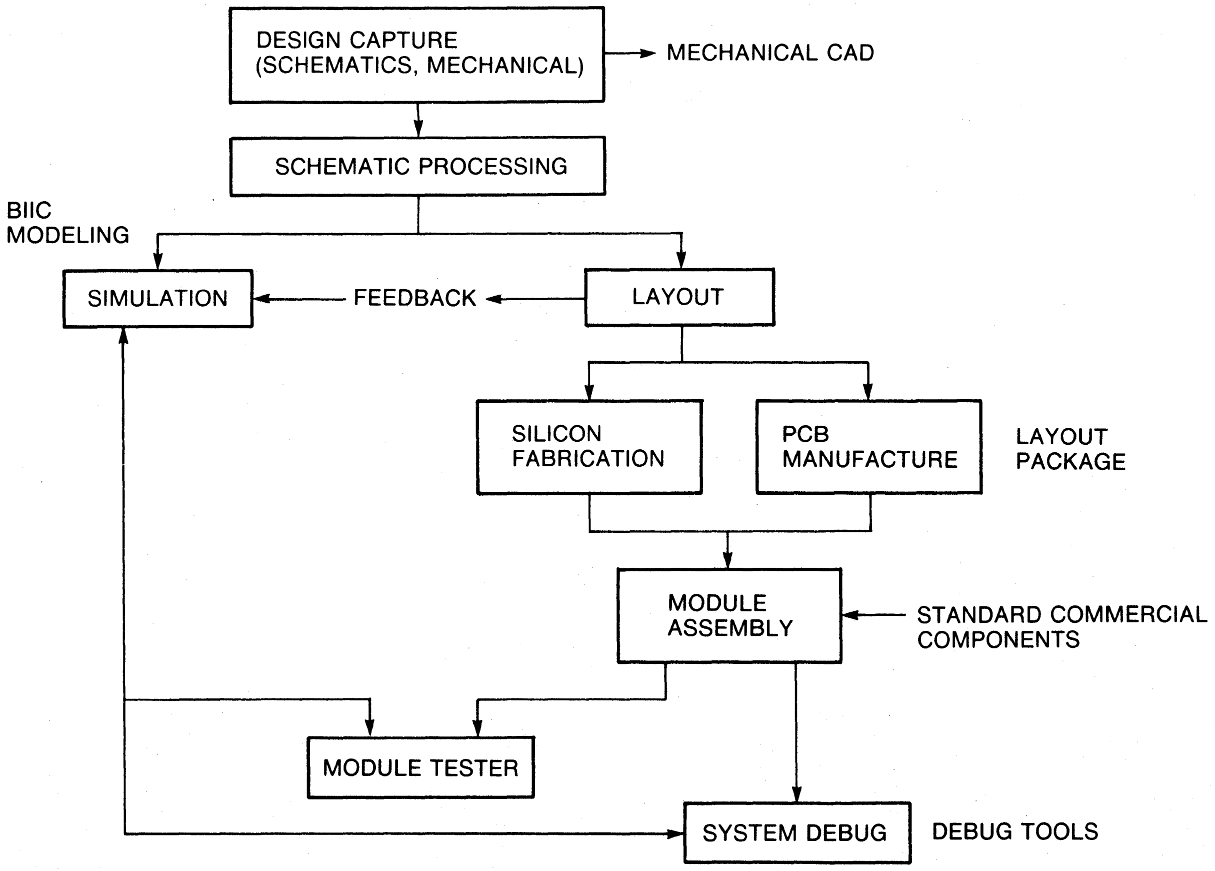


FIGURE 1-9 *Design Process*

The design and implementation of modules with custom VLSI components may require a sophisticated design process that is dependent on computer-aided design tools.

Design Analysis

As you begin to plan your design for the VAXBI bus, you must make some basic system-level decisions based on tradeoffs of complexity, performance, and cost. The *VAXBI System Reference Manual* provides the framework for your decisions.

- Your option belongs to which node class or classes (processor, memory, or adapter) as defined by the VAXBI SRM?
- What type of design is needed—master, slave, or master/slave?

  Use of bus bandwidth may be more efficient if the node has only a master port; the bandwidth is limited only by the memory being accessed.
- Which transactions will you support?

  Using only longword transfers reduces product complexity and cost; however, the maximum bandwidth on the VAXBI will not be attained.

  Most I/O adapters use octaword transactions as much as possible to attain the maximum VAXBI bandwidth. However, note that the data buffers in memory space might not be octaword aligned, but octaword transactions must be octaword aligned.
- How will the software interface function?

  Consider the tradeoffs between complex software drivers and hardware complexity.

The following list directs you to sections of the VAXBI SRM that are relevant to decisions that an adapter designer must make.

- What does the VAXBI SRM require of adapter nodes? See node class definitions in Section 8.1 and node class requirements in Section 8.2. Each node class has required sets of transactions that it must issue and to which it must respond (MIS, MRS).

  As a class, adapters must respond to the STOP command (see Section 5.5).

  Consider what transactions your adapter will generate to communicate with memory.
- Application Note 1 can help you decide what type of adapter you should design. Three types of adapters are described: programmed I/O, DMA, and bus adapters. Consider the architectural tradeoffs.
- What requirements in I/O space must an adapter meet? (See Section 8.2.3.)
- Use of address space is described in Chapter 2. Will the option use window space? Can the BIIC General Purpose Registers be used instead of implementing a slave port interface?
- How will processor/adapter communication be carried out? The two nodes can issue transactions to each other, or they can communicate indirectly by depositing messages in shared memory space.
- The VAXBI interlock protocol must be used for interlock operations (see Section 5.2.2).
- Even if you observe all VAXBI requirements in your adapter design, you can introduce protocols that cannot be carried out by the system software. For example, processor nodes are often limited as to the interlock protocols they can generate. Adapters that require interlock protocols should not require protocols that the processor cannot perform.
- How is the node to request interrupts? (See Section 5.4.) Determine the number of vectors, the vector format, and consider implementation issues. What are the tradeoffs of internal vs. external vectors? An internal vector implementation is simpler, since the BIIC provides the logic required.

  Nodes with master ports will probably find it advantageous to request interrupts by setting force bits rather than by asserting INT lines. Using the existing master port logic and the force bits avoids the need to implement INT driver and control logic.
- Self-test requirements are described in Section 11.1. In addition to self-test of the BIIC which is automatic, the node must implement a self-test of the rest of the module.

  Will the node support two self-test lengths and therefore have to monitor BI STF L? (See Application Note 4, Section 4.2.)

Self-test requirements for multimodule nodes are given in Application Note 4.

- What is required for initialization on power-up? (See Section 11.1.7 and Application Note 7.)

  Nodes must not use "RC time constant type" reset circuits for power-on reset. Instead, nodes should use BCI DC LO L.

- Error-handling requirements are given in Section 11.2.

- Electrical requirements not implemented by the BIIC: All nodes must drive the BI BAD L line and this driver is specified by the user.

  Section 12.5 gives the electrical requirements for asynchronous control line drivers. Section 11.1.4 explains how the BI BAD L line is to be used. (See also Section 4.6 of Application Note 4.)

- What power dissipation, voltage, and current will the node require? Chapter 13 gives the VAXBI requirements.

- What node documentation is required?

  Initialization for node registers (see Application Note 1, page AN1-1).

  Node registers whose contents could be changed in response to a STOP command (see Section 5-5).

Design Tips and Topics

This section lists areas that some designers had difficulty with as VAXBI designs progressed. All of the following will not apply to all designs, since some relate to only master or slave port interface functions. The letters M, S, and I indicate whether the item applies to master port interfaces, slave port interfaces, or interrupt port interfaces. Applicable sections of the *VAXBI System Reference Manual* are cited.

For Your Consideration

1. The BIIC "owns" the BCI. The user interface must wait for the BIIC to assert BCI MDE L and BCI SDE L before it sends out command/address information or data onto the BCI lines. (M, S, I)

2. A slave must check parity through to destination. Slaves must not write data received with bad parity from the VAXBI. (S, I)

3. A node should not default to the STALL BCI RS code when it is not targeted as the slave. (S, I)

4. The following is a proven implementation of command latching: Drive the latch enable input of the command latch with a signal derived by ANDing BCI CLE H with T150 H. T150 H is a clock signal that is asserted high at T150 and deasserted at T0/T200. This procedure allows data on the output of the command latch to be held valid until T150 (plus delays). (S, I)

5. The trailing ACK may be a problem for slaves. Slaves must be able to respond to a new transaction while still supplying acknowledgments for the previous transaction. (S, I)

6. RETRY handling for masters is aided by the BIIC in that internal buffering stores the command/address and first longword of write-type data. (M)

7. Interfaces to nonpended buses should consider the implications of issuing RETRY rather than STALL. (Ap. Note 5) (S, I)

8. Unused BCI input lines must be tied to either +5V or ground as appropriate. (M, S, I)

9. The user interface must drive the node ID on BCI I<3:0> H while BCI DC LO L is asserted; no default is permitted. (M, S, I)

10. Self-test cautions: While performing self-test, nodes should be prepared for CLE assertions that result from transactions initiated by other nodes. (M, S, I)

11. To determine if the BIIC passes self-test, you can monitor the BCI EV lines for the Self-Test Passed (STP) EV code. Clock

the output of the EV code decoder with T100 going high into a flip-flop. BCI DC LO L should clear the flip-flop. (M, S, I)

12. Clock skew/loading cautions: An option designer should take into account the worst-case skew and loading characteristics regarding the VAXBI clock receiver. (Ap. Note 6) (M, S, I)
13. BCI DC LO L must be used to monitor DC power. Nodes must not use other reset methods such as the "RC time constant type." (Section 4.4.2 and Ap. Note 7) (M, S, I)
14. BCI AC/DC loading must be considered. (Chapter 20 and Ap. Note 6) (M, S, I)
15. The RXCD Register address location is reserved and requires the slave interface to either map around the address and not respond or respond with the BSY bit set. (S)
16. Intranode transactions (both loopback and VAXBI) are limited to longword length. (M)
17. Memories must respond to wrapped read-type transactions. (S)
18. Most designs find that a good time to sample BIIC-driven data is at T25. The user interface must be ready to take the data accounting for adequate setup/hold time and clock skew. (M, S, I)
19. The proper implementation of interlock transactions should be noted, particularly for error cases. (Section 5.2.2) (S)
20. In module layout keep within the 0.2-inch border requirement. (Section 13.1.2) (M, S, I)
21. If external interrupt vectors are used with the BIIC, at least one STALL is needed before nodes issue ACK or RETRY in response to an IDENT command. (Section 7.14) (S, I)
22. It is advisable to avoid external vectors, since the use of external vectors can complicate a design. (I)
23. How you choose to implement interrupts depends on how many interrupt vectors you need. Application Note 3 in the VAXBI SRM discusses several alternatives. Devices requiring only a single interrupt level should use the scheme outlined in Section 3.8.2 of the application note. (I)
24. Read-type transactions targeting I/O space must have no side effects. (Section 8.2.3) If you find that this rule imposes significant costs, you may have an indication that you ought to consider a radically different design (communication through shared memory and no slave port). (S)
25. The complexity of pipelined master port interface I/O designs probably outweighs performance gains. (M)
26. Various functions described in the VAXBI SRM were never intended to apply to I/O adapters. Therefore, I/O adapter designs need not be concerned about the following:
    - Issuing IPINTR transactions.
    - Issuing STOP transactions.
    - Doing reads and writes to other nodes' nodespaces.
    - Setting the NRST bit. Use of the node reset function is intended for operating systems.
27. Loopback transactions can be used by a device to access its own nodespace without the knowledge of its node ID.

BIIC Benefits

The following items suggest ways to maximize the use of the BIIC:

1. Using the GPRs and EV (event) codes to avoid need for a slave port interface. The EV codes are the BIIC's method of communicating status changes. To let the BIIC provide the slave functions, you can have the CPU write a node's BIIC GPRs. The adapter detects changes in these registers by monitoring the IRW (Internal Register Written) event code, which tells that an internal register has been modified.

2. Using internal vectors to avoid external vector logic and complexity.
3. Using the force bits instead of the BCI INT<7:4> L lines for interrupt generation—potential saving for all master port nodes.
4. Defaulting to "ACK" on the BCI RS<1:0> L lines to support STOP command acknowledgment; applies to nodes without any other need for a slave port interface. To avoid issuing ACK responses to commands not targeted to this node, the BCICSR should have only the STOP Enable bit set.

Figure 1-10 suggests a method to implement prefetching of master port BCI data. This method allows the use of relatively slow storage elements for the BCI data buffer, since prefetching changes the required access time from approximately 35 to 190 nanoseconds.

Hardware Cautions

You may be tempted to disregard some critical VAXBI requirements as you put together a test system. DIGITAL designers have learned from experience that there are good reasons for using the VAXBI hardware as it was intended to be used. Beware of the hazards that you may encounter as you handle the hardware.

- Use approved pin sockets for the BIIC. Other types of sockets may cause electrical integrity problems.
- Always attach an ESD wrist strap when handling BIICs and modules.
- Use a standard gate array chip puller to remove a BIIC safely, so that you do not bend the pins or chip the ceramic.
- Do not disassemble the card cage. The alignment of the connector block and card cage is critical for correct bus operation.

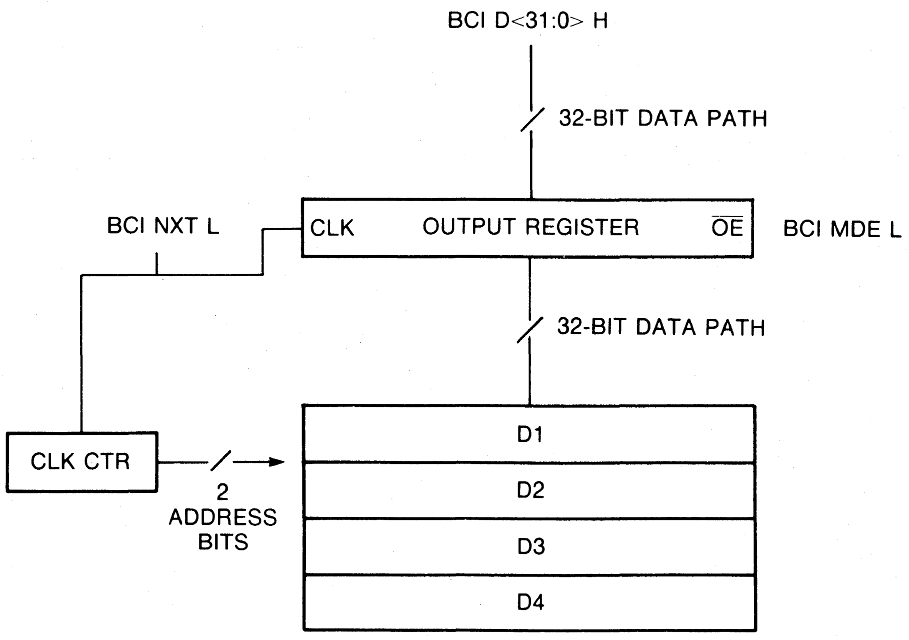


FIGURE 1-10 *Method of Prefetching Master Port BCI Data*

1. *Counter is initially loaded with the address of the first data longword (D1).*
2. *The first asserting edge of the BCI NXT L loads the output register with D1 data and at the same time increments the counter to D2. "Valid" D2 data is required when BCI NXT L asserts in the next cycle, a minimum of 190 ns (Taap as specified in Section 20.3).*
3. *This process continues for D3 and D4 and works for retried transactions as well.*

Chapter 2

The Instrument Control Adapter

by Design Analysis Associates

This chapter describes a VAXBI adapter to the IEEE-488 bus and the STD Bus. The adapter is a master-port-only design and uses a Z80 microprocessor. The project was commissioned by DIGITAL to serve as a design example for VAXBI option designers. The text describes the design process and cites portions of the *VAXBI System Reference Manual* that are relevant to design decisions.

Contents

Design Requirements and Design Decisions

This section introduces the Instrument Control Adapter (ICA), an interface to DIGITAL'S VAXBI bus, that was designed by Design Analysis Associates (DAA). The focus is on the product requirements for the adapter and the design decisions that were made based on the requirements.

Design Requirements

As with any design, the ICA design started as a need. The need was for a VAXBI to IEEE-488 adapter. This basic need was derived from DIGITAL's desire to support the VAXBI bus to a level comparable to that of the Q-bus and UNIBUS. Both the Q-bus and UNIBUS have IEEE-488 interfaces. The following market considerations supported this choice:

- The IEEE-488 is used extensively in the European market.
- Many laboratory instruments have IEEE-488 interface capability.
- VAX computers are presently being used in laboratories for instrumentation control.

These observations aided in defining the requirements for the ICA. The product requirements were defined early in the design process and are as follows:

1. The product must be an adapter. The determination of the node type is the first decision made in designing a node for the VAXBI bus. Three types of nodes are defined in the *VAXBI System Reference Manual:* processor, memory, and adapter nodes. The VAXBI SRM imposes certain requirements for each type of node. The only requirement for adapters is that they must respond to STOP transactions. (See Chapter 8 of the VAXBI SRM.)
2. The adapter must interface to the VAXBI bus in a manner that complies with requirements given in the VAXBI SRM. The requirements are in three general areas:
   - Operational requirements
   - Electrical requirements
   - Mechanical requirements
3. The adapter must interface to the IEEE-488 bus in a manner that complies with the IEEE-488 specification.
4. The adapter must have at least one IEEE-488 port; additional ports would be desirable.
5. The adapter must not depend on the VAX host to perform extensive control functions.
6. The adapter must be designed so that both development and production costs are minimized. This requirement implies that devices such as custom gate arrays or custom VLSI devices cannot be used because of their extensive development costs. Therefore, standard commercially available parts must be used. (This requirement is imposed by the limited development resources of DAA. For adapter design in general, the use of gate arrays can provide the designer with a method to improve performance while still remaining within the space requirements of a VAXBI module. It is likely that many, if not most, adapters will have custom gate arrays.)
7. The adapter must be designed so that complexity is kept to the minimum level necessary to meet the requirements. This requirement further strengthens the case against the use of custom gate arrays and custom VLSI devices. (This requirement is also specific to this particular adapter and the limited development resources. There are certainly applications where performance is paramount and complexity will likely result from the efforts to maximize the performance.)
8. There is no specific speed requirement. However, the data throughput should be as great as possible given the other requirements.
9. The adapter must be testable both for production and field service environments.

These requirements were used by the adapter design team to drive the design process. The next step in this process is to examine the effects the requirements have on the design options that are available. From this the specification can be established.

Design Decisions

In this section we discuss how the requirements presented in the previous section affected design decisions for the ICA.

The first requirement is that the adapter must interface to the VAXBI bus. The primary effect this requirement has on any VAXBI adapter design is to influence the design to be done using a VAXBI module with the VAXBI Corner and its associated functionality packaged in a VAXBI backplane and card cage.

The VAXBI Corner, with the BIIC and other circuitry, satisfies the requirement that the adapter interface to the VAXBI bus. The functionality of the VAXBI Corner with a minimum of additional circuitry is capable of handling all the VAXBI bus interface requirements. The VAXBI Corner satisfies both operational requirements and electrical signal requirements imposed by the VAXBI SRM. The designer's problem is thus reduced to having to interface to the VAXBI Corner, which is considerably less complex than interfacing to the VAXBI bus directly. The decision to use a VAXBI backplane, card cage, and module may seem almost trivially obvious, but it does influence the design by establishing the form factor to which the design is restricted. In addition, it assures that many of the electrical signal and power requirements and mechanical requirements are met.

The requirement that the adapter must not depend on the VAX host to perform extensive control functions influences the determination of the type of adapter that is to be designed. The VAXBI SRM defines three types of adapters: programmed I/O, direct memory access (DMA), and bus adapters (see Application Note 1 of the VAXBI SRM). Based on these definitions and the requirement not to depend on the VAX host to perform extensive control functions, the adapter could not be of the programmed I/O type.

Of the two remaining choices for the adapter type, DMA and bus, the bus adapter is generally more complicated. A bus adapter must be capable of supporting a master on either bus taking control of both buses for direct memory access operations. To do so, the adapter must arbitrate for both buses. When the two buses are very different, as is the case with the VAXBI and IEEE-488 buses, the adapter design can be very complex. For this adapter there was no requirement that a master on the IEEE-488 bus be capable of directly accessing devices on the VAXBI bus, thus eliminating the need for a bus adapter. The DMA-type adapter then is the logical choice for the ICA.

A DMA-type adapter requires some intelligence so that it can initiate the transactions necessary to transfer data to and from VAX memory. The adapter must also be capable of performing data transfer operations to the IEEE-488 bus. The requirement for intelligence, combined with the nonspecific speed requirement, led to the decision to use a microprocessor as the central control element of the adapter. This decision is also supported by the requirement to keep cost and complexity to a minimum. The microprocessor approach provides the intelligence needed for a DMA-type adapter; however, this approach could limit the data transfer rate if speed were a critical requirement. (When the requirements were established for this adapter, the BCAI chip was not available for consideration. The use of the BCAI chip in adapters provides considerable functionality in terms of data transfer control. The BCAI could be used with a microprocessor to provide increased data transfer rates while retaining the intelligence and flexibility associated with the microprocessor approach.)

The use of a microprocessor also fits in well with the decision to implement a DMA-type adapter. The microprocessor is used to control data transfer operations by buffering the data and then generating the transactions necessary to complete the data transfer from one bus to the other. This is also consistent with the requirement not to depend on the VAX host for extensive control functions. The decision to use a microprocessor led to a protocol by which the complexity could be limited. The protocol relies on command and response packets in queues in VAX memory that can be manipulated by either the VAX or the ICA. Since these packets are exchanged through the VAX memory, the VAX processor does not need to perform write-type transactions to a slave port in the ICA. This eliminated the need to implement a slave port user interface in the adapter, thus limiting the complexity of the adapter. Although the adapter requires only a master port user interface, the BIIC allows the node to respond as a slave. (See Chapter 18, BIIC Operation, in the VAXBI SRM: Sections 18.3.1 and 18.3.2 describe BIIC internal register reads and writes; Section 18.3.3 describes how the BIIC responds to an IDENT transaction.)

Once the decision was made to use a microprocessor for control, the remaining design decisions were influenced by that. Use of a microprocessor, in addition to the requirement to keep production costs to a minimum, drove the decision to use 74LS-series parts to interface to the VAXBI Corner. The interface to the VAXBI Corner is treated as an I/O port mapped into the I/O space of the Z80 microprocessor.

We decided to support only longword transactions from the master port. This decision was made to reduce complexity, since the data transfer rate was not a consideration. (Here is another case where use of the BCAI chip could reduce the complexity in implementing quadword and octaword transfers.) Doing only longword transactions reduced complexity in several ways:

- The need to store additional data words was eliminated.
- The logic necessary to assert successive longwords on the BCI data lines was eliminated.
- The logic necessary to control RETRY of longer than longword transfers was eliminated. [The BIIC has all the functionality necessary to retry longword transactions without further action by the master port interface other than to reassert the transaction request (see Section 18.2 of the VAXBI SRM).]

To generate interrupts we decided to use the User Interface Interrupt Control Register (UINTRCSR) rather than the BCI INT<7:4>(L) signal lines (see Section 7.14 of the VAXBI SRM). The microprocessor can use loopback transactions to the UINTRCSR eliminating the need for circuitry to drive the BCI INT<7:4>(L) signal lines and the logic necessary to determine when they should be driven.

The intelligence of the microprocessor also provided a method for error handling. Rather than having extensive hardware to capture and decode the event codes and additional hardware to respond to the event codes, the microprocessor handles this. Hardware is still needed to capture the event codes at the proper time, but the microprocessor can examine the event codes and take the proper action. This approach also allows the capability of having the error-handling routines be a function of the host and the software that is running on the host. The host can down-line load the error-handling routines to the microprocessor to be compatible with the rest of the system.

The requirement to interface to the IEEE-488 bus, combined with the decision to use a microprocessor, drove the decision to use commercially available IEEE-488 support chips on the microprocessor bus as the interface to the IEEE-488 bus.

At this point all functional requirements were satisfied. When a preliminary parts count and layout were done, we realized that space was still available for additional functionality. We then decided to have a total of four IEEE-488 ports. This decision was influenced by the requirement for testability. With four ports implemented, the ports can drive each other in pairs for testing purposes.

Since space was still available, we felt that an RS-232 interface would provide additional flexibility for debug and testing. With an RS-232 interface a terminal could be attached to the adapter directly. The terminal could then be used to control the microprocessor without the need for a VAX host during debug or standalone testing.

Even with the RS-232 addition, space was available. We then decided that since the Z80 is directly compatible with the Standard Bus (STD Bus), a STD Bus interface would be simple to add, and possible applications of the ICA would be increased. The only additional parts required would be buffers to drive the bus from the on-board Z80 bus off the board. Addition of the STD Bus interface provided additional instrument-control capability as well as the other device interfaces available on the STD Bus. With the addition of this bus the adapter can operate as an independent processor using the Z80 for software development, debug, and testing purposes. The STD Bus provides a means to test a significant portion of the functionality of the adapter without the need for a VAXBI system.

Functional and Operational Description

This section describes the Instrument Control Adapter (ICA) as it is implemented. The presentation is in a top-down approach beginning with a general overall description. As the description gets more detailed, the emphasis is on the specific functions that relate to the VAXBI Corner and other VAXBI bus issues. Since this document is primarily concerned with VAXBI adapter design, details of the IEEE-488 are not presented.

High-Level Functional Description of the ICA

The ICA derived its name from the early phases of the design. At first only the IEEE-488 interface was to be implemented, and since the IEEE-488 is used extensively for instrument interfaces, the name was chosen. During the course of the design, the STD Bus interface was added. This was done primarily because of the decision to use a Z80 microprocessor to control the adapter which made implementation of the STD Bus interface relatively simple. The RS-232 interface was added with minimal design impact and increased the flexibility of the adapter. These interfaces are shown in Figure 2-1.

The functionality of the ICA is divided into three primary areas:

- The Z80 microprocessor and its associated functions
- The VAXBI Corner and the user interface to the corner
- The IEEE-488, RS-232, and STD Bus external interfaces

Associated with the Z80 hardware functionality is the software that runs in the Z80.

The primary control activities of the ICA are handled by a Z80 microprocessor. It controls command packet processing, response packet generation, and data transfer. The user interface to the VAXBI Corner uses 74LS-series devices under control of the Z80. These devices provide the speed required to interface to the VAXBI Corner while still allowing the Z80

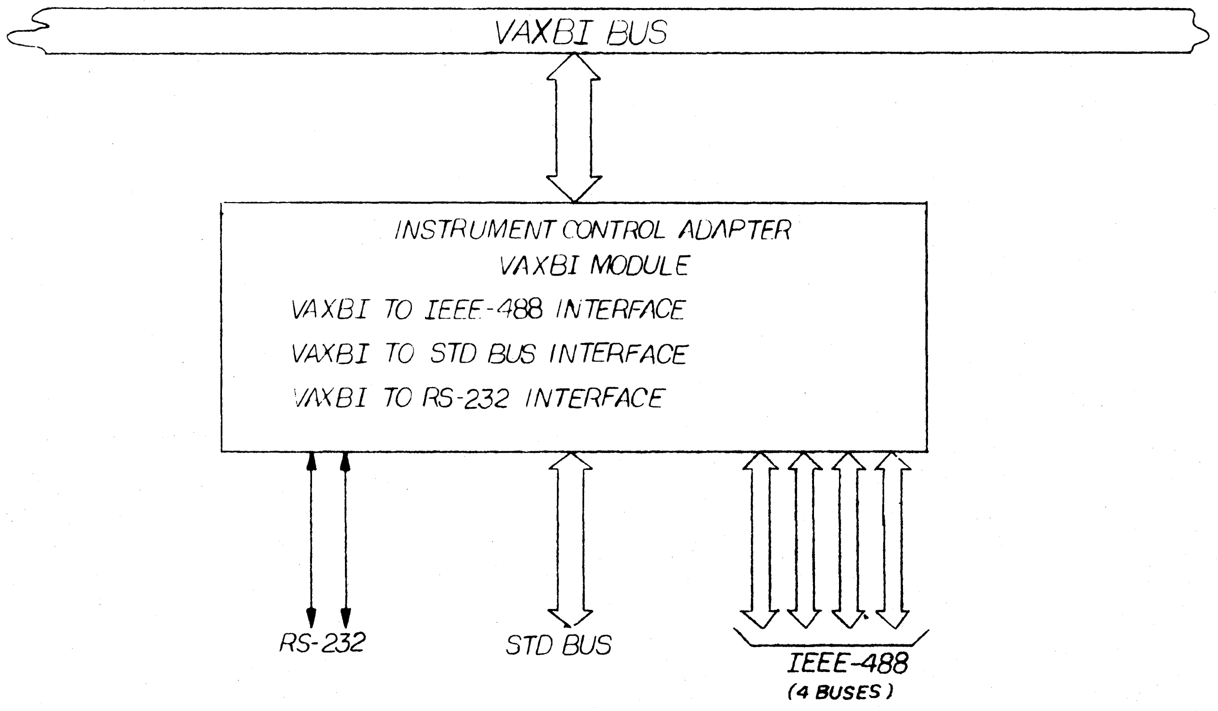

VAXBI BUS
INSTRUMENT CONTROL ADAPTER
VAXBI MODULE
VAXBI TO IEEE-488 INTERFACE
VAXBI TO STD BUS INTERFACE
VAXBI TO RS-232 INTERFACE
RS-232
STD BUS
IEEE-488
(4 BUSES)

FIGURE 2-1 *ICA First-Level Functional Partition*

to control the functions of the adapter. A functional partition showing this architecture is shown in Figure 2-2.

The user-implemented portion of the ICA—that is everything except the VAXBI Corner—is a master-port-only design. The transactions that the adapter responds to as a slave are writes to the BIIC internal General Purpose Register 0 (GPR0) and the STOP and IDENT transactions. The writes to GPR0 are handled entirely by the BIIC. This register is used to pass addresses for locating control information to the ICA. The STOP transaction requires only that the slave port generate an ACK response and the user portion perform all required STOP activities. Because the ACK must be generated, the ICA does act—in this one instance—as if it has a slave port interface. However, there are no addresses implemented for the slave port. The VAXBI bus protocol for STOP transactions is handled by the BIIC.

The ICA communicates with the host processor using command and response packets. These packets are located in queues in the host memory which can be manipulated by either the host or the ICA. The Z80 controls the activities of the other functions of the ICA through the use of reads and writes to its own I/O addresses.

Z80 Microprocessor Functions

Figure 2-3 shows a detailed functional partition of the ICA including the Z80 microprocessor and its related functions. Note that all other functions of the adapter are tied to the Z80 through either data or control paths. Figure 2-4 shows a detailed functional partition of the Z80 and its related functions.

Figure 2-5 shows a detailed functional partition of the I/O bus buffer and register select functions. The I/O bus buffer is used to buffer the memory bus of the Z80 from the I/O bus that drives IEEE-488 interfaces and the VAXBI Corner user interface. The register select function decodes the addresses from the Z80 and generates the necessary signals to control the user interface and the IEEE-488 interface.

User Interface to VAXBI Corner

The user interface to the VAXBI Corner implements a control function and a data function. The control function provides the necessary interaction between the VAXBI Corner and the user-implemented functions of the ICA (see Figure 2-6). The data function provides the data path between the VAXBI Corner and the user functions (see Figure 2-7).

Control Function

The control function has seven subfunctions:

- Master Transaction Request Control
- Reset and Clock
- Self-Test Status
- Master Transaction Control
- Transaction Done Indicator
- Master Event Code Readback
- Internal Transaction Detect

Additionally, BCI INT<7:4>(L) signals are pulled up to +5VDC. The response lines are hardwired to the ACK code. This is done because the only time the BIIC looks for a response from the slave port in the ICA is in response to a STOP command. IDENT transactions in the ICA use the internal vector, and the BIIC generates the confirmation code independent of the response lines. The BCI INT<7:4>(L) interrupt capability is not used by the ICA, thus pulling the INT lines to their deasserted (H) state assures that no spurious interrupts are generated. Interrupts are generated by use of the User Interface Interrupt Control Register.

Master Transaction Request Control

The Master Transaction Request Control subfunction consists of a 74LS175 inverting register. It is mapped to I/O address FF of the Z80. This register is loaded on the deasserting edge of LDBCIRQST(L) that is generated by the register select function. The logic state of the Z80 I/O data bus signals, IOD<7:0>(H), determines the state of the outputs. Specifically, BCI RQ<1:0>(L) are generated by IOD<1:0>(H), and BCI MAB(L) is generated by IOD<2>(L). An additional output from this register, IOEXP(L), is not related to the VAXBI Corner. This signal, which is related to the STD Bus interface, is generated here for convenience and does not affect the VAXBI Corner.

Reset and Clock

The Reset and Clock subfunction consists of 74S04 inverters used to buffer the BCI PHASE(L) and BCI DC LO(L) signals for use throughout the adapter. BCI PHASE(L) is

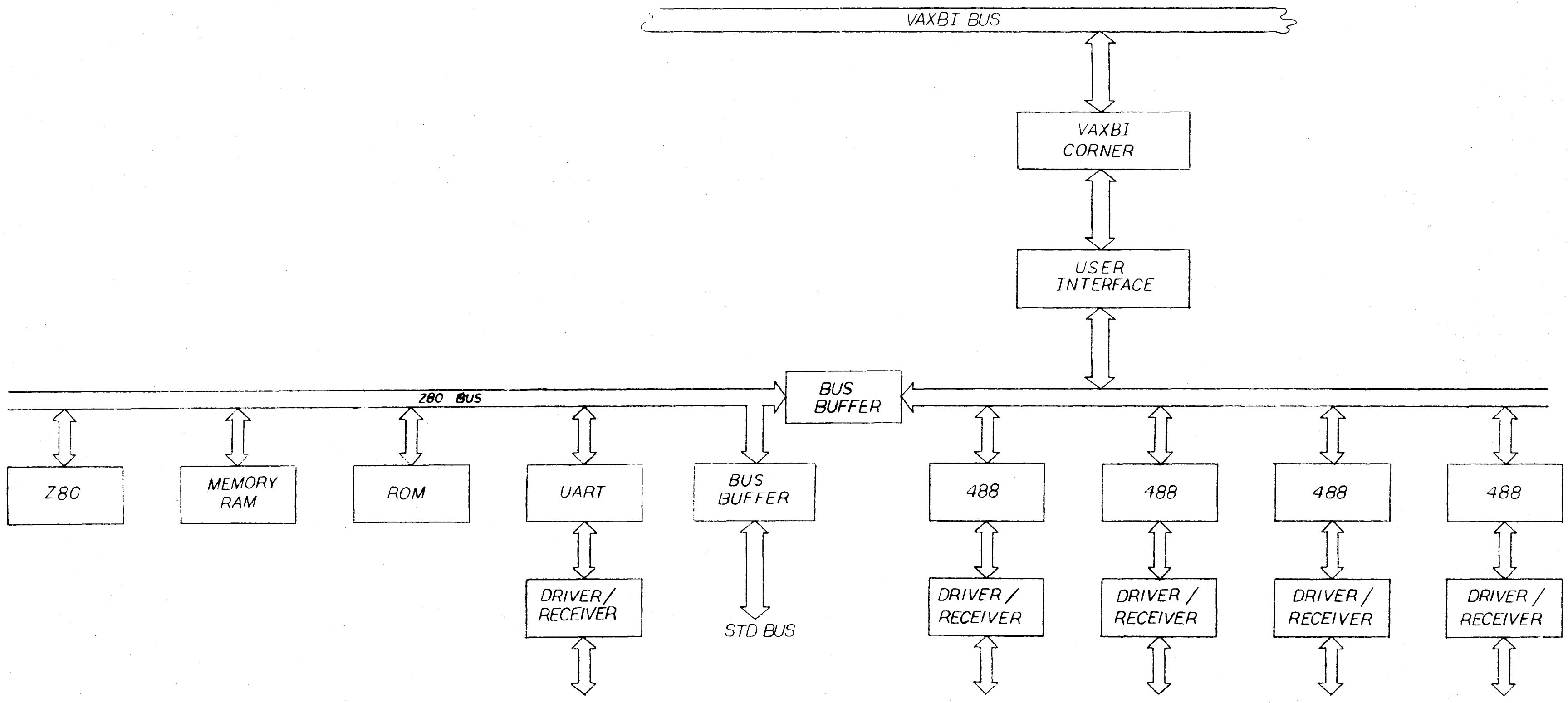


FIGURE 2-2 *ICA Second-Level Functional Partition*

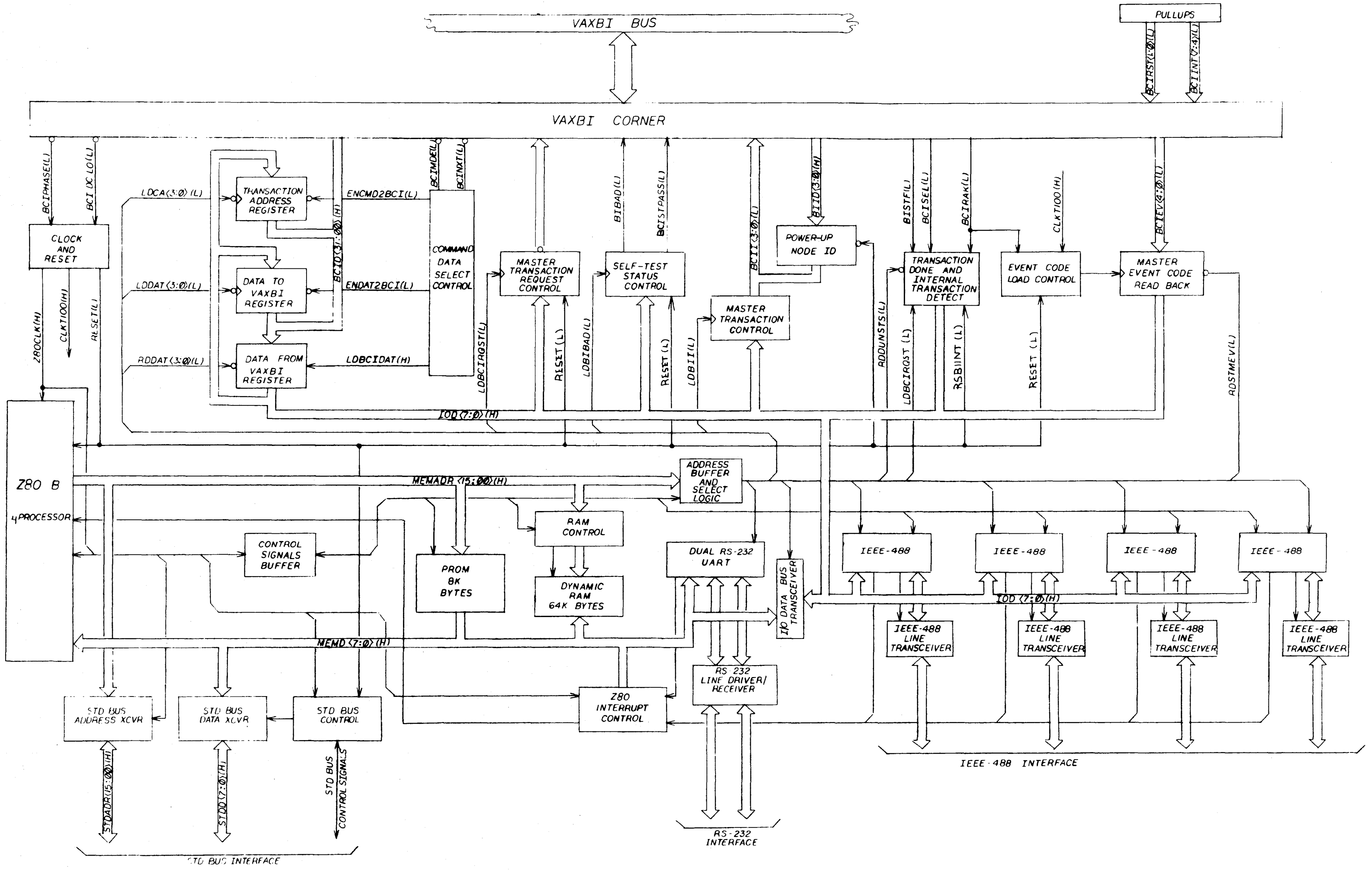


FIGURE 2-3 *ICA Detailed Functional Partition*

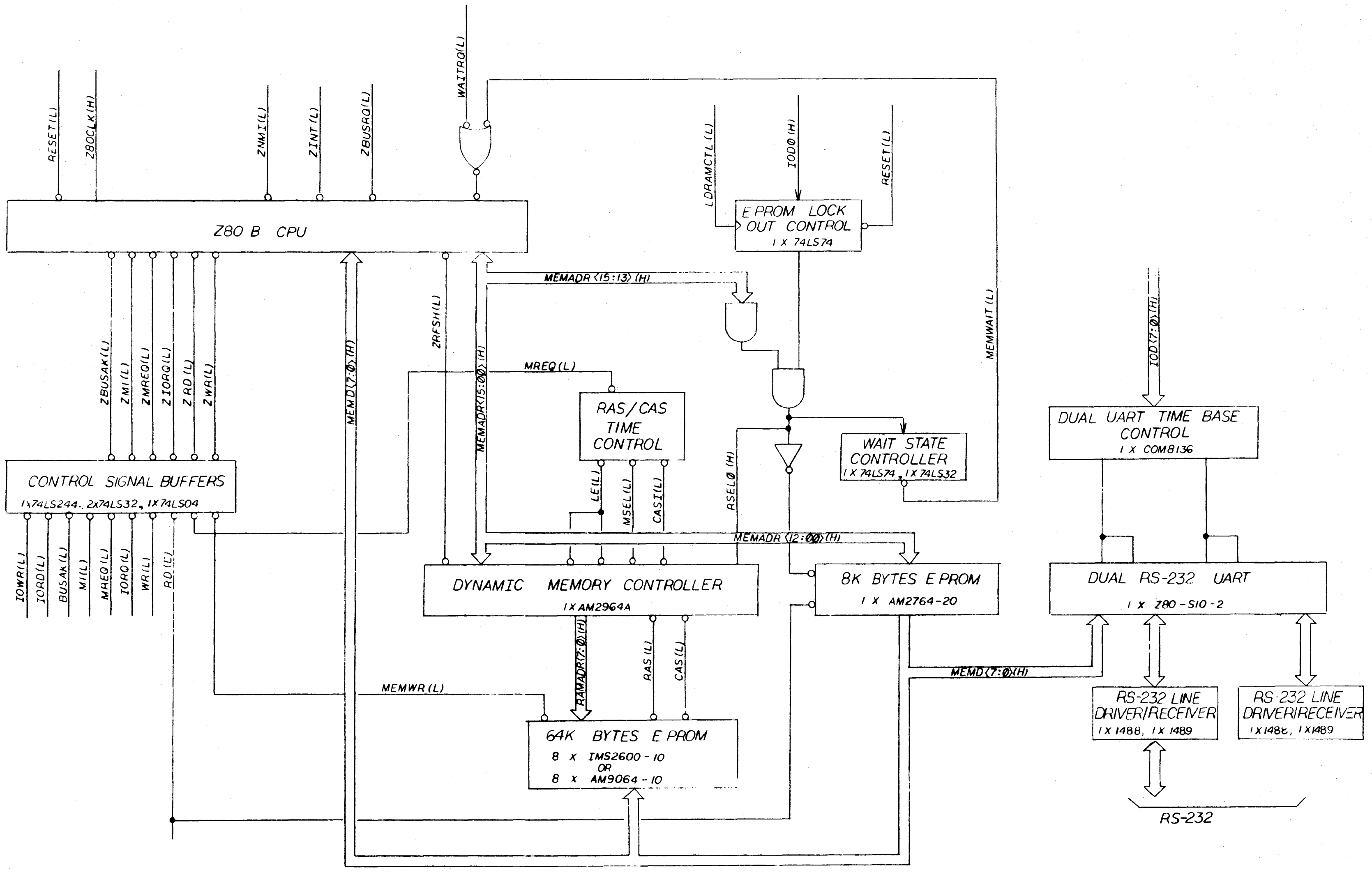


FIGURE 2-4 *ICA Z80 Processor and RS-232 Detailed Functional Partition*

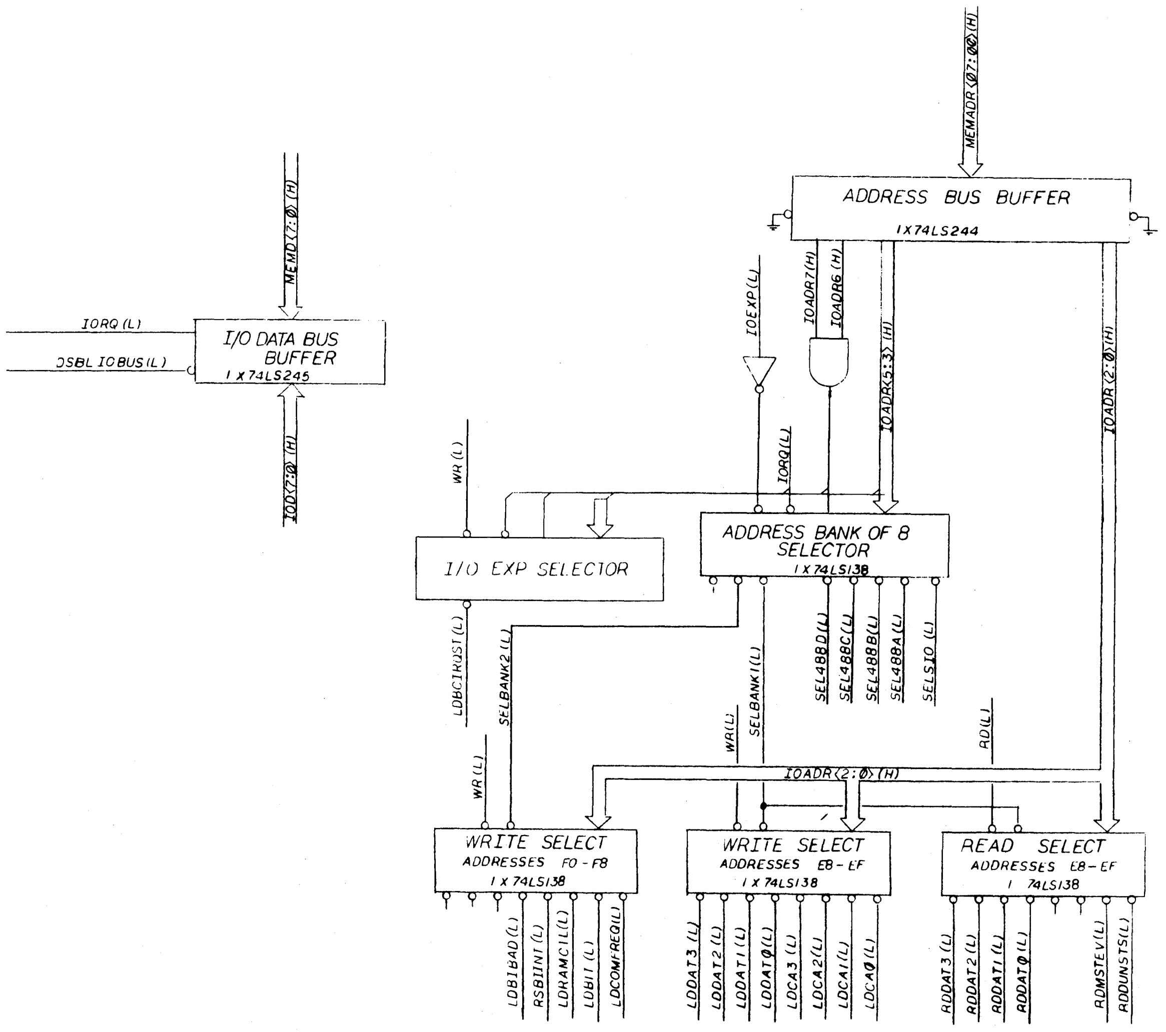


FIGURE 2-5 *ICA I/O Bus Buffer and Register Select Detailed Functional Partition*

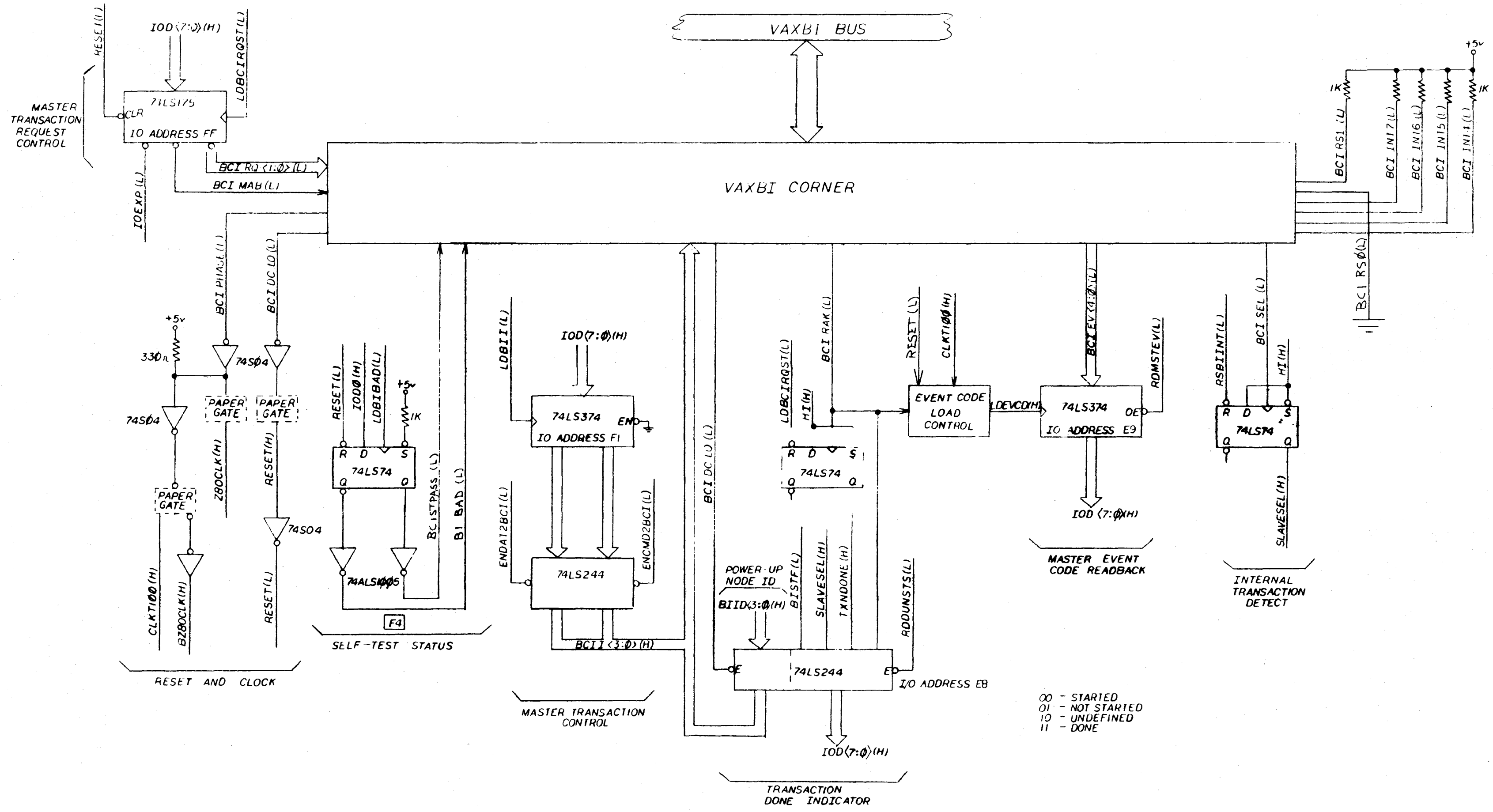


FIGURE 2-6 *ICA VAXBI Corner User Interface Control Function Detailed Functional Partition*

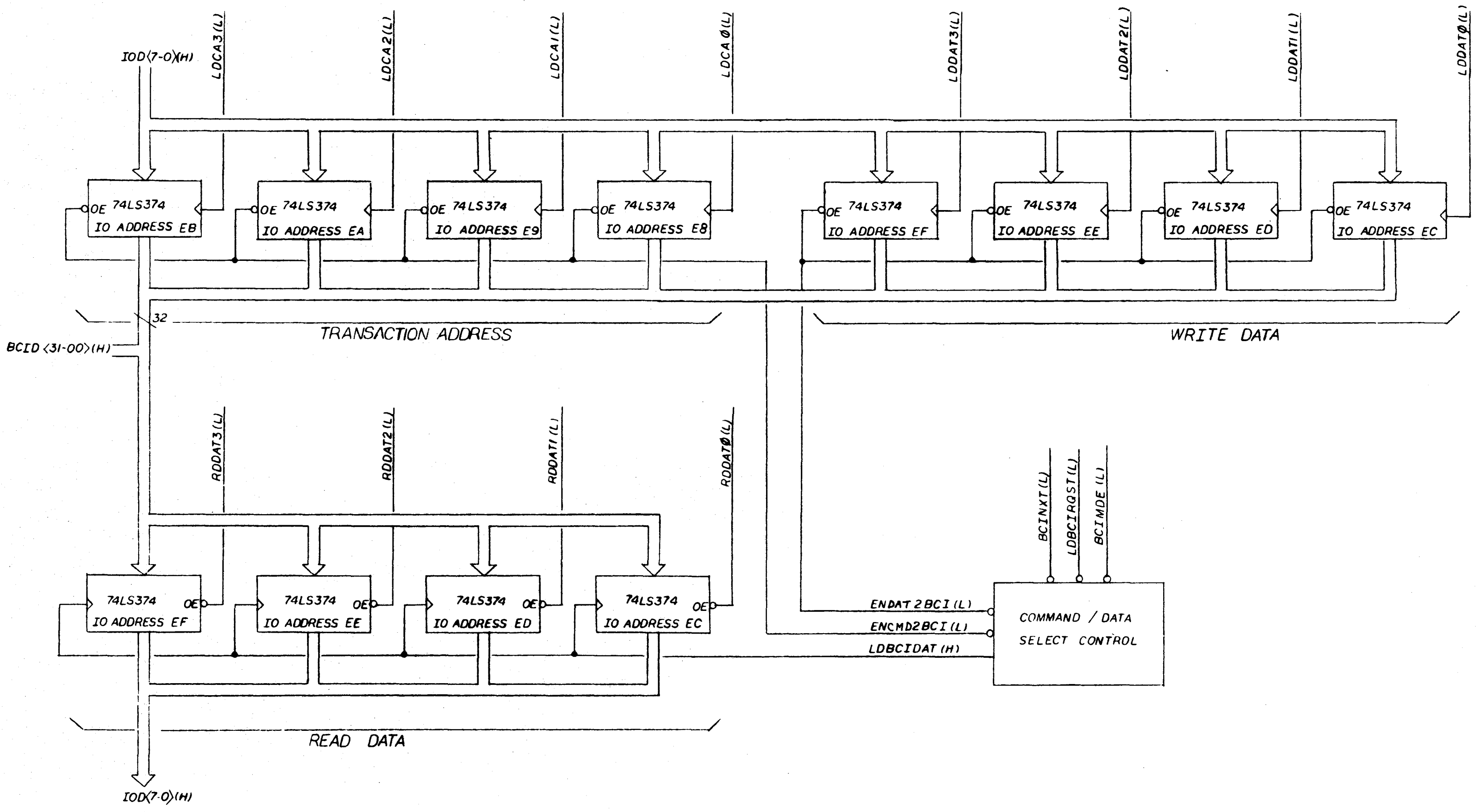


FIGURE 2-7 *ICA BCI Transaction Address and Data Registers Detailed Functional Partition*

inverted and pulled up to +5VDC through a 330-ohm resistor to provide the proper operating levels for the Z80CLK(H) signal. The CLKT100(H) signal is used to clock control circuitry associated with the VAXBI Corner user interface. The BZ80CLK(H) signal is used to clock control circuitry associated with the Z80 functions. The Z80CLK(H) and BZ80CLK(H) signals, which drive the Z80 and its associated functions, are 5 MHz clocks.

The design decision to use the VAXBI timing signals as the Z80 clock was based on the need to synchronize the Z80 to the BIIC. Synchronizing the Z80 to the BIIC is important because of the synchronous nature of the BCI that requires signal assertions to be synchronous. Synchronization is most important when the transaction request is asserted on the BCI RQ<1:0>(L) lines. The other BCI signals are either set up before they are actually used by the BIIC, or they are synchronized using the signals generated by the BIIC, such as BCI MDE(L). Use of the 5 MHz clocks also satisfies the requirement of the IEEE-488 support chips to operate at 5 MHz. The RESET(L) signal is used to reset the adapter to a known state. The RESET(L) signal is simply the BCI DC LO(L) signal buffered and renamed, as indicated by the "paper gate," to a name indicative of its function. (The paper gate is a documentation technique used to change the name of a signal without changing its electrical characteristics.)

Self-Test Status

The Self-Test Status subfunction, which is controlled by the Z80, consists of a 74LS74 D-type flip-flop and two 74LS1005 inverting buffers. Whenever an initialization sequence is started, the RESET(L) signal is asserted. This action resets the Self-Test Status flip-flop which asserts BI BAD(L) and deasserts BCI STPASS(L). The Z80 after successful completion of self-test sets the flip-flop by writing to its I/O address F4, which asserts LDBIBAD(L) and IOD<0>(H). LDBIBAD(L) is generated by the register select logic.

Master Transaction Control

The Master Transaction Control subfunction generates the BCI I<3:0>(H) signals. It consists of a 74LS374 8-bit register and a 74LS244 8-bit tri-state buffer. The register is loaded with the command code and the write mask, if required, on the asserting edge of LDBII(L). LDBII(L) is generated by the register select logic when the Z80 performs a write to its I/O address F1. The register is loaded from the IOD<7:0>(H) lines with bits <3:0> corresponding to the command code and bits <7:4> corresponding to the mask. Since the ICA only generates longword transactions, only one mask is required for each transaction. The command is enabled onto the BCI I<3:0>(H) lines by the assertion of ENCMD2BCI(L), which is generated by the command/data select control logic. The mask is enabled by ENDAT2BCI(L), which is also generated by the command/data select control logic.

Transaction Done Indicator

The Transaction Done Indicator subfunction provides information to the Z80 about the state of the VAXBI Corner user interface. When the Z80 generates a read to its I/O address location E8, the register select control function asserts RDDUNSTS(L), which enables the contents of the register onto the IOD<3:0>(H) lines. The subfunction consists of a 74LS74 D-type flip-flop and a 74LS244 8-bit tri-state buffer. Four of the bits in the 74LS244 are used to enable the node ID onto the BCI I<3:0>(L) lines while BCI DC LO(L) is asserted during initialization. The other four bits of the buffer are used to enable BI STF(L), SLAVESEL(H), TXNDONE(H), and BCI RAK(L) onto the IOD<3:0>(H) lines, respectively. The TXNDONE(H) signal is asserted on the deasserting edge of BCI RAK(L) and deasserted when LDBCIRQST(L) is asserted. The SLAVESEL(H) signal is discussed below in the paragraph on the Internal Transaction Detect subfunction. Table 2-1 shows the truth table that defines the relationship between TXNDONE(H) and BCI RAK(L) and the state of a transaction.

Table 2-1 Transaction State Truth Table

TXNDONE(H)	BCI RAK(L)	Transaction State
0	1	Not Started
0	0	Started
1	1	Done
1	0	Undefined

Master Event Code Readback

The Master Event Code Readback subfunction consists of the Event Code Load Control state machine and a 74LS374 8-bit tri-state register. The state machine is used to determine the proper time to latch the event code. Its operation is defined by the Mnemonic Documented State (MDS) diagram and logic maps in Figure 2-8. The BCI EV<4:0>(L) lines are loaded into the register on the asserting edge of LDEVCD(H) from the state machine. The contents of the register are asserted onto the IOD<7:0>(H) lines when RDMSTEV(L) is asserted by the register select logic. RDMSTEV(L) is asserted when the Z80 generates a read to its I/O address E9.

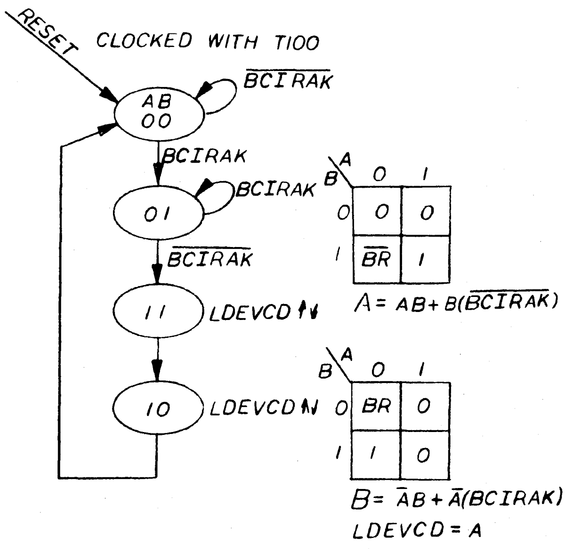


FIGURE 2-8 *ICA MDS Diagram and Logic Maps for Master Event Code Readback Controller*

Internal Transaction Detect

The Internal Transaction Detect subfunction consists of a single 74LS74 D-type flip-flop. It generates the SLAVESEL(H) signal which is available to the Z80 and indicates when a transaction is directed to this node. The only transactions that the ICA responds to as a slave are transactions to the internal registers of the BIIC and the STOP and IDENT transactions. These transactions are handled by the BIIC. (For STOP transactions the BIIC recognizes the ACK response that is hardwired to the BCI RS<1:0>(L) lines. For IDENT transactions an internal vector is used and the BIIC generates the confirmation code independent of the state of the BCI RS lines.)

Even though these transactions are handled by the BIIC, the Z80 must know about them. The Internal Transaction Detect subfunction provides this capability. The BICSREN bit in the BCI Control and Status Register must be set to enable this capability. SLAVESEL(H) is asserted on the deasserting edge of BCI SEL(L) and is deasserted under control of the Z80 when RSBIINT(L) is asserted from the register select logic. RSBIINT(L) is asserted when the Z80 generates a write to its I/O address F3.

Data Function

The data function (shown in Figure 2-7) consists of three 32-bit data registers—each of which is comprised of four 74LS374 8-bit tri-state registers—and a Command/Data Select Control subfunction. The three 32-bit registers are designated for command/address data, write data, and read data. The implementation details for the Command/Data Select Control subfunction are shown in Figure 2-9. This subfunction controls the enabling of command/address and write data onto the BCI D<31:0>(H) lines and latching of read data from the BCI D<31:0>(H) lines.

The Z80 loads the transaction address data into the transaction address register eight bits at a time. Four control signals, LDCA<3:0>(L), generated by the register select logic when the Z80 generates a write to its I/O addresses E8 through EB, load each byte into the register. Similarly, for the write data, the four signals, LDDAT<3:0>(L) corresponding to writes to I/O addresses EC through EF, control loading of the register. Read data is enabled from the register onto the IOD<7:0>(H) lines eight bits at a time with the four RDDAT<3:0>(L) signals corresponding to reads to I/O addresses EC through EF.

Data transfers to and from the BCI D<31:0>(H) lines are controlled by the Command/Data Select Control subfunction (see Figure 2-9). The subfunction consists of a 74LS74 D-type flip-flop and two 74LS32 2-input OR-gates used as AND functions. When a transaction request is initiated in the Master Transaction Request Control subfunction by the assertion of LDBCIRQST(L), the flip-flop is reset. This enables the AND function that generates ENCMD2BCI(L). When BCI MDE(L) is asserted, ENCMD2BCI(L) is asserted which enables the transaction address onto the BCI data lines. On the asserting edge of BCI NXT(L), the flip-flop is set, and the AND function which generates ENDAT2BCI(L) is enabled. When BCI MDE(L) is asserted, ENDAT2BCI(L) is asserted, which enables write data onto the BCI data lines. Since all transactions of the ICA are longword, the assertion of BCI NXT(L) only occurs once. BCI NXT(L) is used to load the read data register from the BCI data bus.

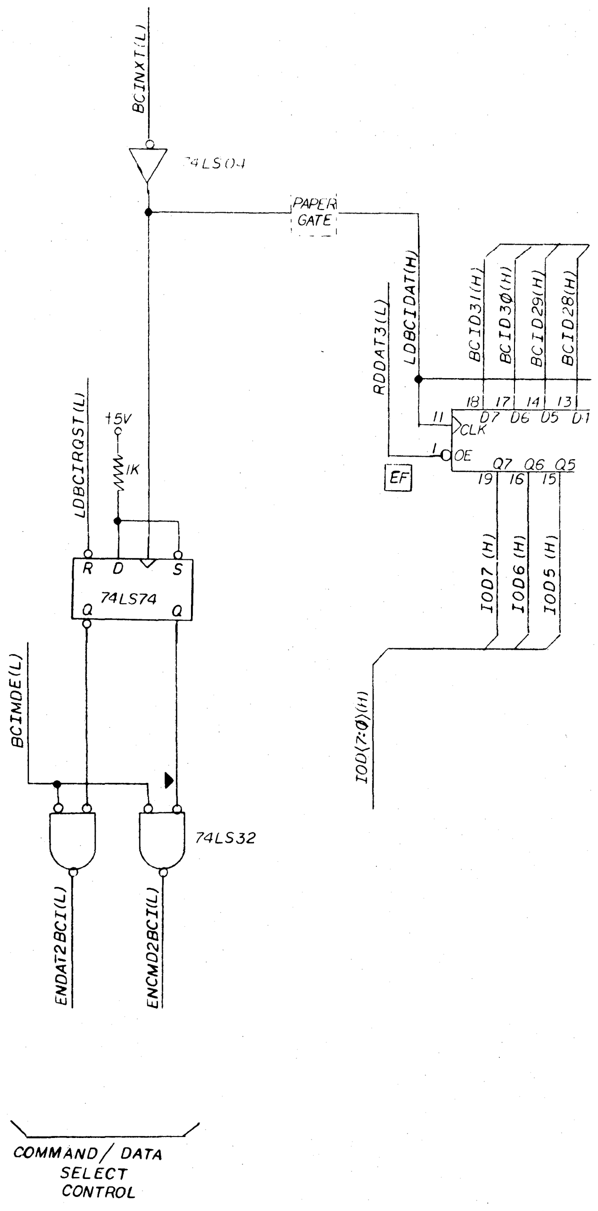


FIGURE 2-9 *Command/Data Select Control Subfunction Details*

External Interfaces

The ICA supports three types of external interfaces:

- RS-232 (Two of this type.)
- STD Bus (One of this type.)
- IEEE-488 (Four of this type.)

The RS-232 interface is generated directly from the Z80 memory data bus with the time base being controlled by the I/O bus. The RS-232 interface is shown in Figure 2-4 in the detailed functional partition of the Z80 microprocessor functions.

Figure 2-10 shows a detailed functional partition of the STD Bus interface. The STD Bus interface is driven directly by the Z80 bus and its associated controls.

Figure 2-11 shows a detailed functional partition of the four IEEE-488 interfaces. These interfaces are controlled by the Z80 I/O bus.

Z80 Software Functions

The Z80 software provides the control intelligence for the ICA. The Z80 software does the following:

- Controls the external interfaces and the user interface to the VAXBI Corner
- Generates the command codes and addresses for VAXBI transactions
- Handles all data transfers between the BCI and the external interfaces
- Performs user self-test and initialization

The following sections describe the operation of the Z80 and how the Z80 relates to other elements of a VAXBI system.

Communication with the VAX

The VAX and the ICA talk to each other using a system of packets. Command packets are filled by the VAX and tell the ICA what to do, and response packets are filled by the ICA and tell the VAX what to do.

Communication with the VAX is done by using four queues in VAX memory:

- Command Packet Queue
- Empty Command Packet Queue
- Response Packet Queue
- Empty Response Packet Queue

To issue a command to the controller, the VAX grabs an empty command packet from the head of the Empty Command Packet Queue, fills it, and places it on the tail of the Command Packet Queue. Similarly, when the Z80 wishes to communicate with the VAX, it grabs an empty response packet from the Empty Response Packet Queue, fills it, and places it on the Response Packet Queue. A packet is

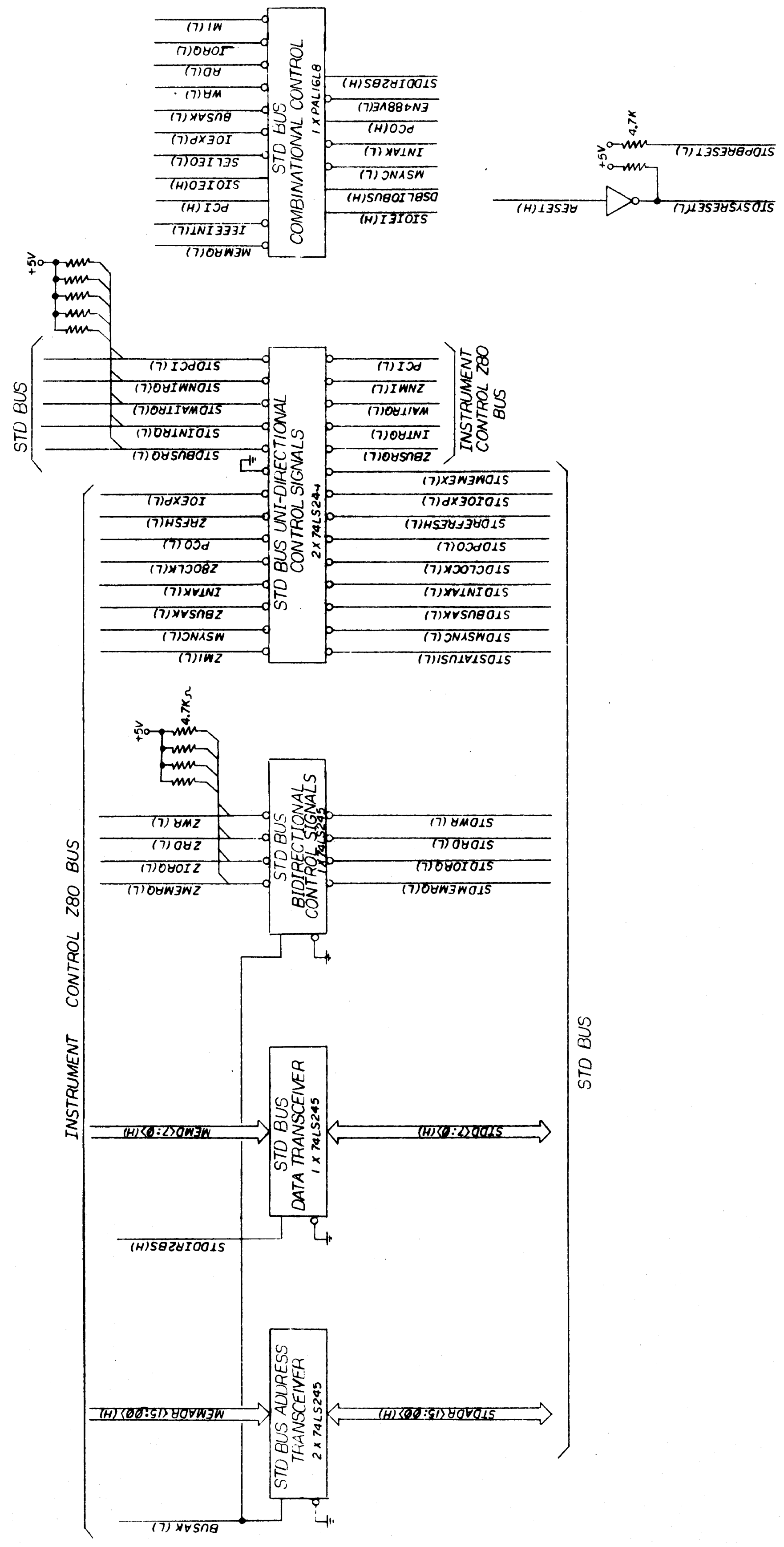


FIGURE 2-10 ICA STD Bus Interface Detailed Functional Partition

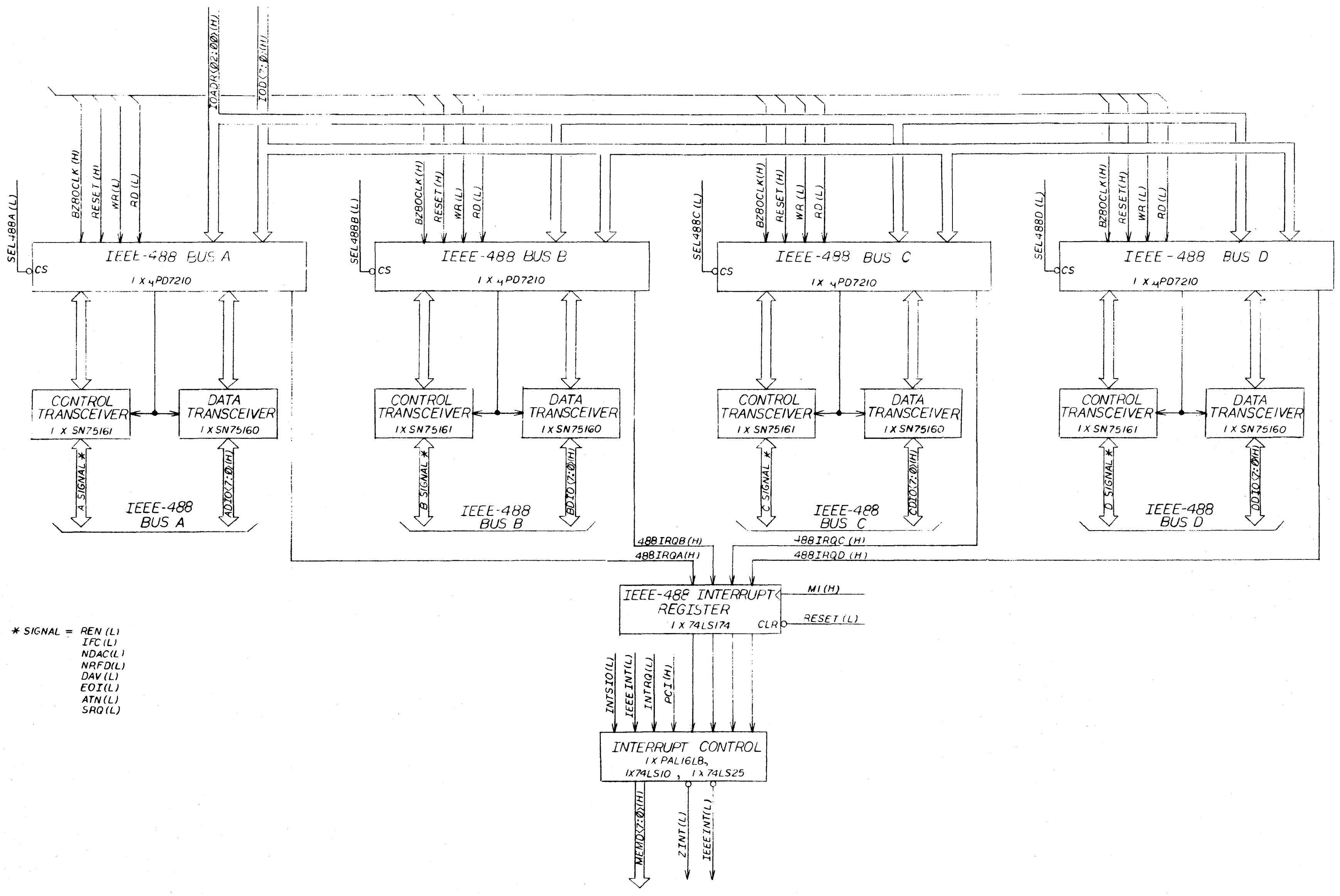


FIGURE 2-11 *ICA IEEE-488 Bus Detailed Functional Partition*

returned to the respective empty queues when the recipient of the packet has completed processing it. All queue operations observe proper VAX protocol as defined in the *VAX-11 Architecture Reference Manual.*

Command Packets

Command packets are placed on the Command Packet Queue by the VAX to indicate that an operation is to be performed by the ICA. The format of the command packet is shown in Figure 2-12. Bits <7:0> specify the unit number to which the request is directed, bits <15:8> specify the operation to be performed by the unit, and bits <31:16> must be zero (MBZ).

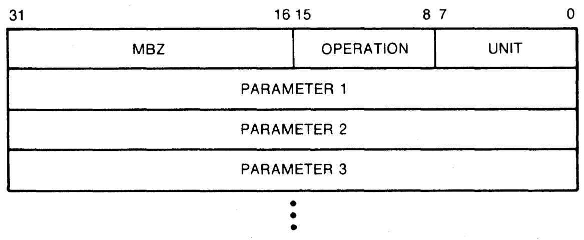


FIGURE 2-12 *Command Packet Format*

Unit numbers are defined as follows:

0 STD Bus interface

1 IEEE-488 Bus 1 Controller/Command Interpreter

2 IEEE-488 Bus 1 Talker

3 IEEE-488 Bus 1 Listener

4 IEEE-488 Bus 2 Controller/Command Interpreter

5 IEEE-488 Bus 2 Talker

6 IEEE-488 Bus 2 Listener

7 IEEE-488 Bus 3 Controller/Command Interpreter

8 IEEE-488 Bus 3 Talker

9 IEEE-488 Bus 3 Listener

10 IEEE-488 Bus 4 Controller/Command Interpreter

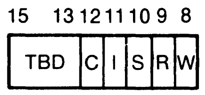
11 IEEE-488 Bus 4 Talker

12 IEEE-488 Bus 4 Listener

13 RS-232 Interface 1

14 RS-232 Interface 2

Bits <15:8> of the first longword of the command packet, which specify the operation to be performed, are broken down as follows:



W If this bit is set, a Write operation is to be performed; that is, data will be written from the VAX to the unit.

R If this bit is set, a Read operation is to be performed; that is, data will be read from the unit to the VAX.

If both the R and W bits are set, a Write operation will be performed; however, the user process has write access to the buffer and the buffer can be modified by the unit.

S If this bit is set, a Set Mode/Set Characteristics or a Read or Write with function modifiers is to be performed.

I If this bit is set, a Unit Initialize is to be performed.

C If this bit is set, a Cancel I/O is to be performed.

Figure 2-13 shows the command packet format for read and write operations.

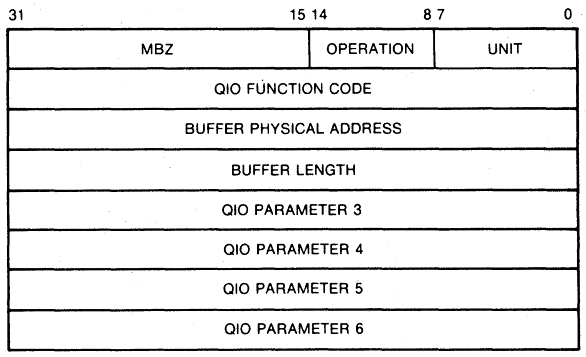


FIGURE 2-13 *Command Packet Format for Read and Write Operations*

Response Packets

Response packets are placed on the Response Packet Queue by the controller to indicate that the VAX must perform an operation on behalf of the ICA; for example, to notify the VAX that an I/O operation has been completed or to give interrupt status.

The format of the response packet is shown in Figure 2-14. Bits <7:0> specify the unit number that generated the interrupt. Bits <14:8> give an 8-bit unit-dependent reason for the interrupt. Bits <23:15> give an 8-bit unit-dependent status value. Bits <31:24> give an 8-bit value used to request special action by the VMS driver.

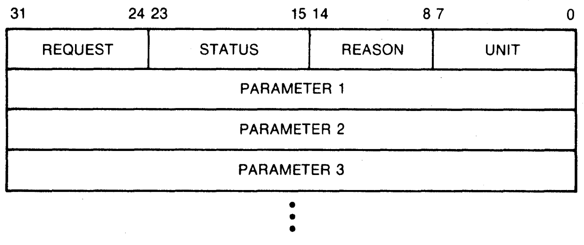


FIGURE 2-14 *Response Packet Format*

Controller Devices

The VMS driver performs the device-independent functions that a VMS driver normally does, and the controller performs the device-dependent functions. Firmware in the ICA is divided into several tasks executing in round robin order—each of which handles a single device. In this way new device firmware can easily be added to handle devices attached to the STD Bus on the ICA. The VMS driver dynamically creates and destroys devices in a manner similar to the DEC mailbox and Ethernet drivers to support devices in the ICA. When firmware is loaded into the ICA to support a device, a VMS unit is created to allow application software to talk to the unit's firmware. If the device is no longer needed, its firmware may remove the unit from the unit queue in the ICA and stop its tasks. The VMS driver will delete the device created to talk to that unit's firmware.

A library of common subroutines is included in the ICA's on-board ROM. These subroutines include response packet generation, VAX memory-handling utilities, and the multitasking kernel routines to create and destroy processes. These routines are reentrant and may be called by normal processes and by interrupt service routines without ill effect.

Firmware routines for the Z80 drive the four IEEE-488 buses and the two RS-232 interfaces and allow the VAX to read and write STD Bus memory and I/O ports. The STD Bus firmware is in ROM, and the rest of the firmware is loaded by the VAX at system start-up time.

The ICA Implementation of VAXBI Protocol

The BIIC handles most of the VAXBI protocol. The user-implemented portion of the ICA is responsible for the protocol associated with initialization. The user-implemented portion of the ICA communicates with the VAXBI bus through the BIIC. Communicating with the BIIC is done through the user interface to the BCI and its associated protocol.

Performing I/O operations to the ICA consists of a series of activities that are controlled by the VAX, the BIIC in the ICA, the Z80 in the ICA, and the device on the external bus. Table 2-2 gives a summary of these activities. Figure 2-15 shows the elements of the ICA and its associated VAX host that are used in the execution of I/O operations as they relate to the VAXBI bus and its protocol. The sections that follow give examples of the activities associated with transaction handling in the ICA.

Transfer of Data from an External Interface to VAX Memory

Data can be transferred from an external interface to VAX memory in two ways:

- The device on the external interface can inform the ICA that it has data to be transferred to the VAX.
- The VAX can request that the ICA read data from an external device.

The following example assumes the second case. Suppose a logic analyzer is attached to the IEEE-488 Bus A, and the VAX wants to read the contents of the logic analyzer memory. The assumption is also made that the logic analyzer has been previously set up to be the IEEE-488 Talker and the IEEE-488 Bus A in the ICA set up to be the Listener. The following sequence of events occurs:

1. The VAX sets up a data buffer in VAX memory for the data that is to be transferred.
2. The VAX builds the appropriate command packet on the Empty Command Packet Queue. This command packet includes the information necessary for the ICA to determine that a transfer of data from an external device to the VAX is required, which port the device is attached to, and where to put the data.

Table 2-2 Operation of I/O Requests

VAX Activities	Z80 Activities
▪ Accepts QIO from the user and validates parameters	
▪ Verifies process privilege based upon the type of I/O operation (e.g., Write Logical operations require the Log_IO privilege)	
▪ Locks any associated buffer into physical memory and translates the virtual addresses of the buffer	
▪ Obtains empty command packet from Empty Command Packet Queue	
▪ Writes command information into the command packet and places the packet on the Command Packet Queue	
▪ Writes the address of the Communication Queues Header Block in GPR0	
	▪ Receives notification that GPR0 has been written
	▪ Removes command packet from Command Packet Queue and copies contents into Z80 memory
	▪ Returns the empty command packet to the Empty Command Packet Queue
	▪ Calls the routine to service the VAX
	▪ The VAX service routine does whatever is required and returns
	▪ Removes packet from Empty Response Packet Queue
	▪ Builds response packet
	▪ Places response packet on Response Packet Queue
	▪ Interrupts the VAX if the Response Packet Queue is empty (it is assumed that the VAX is busy emptying the queue if it is not)
▪ Receives interrupt	
▪ Removes response packet from the Response Packet Queue	
▪ Does whatever is required	
▪ Places empty response packet on the Empty Response Packet Queue	
▪ If the response queue is not empty, repeats operation for the next packet	

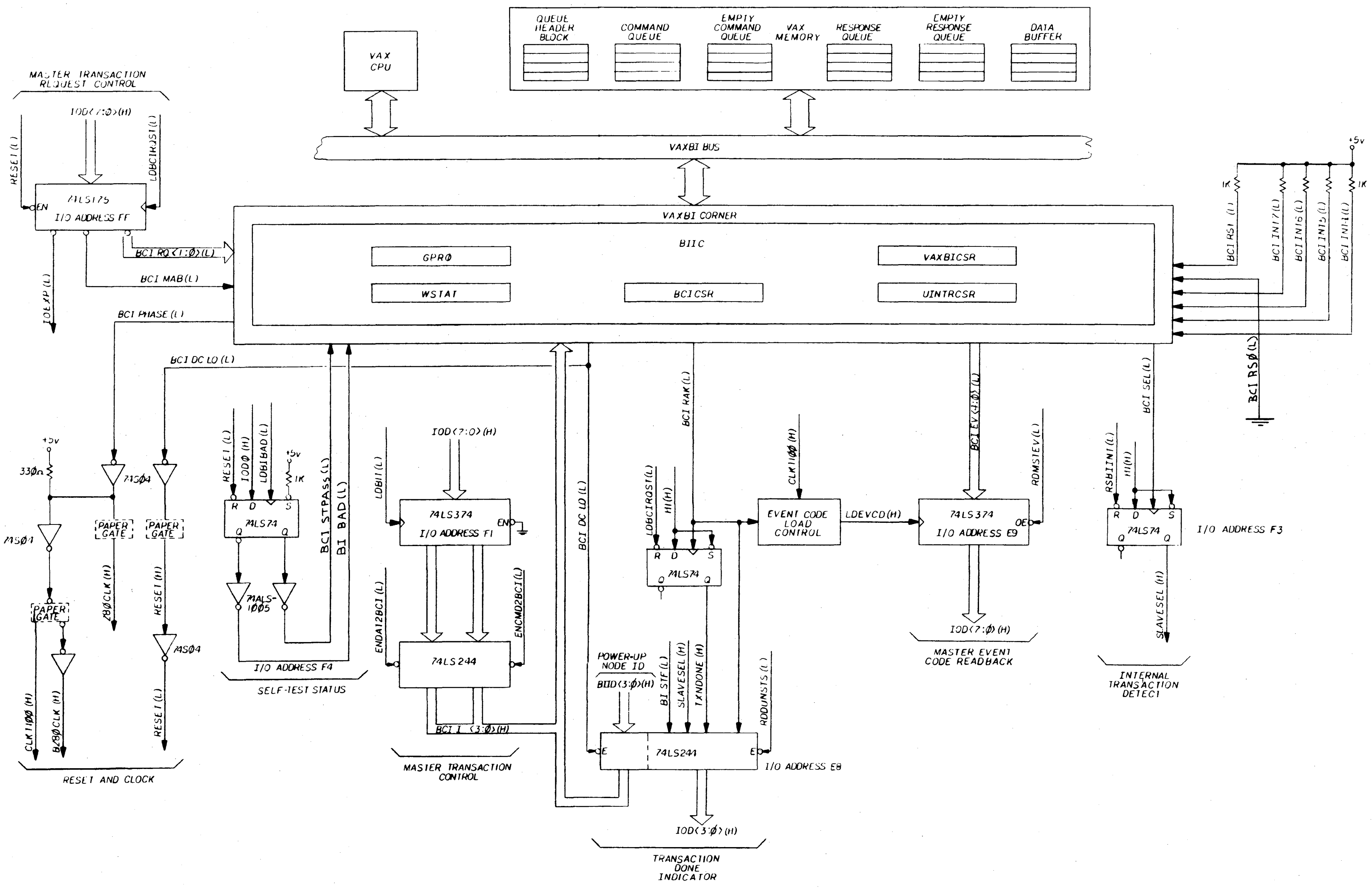


FIGURE 2-15 *Elements of the ICA and VAX Host Used in I/O Operations*

3. The VAX then moves the command packet from the Empty Command Packet Queue to the bottom of the Command Packet Queue. If no other command packets are in the queue, the command packet will be on the top of the queue.
4. The VAX then writes to General Purpose Register 0 (GPR0) in the BIIC of the ICA. The BIIC handles the transaction protocol associated with the write to GPR0. Additionally, since the BICSREN bit in the BCI Control and Status Register (BCICSR) is set, which is done at initialization, the BIIC generates the BCI SEL(L) signal. This sets the SLAVESEL(H) signal (see Figure 2-6). The BIIC sets the GPR0 bit in the Write Status Register (WSTAT).
5. The Z80 determines when a transaction has occurred by reading the register at its I/O address E8. When this is done, the the I/O register select logic generates the RDDUNSTS(L) signal, which enables BI STF(L), SLAVESEL(H), TXNDONE(H), and BCI RAK(L) onto the IOD<3:0>(H) lines, respectively.
6. If a transaction has occurred, the Z80 reads the WSTAT to determine that GPR0 has been written. This is done by generating a loopback READ transaction to the WSTAT.

   The following sequence generates a loopback READ transaction to the WSTAT.

   (1) The Z80 writes the command code to the Master Transaction Control register. This is done with a write to I/O address F1, which asserts LDBII(L), which loads the register with the data on IOD<7:0>(H). For this instance IOD<7:0>(H) would be as shown below:

IOD bits	7	6	5	4	3	2	1	0
	X	X	X	X	L	L	L	H

   The "x" represents a "don't care" condition for a READ transaction. For a WMCI or UWMCI transaction, bits <7:4> would contain the write mask. The "LLLH" code is the command code for a READ transaction.

   (2) The Z80 then does a sequence of four writes to its I/O space. These four writes correspond to the four bytes of the transaction address. The four bytes are located at the four consecutive I/O addresses E8 through EB. Writing to these addresses causes the I/O register select logic to assert the LDCA0(L) through LDCA3(L) signals, respectively. The two most significant bits of I/O address EB must contain the longword transaction size code "LH." The address of WSTAT is 2C. Since the write to WSTAT is done as a loopback transaction, the base address is not required and would be ignored by the BIIC. Thus, the transaction address would be as follows:

I/O address	EB								EA							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	H	L	L	L	L	L	L	L	L	L	L	L	L	L	L
BCI D bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

I/O address	E9								E8							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	L	L	L	L	L	L	L	L	L	H	L	H	H	L	L
BCI D bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

   (3) The Z80 then requests a loopback transaction by writing the loopback transaction request code "LH" to the Master Transaction Request Control register. A write to I/O address FF causes the I/O register select logic to assert LDBCIRQST(L) signals, which loads the register

from the IOD<7:0>(H) lines. This signal also initializes the command/data select control logic for selection of the transaction address word when the BIIC asserts BCI MDE(L) and initializes the Transaction Done Indicator. For this example, the IOD lines would be as follows:

IOD bits	7	6	5	4	3	2	1	0
	X	X	X	X	?	L	H	L

The "?" denotes the state of the IOEXP bit which does not relate to the BCI. Note that the register is an inverting register.

(4) The BIIC will then assert BCI MDE(L). This causes the command/data select control function to assert ENCMD2BCI(L) which enables the contents of the transaction address register onto the BCI D<31:0>(H) lines and the low-order four bits of the Master Transaction Control register onto the BCI I<3:0>(H) lines. The state of the select control function is changed so that the next assertion of BCI MDE(L) would enable write data onto the data lines in the case of a write-type transaction.

(5) At this point the Z80 can monitor the state of the transaction by reading the Transaction Done Indicator register. This is done by reading I/O address E8 which causes the I/O register select logic to assert RDDUNSTS(L), which enables the contents of the register onto the IOD<3:0>(H) lines.

(6) When the data is available on the BCI master port, the BIIC asserts BCI NXT(L). This signal is used directly to load the Read Data register with the data on the BCI D<31:0>(H) lines.

(7) The BIIC then deasserts BCI RAK(L), which sets the Transaction Done flip-flop to indicate that the transaction is done. The Z80 must read this register as described above to make this determination.

(8) The Z80 can then read the contents of the Read Data register one byte at a time. The Read Data register is located at addresses EC through EF, LSB through MSB respectively, in the I/O address space of the Z80. When these reads are executed, the I/O register select logic asserts RDDAT0(L) through RDDAT3(L), which asserts the contents of the four byte registers of the 32-bit data register. For the particular case of the WSTAT register, only the MSB is required and only the MSB of the Read Data register needs to be read by the Z80.

(9) The Z80 can determine from the contents of the WSTAT if the GPR0 register has been written. If it has been written, the contents of the MSB will appear as shown below on the IOD lines when the MSB is read by the Z80:

IOD bits	7	6	5	4	3	2	1	0
	L	L	L	H	L	L	L	L

If the GPR0 has not been written, all bits will be "L." The only other VAXBI transactions that the ICA receives are the STOP and IDENT transactions. (See Sections 5.5, 5.4, 18.3.5, and 18.3.8 of the VAXBI SRM.) At this point the Z80 knows that the GPR0 has been written.

7. The GPR0 now contains the address in VAX memory of the Queue Header Block. Since the address of the Queue Header Block does not change, it is only read once after initialization and then stored by the Z80 for future use. Read-

ing the contents of the GPR0 is done in the same way as described for reading the WSTAT register except that the address for the GPR0 is F0 and all four bytes are needed by the Z80.

8. The Z80 then uses this address to acquire the Queue Header Block. Reading the Queue Header Block is done using VAXBI READ transactions. The steps are the same as for the loopback reading of an internal register. The differences are that the VAXBI transaction request code is substituted for the loopback transaction request code and the transaction address is that which was contained in the GPR0 register. Since the ICA does only longword transactions, four separate transactions are required to acquire the four longwords that comprise the Queue Header Block. Once initialized the Queue Header Block does not change; therefore, it need only be read once by the ICA and saved by the Z80 in its memory. This saves time on subsequent activities that require the Queue Header Block.

9. Since GPR0 was written to by the VAX, the Z80 knows that a command packet is ready on the Command Packet Queue. Therefore, the Z80 performs a VAXBI IRCI to the first longword in the queue header. This read will be the same as when reading the Queue Header Block except that the transaction address is that of the Command Packet Queue given by the Queue Header Block. Bit <0> of this word is the lock bit for the queue. If set, it means that another process is using the queue and it is not available. The Z80 then generates a VAXBI UWMCI to write the word back to the queue header. The Z80 continues this sequence until bit <0> is cleared by the process using the queue. When the queue is available, the Z80 sets bit <0> to lock the queue for its use and generates a VAXBI UWMCI to write the queue header longword back to the queue header. The Z80 then performs a VAXBI READ transaction to the address of the first longword of the command packet. For this specific example, the contents of the first longword of the command packet is as follows:

I/O address	EB								EA							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
BCI D bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

I/O address	E9								E8							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	L	L	L	L	L	H	L	L	L	L	L	L	L	H	L
BCI D bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

This first word of the command packet informs the ICA that a read operation is required; that is, a transfer of data from the unit, the logic analyzer, to the VAX.

10. The ICA then reads the next longword of the command packet, which contains the QIO Function Code.

11. The ICA then reads the next longword of the command packet, which contains the Buffer Physical Address, which tells the ICA where to put the data that it retrieves from the logic analyzer.

12. The ICA then reads the next four longwords of the command packet, which contain four QIO parameters. This operation requires four longword READ transactions.

13. The ICA then returns the "used" command packet to the Empty Command Packet Queue and unlocks the queue.

14. The Z80 in the ICA now has the information necessary to begin transferring data from the logic analyzer to VAX memory. The transfer is done by alternately reading data from the logic analyzer and then generating VAXBI WRITE transactions to the buffer defined in the command packet.

15. When the buffer is full, the ICA informs the VAX by creating a response packet. This is done by building the response packet on the Empty Response Packet

Queue through a series of VAXBI write operations to the address of the Empty Response Packet Queue defined in the Queue Header Block.

16. The ICA then moves the response packet from the Empty Response Packet Queue to the Response Packet Queue.
17. By generating an interrupt, the ICA informs the VAX that the Response Packet Queue contains a response. The interrupt is generated by a loopback write transaction to the User Interface Interrupt Control Register (UINTRCSR) in the BIIC.
18. The VAX recognizes the INTR and issues an IDENT to acquire the interrupt vector.
19. The ICA responds to the IDENT with an interrupt vector. Responding to the IDENT is handled by the BIIC.
20. The VAX uses the interrupt vector to locate the interrupt service routine. The interrupt service routine tells the VAX that a response packet is available on the Response Packet Queue.
21. The VAX reads the response packet and performs the action required.

This completes the activities required to read data from the logic analyzer.

Transfer of Data from VAX Memory to an External Interface

Transfer of data from VAX memory to an external interface, or unit, is referred to as a write operation at the system level. At the VAXBI level such a transfer requires many read- and write-type transactions to complete. This type of transfer can be initiated in two ways:

- The device on the external interface can inform the ICA that it requires data from the VAX.
- The VAX can request that the ICA write data to the external interface.

For this example let's assume the second case. As for the example in the previous section, suppose a logic analyzer is attached to the IEEE-488 Bus A, and the VAX wants to set up the logic analyzer for data capture. The assumption is also made that the logic analyzer has been previously set up to be the IEEE-488 Listener and the IEEE-488 Bus A in the ICA set up to be the Talker. The sequence of events is similar to that for the previous example. To highlight the differences and avoid repeating unnecessarily, the following sequence of steps and the associated step numbers refer to the steps in the previous example of a read operation.

1. The VAX sets up a data buffer in VAX memory and loads it with the data that is to be transferred.
2. This step is the same except that the command packet indicates that the transfer is from VAX memory to the unit.
3. The VAX puts a command packet on Command Packet Queue. (Same as read operation.)
4. The VAX writes to GPR0. (Same as read operation.)
5. The Z80 determines that the transaction has occurred. (Same as read operation.)
6. The Z80 reads the WSTAT register. (Same as read operation.)
7. The Z80 reads GPR0, if required. (Same as read operation.)
8. The Z80 acquires the Queue Header Block, if required. (Same as read operation.)
9. The Z80 acquires the first longword of the command packet. This is the same as for the read operation except that the longword will have the following contents:

I/O address	EB								EA							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L	L
BCI D bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

I/O address	E9								E8							
IOD bits	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	L	L	L	L	L	L	L	H	L	L	L	L	L	L	H	H
BCI D bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

10. The ICA reads the QIO Function Code. (Same as read operation.)
11. The ICA reads the Buffer Physical Address. (Same as read operation.)
12. The ICA reads the remaining four longwords of command packet. (Same as read operation.)
13. The ICA returns empty command packet. (Same as read operation.)
14. This step differs from a read operation. In this case the ICA performs VAXBI read transactions to the data buffer in VAX memory to acquire the write data. The ICA then transfers this data, using IEEE-488 Bus A, to the logic analyzer.
15. The ICA builds the response packet. (Same as read operation.)
16. The ICA moves the response packet to the Response Packet Queue. (Same as read operation.)
17. The ICA interrupts the VAX. (Same as read operation.)
18. The VAX issues an IDENT. (Same as read operation.)
19. The ICA responds to the IDENT. (Same as read operation.)
20. The VAX performs the interrupt service routine. (Same as read operation.)
21. The VAX reads the response packet and performs the action required. (Same as read operation.)

ICA Handling of STOP Transactions

In addition to performing the VAXBI transactions necessary for read and write operations, the ICA must respond to STOP transactions from the VAXBI because STOP is in the Must Respond Set (MRS) of adapters. However, the STOP Enable bit in the BCICSR register must be set for the BIIC to recognize the STOP transaction and pass it through to the slave port. The following sequence of events describes how the ICA responds to a STOP transaction:

1. During initialization of the ICA, the Z80 must reset the INIT bit in the VAXBI Control and Status Register (VAXBICSR) by executing a loopback WRITE transaction to the BIIC and writing a one to the bit to clear it. This bit will be read by the Z80 in the ICA to recognize the occurrence of a STOP transaction.
2. The VAX generates a STOP transaction.
3. The BIIC handles the VAXBI protocol associated with the STOP transaction and sets the INIT bit in the VAXBICSR. Additionally, since the BICSREN bit in the BCICSR is set (which is done at initialization), the BIIC generates the BCI SEL(L) signal. This action sets the SLAVESEL(H) signal.
4. The Z80 determines when a transaction has occurred by reading the register at its I/O address E8 as described in Step 5 of the read operation.
5. The Z80 reads the WSTAT register to determine if the GPR0 register has been written as described in Step 6 of the read operation. However, for a STOP transaction all bits in the WSTAT register will be "L" indicating that GPR0 has not been written.
6. The Z80 reads the VAXBICSR using a loopback transaction in the same way as reading the GPR0 register. The address for the VAXBICSR is 04. If a STOP transaction has occurred, bit 13, the INIT bit, will be "H."
7. The Z80 then stops executing and remains in this "stopped" state until it is reset by the VAX.

ICA Initialization and Self-Test

At power-up or when a node or system reset occurs, the user-implemented portion of the ICA must do the following:

- Assert the node ID on the BCI I<3:0>(H) lines while BCI DC LO(L) is asserted.
- Load the Device Register with the ICA identification code.
- Execute a self-test sequence to test the user-implemented portion of the ICA.

If the self-test completes successfully, the following must occur in sequence:

- The self-test LEDs must be lit.
- The Broke bit in the VAXBICSR must be cleared.
- The BI BAD(L) line must be deasserted within one microsecond of clearing the

Broke bit. BI BAD(L) is driven with a 74ALS1005 inverting buffer.

The remaining initialization steps are performed by the BIIC as defined in Chapter 6 of the VAXBI SRM.

Self-test in the ICA is performed by the Z80 and consists of the following:

- Reading and writing of alternating one/zero patterns to all Z80 memory locations.
- Generating loopback READ and WRITE transactions to the BIIC General Purpose Registers with alternating one/zero patterns.
- Reading and writing to external devices if applicable.

The Z80 begins the ICA self-test after it determines that the BIIC successfully completed its self-test. This determination is made by the Z80 generating a loopback READ of the VAXBICSR register, which contains the Self-Test Status bit. When set, this bit indicates that the BIIC passed self-test. If the BIIC does not pass self-test, the Z80 does not perform any further activities. The Z80 must also determine if a fast self-test is required. The Z80 determines this by reading its I/O location E8, the address of the Transaction Done Indicator register which also has BI STF(L) as an input. If a fast self-test is required, the Z80 performs a limited subset of the full self-test.

At the successful completion of self-test, the Z80 must generate a loopback WRITE transaction to the VAXBICSR to clear the Broke bit. The Z80 then deasserts BI BAD(L) and asserts BCI STPASS(L) by executing a write to its I/O address 35 with IOD0(H) asserted "H." Asserting BCI STPASS(L) is required to light the self-test LEDs.

Design Analysis Associates is an electrical engineering consulting firm based in Logan, Utah (75 West 100 South, Logan, Utah 84321). The firm specializes in the design of digital, analog, and software systems for both government and private industry and offers seminars on systematic design methods.

Chapter 3
VAXBI Module Layout Guide

by VAXBI Development Group

This chapter serves as a guide for engineers and module layout designers as they design options and build VAXBI modules. It points out problem areas and reports some of DIGITAL's experiences. References are made to the appropriate module control drawings.

Contents

(continued)

Contents (continued)

Overview of VAXBI Modules

VAXBI modules are new DIGITAL standard modules for VAXBI systems. These higher performance printed circuit boards incorporate new technology and techniques. This chapter describes VAXBI standard and expansion modules and presents design implications that arise from this new size printed circuit board.

Along with the *VAXBI System Reference Manual* and this notebook, you received the VAXBI Base Layout Package, which consists of databases and module control drawings. This chapter comments upon the drawings and gives guidelines for building modules, but it should not be used in place of the drawings. The module control drawings are the authoritative specifications. Make sure that you have the latest revision. This chapter refers to the following module control drawings:

- T1999 – Standard Module Control Drawing
- T1996 – Expansion Module Control Drawing
- ELEN 633 – Layup Specification
- ELEN 626 – Mechanical Outline Specification

Please read and study this chapter. It has been prepared to help engineers and module layout designers avoid practices that could lead to divergence from the specification or introduce problems into VAXBI systems. To assure compatibility, the VAXBI specifications describe several areas that have not been defined in the past:

- The bus interface, including components used in the VAXBI Corner and signal integrity characteristics of the module etch
- The module layup and layout of the bus interface area
- Length and capacitance restrictions on component interconnections
- Special routing for the VAXBI clock circuitry to ensure uniform clock timing

VAXBI Module Glossary

BIIC – Bus interconnect interface chip; DIGITAL's custom ZMOS chip (packaged in a 133 pin grid array) that serves as the VAXBI primary interface.

VAXBI Corner – Portion of a VAXBI module where the BIIC resides.

VAXBI module – Any printed circuit board mechanically compatible with VAXBI hardware. There are two types:

- **Standard VAXBI module** – Module with a VAXBI Corner, which conforms to these guidelines sufficiently.
- **VAXBI expansion module** – Module without a VAXBI Corner.

VAXBI node – A VAXBI interface that occupies one of sixteen logical locations on a VAXBI bus. A VAXBI node consists of one or more VAXBI modules.

VAXBI option – A VAXBI node. There are three types:

- **Single-board VAXBI option** – One standard VAXBI module.
- **Multiboard VAXBI option** – One standard VAXBI module plus one or more VAXBI expansion modules.
- **Remote VAXBI option** – A single- or multiboard VAXBI option plus some remote electronics not on VAXBI modules.

Boundary area – The strip between the VAXBI Corner and the user-configurable area of a VAXBI module.

Component side – Side of module on which the BIIC is mounted.

Connector area – A strip on the module edge reserved for the connector.

Rim – A strip around the three nonconnector module edges available to the user only for module lettering.

User-configurable area – Area of a VAXBI module not reserved for the VAXBI Corner, the connector area, or the rim.

Standard VAXBI Module Anatomy

Figure 3-1 shows the orientation of a VAXBI module as shown in the VAXBI SRM. The right edge of the module plugs into a VAXBI card cage. The module size was chosen so that the VAXBI and related hardware are acceptable to the European market. VAXBI card cages are compatible with Eurocages.

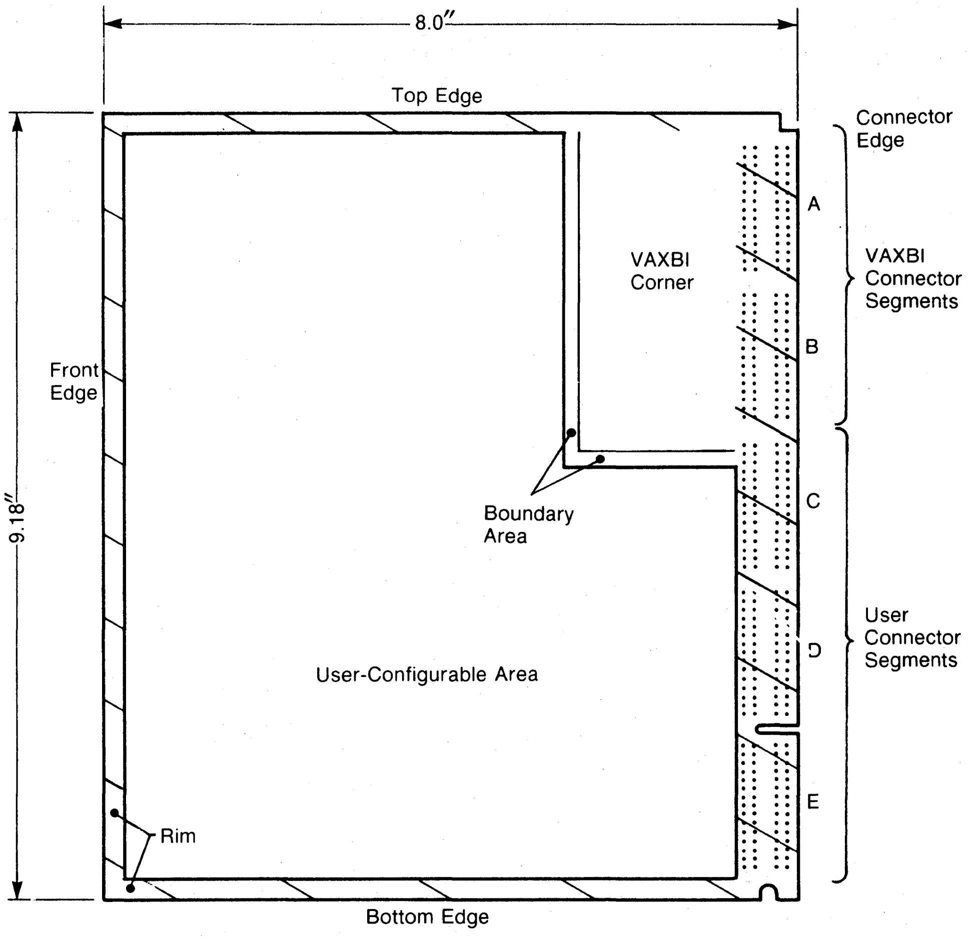


FIGURE 3-1 *VAXBI Module, Mechanical Orientation (component side)*

Module Layout Definitions

Figure 3-2 shows how a typical layout system views a standard VAXBI module. Layout systems view the component side of the board. The right edge of the board and all pin holes are on 0.1" centers. The left edge of the board is off all grids; however, the mechanical features are on grid with reference to the right edge. The A and B connector segment vias are off grid by 0.025" or 0.075". The user connector segments C, D, and E are on grid; however, the designer has the option to move these connector vias off grid to improve routability.

Most holes and features align on a 0.025" grid pattern (see ELEN 626 for exceptions).

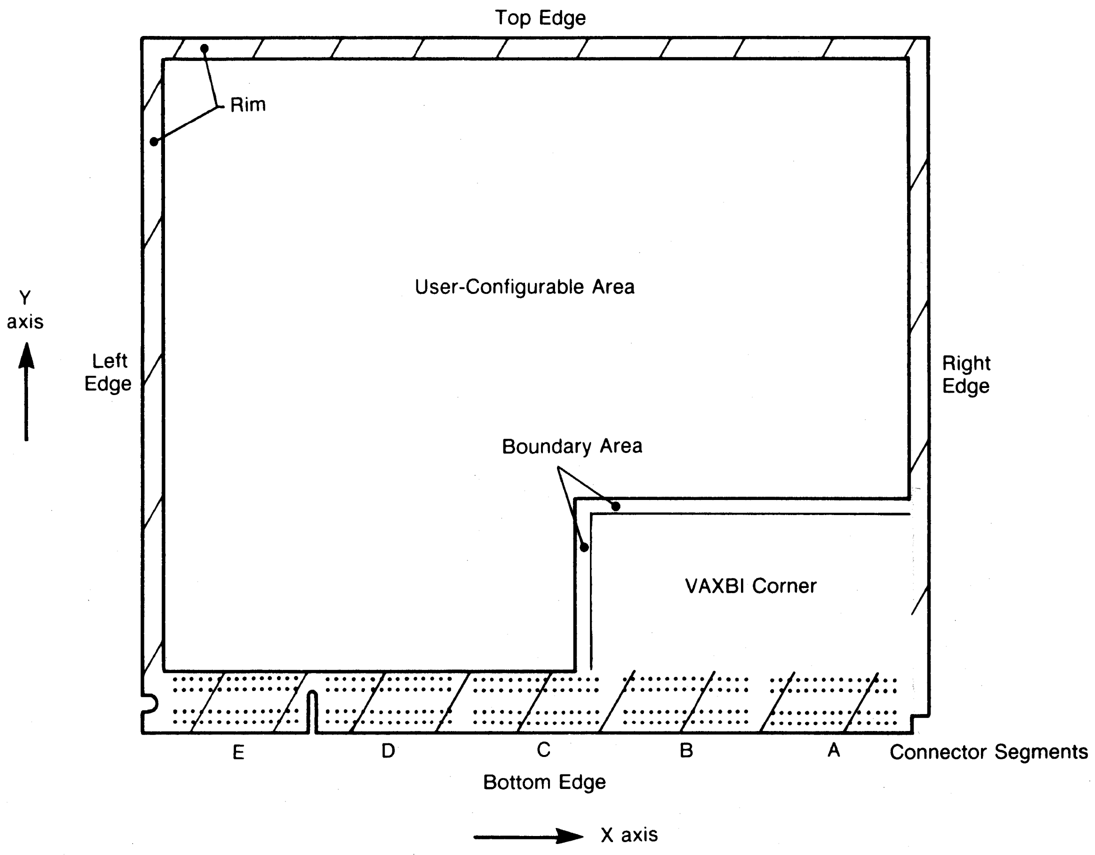
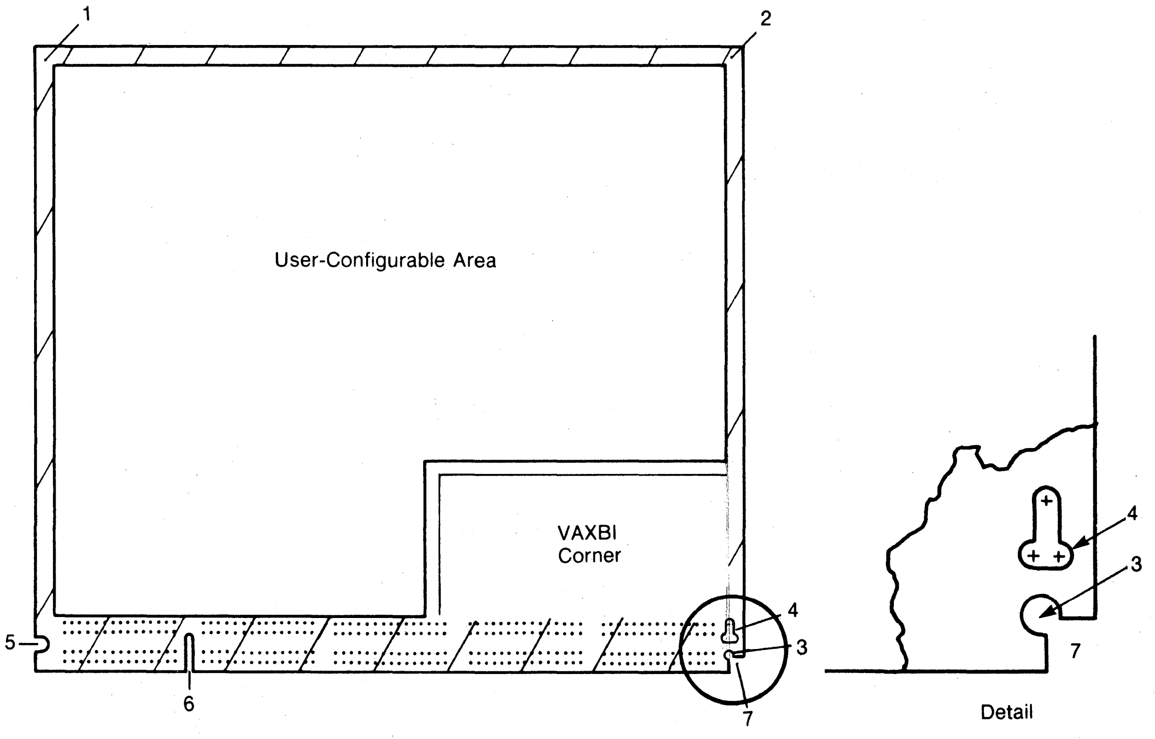


FIGURE 3-2 *Layout System View of Standard VAXBI Module (component side)*

Figure 3-3 shows several important profile and keying features located near the module edges. Details of the corner areas are shown on pp. 3-12 and 3-13. In addition to these physical requirements, the connector edge of VAXBI modules is beveled (see ELEN 626, sheet 2, view C-C).

A 50-mil clearance area (feature outline plus .050") around mechanical features should be incorporated in the power and ground layers (layers 4, 5, 6, and 7). The clearance area reduces the possibility of interlayer shorting when you install mechanical features such as keying shapes.



1,2,3 Drilled circular nonplated tooling holes (0.157″diameter)
4 Slot
5,6 Curved notches, routed
7 Cut out square notch, 0.2″on a side

FIGURE 3-3 *Module Profile and Keying Features*

Table 3-1 summarizes some of the important numbers from Module Control Drawing T1999.

Connector Area

A VAXBI connector contains five 60-pin segments in one 300-pin connector. These segments are called A through E. Segments A and B are for the VAXBI bus. The C, D, and E segments are for external connections. All connector pins contact gold fingers on the module surface layers (two rows of 15 pins in each segment on each side of the module).

The critical connector area of the module is 9.18" by 0.65". Component outlines must not be placed within 0.650" of the connector edge to provide clearance for the VAXBI edge connector. Figure 3-4 shows the required component clearance. (See T1999 or T1996.)

Table 3-1 Summary of Configuration Details from T1999

Item	Description
Area (nominal on one layer)	73.44 square inches (8.0" X 9.18")
Etch signal runs	Primarily internal
Layers	10 (2 power, 2 ground, 2 cap, 4 signal)
Power dissipation	50 watts per slot, maximum acceptable (3-board option, 150 watts max.)
Slots in each VAXBI cage	6, on 0.8" centers, guides 0.2" deep, max.
Standard PWB material	FR-4 (dielectric constant = 4.1 to 4.9, 4.2 typical)
Thickness after lamination	0.083" to 0.093"
Thickness after plating and etching	0.103" max. over gold fingers, 0.093" plus or minus 10% elsewhere
Tolerance on board dimensions	7.995" to 8.007" X 9.175" to 9.187"
Maximum warpage	0.005" per in. (within 0.65" of connector)

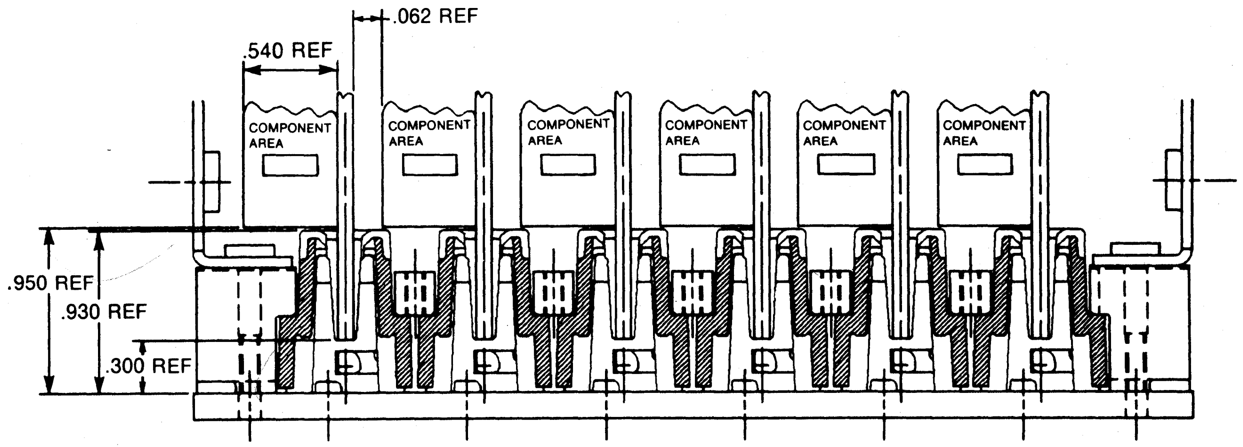


FIGURE 3-4 Clearance Required for Connector

Figure 3-5 shows details of a module connector segment before the plating bar is severed. Two rows of .080"-square gold finger tabs are required on each side of VAXBI modules. One row is centered 0.3" above the connector edge of the module, and the other is centered 0.4" above. Each gold tab center is 0.1" to the right of the last. The gold tabs in the lower row are staggered 0.05" to the left of those in the upper row (see Figure 3-6).

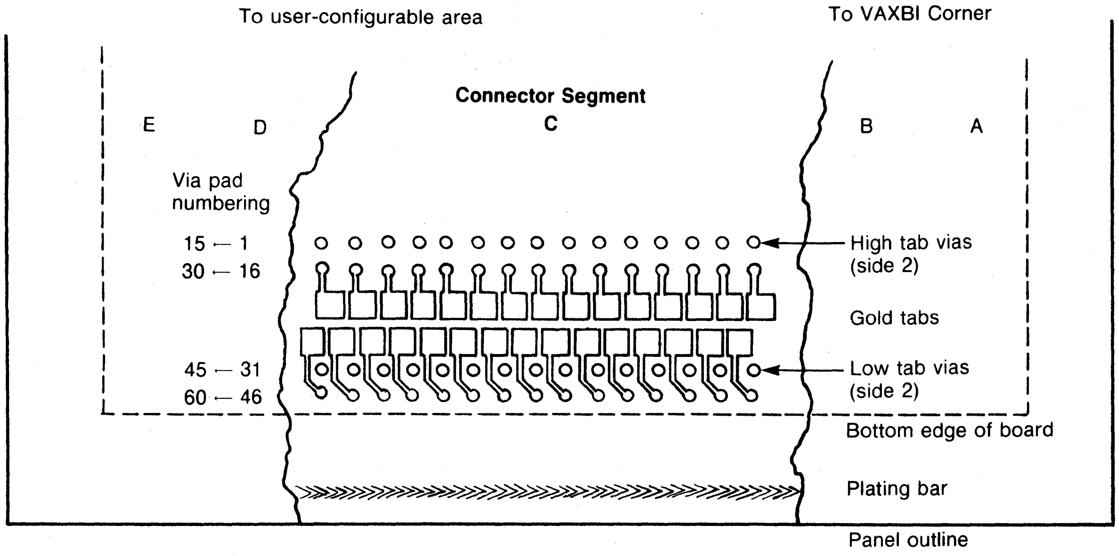


FIGURE 3-5 *Connector Segment*

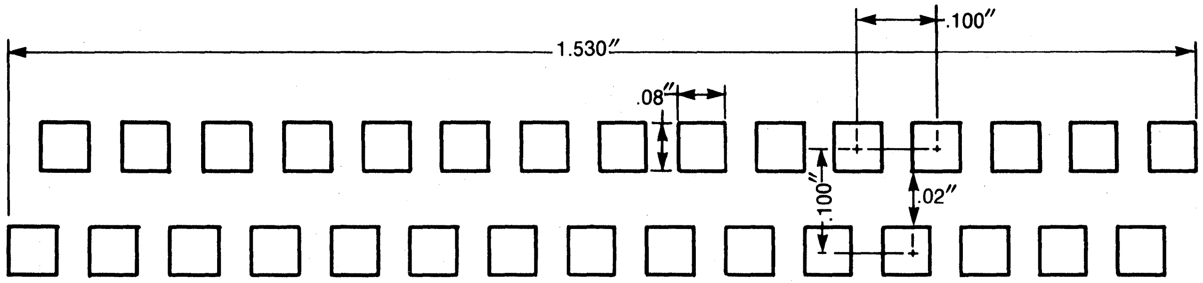


FIGURE 3-6 *Dimensions of Gold Finger Tabs*

Gold plating also covers the four rows of vias above and below the gold fingers on both sides of the board. Each gold finger is prerouted to a single via pad above or below using 0.015" etch on layers 1 and 10. Each via pad is also prerouted internally in 5- or 6-mil etch to a plating via outside the circuit outline (see Figure 3-7). (Layout was done with 6-mil etch, but you can use 5-mil etch.) The plating vias are removed after gold deposition by profiling the board along the bottom line, but the thin plating etch remains on the signal layers, up to the circuit edge. The layout designer must avoid the plating bar connections in the I/O segments when making connections to the via pads from the user-configurable area.

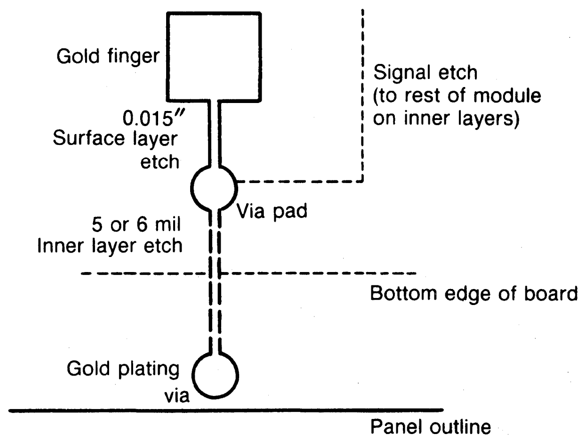

Gold finger
0.015" Surface layer etch
Signal etch (to rest of module on inner layers)
Via pad
5 or 6 mil Inner layer etch
Bottom edge of board
Gold plating via
Panel outline

FIGURE 3-7 *Via Pad and Gold-Plating Via*

Manufacturing Standard VAXBI Plating Feature

DIGITAL manufacturing normally adds the plating feature to the module. In the customer layout package the plating feature has been added to the Gerber plot tapes for ease of implementation. We recommend that you use this feature, but you can remove it and implement your own for segments C, D, and E. *You must not change the routing or layer assignment of the etch on connector segments A and B.* Note that the gold plating covers the four rows of vias.

See ELEN 626 for the finger detail and circuit outline.

I/O Connector Segments C, D, and E

Connector segments C, D, and E are available to the user. You must not omit the tabs in the I/O segments; doing so could damage or contaminate the VAXBI connector. Do not connect to them if they are unused. All tabs and vias in the I/O segments must be gold plated.

Connector Segments A and B

The VAXBI connector segments A and B are not configurable by the user. These two segments appear similar to the pattern provided in the I/O segments, but they are modified slightly for RLC purposes. In the A and B segments, the via pads do not conform to the regular pattern of the other segments. Some gold fingers have been deliberately shorted together. (See the finger detail drawing [D-UA-T1999-00].) Care has been taken to ensure uniform copper metalization on all layers, especially power and ground. The ground and power planes have been custom tailored to form properly around the critical VAXBI signals.

Refer to the module control drawings to see the detailed layout on the 10 layers. Note the custom gold finger shapes, the custom routing to the connection via pads and plating bar, and the layer assignment of connections.

All VAXBI modules must have gold finger tabs for all 300 pins on the connector, whether those pins are used or not. This is to prevent buildup of debris on the connector, which could occur if the connector made contact with unplated areas of VAXBI modules. Table 3-2 lists the connector pins in segments A and B of VAXBI modules.

Table 3-2 Connector Segments A and B Signal Names

Pin	Signal Name	Pin	Signal Name
A01	BI D29 L	A31	GND
A02	BI D28 L	A32	PASS THRU*
A03	GND	A33	5VBB
A04	BI D27 L	A34	5VBB
A05	BI D25 L	A35	5VBB
A06	BI D23 L	A36	GND
A07	BI D21 L	A37	PASS THRU
A08	BI D19 L	A38	GND
A09	BI D15 L	A39	PASS THRU
A10	BI D24 L	A40	GND
A11	BI D13 L	A41	PASS THRU
A12	BI D11 L	A42	GND
A13	+5.0V	A43	+5.0V
A14	BI D07 L	A44	+5.0V
A15	BI D06 L	A45	BI D08 L
A16	BI D31 L	A46	GND
A17	BI D30 L	A47	5VBB
A18	GND	A48	5VBB
A19	GND	A49	GND
A20	BI D26 L	A50	GND
A21	BI D22 L	A51	5VBB
A22	BI D20 L	A52	PASS THRU
A23	BI D18 L	A53	GND
A24	BI D17 L	A54	PASS THRU
A25	BI D14 L	A55	GND
A26	BI D12 L	A56	BI SPARE L
A27	BI D10 L	A57	GND
A28	+5.0V	A58	+5.0V
A29	+5.0V	A59	GND
A30	BI D16 L	A60	BI D03 L
B01	BI D00 L	B31	BI D02 L
B02	BI P0 L	B32	GND
B03	BI I1 L	B33	BI ID2 H
B04	BI CNF2 L	B34	GND
B05	BI BSY L	B35	BI ID0 H
B06	BI NO ARB L	B36	BI STF L
B07	BI I3 L	B37	−12.0V
B08	−5.2V	B38	−2.0V
B09	BI DC LO L	B39	−2.0V
B10	GND	B40	BI AC LO L
B11	GND	B41	BI TIME − L

Table 3-2 Connector Segments A and B Signal Names (Continued)

Pin	Signal Name	Pin	Signal Name
B12	GND	B42	BI PHASE − L
B13	GND	B43	GND
B14	Module cannot use*	B44	GND
B15	Module cannot use	B45	Module cannot use
B16	BI D04 L	B46	BI D05 L
B17	BI D01 L	B47	BI ID3 H
B18	BI I2 L	B48	GND
B19	BI I0 L	B49	BI ID1 H
B20	BI CNF1 L	B50	+12.0V
B21	BI D09 L	B51	BI BAD L
B22	BI CNF0 L	B52	GND
B23	−5.2V	B53	−2.0V
B24	−5.2V	B54	BI RESET L
B25	BI ECL VCC H	B55	GND
B26	BI TIME + H	B56	GND
B27	BI PHASE + H	B57	GND
B28	GND	B58	GND
B29	GND	B59	Module cannot use
B30	Module cannot use	B60	Module cannot use

* Signals labeled "PASS THRU" and "Module cannot use" are reserved for future use by DIGITAL.

Module Rim

On VAXBI modules a rim of 0.2" is reserved for the card guides in the VAXBI card cage. No component outline or overhang, no component pin or connection, and no pad for a pin or via may enter the rim. Devices may be placed tangential to the rim. If staying on a 0.1" grid for pin centers, however, it is necessary to stay 0.3" (0.28" on the left-hand side) from the edge of VAXBI modules. Because of the 0.2" rim restriction, vias must be 0.3" from the edge of the board.

The rim contains two ground pads, one at the left edge and one at the top edge, for use with electrostatic discharge (ESD) clips. The ESD pads are located in the rim on layer 1 and measure 0.3" by 0.14". Each ESD pad has a via hole which connects to a long etch run on layer 2 all the way around the edge of the module to the VAXBI Corner.

Layers 1 and 10 of the module rim are available for lettering and symbology (see Figure 3-8). The standard features like UL block and the DIGITAL logo appear in the rim on side 1. A bar code can be attached along the top rim to show the plant code and serial number in both human and machine-readable form. No etch should be placed in the rim area on layers 1 and 10.

In the user-configurable area, user lettering is permitted only on layers 1 and 10. Lettering can include component reference designators, pin one markers, and other identifiers useful to the designer or assembler. No additional lettering is permitted in the VAXBI Corner area.

VAXBI modules do not have handles.

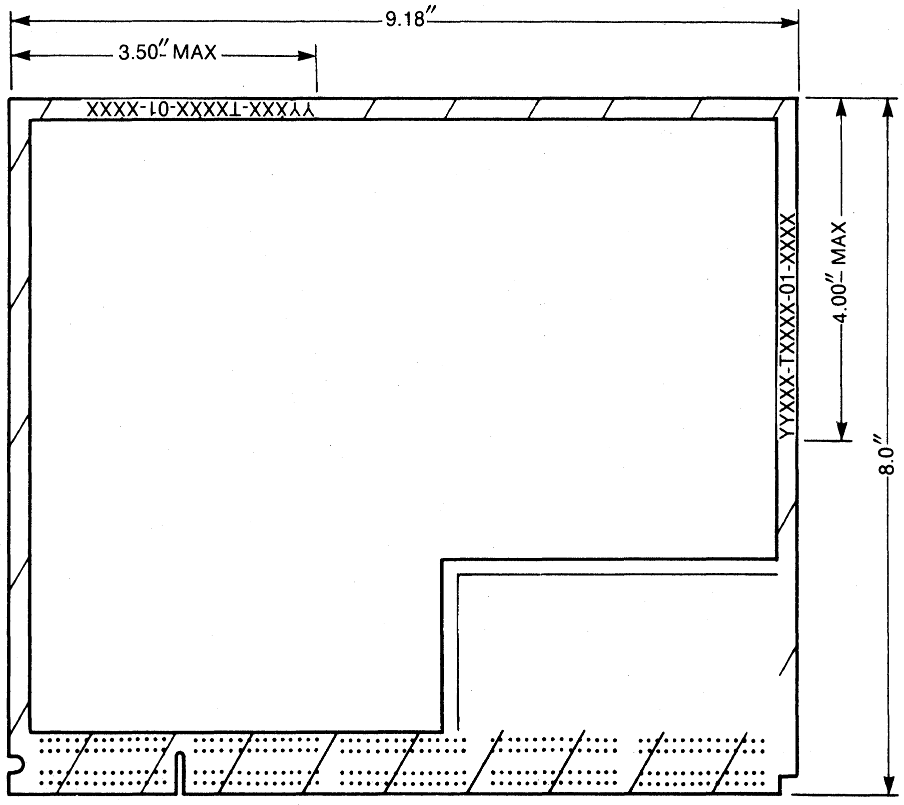


FIGURE 3-8 Nomenclature and Lettering for a VAXBI Module

VAXBI modules usually do not have etch on internal layers in the rim, but routing in the rim is a minor variation from standard if care is taken to avoid the features already residing there:

- Layer 9 has a run for use with the LED.
- Layer 2 has a run for the ESD pads.

Figures 3-9 through 3-12 show details of each corner of a VAXBI module.

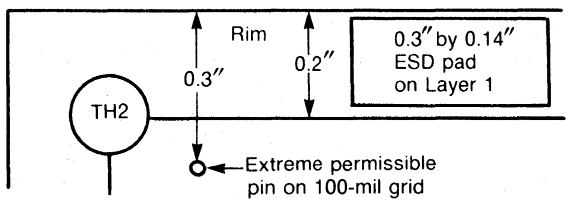


FIGURE 3-9 Upper Left-Hand Corner Details

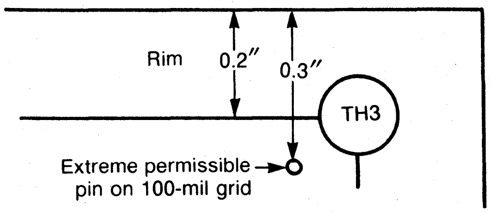


FIGURE 3-10 Upper Right-Hand Corner Details

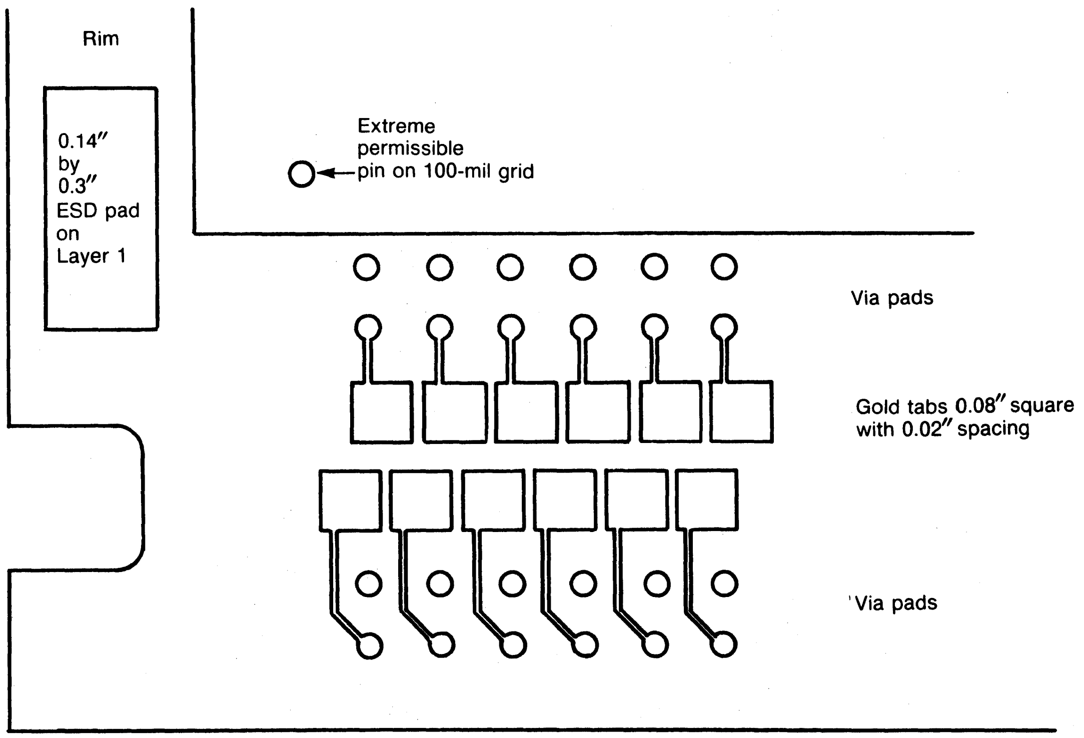

Rim
0.14″ by 0.3″ ESD pad on Layer 1
Extreme permissible pin on 100-mil grid
Via pads
Gold tabs 0.08″ square with 0.02″ spacing
Via pads

FIGURE 3-11 *Lower Left-Hand Corner Details, Connector Segment E*

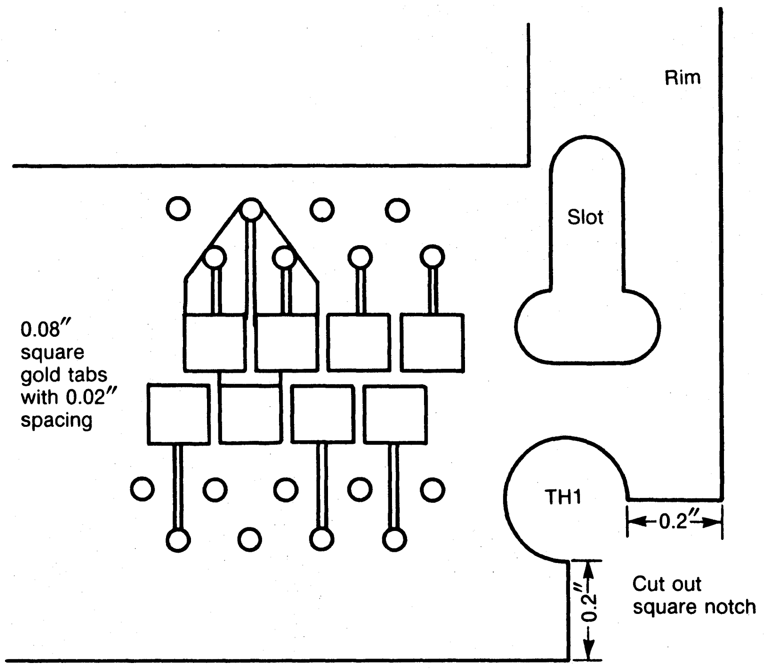


FIGURE 3-12 *Lower Right-Hand Corner Details, VAXBI Corner*

Electrical Characteristics

Table 3-3 shows the overall electrical limits for a VAXBI module. The standard VAXBI module uses up to six different voltages including 5VBB (battery backup). However, 50 watts is the total combined power input limit per board or board slot.

Voltage Drop

The voltage drop across a VAXBI module is limited for each supply to the worst point. Table 3-3 gives the permissible IR drop for each supply. This specification has been met for prototypes using 2-oz. copper power and ground planes. However, care should be taken to leave sufficient metal on these layers to allow good power distribution. Large pin grid arrays with large pin clearance areas can be a problem, particularly if accompanied by heat-relief pads that take considerable metal around power pins. If there may be a problem, the designer should try to design various clearances down in size. (For example, note the downsizing around the BIIC.)

Be careful to maintain minimum drill size for power and ground pins; see T1999 for details.

Impedance

Impedance of etch runs on VAXBI modules is unspecified. Normally, TTL-compatible impedances are assumed. If you design VAXBI modules that require other impedances, then the board layup, the line widths, or other board features can be modified to remain within the specifications.

Table 3-3 Overall Electrical Limits

Voltage	Maximum Current (amps)	Maximum Power (watts)	Maximum IR Drop (millivolts)
Ground	10.0	N.A.	15
+5.0V	10.0	50	20
5VBB	7.0	35	20
−5.2V	5.0	26	20
−2.0V	3.3	6.6	15
+12.0V	1.0	12	15
−12.0V	1.0	12	15

Standard VAXBI Module and the VAXBI Corner

The VAXBI Corner rectangle is an absolute barrier to placement and routing by the user. The user should think only of connecting to the double row of connection points in the VAXBI boundary area. The only exceptions to this rule are the power plane splits as described on page 3-28.

The VAXBI Corner revision is etched on layer 1, and the pad pattern on layer 1 is the hole pattern for the Corner. This pattern is not necessarily repeated on other layers, where holes may be unaccompanied by pads for capacitance reasons.

VAXBI Corner Parts

Table 3-4 lists the VAXBI Corner parts, and Figure 3-13 shows the VAXBI Corner component layout. Component C995 extends out of the Corner area and blocks placement of devices in a rectangle next to the boundary area (see Figure 3-13), but this does not affect routing and applies only to designs using −2.0 volts. Otherwise, C995 is depopulated and can be ignored. The component pins for C995 are in the BCI connector boundary at pins 66 and 74.

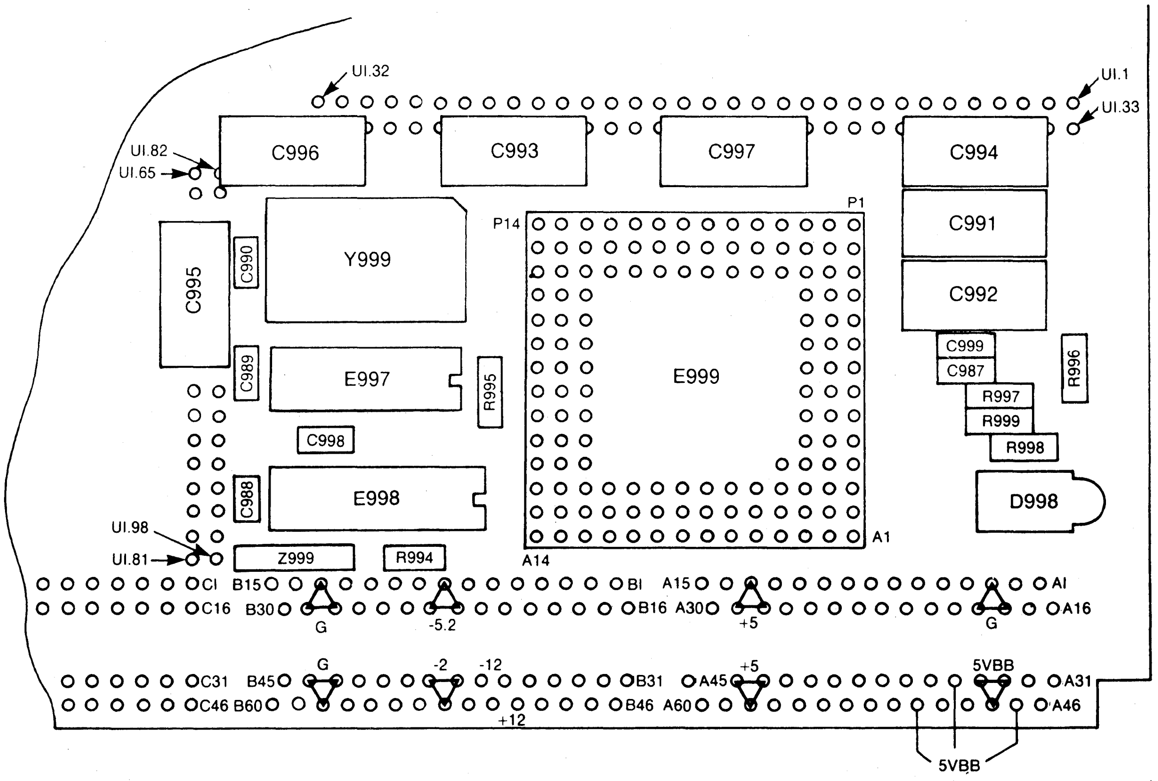


FIGURE 3-13 *VAXBI Corner Placement*

Table 3-4 VAXBI Corner Parts List

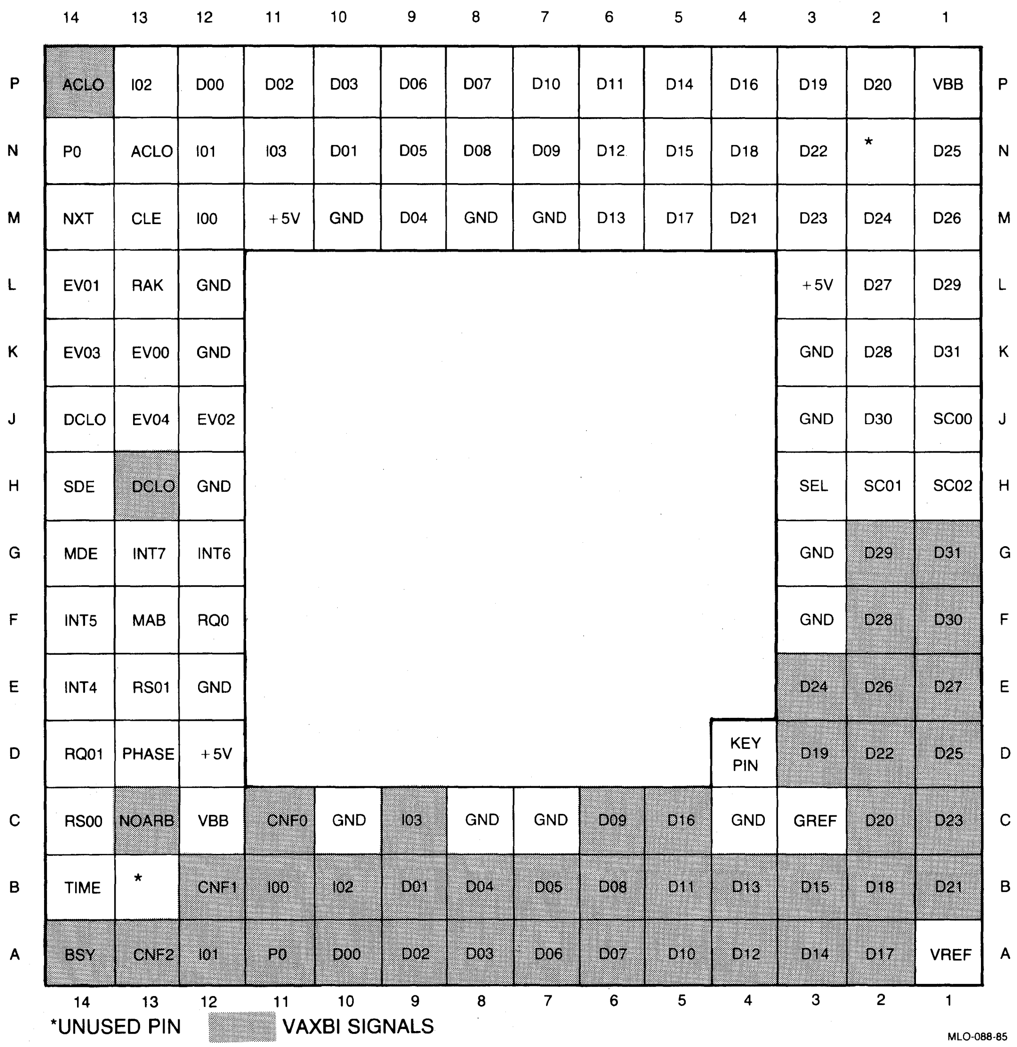
Reference Designator	Description	Notes
C987	0.047 uF, 50V, cap.	+5V
C988	0.047 uF, 50V, cap.	+5V
C989	0.047 uF, 50V, cap.	+5V
C990	0.047 uF, 50V, cap.	+5V
C991	8.0 uF, 25V, cap.	+5V
C992	8.0 uF, 25V, cap.	+5V
C999	0.01 uF, 50V, cap.	BIIC VBB
C996	8.0 uF, 25V, cap.	−12V; Note 1
C997	8.0 uF, 25V, cap.	−5.2V; Note 1
C995	8.0 uF, 25V, cap.	−2V; Note 1
C994	8.0 uF, 25V, cap.	5VBB; Note 1
C993	8.0 uF, 25V, cap.	+12V; Note 1
C998	0.047 uF, 50V, cap.	ECL Vcc; Note 2
E999	78732 BIIC	
E998	78702 BI Clock Receiver	
E997	78701 BI Clock Driver	Note 2
Y999	40 MHz crystal oscillator	Note 2
D998	Yellow LED	Self-test passed; Note 3
R998	25.5 Kohm, 1/4W, 1%	BIIC Vcc reference; Note 4
R999	3.83 Kohm, 1/4W, 1%	BIIC ground reference; Note 4
Z999	1.0 Kohm, 1/8W, 5% SIP	ID pullups; may be out of the corner
R995	1.0 Kohm, 1/4W, 5%	BCI DC LO pulldown
R996	1.0 Kohm, 1/4W, 5%	ESD pad discharge resistor
R997	270 ohm, 1/4W, 5%	STPASS LED; Note 3
R994	470 ohm, 1/4W, 5%	Clock disable sense; Note 2

NOTES

1. Capacitors for unused voltages can be omitted.
2. The part can be omitted in VAXBI modules that never drive the clock signals.
3. D998, R997, and D999 (which is not in the VAXBI Corner) are required on all VAXBI modules.
4. The maximum temperature coefficient is 100 PPM/degrees C.

The custom ZMOS BIIC pins are labeled by row and column as shown in Figure 3-14. A keying pin at D4 ensures proper insertion. For the BIIC, we recommend the use of in-situ socketing devices because they can fit many package shapes, are very low profile, and especially because they have negligible capacitance and inductance.

For example, the Augat Holtite (Augat P/N 8134-HC-5P2 or P3) is commercially available in tin or in gold. The Mark Eyelet socket has also been qualified for use in the VAXBI Corner. The Mark Eyelet holes are .040" +.003", −.003". The Augat Holtite requires .041" +.002", −.002". This leads to an incompatibility which has been resolved by specifying the holes as .041" +.002", −.003". If, however, the PCB finished hole size is .044", you can waiver the hole size and use oversize Holtites. No similar recovery process is currently available for Mark Eyelet sockets. The Augat part is a press-fit component, whereas the Mark Eyelet part is soldered.



	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
P	▨ ACLO	I02	D00	D02	D03	D06	D07	D10	D11	D14	D16	D19	D20	VBB	P
N	P0	ACLO	I01	I03	D01	D05	D08	D09	D12	D15	D18	D22	*	D25	N
M	NXT	CLE	I00	+5V	GND	D04	GND	GND	D13	D17	D21	D23	D24	D26	M
L	EV01	RAK	GND									+5V	D27	D29	L
K	EV03	EV00	GND									GND	D28	D31	K
J	DCLO	EV04	EV02									GND	D30	SC00	J
H	SDE	▨ DCLO	GND									SEL	SC01	SC02	H
G	MDE	INT7	INT6									GND	▨ D29	▨ D31	G
F	INT5	MAB	RQ0									GND	▨ D28	▨ D30	F
E	INT4	RS01	GND									▨ D24	▨ D26	▨ D27	E
D	RQ01	PHASE	+5V								KEY PIN	▨ D19	▨ D22	▨ D25	D
C	RS00	▨ NOARB	VBB	▨ CNF0	GND	▨ I03	GND	GND	▨ D09	▨ D16	GND	GREF	▨ D20	▨ D23	C
B	TIME	*	▨ CNF1	▨ I00	▨ I02	▨ D01	▨ D04	▨ D05	▨ D08	▨ D11	▨ D13	▨ D15	▨ D18	▨ D21	B
A	▨ BSY	▨ CNF2	▨ I01	▨ P0	▨ D00	▨ D02	▨ D03	▨ D06	▨ D07	▨ D10	▨ D12	▨ D14	▨ D17	VREF	A
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	

*UNUSED PIN ▨ VAXBI SIGNALS

MLO-088-85

FIGURE 3-14 *BIIC Pin Grid Array (top view)*

VAXBI Corner Etch

It is advisable to use the design width specified for etch. Because the VAXBI Corner is sensitive to capacitance, it is undesirable to use etch wider than the design width. On the other hand, the danger in thinner etch is that the percent variation rises, which leads to a range of values for IR drop and capacitance, and may not produce uniform metalization in some manufacturing processes. Not enough is known to specify a tolerance, but it should be as fine as the manufacturing process can normally use.

One exception to maintaining the design width of etch are the lines to the plating vias. In principle, they could be as small as 0.001". DIGITAL found the 0.006" width to be the smallest practical design width consistent with a subtractive, wet chemical etch process for circuit fabrication.

In the VAXBI Corner two etch lines must never pass between the same pair of pads on the same layer. A wide variety of angles can be employed. In some cases etch patterns for more than one line have been handcrafted to produce identical skews. Signal vias are not permitted in the Corner; however, signal vias can be in the connector and boundary areas of the Corner.

VAXBI Corner Connectivity

Table 3-5 gives the connectivity for the VAXBI Corner. For signals on the BCI side of the BIIC the table indicates whether connections are required, prohibited, or optional.

Components in the VAXBI Corner are identified below:

Package	**Description**
E999	78732 BII
E998	78702 Clock Receiver
E997	78701 Clock Driver
Z999	SIP, 1.0 Kohm
R997	STPASS LED, 270 ohms
R994	Clock disable sense, 470 ohms

Table 3-5 Corner Connectivity

Signal Name	Package and Pin No.	Module Connector Pin No.	Module BCI I/O Pin No.	Usage Code
BI D30 L	E999,F-1	A-17	–	N/A
BI D29 L	E999,G-2	A-01	–	N/A
GND	E999,G-3/F-3	@	–	N/A
BI D28 L	E999,F-2	A-02	–	N/A
BI D27 L	E999,E-1	A-04	–	N/A
BI D26 L	E999,E-2	A-20	–	N/A
BI D25 L	E999,D-1	A-05	–	N/A
BI D24 L	E999,E-3	A-10	–	N/A
BI D23 L	E999,C-1	A-06	–	N/A
BI D22 L	E999,D-2	A-21	–	N/A
BI D21 L	E999,B-1	A-07	–	N/A
BI D20 L	E999,C-2	A-22	–	N/A
BI D19 L	E999,D-3	A-08	–	N/A
BCI VCCREF H	E999,A-1	–	–	N/A
BCI GNDREF H	E999,C-3	–	–	N/A
BI D18 L	E999,B-2	A-23	–	N/A
BI D17 L	E999,A-2	A-24	–	N/A
GND	E999,C-4	@	–	N/A
BI D16 L	E999,C-5	A-30	–	N/A
BI D15 L	E999,B-3	A-09	–	N/A
BI D14 L	E999,A-3	A-25	–	N/A
BI D13 L	E999,B-4	A-11	–	N/A
BI D12 L	E999,A-4	A-26	–	N/A
BI D11 L	E999,B-5	A-12	–	N/A
BI D10 L	E999,A-5	A-27	–	N/A
BI D09 L	E999,C-6	B-21	–	N/A
BI D08 L	E999,B-6	A-45	–	N/A
BI D07 L	E999,A-6	A-14	–	N/A
BI D06 L	E999,A-7	A-15	–	N/A
BI D05 L	E999,B-7	B-46	–	N/A
GND	E999,C-7,C-8	@	–	N/A
BI D04 L	E999,B-8	B-16	–	N/A
BI D03 L	E999,A-8	A-60	–	N/A
BI D02 L	E999,A-9	B-31	–	N/A
BI D01 L	E999,B-9	B-17	–	N/A
BI D00 L	E999,A-10	B-01	–	N/A
BI P0 L	E999,A-11	B-02	–	N/A
BI I3 L	E999,C-9	B-07	–	N/A
BI I2 L	E999,B-10	B-18	–	N/A
BI I1 L	E999,A-12	B-03	–	N/A
BI I0 L	E999,B-11	B-19	–	N/A
GND	E999,C-10	@	–	N/A
BI CNF2 L	E999,A-13	B-04	–	N/A
BI CNF1 L	E999,B-12	B-20	–	N/A
BI CNF0 L	E999,C-11	B-22	–	N/A

Table 3-5 Corner Connectivity (Continued)

Signal Name	Package and Pin No.	Module Connector Pin No.	Module BCI I/O Pin No.	Usage Code
N/C (E999-B13)	E999,B-13	–	U1.57	X
VBB GEN L	E999,C-12	–	–	N/A
+5.0 (VDD)	E999,D-12	–	–	N/A
BI BSY L	E999,A-14	B-05	–	N/A
BI NO ARB L	E999,C-13	B-06	–	N/A
BCI TIME L	E999,B-14	–	–	N/A
GND	E999,E-12	@	–	N/A
BCI PHASE L	E999,D-13	–	–	N/A
BCI RS0 L	E999,C-14	–	U1.93	R
BCI RS1 L	E999,E-13	–	U1.83	R
BCI RQ0 L	E999,F-12	–	U1.92	R
BCI RQ1 L	E999,D-14	–	U1.69	R
BCI MAB L	E999,F-13	–	U1.84	R
BCI INT4 L	E999,E-14	–	U1.68	R
BCI INT5 L	E999,F-14	–	U1.67	R
BCI INT6 L	E999,G-12	–	U1.71	R
BCI INT7 L	E999,G-13	–	U1.73	R
BCI MDE L	E999,G-14	–	U1.91	R
BCI SDE L	E999,H-14	–	U1.72	O
GND	E999,H-12	@	–	N/A
BI DC LO L	E999,H-13	B-09	U1.53	O
BCI DC LO L	E999,J-14	–	U1.65	R
BCI EV4 L	E999,J-13	–	U1.82	R
BCI EV3 L	E999,K-14	–	U1.31	R
BCI EV2 L	E999,J-12	–	U1.58	R
BCI EV1 L	E999,L-14	–	U1.60	R
BCI EV0 L	E999,K-13	–	U1.30	R
BCI NXT L	E999,M-14	–	U1.59	R
BCI RAK L	E999,L-13	–	U1.28	R
GND	E999,K-12,L-12	@	–	N/A
BCI CLE H	E999,M-13	–	U1.27	R
BCI P0 H	E999,N-14	–	U1.64	O
BI AC LO L	E999,P-14	B-40	U1.24	O
BCI AC LO L	E999,N-13	–	U1.32	O
+5.0V (PWR)	E999,M-11	–	–	N/A
BCI I0 H	E999,M-12	–	U1.26	R
GND (VSS)	E999,M-10	@	–	N/A
BCI I1 H	E999,N-12	–	U1.62	R
BCI I2 H	E999,P-13	–	U1.63	R
BCI I3 H	E999,N-11	–	U1.29	R
BCI D00 H	E999,P-12	–	U1.61	R
BCI D01 H	E999,N-10	–	U1.23	R
BCI D02 H	E999,P-11	–	U1.56	R
BCI D03 H	E999,P-10	–	U1.55	R
BCI D04 H	E999,M-9	–	U1.25	R

Table 3-5 Corner Connectivity (Continued)

Signal Name	Package and Pin No.	Module Connector Pin No.	Module BCI I/O Pin No.	Usage Code
BCI D05 H	E999,N-9	–	U1.22	R
BCI D06 H	E999,P-9	–	U1.20	R
BCI D07 H	E999,P-8	–	U1.51	R
BCI D08 H	E999,N-8	–	U1.52	R
GND	E999,M-7,M-8	@	–	N/A
BCI D09 H	E999,N-7	–	U1.19	R
BCI D10 H	E999,P-7	–	U1.50	R
BCI D11 H	E999,P-6	–	U1.49	R
BCI D12 H	E999,N-6	–	U1.18	R
BCI D13 H	E999,M-6	–	U1.21	R
BCI D14 H	E999,P-5	–	U1.48	R
BCI D15 H	E999,N-5	–	U1.17	R
BCI D16 H	E999,P-4	–	U1.47	R
BCI D17 H	E999,M-5	–	U1.45	R
BCI D18 H	E999,N-4	–	U1.16	R
BCI D19 H	E999,P-3	–	U1.46	R
BCI D20 H	E999,P-2	–	U1.13	R
BCI D21 H	E999,M-4	–	U1.44	R
BCI D22 H	E999,N-3	–	U1.15	R
N/C (E999-N2)	E999,N-2	–	U1.14	R
VBB GEN L	E999,P-1	–	–	N/A
+5.0V (VDD)	E999,L-3	–	–	N/A
BCI D23 H	E999,M-3	–	U1.43	R
GND	E999,K-3	@	–	N/A
BCI D24 H	E999,M-2	–	U1.10	R
GND (VSS)	E999,J-3	@	–	N/A
BCI D25 H	E999,N-1	–	U1.11	R
BCI D26 H	E999,M-1	–	U1.12	R
BCI D27 H	E999,L-2	–	U1.42	R
BCI D28 H	E999,K-2	–	U1.8	R
BCI D29 H	E999,L-1	–	U1.41	R
BCI D30 H	E999,J-2	–	U1.7	R
BCI D31 H	E999,K-1	–	U1.9	R
BCI SEL L	E999,H-3	–	U1.39	O
BCI SC0 L	E999,J-1	–	U1.40	O
BCI SC1 L	E999,H-2	–	U1.6	O
BCI SC2 L	E999,H-1	–	U1.38	O
BI D31 L	E999,G-1	A-16	–	N/A
None	E999,D-4	(Keying Pin)	–	N/A
BI ID0 H	Z999,5	B-35	U1.98	R
BI ID1 H	Z999,4	B-49	U1.80	R
BI ID2 H	Z999,3	B-33	U1.97	R
BI ID3 H	Z999,2	B-47	U1.79	R
BI STF L	–	B-36	U1.86	O
BI BAD L	–	B-51	U1.70	R

Table 3-5 Corner Connectivity (Continued)

Signal Name	Package and Pin No.	Module Connector Pin No.	Module BCI I/O Pin No.	Usage Code
BI SPARE L	–	A-56	U1.37	X
BI TIME + H	E998,3	B-26	–	N/A
BI TIME – L	E998,4	B-41	–	N/A
BI PHASE + H	E998,5	B-27	–	N/A
BI PHASE – L	E998,6	B-42	–	N/A
BI ECL VCC H	–	B-25	–	N/A
BCI CK DIS L	R994,1	–	U1.81	O
BI RESET L	–	B-54	U1.94	O
MOD GND PLANES	–	A-03	U1.66	O
MOD GND PLANES	–	A-18	–	N/A
MOD GND PLANES	–	A-19	–	N/A
MOD GND PLANES	–	A-31	–	N/A
MOD GND PLANES	–	A-36	–	N/A
MOD GND PLANES	–	A-38	–	N/A
MOD GND PLANES	–	A-40	–	N/A
MOD GND PLANES	–	A-42	–	N/A
MOD GND PLANES	–	A-46	–	N/A
MOD GND PLANES	–	A-49	–	N/A
MOD GND PLANES	–	A-50	–	N/A
MOD GND PLANES	–	A-53	–	N/A
MOD GND PLANES	–	A-55	–	N/A
MOD GND PLANES	–	A-57	–	N/A
MOD GND PLANES	–	A-59	–	N/A
MOD GND PLANES	–	B-10	–	N/A
MOD GND PLANES	–	B-11	–	N/A
MOD GND PLANES	–	B-12	–	N/A
MOD GND PLANES	–	B-13	–	N/A
MOD GND PLANES	–	B-28	–	N/A
MOD GND PLANES	–	B-29	–	N/A
MOD GND PLANES	–	B-32	–	N/A
MOD GND PLANES	–	B-34	–	N/A
MOD GND PLANES	–	B-43	–	N/A
MOD GND PLANES	–	B-44	–	N/A
MOD GND PLANES	–	B-48	–	N/A
MOD GND PLANES	–	B-52	–	N/A
MOD GND PLANES	–	B-55	–	N/A
MOD GND PLANES	–	B-56	–	N/A
MOD GND PLANES	–	B-57	–	N/A
MOD GND PLANES	–	B-58	–	N/A
MOD PWR +5.0V	–	A-13	–	N/A
MOD PWR +5.0V	–	A-28	–	N/A
MOD PWR +5.0V	–	A-29	–	N/A
MOD PWR +5.0V	–	A-43		N/A
MOD PWR +5.0V	–	A-44	–	N/A
MOD PWR +5.0V	–	A-58	–	N/A

Table 3-5 Corner Connectivity (Continued)

Signal Name	Package and Pin No.	Module Connector Pin No.	Module BCI I/O Pin No.	Usage Code
MOD PWR 5VBB	–	A-33	U1.1	O
MOD PWR 5VBB	–	A-34	–	N/A
MOD PWR 5VBB	–	A-48	–	N/A
MOD PWR 5VBB	–	A-35	–	N/A
MOD PWR 5VBB	–	A-47	–	N/A
MOD PWR 5VBB	–	A-51	U1.34	O
MOD PWR −5.2V	–	B-08	–	N/A
MOD PWR −5.2V	–	B-23	–	N/A
MOD PWR −5.2V	–	B-24	–	N/A
MOD PWR −2.0V	–	B-38	U1.74	O
MOD PWR −2.0V	–	B-39	–	N/A
MOD PWR −2.0V	–	B-53	–	N/A
MOD PWR +12.0V	–	B-50	U1.85	O
MOD PWR −12.0V	–	B-37	U1.87	O
Reserved	–	B-14	–	N/A
Reserved	–	B-15	–	N/A
Reserved	–	B-30	–	N/A
Reserved	–	B-45	–	N/A
Reserved	–	B-59	–	N/A
Reserved	–	B-60	–	N/A
Reserved	–	A-32	U1.33	X
Reserved	–	A-37	U1.2	X
Reserved	–	A-39	U1.3	X
Reserved	–	A-41	U1.4	X
Reserved	–	A-52	U1.35	X
Reserved	–	A-54	U1.36	X
N/C (E998-13)	E998,13	–	U1.96	X
N/C (E998-14)	E998,14	–	U1.95	X
BCI STPASS L	R997,1	–	U1.5	R
BCI PHASE L	E998,8	–	U1.75	R
BCI TIME H	E998,1	–	U1.76	O
BCI TIME L	E998,15	–	U1.77	R
BCI PHASE H	E998,10	–	U1.78	R
Open Via	–	–	U1.54	X
Open Via	–	–	U1.88	X
Open Via	–	–	U1.89	X
Open Via	–	–	U1.90	X

Key:

R = Required connection on T1999 modules.
X = No connection permitted.
O = Optional connection; designer's choice.
N/A = Not applicable; the signal does not appear at Corner boundary.
@ = Module ground planes (layers 4 and 7).

VAXBI Corner Hole and Pad Sizes

Table 3-6 shows the hole and pad sizes that can be used in the Corner. The preferred tolerance limits on pads is +0.004" or −.002" on external layers, and +.002", −.003" on internal layers. The preferred tolerance limits for holes is + or − 0.003" except the .041" holes to accommodate in-situ, low-profile, press-fit sockets to accommodate replacement of failed or revised chip components. The hole tolerance for press-fit sockets must be +.002/−0.003". The connector via pads are small to reduce capacitance. The ground and power layer clearances (negative pads) have been reduced because empirical measurement of layers fabricated with larger negative pads showed excessive voltage drop characteristics. For pin 1 reference on layers 1 and 10, a .060"-square pad is permitted.

Table 3-6 VAXBI Corner Hole and Pad Sizes (in inches)

Use Within VAXBI Corner	Nominal Plated-Thru Hole	Maximum Drill Size	Design Nominal Pad Sizes (by layer) 1, 10	2,3,8,9	4,5,6,7
Device pin holes	.041	.047	.060	.060	.080
Discretes, boundary	.038	.044	.060	.060	.080
Connector tab pads	.032	.038	.060	.060	.070
Or if no pins in hole and surface-mount technology	.020	.025	.040	.040	.060

Note: For plated-thru holes with no pins, power and ground connections must be .032" min.

VAXBI Corner Boundary Area

The VAXBI Corner contains an L-shaped pattern of 98 signal connections, which are for user interface to the VAXBI bus (see Figure 3-15). In the figure, the revision field (B1) refers to the database revision of the Corner, not to the functional revision of the Corner.

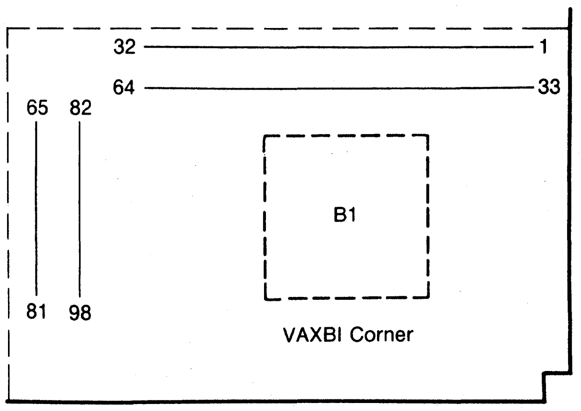


FIGURE 3-15 VAXBI Corner Connection Points

Table 3-7 lists the signals available on the virtual connector, at the boundary between the VAXBI Corner and the user-configurable area.

A capacitance restriction of 50 pF must be met for the 53 BIIC signal lines of the VAXBI boundary to meet the BIIC timing specifications with no derating. (The BIIC requires that the capacitive load be less than 100 pF.) Since less than 10 pF has been used up in the Corner already, at least 40 pF remains in the capacitance budget for each signal. Derating information for BIIC AC timing parameters for designs with capacitance above 50 pF and less than 100 pF is in Part Two of the VAXBI SRM. See the discussion in this chapter on capacitance, page 3-32.

We recommend that you conform to the naming conventions used in the table. The layer listed is the layer at which a printed circuit connects to the virtual connector from something within the VAXBI Corner. Normally, pads will exist at all layers for each virtual pin in the virtual connector, but these pads may be deleted (to reduce capacitance) for all but the listed layer.

It is permissible not to use the connecting points as pads or vias, but simply as routing targets existing only on the layer on which they come in from the Corner. In this case, they can be routed through on a different layer, and capacitance is reduced. The points are preferable as vias because they provide an easy point at which to test the VAXBI Corner. The designer should be aware that large clearance holes around these vias may result in insufficient copper remaining on power and ground layers to maintain voltage drop within tolerable limits. Unused pads can be deleted and the vias removed if desired.

Connection pads 66 and 74 are the pin holes for capacitor C995. The pads are a different size and must not be omitted. These and other open vias in the boundary area are not available to users.

When connecting to the lower row of connecting points, the user should be aware of the danger of shorting to prerouted connections from the Corner to the upper points. Table 3-7 indicates where the prerouting has been done. Refer to the module control drawings to see whether a particular prerouted connection goes to the left or right of the inner connection point below, before routing these connections or preparing blockage areas for a routing process.

The BI AC LO L and BI DC LO L signals are provided to allow designs to drive these BI signals. Designs should not monitor these lines directly off the VAXBI bus. The BCI AC LO L and BCI DC LO L signals from the BIIC are the only signals that are to be monitored for power status.

Table 3-7 VAXBI Virtual Connector Signals

Pin	Signal Name	Routed From	Layer	Notes
U1.01	5VBB	ZIF A-35	6	
U1.02	PASS THRU	ZIF A-37	8	Reserved for DIGITAL
U1.03	PASS THRU	ZIF A-39	8	Reserved for DIGITAL
U1.04	PASS THRU	ZIF A-41	8	Reserved for DIGITAL
U1.05	BCI STPASS L	LED, R997	9	
U1.06	BCI SC1 L	BIIC, H-2	9	< 100 pF capacitance
U1.07	BCI D30 H	BIIC, J-2	2	< 100 pF capacitance
U1.08	BCI D28 H	BIIC, K-2	2	< 100 pF capacitance
U1.09	BCI D31 H	BIIC, K-1	3	< 100 pF capacitance
U1.10	BCI D24 H	BIIC, M-2	8	< 100 pF capacitance
U1.11	BCI D25 H	BIIC, N-1	3	< 100 pF capacitance
U1.12	BCI D26 H	BIIC, M-1	2	< 100 pF capacitance
U1.13	BCI D20 H	BIIC, P-2	3	< 100 pF capacitance
U1.14	N/C (E999-N2)	BIIC, N-2	2	Reserved for DIGITAL
U1.15	BCI D22 H	BIIC, N-3	2	< 100 pF capacitance

Table 3-7 VAXBI Virtual Connector Signals (Continued)

Pin	Signal Name	Routed From	Layer	Notes
U1.16	BCI D18 H	BIIC, N-4	2	< 100 pF capacitance
U1.17	BCI D15 H	BIIC, N-5	2	< 100 pF capacitance
U1.18	BCI D12 H	BIIC, N-6	2	< 100 pF capacitance
U1.19	BCI D09 H	BIIC, N-7	2	< 100 pF capacitance
U1.20	BCI D06 H	BIIC, P-9	3	< 100 pF capacitance
U1.21	BCI D13 H	BIIC, M-6	9	< 100 pF capacitance
U1.22	BCI D05 H	BIIC, N-9	2	< 100 pF capacitance
U1.23	BCI D01 H	BIIC, N-10	2	< 100 pF capacitance
U1.24	BI AC LO L	BIIC, P-14	3	User interface driver only
U1.25	BCI D04 H	BIIC, M-9	8	< 100 pF capacitance
U1.26	BCI I0 H	BIIC, M-12	8	< 100 pF capacitance
U1.27	BCI CLE H	BIIC, M-13	9	< 100 pF capacitance
U1.28	BCI RAK L	BIIC, L-13	9	< 100 pF capacitance
U1.29	BCI I3 H	BIIC, N-11	2	< 100 pF capacitance
U1.30	BCI EV0 L	BIIC, K-13	9	< 100 pF capacitance
U1.31	BCI EV3 L	BIIC, K-14	9	< 100 pF capacitance
U1.32	BCI AC LO L	BIIC, N-13	2	< 100 pF capacitance
U1.33	PASS THRU	ZIF A-32	8	Reserved for DIGITAL
U1.34	5VBB	ZIF A-51	6	
U1.35	PASS THRU	ZIF A-52	8	Reserved for DIGITAL
U1.36	PASS THRU	ZIF A-54	8	Reserved for DIGITAL
U1.37	BI SPARE L	ZIF A-56	8	Reserved for DIGITAL
U1.38	BCI SC2 L	BIIC, H-1	2	< 100 pF capacitance
U1.39	BCI SEL L	BIIC, H-3	9	< 100 pF capacitance
U1.40	BCI SC0 L	BIIC, J-1	9	< 100 pF capacitance
U1.41	BCI D29 H	BIIC, L-1	3	< 100 pF capacitance
U1.42	BCI D27 H	BIIC, L-2	2	< 100 pF capacitance
U1.43	BCI D23 H	BIIC, M-3	9	< 100 pF capacitance
U1.44	BCI D21 H	BIIC, M-4	8	< 100 pF capacitance
U1.45	BCI D17 H	BIIC, M-5	9	< 100 pF capacitance
U1.46	BCI D19 H	BIIC, P-3	3	< 100 pF capacitance
U1.47	BCI D16 H	BIIC, P-4	3	< 100 pF capacitance
U1.48	BCI D14 H	BIIC, P-5	3	< 100 pF capacitance
U1.49	BCI D11 H	BIIC, P-6	3	< 100 pF capacitance
U1.50	BCI D10 H	BIIC, P-7	3	< 100 pF capacitance
U1.51	BCI D07 H	BIIC, P-8	3	< 100 pF capacitance
U1.52	BCI D08 H	BIIC, N-8	2	< 100 pF capacitance
U1.53	BI DC LO L	BIIC, H-13	8	User interface driver only
U1.54	Open Via		-	Reserved for DIGITAL
U1.55	BCI D03 H	BIIC, P-10	3	< 100 pF capacitance
U1.56	BCI D02 H	BIIC, P-11	3	< 100 pF capacitance
U1.57	N/C (E999-B13)	BIIC, B-13	9	Reserved for DIGITAL
U1.58	BCI EV2 L	BIIC, J-12	9	< 100 pF capacitance

Table 3-7 VAXBI Virtual Connector Signals (Continued)

Pin	Signal Name	Routed From	Layer	Notes
U1.59	BCI NXT L	BIIC, M-14	9	< 100 pF capacitance
U1.60	BCI EV1 L	BIIC, L-14	9	< 100 pF capacitance
U1.61	BCI D00 H	BIIC, P-12	2	< 100 pF capacitance
U1.62	BCI I1 H	BIIC, N-12	2	< 100 pF capacitance
U1.63	BCI I2 H	BIIC, P-13	2	< 100 pF capacitance
U1.64	BCI P0 H	BIIC, N-14	3	< 100 pF capacitance
U1.65	BCI DC LO L	BIIC, J-14	9	< 100 pF capacitance
U1.66	GND	ZIF (GND)	4/7	Lead hole for C10
U1.67	BCI INT5 L	BIIC, F-14	3	
U1.68	BCI INT4 L	BIIC, E-14	3	
U1.69	BCI RQ1 L	BIIC, D-14	3	
U1.70	BI BAD L	ZIF B-51	8	
U1.71	BCI INT6 L	BIIC, G-12	9	
U1.72	BCI SDE L	BIIC, H-14	9	< 100 pF capacitance
U1.73	BCI INT7 L	BIIC, G-13	9	
U1.74	−2.0V	ZIF (B-38,39,53)	5	Lead hole for C10
U1.75	BCI PHASE L	Clk Rec Pin 8	3	See Application
U1.76	BCI TIME H	Clk Rec Pin 1	9	Note 6 in SRM for
U1.77	BCI TIME L	Clk Rec Pin 15	3	more information.
U1.78	BCI PHASE H	Clk Rec Pin 10	3	
U1.79	BI ID3 H	ZIF B-47	9	
U1.80	BI ID1 H	ZIF B-49	9	
U1.81	BCI CK DIS L	R994,1	2	Disabled driver sensing
U1.82	BCI EV4 L	BIIC, J-13	9	< 100 pF capacitance
U1.83	BCI RS1 L	BIIC, E-13	3	
U1.84	BCI MAB L	BIIC, F-13	3	
U1.85	+12.0V	ZIF B-50	8	
U1.86	BI STF L	ZIF B-36	8	
U1.87	−12.0V	ZIF B-37	8	
U1.88	Open Via		-	Reserved for DIGITAL
U1.89	Open Via		-	Reserved for DIGITAL
U1.90	Open Via		-	Reserved for DIGITAL
U1.91	BCI MDE L	BIIC, G-14	9	< 100 pF capacitance
U1.92	BCI RQ0 L	BIIC, F-12	9	
U1.93	BCI RS0 L	BIIC, C-14	9	
U1.94	BI RESET L	BI ZIF B-54	8	
U1.95	N/C (E998-14)	Clk Rec Pin 14	8	Reserved for DIGITAL
U1.96	N/C (E998-13)	Clk Rec Pin 13	3	Reserved for DIGITAL
U1.97	BI ID2 H	ZIF B-33	9	
U1.98	BI ID0 H	ZIF B-35	9	

VAXBI Corner Ground and Power Planes

The ground planes (layers 4 and 7) and the power planes (layers 5 and 6) are treated as negative layers. The ground layers are identical. Note that they contain custom clearance areas within the VAXBI Corner. The user need not alter the ground planes at all in normal circumstances (except to provide the normal clearance area around pin holes not grounded), and should never do so in the VAXBI Corner. Outside the Corner, the user may reconfigure these planes.

The power planes are divided into various voltages in a way required by the placement of devices in the Corner. Alteration of these planes within the Corner can cause the bus to fail. The user should only be concerned with the boundary locations of the different voltages when deciding how to divide up the power planes in the user-configurable area. It is acceptable to change the configuration of the power plane splits within the Corner to improve the IR drop for a particular voltage. However, this is not a simple process and voltage shorts and lack of power supply connectivity can result. If these changes are made, you must be especially careful when blanking off supplies around the connector area that connections to the plating features are not disconnected.

The power planes (layers 5 and 6) are already divided within the VAXBI Corner. The default is that all these voltages are blanked off from the user-configurable space. You must block off any voltages not desired and distribute the others around the board in a manner appropriate to the VAXBI module being designed. Since the power layers are negative, you must draw a dividing line across the inner row of VAXBI Corner boundary area connection points on the appropriate layer. Failure to do this operation properly can short voltages or not connect voltages to the module.

Do not isolate voltage connections on the four rows of vias in the connector area, even if a particular voltage is not used. These vias are used to carry current for gold plating the fingers on layers 1 and 10.

The voltages available on layer 5 are as follows:

- 5VBB
- +5.0V
- −12.0V
- −2.0V

The voltages available on layer 6 are as follows:

- 5VBB
- +5.0V
- −5.2V
- +12.0V
- −12.0V

Figure 3-16 shows two examples of how the power plane splits can be used. Ordinarily, you should use a 0.020" or 0.025" line to make these cuts. Remember to make the negative lines which close off unused voltages. For the VAXBI Corner, the voltage apportionment is defined, and the user may not change it. The default case when the user obtains the database is for all voltages to be blanked off from the user-configurable area.

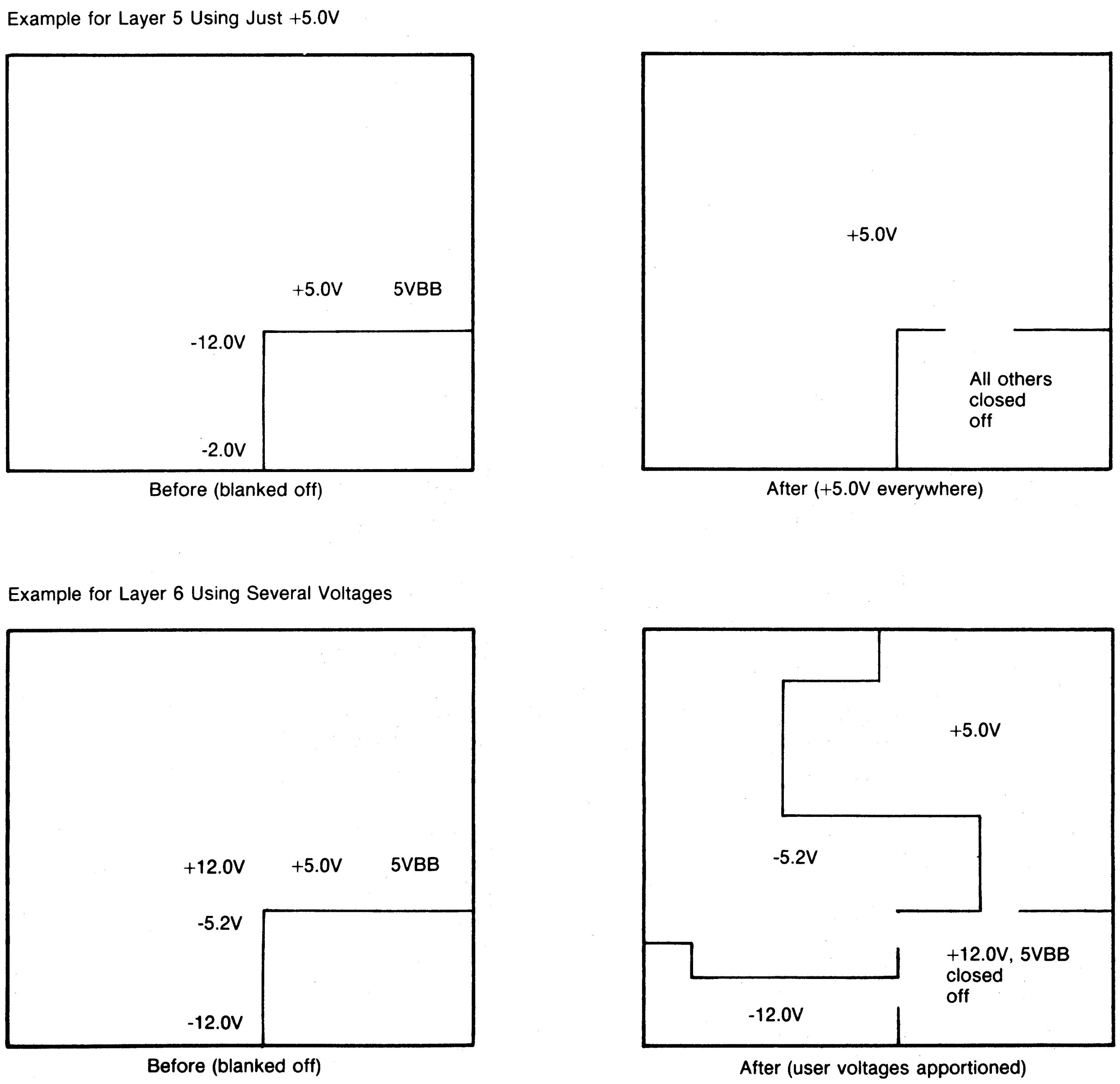


__FIGURE 3-16__ Examples of Power Plane Splits

User-Configurable Area

Figure 3-17 shows the area of a standard VAXBI module that is available to the user. This area is 55.6 square inches. The operative pin grid is 0.100".

The user-configurable area is irregularly shaped, so even if your numerical calculations indicate that enough space is available, the required components may not fit. When designing a standard VAXBI module layout, you should try to fit the devices into the actual shape. Rectangular packages will normally be placed either all vertical or all horizontal for ease of assembly and inspection. Table 3-8 gives the typical number of components that a standard VAXBI module can accommodate in DIP equivalents.

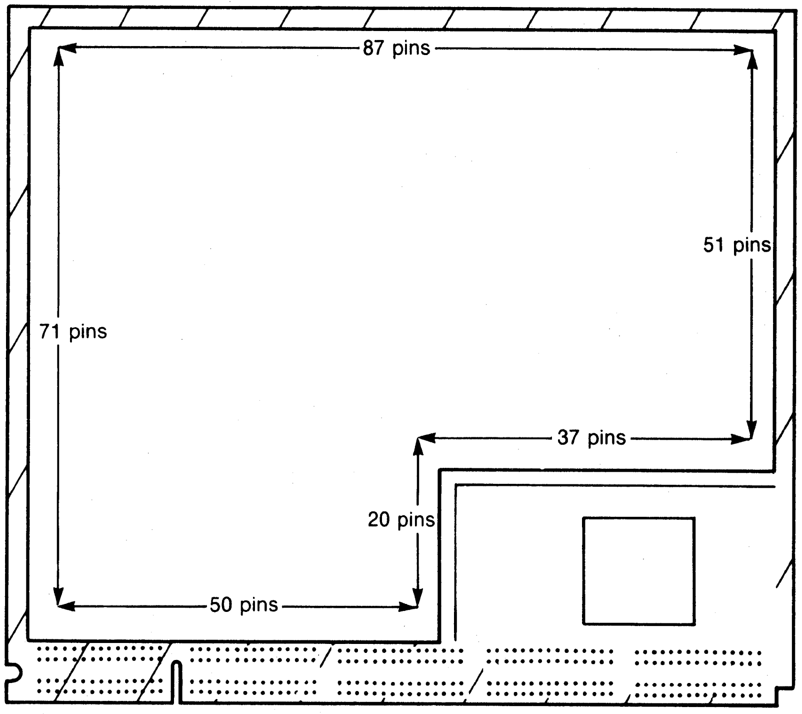


FIGURE 3-17 *Placement Area*

Table 3-8 Number of Components That Fit in User-Configurable Area

Orientation	**Vert**	**Vert**	**Vert**	**Vert**	**Horiz**	**Horiz**	**Horiz**	**Horiz**	**–**
DIP-size (pins)	14	16	14	16	14	16	14	16	15x15 PGAs
Package spacing (pins)	0	0	1	1	0	0	1	1	2
Package count	183	150	122	105	179	150	134	112	21

Required Self-Test LED

All VAXBI modules must have a yellow LED (such as DIALIGHT P/N 550-0306), reference designator D999, along the edge of the module opposite the connector edge. Figure 3-18 shows the position.

VAXBI nodes are required to test themselves and turn on this LED when they are good. The LED cannot be moved in the vertical direction. It is best to leave the LED where it is, but if this location interferes with the layout, the LED can be moved in the horizontal direction, as long as its entire outline is between 2.50" and 8.75" from the right-hand edge. It cannot go farther to the right because a securing device would block it from view.

No other yellow LED may be placed along the top of the module, so that the yellow LED will always be for self-test. Note that the etch run connected to the LED must be moved along with it. Do not confuse D999 with the other LED, D998, which is in a fixed location in the VAXBI Corner.

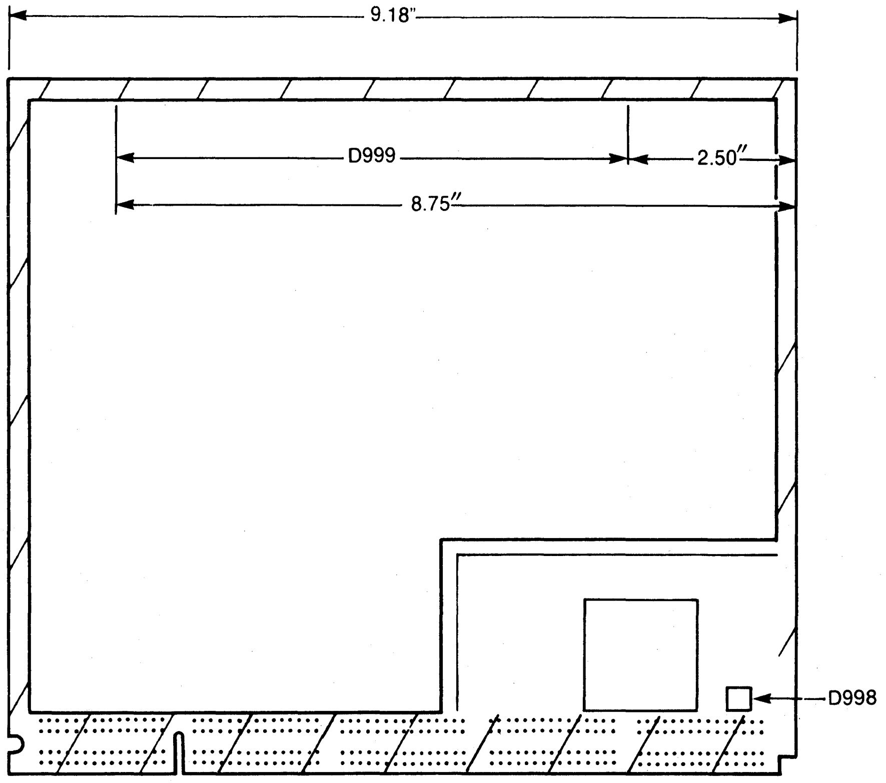


FIGURE 3-18 *Self-Test LED Location*

Constraints on Parts

VAXBI modules must fit in a slot on a 0.8" center, so except for very thin devices, components should only be mounted on the front (component) side of the module. There is a potential exception to this for a multiboard VAXBI node if certain rules are followed so that the boards can be placed in adjacent slots and be cabled together.

VAXBI modules may not have attached cables, but must interconnect through the connector zones C, D, and E. They also may not have DIP switches or other mechanical adjustments. They may not have pots, trimmers, or other electromechanical adjustments. (A VAXBI extender module is being developed that does not degrade the operating characteristics of the bus.)

We highly recommend in-situ socketing devices, because soldered components are not easy to remove from a VAXBI module, and it is unlikely that the user will wish to scrap a board just because a chip has failed or has been revised. This is particularly true for devices with more than 64 pins.

Capacitance-Restricted Signal Specifications

The maximum capacitance allowed on a BIIC signal line is less than 100 pF. However, for designs with capacitance above 50 pF and less than 100 pF, the timing can be derated.

DIGITAL measured one bare module from the BIIC up to and including the via in the boundary area. (The module measured was a Rev. 2 foundation module with 2-oz./sq" in copper on ground planes and 1-oz./sq" copper on signal layers.) Table 3-9 gives capacitance measurements for the 53 BIIC signals. The total capacitance used up in the VAXBI Corner is less than 10 pF in every case. To determine the remaining capacitance budget on that module, you subtract the figure in the capacitance column from 50. The values can vary for different batches of modules.

Should the signal capacitance load of any of these signals exceed 50 pF (but be less than 100 pF) there is a possibility that the particular VAXBI option can still function if there is enough margin in its timing. Essentially, this means derating the signal from the BIIC specification. The following derating formulas are offered as a guideline:

Tr or Tf (50 pF < Cl < 100 pF) =
Tr or Tf (50 pF) + Cl / 17.8 − 2.8

Tp1 or Tp2 or Tp3 (50 pF < Cl < 100 pF) =
Tp1 or Tp2 or Tp3 (50 pF) + Cl / 5.5 − 9.1

Table 3-9 Capacitance Measurements of BCI Signals

Pin	Signal Name	Source	Layer	Etch (in.)	Capacitance (pF)
U1.6	BCI SC1 L	E999, H2	9	1.56	5.2
U1.7	BCI D30 H	E999, J2	2	1.31	4.1
U1.8	BCI D28 H	E999, K2	2	1.06	4.0
U1.9	BCI D31 H	E999, K1	3	1.00	5.4
U1.10	BCI D24 H	E999, M2	8	0.63	4.0
U1.11	BCI D25 H	E999, N1	3	0.56	4.0
U1.12	BCI D26 H	E999, M1	2	0.69	3.3
U1.13	BCI D20 H	E999, P2	3	0.47	3.8
U1.15	BCI D22 H	E999, N3	2	0.66	3.1
U1.16	BCI D18 H	E999, N4	2	0.66	3.1
U1.17	BCI D15 H	E999, N5	2	0.66	3.1
U1.18	BCI D12 H	E999, N6	2	0.66	3.1
U1.19	BCI D09 H	E999, N7	2	0.66	3.1
U1.20	BCI D06 H	E999, P9	3	0.56	3.8
U1.21	BCI D13 H	E999, M6	9	1.06	4.1
U1.22	BCI D05 H	E999, N9	2	0.69	3.2
U1.23	BCI D01 H	E999, N10	2	0.78	3.3
U1.25	BCI D04 H	E999, M9	8	1.03	5.0
U1.26	BCI I0 H	E999, M12	8	0.97	4.6
U1.27	BCI CLE H	E999, M13	9	2.00	3.7
U1.28	BCI RAK L	E999, L13	9	1.03	4.1
U1.29	BCI I3 H	E999, N11	2	1.22	4.2
U1.30	BCI EV0 L	E999, K13	9	1.22	4.5
U1.31	BCI EV3 L	E999, K14	9	1.22	4.4
U1.32	BCI AC LO L	E999, N13	2	1.31	4.3
U1.38	BCI SC2 L	E999, H1	2	1.13	3.8
U1.39	BCI SEL L	E999, H3	9	1.44	3.9
U1.40	BCI SC0 L	E999, J1	9	1.03	4.1

Table 3-9 Capacitance Measurements of BCI Signals (Continued)

Pin	Signal Name	Source	Layer	Etch (in.)	Capacitance (pF)
U1.41	BCI D29 H	E999, L1	3	0.72	4.4
U1.42	BCI D27 H	E999, L2	2	1.63	3.4
U1.43	BCI D23 H	E999, M3	9	0.94	3.1
U1.44	BCI D21 H	E999, M4	8	0.63	3.8
U1.45	BCI D17 H	E999, M5	9	0.56	3.7
U1.46	BCI D19 H	E999, P3	3	0.44	3.4
U1.47	BCI D16 H	E999, P4	3	0.44	3.4
U1.48	BCI D14 H	E999, P5	3	0.44	3.4
U1.49	BCI D11 H	E999, P6	3	0.44	3.4
U1.50	BCI D10 H	E999, P7	3	0.41	3.4
U1.51	BCI D07 H	E999, P8	3	0.44	3.4
U1.52	BCI D08 H	E999, N8	2	0.53	2.9
U1.55	BCI D03 H	E999, P10	3	0.53	3.7
U1.56	BCI D02 H	E999, P11	3	0.50	4.0
U1.58	BCI EV2 L	E999, J12	9	0.72	4.1
U1.59	BCI NXT L	E999, M14	9	1.03	3.3
U1.60	BCI EV1 L	E999, L14	9	0.94	3.7
U1.61	BCI D00 H	E999, P12	2	1.06	3.8
U1.62	BCI I1 H	E999, N12	2	1.13	4.0
U1.63	BCI I2 H	E999, P13	2	1.31	3.8
U1.64	BCI P0 H	E999, N14	3	1.13	5.5
U1.65	BCI DC LO L	E999, J14	9	1.41	6.6
U1.72	BCI SDE L	E999, H14	9	1.34	4.6
U1.82	BCI EV4 L	E999, J13	9	1.63	5.2
U1.91	BCI MDE L	E999, G14	9	1.22	4.9

The four clock receiver signals are also restricted in total skew (see Application Note 6 in the VAXBI SRM for details). The skew varies with the timing of the particular VAXBI option. These signals drive devices within the Corner and go to the connector segments. There are no measured values yet for the capacitance already used. The simulated capacitance values for these signals up to the boundary area are given in Table 3-10.

Table 3-10 Simulated Clock Run Capacitances

Pin	Signal Name	Receiver Pin	Capacitance (pF)
U1.75	BCI PHASE L	E998-8	17.2
U1.76	BCI TIME H	E998-1	18.3
U1.77	BCI TIME L	E998-15	15.8
U1.78	BCI PHASE H	E998-10	18.4

Capacitance Estimation

Following are some rules of thumb for estimating capacitance on critical signals.

Using schematic and preliminary placement sketch:

- Add component capacitances and 2 pF per via and 2-4 pF per inch. This gives an approximation.

From artwork:

- Add component capacitances and 1.5 pF per via pad (2 pF if on adjacent layer also). Then add 2-3 pF per inch for runs on layers 2 and 9, depending upon the density of nearby metallic features or 3-4 pF on layers 3 and 8.

From physical prototype VAXBI modules:

- Use a capacitance meter to measure the worst-case signal. Manufacturing tolerances may cause variation in measured values.

Signal Capacitance Reduction Techniques

During logic design:

- Choose components with low capacitance loading.

During module layout:

- Place components driven from the VAXBI Corner near the boundary area.
- Use the shortest, thinnest etch possible.
- Prefer layers 2 and 9 to layers 3 and 8.
- Avoid vias, especially to adjacent layers.
- Delete unused pads.
- Spread lines apart.
- Change component placement to reduce etch run lengths.

VAXBI Expansion Module

Some options are too complicated to fit on a single VAXBI module. In the case of an option that consists of a standard VAXBI module and a subsystem remote from the VAXBI, connector segments C, D, and E can be attached to cables or other mechanisms for use outside the VAXBI cage. If an option consists of two or more VAXBI modules, segments C, D, and E on a standard VAXBI module can connect to one or more VAXBI expansion modules. In each case, only one module is a standard VAXBI module. The others do not contain a VAXBI Corner.

The control drawing for the VAXBI expansion module is T1996. Figure 3-19 shows how a typical layout system views a VAXBI expansion module. Layout systems view the component side of the board.

The right edge of the board and all pin holes are on 0.1" centers. The left edge of the board is off all grids; however, the mechanical features are on grid with reference to the right edge. The A and B connector segment vias are off grid by 0.025" or 0.075". The user connector segments C, D, and E are on grid; however, the designer has the option to move these connector vias off grid to improve routability.

Most holes and features align on a 0.025" grid pattern (see ELEN 626).

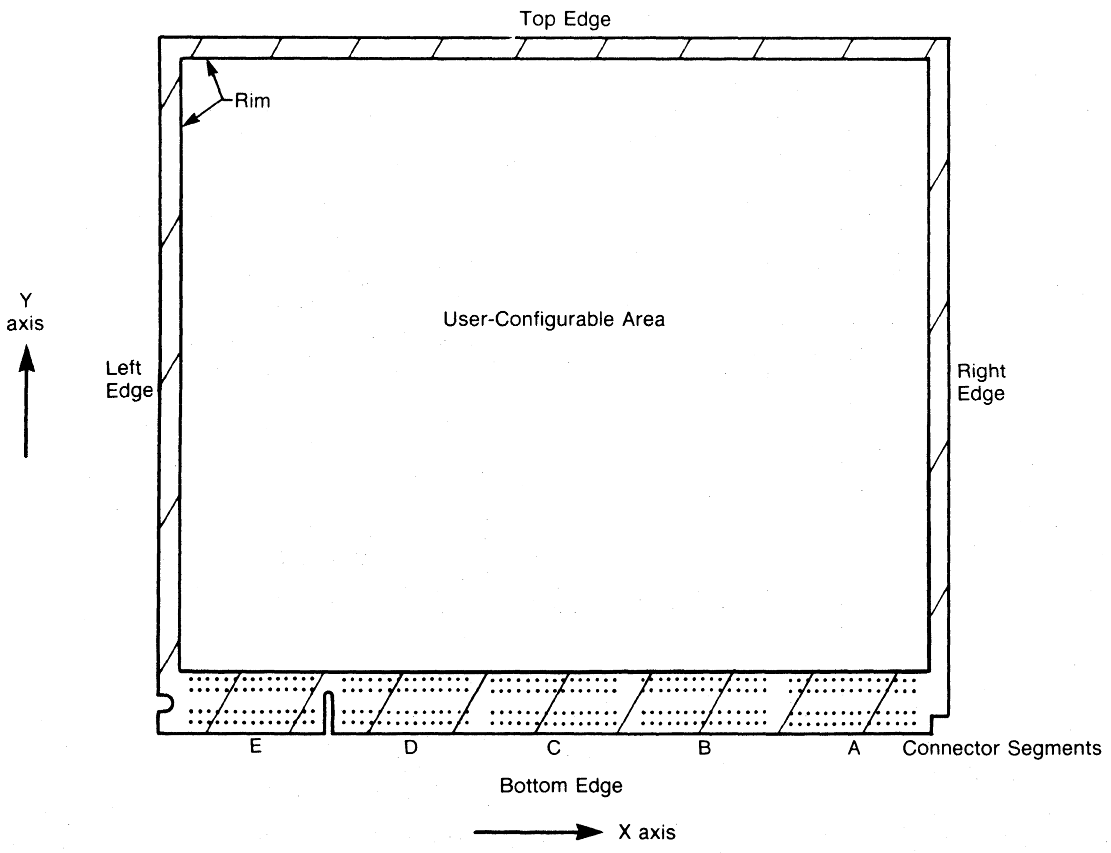


FIGURE 3-19 *Layout System View of VAXBI Expansion Module (component side)*

Components

Four components are required on a VAXBI expansion module:

- R998 270-ohm resistor (for LED circuit)
- R999 1-Kohm resistor (for ESD pads)
- D999 LED
- D998 LED

Figure 3-20 shows the position of the components. R999, R998, and D998 are in the lower right-hand corner of the module. D999 is in the user-configurable area in the same location as on a standard VAXBI module.

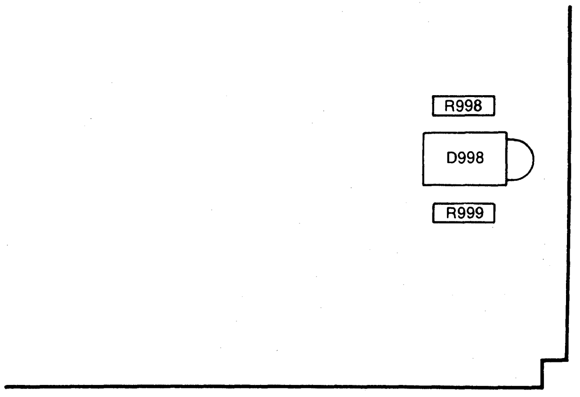


FIGURE 3-20 VAXBI Expansion Module (lower right-hand corner detail)

Summary of VAXBI Expansion Modules

VAXBI expansion modules have many of the same features as a standard VAXBI module.

- Module outline
- Tooling holes
- Rim reserved on outer layers
- Three user connector segments
- Height and width tolerances
- Notches and shapes
- ESD pads and discharge resistor
- Self-test LEDs (driven from the option's standard module)
- Powered through A and B segments

An expansion module differs from a standard VAXBI module as follows:

- No VAXBI Corner
- No VAXBI boundary area
- Configuration of power plane(s)
- A and B connector segments
- Plating connections for clock signals

Restrictions

The gold finger tabs for segments A and B must exist on VAXBI expansion modules but must not be used, except for the power pins to bring appropriate voltages onto the module. Only the six asynchronous bus signals are available for connection. All other signals must not be connected.

Asynchronous Signals in A and B Connector Segments

An expansion module must not drive or receive any VAXBI synchronous signals or VAXBI clocks: there can be no connection to any pin in the A or B zone of the connector other than to pins that supply DC voltages (labeled GND or voltage) and to the VAXBI asynchronous control signals. BI AC LO and BI DC LO may be driven through suitable drivers by VAXBI expansion modules, but they should not be otherwise loaded.

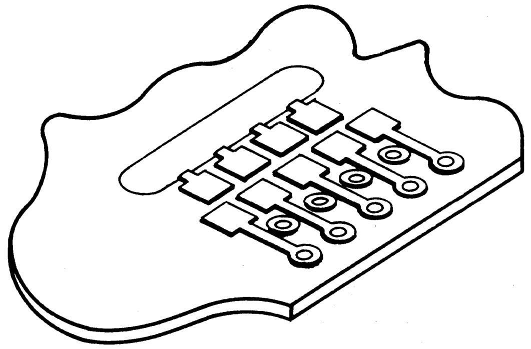
Only the following signals can be routed in the A and B connector segments:

Signal Name	Connector Pin
BI AC LO L	B-40
BI DC LO L	B-09
BI BAD L	B-51
BI RESET L	B-54
BI STF L	B-36
Spare line	A-56

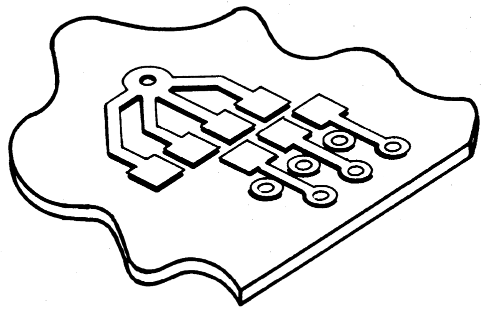
Critical Signal Plating Feature

Because of the critical nature of the VAXBI signals, it is necessary to be able to plate the pads on the A and B connectors without generating long stubs that will cause mismatches. This is particularly important for 36-slot systems. To achieve the 36-slot configuration, critical signals are connected in groups to a single via. The etch runs must be cut following board fabrication and gold plating of the connectors. Cutting etch runs can be done in three ways: (See T1996 and ELEN 626 for full details.)

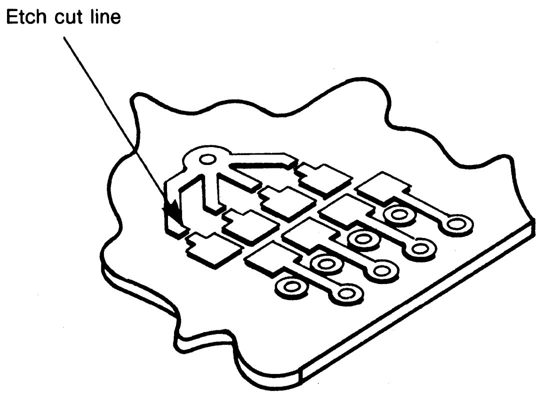
- **Method 1** Rout a slot above the connector removing the shorted etch.



- **Method 2** Drill rout the via and pad to produce an open circuit.



- **Method 3** Cut the etch above the connector to produce a disconnect.



Experiences in Building VAXBI Modules

This section discusses considerations and constraints that led to the design of the VAXBI Corner. It also describes problems that must be considered should one wish to develop a new layup for a VAXBI module.

Constraints

Mechanical Constraints

The base layup, which was used for the foundation module, has the power/ground core placement and thicknesses optimized for maximum mechanical stability in terms of stiffness and warp.

Data and Signal Line Constraints

The VAXBI data and control signals are driven by the BIIC, a ZMOS device. The most sensitive electrical parameter for these signals is the total capacitance that the BIIC must charge and discharge when switching signal states. Each system component (module, backplane, BIIC) has been specified with a maximum capacitance to guarantee system performance. The standard layout for the VAXBI Corner uses fine-line PCB technology and routes these signals on layer 2, which is the signal layer farthest from the ground plane. The maximum allowable line capacitance for the VAXBI signals is 1.9 pF per inch.

Clock Line Constraints

The VAXBI clock system uses ECL technology to minimize skews. The overall design of the system is a compromise since the loading on the clock lines is a function of both the number of backplane segments and the number (and position) of VAXBI modules in a system. The clock lines run on layers 8 and 9 and must present capacitive (and hopefully impedance) loading to the clock bus lines on the backplanes. The loading should be approximately 40 ohms and 4.0 pF per inch.

The following assumptions were made in modeling each layup:

- H1D layout rules limit the number of conductors running in each routing channel to one. H2D layout rules permit two conductors per channel.
- "Spacing" between conductors refers to the open distance between them, not to their center-to-center spacing.
- Conductors are trapezoids with the dimension of the smaller side always assumed to be one conductor thickness less than the dimension of the larger side. Hence, a 10-mil line of 1-oz. copper would have widths of 10 and 8.6 to 9 mils.
- All adjacent prepreg and core material have the same dielectric constant, which is assumed to be 4.2 (the effective Er of typical boards that have been measured).
- Conductors imbed into the prepreg during fabrication and do not add any overall thickness to a module. Therefore, with a layup as shown in Figure 3-21 with 1-oz. copper conductors (1.4 mils thick), the overall thickness would be 26 mils and NOT 28.8 mils.

The core has signal runs on both sides and is sandwiched between the power planes and surface layer with prepreg. The narrow part of the trapezoids is oriented toward the closest power plane or surface layer, as shown in Figure 3-21.

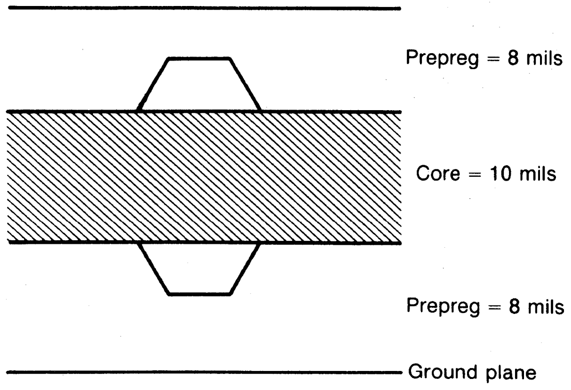


FIGURE 3-21 *Conductor Modeling Diagram*

Module Layup

To become qualified suppliers for VAXBI modules, vendors must supply a layup of the proposed layering thicknesses for DIGITAL's approval. Table 3-11 describes the 10 layers. Figure 3-22 shows a good starting layup in which cores are copper clad and laminated (prepreg means insulation core). See ELEN 633 for complete specifications.

Table 3-11 Layer Assignment

Layer	Purpose	Limitations
1	Cap layer, tin-lead plating, pads for thru-hole plating	Outer layer (0.25–2.0-oz. copper/sq. ft)
2	Signal routing	Bi-core layers
3	Signal routing	(1-oz. copper/sq. ft)
4	Signal ground and AC ground	Bi-core layers
5	Power	(copper user configurable, 1, 2, or 3-oz./sq. ft, 2-oz. preferred)
6	Power	Bi-core layers
7	Signal ground and AC ground	(copper user configurable, 1, 2, or 3-oz./sq. ft, 2-oz. preferred)
8	Signal routing	Bi-core layers
9	Signal routing	(1-oz. copper/sq. ft)
10	Cap layer, tin-lead plating, pads for thru-hole plating	Outer layer (0.25–2.0-oz. copper/sq. ft)

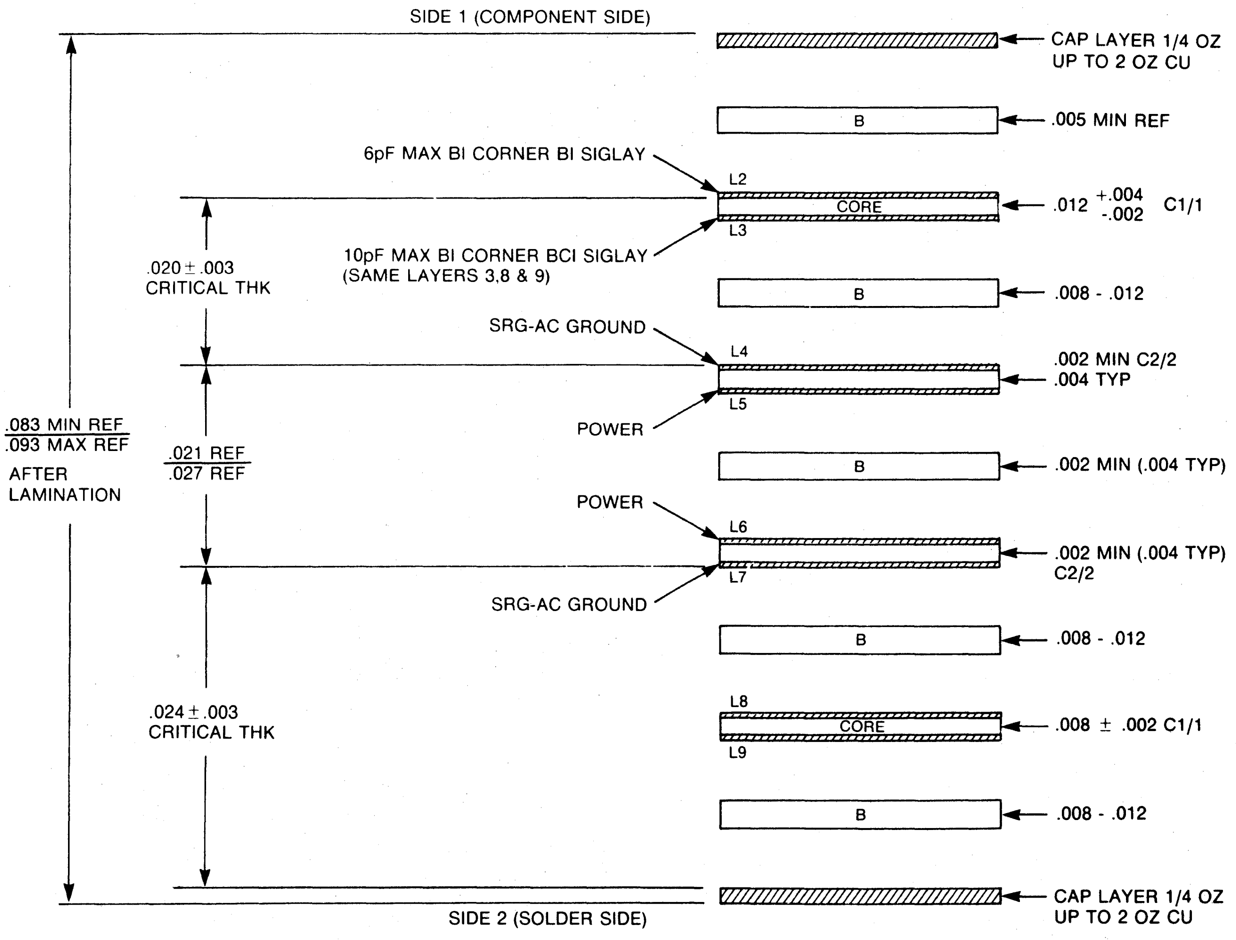


FIGURE 3-22 *Starting Layup*

Layup Variations

The starting layup in Figure 3-22 has rather loose tolerances and is intended as a guideline rather than as an optimum layup. Two layups are acceptable for the VAXBI Corner: the TTL layup and the base layup. These layups have been used on boards built internally and externally. The boards have been measured extensively. Both layups are compromises from what most designers would consider ideal due to a number of different but necessary constraints. The TTL layup represents an effort to accommodate most options with a single case.

The use of layups other than those recommended could lead to modules that would prohibit a VAXBI system from working in a configuration-independent fashion. Deviations from the specifications could lead to compatibility problems in the field long after a module has been introduced and appears to be working properly in a system.

Base Layup

The base layup shown in Figure 3-23 is designed to handle both TTL and ECL technologies and has been implemented on the VAXBI foundation modules, type A and type B. The layup is characterized by two "high" impedance layers, L2 and L9, and two "low" impedance layers, L3 and L8.

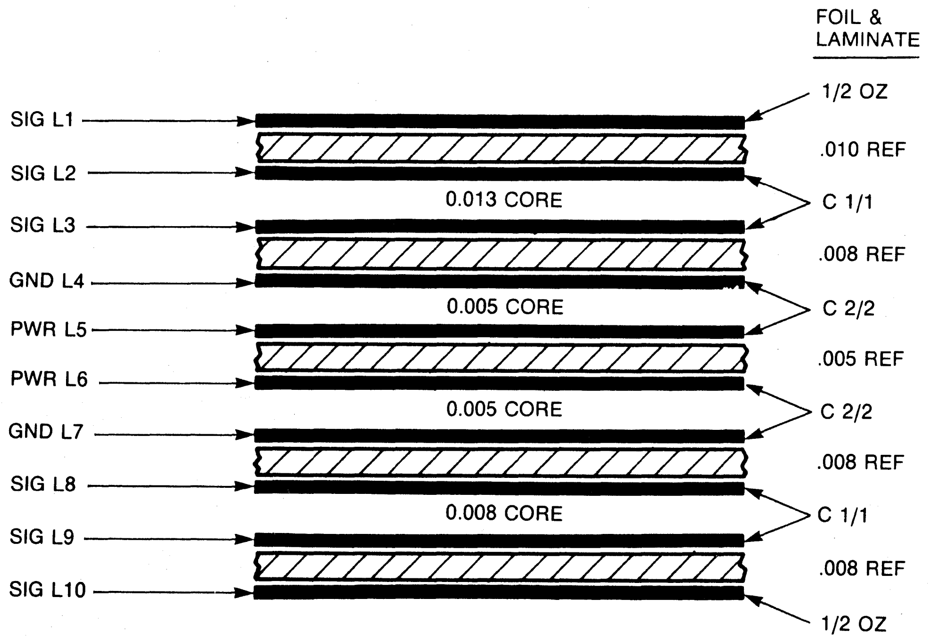


FIGURE 3-23 *Base Layup Diagram*

Table 3-12 gives the simulated signal line impedance range for each layer, assuming the line widths shown and 10-mil spacing. The maximum impedance values (minimum capacitance) occur with H1D layout rules and no adjacent layer signal runs. The minimum impedance values (maximum capacitance) occur with H2D layout rules and adjacent layer signals.

All impedances shown are nominal values. Impedance of an etch run on a given module can easily vary plus or minus 15% from these values with normal etching tolerance and proximity of other metal conductors. These figures are expected to provide adequate impedance control for normal TTL and MOS logic. If tighter impedance control is thought to be needed, you must carefully specify it, have each module tested, and pay higher PCB manufacturing costs.

Table 3-12: Signal Line Impedance Range for Base Layup

	H1D, No Adjacent Layer Signals		**H2D, With Adjacent Layer Signals**	
Layer	**Z(max.) Ohms**	**C(min.) pF/ in.**	**Z(min.) Ohms**	**C(max.) pF/ in.**
L2 (6 mils)	93	1.8	82	2.2
L3 (6 mils)	56	3.0	53	3.4
L8 (6 mils)	56	3.0	51	3.6
L9 (6 mils)	87	1.9	76	2.4

VAXBI Corner Data and Control Characteristics

BIIC data and control lines must run on layer 2. Using the VAXBI Corner layup will result in the following characteristics.

Layer	Z Ohms	C pF/ in.
L2 (6 mils)	93	1.8

VAXBI Clock Characteristics

The VAXBI clock lines must run on layers 8 and 9. Using the VAXBI Corner layup will result in the following characteristics.

Layer	Z Ohms	C pF/ in.
L8 (8 mils)	43	4.2
L9 (8 mils)	43	4.2

TTL Layup

The TTL layup shown in Figure 3-24 has been optimized for module designs that use TTL and MOS technologies. The difference between the impedances of the signal layers has been minimized to reduce signal reflections for long runs that jump between layers. The layup is characterized by four "high" impedance layers.

Table 3-13 gives the simulated signal line impedance range for each layer, assuming the line widths shown with a line spacing of 11 mils on layers 2 and 9 and a line spacing of 9 mils on layers 3 and 8. The minimum value represents H1D layout rules with no adjacent layer signal runs, and the maximum value represents H2D layout rules with adjacent layer signals.

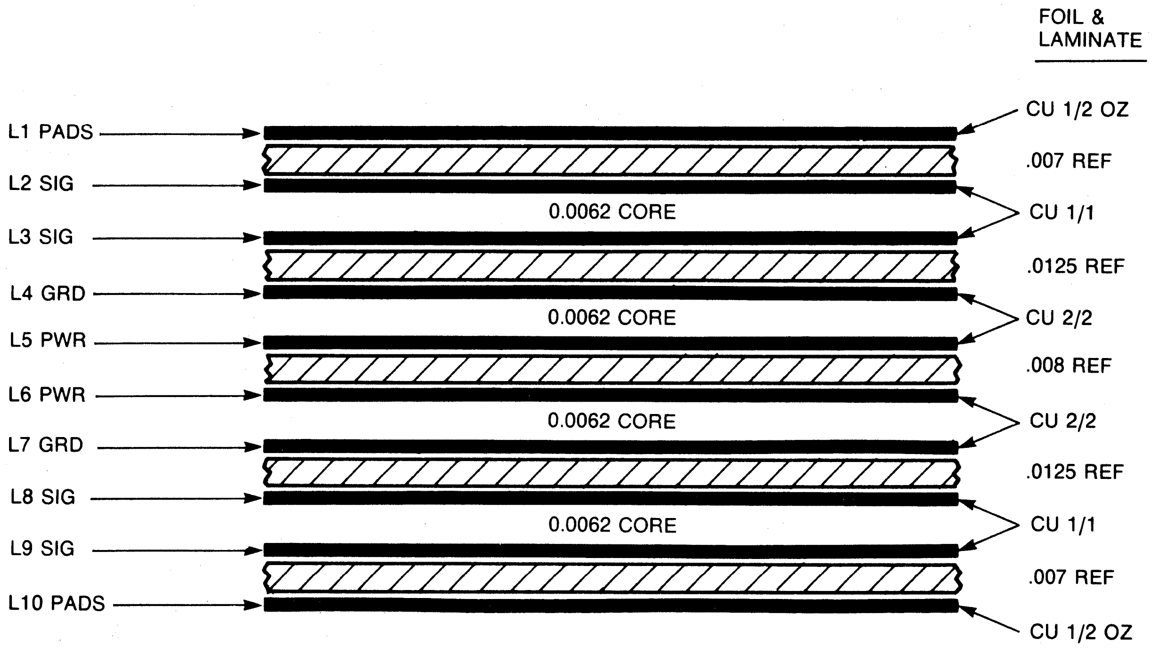


FIGURE 3-24 *TTL Layup Diagram*

Table 3-13: Signal Line Impedance Range for TTL Layup

	H1D, No Adjacent Layer Signals		**H2D, With Adjacent Layer Signals**	
Layer	**Z(max.) Ohms**	**C(min.) pF/ in.**	**Z(min.) Ohms**	**C(max.) pF/ in.**
L2 (8 mils)	90	1.8	71	2.8
L3 (6 mils)	71	2.6	66	3.0
L8 (6 mils)	71	2.6	66	3.0
L9 (8 mils)	90	1.8	71	2.8

All impedances shown are nominal values. Impedance of an etch run on a given module can easily vary plus or minus 15% from these values with normal etching tolerance and proximity of other metal conductors. These figures are expected to provide adequate impedance control for normal TTL and MOS logic. If tighter impedance control is thought to be needed, you must carefully specify it, have each module tested, and pay higher PCB manufacturing costs.

VAXBI Corner Data and Control Characteristics

BIIC data and control lines must run on layer 2. Using the VAXBI Corner layup will result in the following characteristics.

Layer	Z Ohms	C pF/ in.
L2 (6 mils)	94	1.7

VAXBI Clock Characteristics

The VAXBI clock lines must be run on layers 8 and 9. Using the VAXBI Corner layup will result in the following characteristics.

Layer	Z Ohms	C pF/ in.
L8 (8 mils)	43	4.2
L9 (8 mils)	43	4.2

Recommendations

The following comments are included for your guidance in building VAXBI modules:

- Use the TTL-optimized layup as shown in ELEN 633. This layup is slightly less sensitive to routing density variations than the base layup and should meet the capacitance requirements for BI signal lines (according to data provided on test boards). This layup is also better balanced, both electrically and mechanically, since it is a symmetrical layup electrically.
- Due to ground-plane proximity, the impedance of inner layers (L3 and L8) is less sensitive to manufacturing and routing variations, but these layers have somewhat lower typical impedance. Hence, critical signals should be routed on these layers if the lower impedance can be tolerated. Critical lines should also be routed with more attention to surrounding lines in the same routing channel and on adjacent layers, if reasonably constant impedance is to be maintained. The impedance is affected both by capacitance of surrounding copper and by line-width etching variations caused by routing density variations. Manual routing of critical lines may be required, since this may be beyond the capabilities of automated routing. If lines are carefully and uniformly routed, impedance may then be estimated using programs such as H2D. Be sure to include two standard deviations of line width and dielectric spacing.
- Do not specify impedance-controlled modules. The modules cost extra and do not give added value.

A number of enhancements are possible which, while not required, are compatible with the goals of VAXBI module design. We mention several of these here for those designers wishing to go beyond the merely acceptable to the outstanding.

- Use a smaller hole than 0.032" +.005", −.016" for all logic vias, including those in the VAXBI Corner, so long as no intrusive-pin component is to be placed in them. A 20-mil finished hole (25-mil drill), with a 40-mil pad and a 70-mil clearance yields acceptable voltage drop. Be careful not to change the power and ground pins, which have a minimum drill size of 0.032".
- Delete nonfunctional pads, if the design system in use permits it. It is advantageous to mark all pin 1 holes with a square pad on layer 1 for ease of assembly. The ideal configuration of power and ground pads internally is a modified heat relief in the shape of an X, .016" wide and .096" diagonal. This pattern provides the optimum mix of heat relief and solderability. Figure 3-25 shows the recommended shape.
- Spread lines apart as far as possible. The yield statistics and signal integrity are best when spacings are the greatest that can be achieved given the density of the particular option.

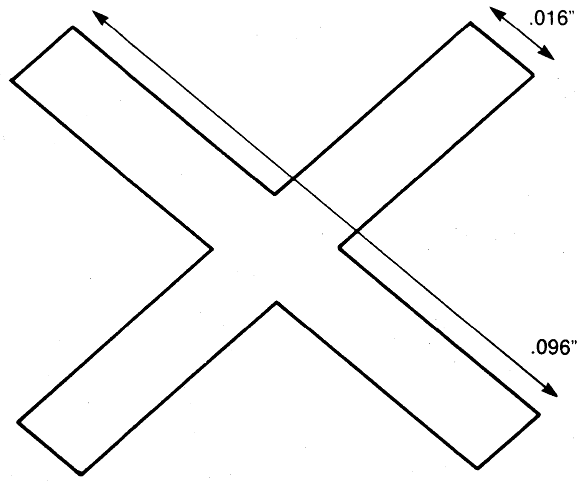


FIGURE 3-25 *Heat-Relief Pad Shape*

Some Commonly Asked Questions and Answers

Question: What are the ABSOLUTE MINIMUM requirements for a standard VAXBI module?

Answer: A module must meet the specifications given in the VAXBI SRM including protocol, speed, electrical, and mechanical requirements. Neither the BIIC nor the standard VAXBI module layout is "technically" required, but there is currently no way to build a VAXBI module without them.

Placement Problems

Question: What if the logic in my VAXBI option won't fit on this size module?

Recommendations:

- Micropackage logic as custom chips or gate arrays.
- Use standard VAXBI module plus VAXBI expansion module(s).
- Scale back the proposed functionality of the VAXBI module.

Not Advisable:

- Reorient the components. (Check with manufacturing before doing this. The costs may be substantial.)
- Go off the 0.1" grid and use more finely placed and routed packages. (Requires new technology not yet developed.)

Mechanically Not Feasible:

- Mount devices on both sides.
- Use a bigger board. (Won't fit in cage. May vary electrically. Corner is not engineered for a bigger board.)
- Violate the rim. (Card won't go in card guides.)

Question: What can I do to minimize the capacitance problem?

Answer: Use the 53 capacitance-restricted signals to drive a few low-capacitance devices, and keep them close to the boundary connection points around the VAXBI Corner.

Routing Problems

Question: What if I can't complete the signal routing?

Answer: Automatic routing experiments have indicated excellent routability of fully packed VAXBI modules on a 0.1" grid, using the 4 given signal layers and 2 lines between pads. However, if your design is not routable, we recommend using 3 lines between pads. (Beware of crosstalk effects.)

Question: What about capacitance?

Recommendations:

- On capacitance-restricted signals, use thinnest, shortest etch possible.
- Use layers 2 and 9 (far from ground plane) over layers 3 and 8 (close to ground plane).
- Delete nonfunctional pads.
- Spread lines apart, on same or adjacent layers.
- Modify placement of chips for shorter etch runs.

Connector Problems

Question: Can I remove unused connector segments or pads?

Answer: No. You will damage the connector in the card cage.

Electrical and Mechanical Problems

Question: My option includes a remote system with a different set of requirements. Can I avoid a VAXBI module?

Answer: No. The VAXBI is length-limited. Cable extensions are discouraged. Use a standard VAXBI module and run your cable from the C, D, or E connector segments.

Question: What is the rim restriction again?

Answer: 0.2" to any feature. This means pins on the grid must be 0.3" from the top or right, 0.28" from the left.

Question: Can I use more than 50 watts?

Answer: No. You would use power allocated for other options, and you may not be able to cool the module adequately.

Question: Can I use a bigger module?

Answer: No; it won't fit in a VAXBI cage.

Question: Can I use a smaller module?

Answer: No; how can you connect it?

Question: Can I use fewer layers to make my design cheaper?

Answer: Absolutely not. The VAXBI Corner cannot be modified in any way. Varying the number of layers changes the impedances presented to the bus which would invalidate the signal integrity.

Question: Can I connect to the VAXBI clocks on the expansion module?

Answer: NO! The clock signals have fixed impedance which permits a full 36-slot system to function. Changing the connection to plating bar or connecting to the clock signals or any other BI signal apart from the six permitted asynchronous signals is a violation of the specifications.

Question: What are the implications of deviating from the module layups presented in this guide?

Answer: We strongly recommend that you follow layups in ELEN 633. Deviation from these layups affects the electrical characteristics of the Corner. Common problems with variants from these layups are failure to comply with the VAXBI electrical specifications and excessive warping of the PCB during wave soldering in manufacturing, causing failure to comply with the VAXBI mechanical specifications.

Appendix A VAXBI BIIC Simulation: Physical Chip Modeling

This appendix gives requirements for physical chip modeling of the BIIC as part of the simulation of a VAXBI option. Using the guidelines presented in this application note, DIGITAL simulated the BIIC using a commercially available simulator with a physical modeling capability.

The VAXBI bus, which has been developed for the next generation of VAX systems, requires new design processes to cope with the increased complexity and to reduce time required for debugging of hardware prototypes. A key factor in providing "high-quality, fast development time" products is the use of logic simulators in the early stages of a product's design. Experiences have shown that a significant proportion of bugs that normally would be discovered at the first hardware prototype stage can be eliminated at the simulation stage of a design. Another benefit derived from simulating a design is the early involvement of diagnostic and microcode development groups on a project. In the past these groups could not debug their code until a hardware prototype was available. Simulations can also provide feedback on fault coverage.

Simulating the VAXBI and BIIC

To simulate a VAXBI option, a designer needs to produce a software model of the option. This model is usually built up by interconnecting several smaller models to produce a larger model. The usual method of producing a simulation model of an option is to produce a schematic of the design and model the integrated circuits within the design. Common devices such as 74S373 are easily modeled on simulators; however, the larger VLSI chips like the BIIC require very large complex models that are slow and only work on one simulator. The goal of DIGITAL's VAXBI simulation strategy was to permit an option designer to simulate the design including the BIIC and not be tied to a single simulator. To achieve this goal, DIGITAL chose to use a physical chip modeling approach.

This approach, which uses the physical device to provide the model for the simulator, has three advantages:

- Reducing the time required to produce an accurate model, which can be measured in days or at maximum weeks, whereas the time required to produce an accurate software model can take years.
- Increasing the speed of execution of the simulation. The speed of a physical model is greater than the speed of execution for the software model.
- Simplifying revision changes. Instead of issuing a new large software model the chip is simply replaced.

Any simulator with a physical modeling system that can provide the features described below can be used to model the BIIC.

Modeling the BIIC

Modeling the BIIC using a physical chip modeling system posed some unique problems associated with simulating a large VLSI chip by this technique. The common technique employed by these systems is to reset the device to a known state, apply the stimulus from the simulator, capture the results, and then pass this data back to the simulator. A cumulative history of the pattern stimulus is thus saved within the modeling system. This method has the following advantages:

- It permits dynamic devices to be modeled.
- It also permits multiple instances of a device in the simulation to be modeled by one physical device.

This method, however, suffers in terms of performance when the reset sequence is long. Also, if a uniqueness is loaded into the chip at reset time, then it is no longer possible to model multiple instances with one device.

The BIIC is an example of such a device. The BIIC has one major advantage in that it is a static device during normal operation and does not require a reset sequence. However, it does require clocks to run it, but they can run at very low speed. Its reset sequence takes approximately 5000 VAXBI cycles. The following requirements must be fulfilled to enable the BIIC to be modeled:

- The chip is dynamic during reset/self-test and must be clocked at a minimum clock rate greater than 1 MHz.
- Some custom logic is necessary to permit correct operation of the BIIC within a physical modeling system.
- All the BI lines must be pulled up by a 270-ohm resistor.
- The node ID is loaded during reset and self-test and must be presented on the BCI I lines for the cycle preceding the removal of BI DC LO L.

Generic BIIC Physical Model

It is not possible to provide a single solution that works for every physical chip modeling system in the marketplace; however, there are general requirements that can be defined to permit a successful model to be produced. Figure A-1 shows a block diagram of a BIIC physical chip model.

The clock logic is responsible for providing high-speed clocks to the BIIC when BI DC LO L is asserted. When BI DC LO L is deasserted, the chip performs a self-test. When the BCI EV codes show self-test passed, the hardware clocks are disabled and the simulator-generated clock is started. It is important at this stage that the transition from the hardware clocks to the software clocks be performed at a known point, because when multiple BIIC models are used, they must all be in phase with each other to permit the simulation to work. The node ID is gated onto the BCI I lines by using BCI DC LO L to enable a tri-state buffer. The node ID can be either a set of switches or signals passed to the hardware from the simulation. The latter method permits the node ID to be changed from the simulation environment. Note that all VAXBI signals (prefix "BI") are pulled up by a 270-ohm resistor.

Generic Simulator and Physical Chip Modeling System Requirements

A simulator must provide certain features to permit simulation of a two-node VAXBI system. It is necessary that the BIIC has certain input signals in a known state at the time when BI DC LO L is deasserted and self-test starts.

Required Simulator Features

To simulate the operation of two BIICs, it is necessary that the two chips can perform all the required BI cycles using the simulator. The activities that occur during an arbitration cycle are the most difficult with respect to the simulator's functional capabilities. During this cycle each BIIC asserts its decoded ID on the BI data lines; however, the simulator must make sure that each chip sees the other node's asserted ID. This is equivalent to supporting "wired AND" functionality in the simulator and the physical modeling system. The BCI D, I, and P0 signals also must be treated as "wired AND" signals. The ability to treat signals as "wired AND" is necessary because these signals have internal pullups associated with them. Although they are bidirectional tri-state lines, if they are described as tri-state to the chip modeling system, they never get detected as being in the high-impedance state because of the pullups internal to the BIIC. Consequently, the physical modeling system senses that the BIIC is always driving these signals and gives an error if any other tri-state device in the simulation attempts to drive these signals.

Both the simulator and physical chip modeling system must support:

- A mode in which the BIIC is treated as a static device.
- A capability to assert known values on multiple signals during reset. This is necessary to prevent the BIIC from going into an undefined mode at reset. (See the *VAXBI System Reference Manual*, Section 15.2.)

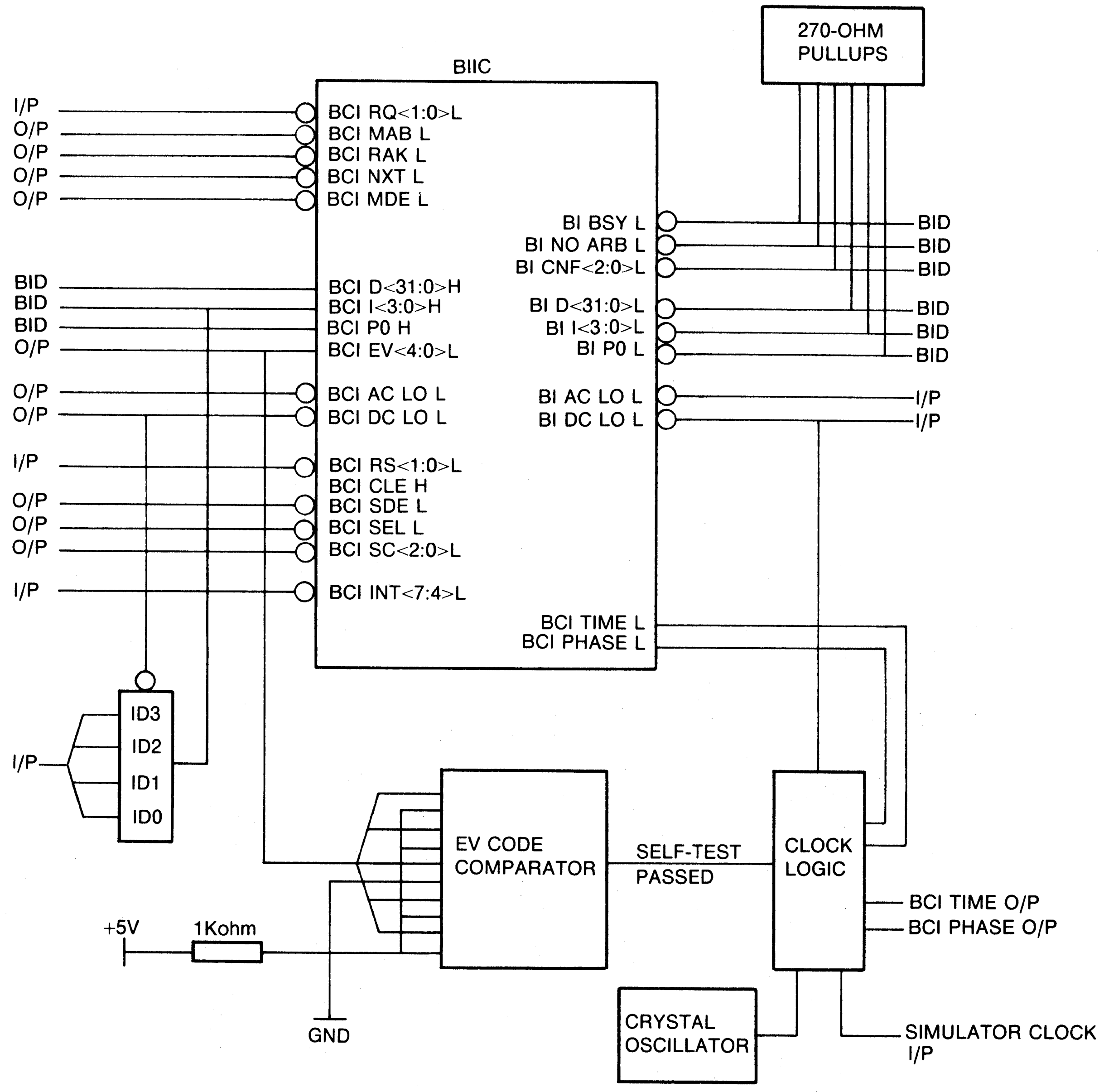
During node reset the following signal must be in the deasserted state (high):

- BCI RQ<1:0> L

If the BCI RQ<1:0> L lines are not deasserted, the BIIC does not perform self-test correctly and the results are undefined.

During node reset the following signals should be in the deasserted state (high):

- BCI MAB L
- BCI RS<1:0> L
- BCI INT<7:4> L



I/P = INPUT FROM THE SIMULATOR
O/P = OUTPUT TO THE SIMULATOR
BID = BIDIRECTIONAL WIRED AND

FIGURE A-1 *Block Diagram of Support Logic Required to Model the BIIC*

Chip Carrier Constraints

Most modeling systems have a standard carrier for mounting devices. The BIIC requires a special carrier to support the custom logic associated with handling reset operations and node ID setup. In addition, since these carriers are usually designed to permit microprocessor devices to be modeled, the power and ground capabilities are often inadequate for proper BIIC operation. The +5V power requirement for the BIIC, termination resistors, and external logic is approximately 10 watts. The BIIC requires approximately 3 watts which represents approximately 600 mA of current consumption. The recommended method of mounting is to have a large power and ground plane in the carrier and as much decoupling as possible on the +5V supply.

Appendix B VAXBI Base Layout Package

The following information is from the cover letter that is distributed with the VAXBI Base Layout Package.

This document describes the contents of the documents and magnetic tapes in this package. This package provides the necessary information to design and build either a Standard VAXBI Module or an Expansion VAXBI Module.

The following documents are supplied:

T1999	Standard Module Control Drawing
T1996	Expansion Module Control Drawing
ELEN 633	Layup Specification
ELEN 626	Mechanical Outline Specification

(References within these documents to DIGITAL internal documents are for the use of DIGITAL manufacturing. These documents are not needed by customers.)

The following magnetic tapes are supplied:

T1999A	Gerber Tape (10 artwork layers)
T1999D	Drill Tape
T1996A	Gerber Tape (10 artwork layers)
T1996D	Drill Tape

This package contains information for two types of VAXBI modules. T1999 and its associated databases are for the standard VAXBI module with a VAXBI Corner; T1996 and its associated databases are for the VAXBI expansion module. ELEN 626, ELEN 633, and the Gerber wheel description apply to both types of VAXBI modules. The Gerber tapes are for the 10 layers of the module and, when plotted, will produce the artwork for the DIGITAL-defined portions of a VAXBI module. The drill tapes contain the necessary data for drilling the holes in the DIGITAL-defined portions of the module. The precise format of the data on these tapes is described below.

The artwork for the module has added certain special features to assist PCB manufacturing. The connector area gold plating feature that DIGITAL manufacturing uses has been added; it can be removed and replaced if desired. Beware that if you supply your own plating feature and it does not work, you must not use the module in a VAXBI backplane because the module will damage the connector if it is not properly plated. Each layer has two targets on it to assist in alignment; DIGITAL's tooling features have not been added.

The intent is that a designer can produce boards either by reading the Gerber data into a CAD system as it is provided to produce a layout or by plotting the artwork and then digitizing it back into the CAD system. (If you intend to do the latter, you should plot the artwork at 2:1 at least. Feedback from users has indicated that 4:1 is preferable.)

Note: A manufacturing problem can exist when removing the finished PCB from the panel in which it was etched. Because some shearing equipment cannot hold the necessary tolerances on the module outline, we recommend routing the PCB out of the panel in these cases.

Gerber Tape Format

The decision to use Gerber tape format for the artwork was made to give the best coverage in terms of photo-plotting systems in use. The format on tape can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999A, T1996A)

ASCII format, absolute coordinates

Coordinate format 4.4 (no leading zeros)

All coordinates are positive X and positive Y. The coordinates are referenced to the Gerber plotter origin, but the artwork is offset by 1.1" in the X direction and 1.7" in the Y direction (DIGITAL manufacturing's standard offset for VAXBI modules). The artwork produced is data as viewed from side 2. The first five layers are wrong reading; the bottom five layers are right reading. It is recommended that you verify that the data does not require a coordinate translation for the plotter system in use. (Note that all pads and traces will be plotted at finished line width; look in T1999/T1996 for line widths.)

The Gerber tape format used works on many Gerber photo-plotters; however, many different plotters exist and several versions of software run on the plotter controllers.

Two problems can occur when attempting to photo-plot the layers.

- The file format is ANSI standard X3.27-1969, which means that most Gerber controllers see ANSI file format information as plot files and consequently give errors. The rule for determining the file which contains a given layer plot file is: F = 3L − 2, where F is the file number to plot and L is the layer to be plotted. For example, to plot layer 6, request the Gerber controller to plot file 16.
- The D-codes for changing the aperture in use are not preceded by a G54 tool select code. If your Gerber photo-plotter controller requires a G54 select code, you have two alternatives:
    - Read the photo-plot data into your VAX and write a program to detect all D-codes greater than or equal to 10 and precede these D-codes with a G54 select code.
    - Request a version of software from Gerber that assumes a G54 whenever a D-code is encountered.

Drill Tape Format

The drill tape format was chosen to give drill information in a format that can be translated into any of the other standard formats in use. The tape format can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999D, T1996D)

Excellon 0 reference, absolute ASCII with ASCII format leaders

The finished hole size is given in the ASCII format leaders. If your manufacturing process requires compensation on drill size, you must change the size in the ASCII format leaders. The data is in the form of absolute coordinates from the first hole. If you wish to read this data into your CAD system, then you need to shift the coordinates by the coordinate of the first hole from the origin that was used for the artwork data. The coordinates are as follows:

For T1999 drill data:

X offset = 1300
Y offset = 1900

For T1996 drill data:

X offset = 1300
Y offset = 1900

Gerber Wheel Data

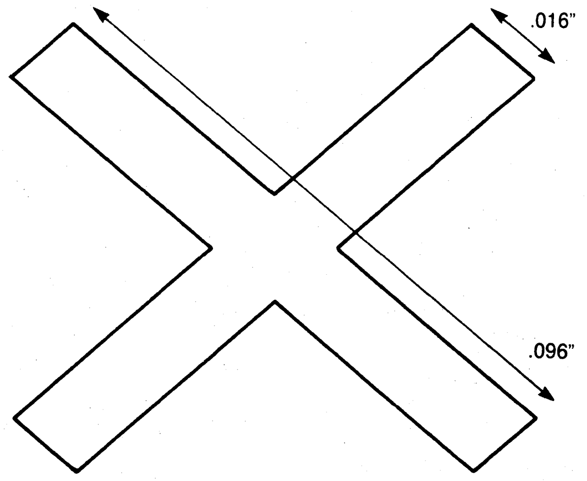
The following table gives details of the Gerber wheel required to photo-plot the Gerber tapes.

Type	Dimension/ Shape	D Code	Position
	100L	D19	10
	50L	D70	11
	25L	D71	12
Line apertures	18L	D13	4
	15L	D21	14
	12L	D23	16
	6L	D72	23
	5L	D29	22
	187C	D10	1
	80C	D14	5
	80S	D16	7
Pad apertures	70C	D15	6
	60C	D27	20
	60S	D18	9
X heat relief	16 x 96X	D73	24

Shape codes:

L	Line aperture
C	Circular pad
S	Square pad
X	X-shaped heat relief pad

The following figure shows the shape of the heat relief pad.



Index

UPDATE NOTICE NO. 1

VAXBI Designer's Notebook

EK-VBIDS-R1-001

This update contains three new chapters and one appendix. The pages enclosed in this package are listed below:

Title page/copyright page
iii/iv
v/blank
4-1 through 4-6
5-1 through 5-12
6-1 through 6-6
C-1 through C-3/blank
Index I-1 through I-3/blank

Digital Equipment Corporation • Maynard, Massachusetts

VAXBI DESIGNER'S NOTEBOOK

EK-VBIDS-RM-001

January 1987
This manual contains Update Notice 1, EK-VBIDS-R1-001.

Digital Equipment Corporation • Maynard, Massachusetts

First Printing, January 1986
Updated, January 1987

The manuscript for this book was created using generic coding and, via a translation program, was automatically typeset. Book production was done by Educational Services Media Communications Group in Bedford, MA.

DEC	MASSBUS	UNIBUS
DECmate	PDP	ULTRIX
DECnet	P/OS	VAX
DECUS	Professional	VAXBI
DECwriter	Rainbow	VAXELN
DIBOL	RSTS	VMS
digital™	RSX	VT
	RT	Work Processor

Contents

Preface

The purpose of this notebook is to provide a focus for those who are involved with designing options for the VAXBI bus. This book attempts to put the *VAXBI System Reference Manual* into perspective, explaining which portions of the VAXBI specification option designers need to be most concerned about. Some portions deal with requirements that are implemented by DIGITAL-supplied hardware. The option designer need only use the specified hardware to comply with many of the requirements that are detailed in the comprehensive specification.

With the proper perspective, designers can understand more quickly what they need to do. To help them understand *how* to go about their task, we will provide design examples. In addition, as design tools become available we will include commentary on their use.

Intended Audience

This notebook is primarily for engineers who design options for VAXBI systems. System architects and others who want to understand DIGITAL's design philosophy for the bus will also find this information useful. Chapter 3, which is a guide to module layout, should be read and studied by module layout designers; this chapter also includes information pertinent to manufacturing.

Structure of This Manual

Chapter 1, Introduction to VAXBI Option Design, provides an overview of the VAXBI bus and of the documentation from an option designer's point of view. The chapter also poses design issues and gives hints for designing options.

Chapter 2, The Instrument Control Adapter, is an example of a VAXBI option design. The design requirements and the resulting design decisions are described. The option is a master-port-only design.

Chapter 3, VAXBI Module Layout Guide, serves as a guide for engineers and module layout designers as they design options and build VAXBI modules. It points out problem areas and reports some of DIGITAL's experiences. References are made to the appropriate module control drawings.

Chapter 4, Migrating Designs to the VAXBI, describes how the VAXBI design philosophy differs from the approaches used for UNIBUS and Q-bus designs. It looks at the functions of the UNIBUS-to-VAXBI adapter to show what a VAXBI option must do.

Chapter 5, Realchip BIIC Model, is a description of Valid Logic Systems' hardware modeling product that is used in simulating designs that use the BIIC chip.

Chapter 6, VAXBI Debug Tool: The DAS91VB, describes a debug tool developed by Tektronix. The DAS91VB is a logic analyzer with two custom VAXBI modules that can be used to verify VAXBI boards during prototype design and manufacturing test and to diagnose VAXBI systems in the field.

Appendix A, VAXBI BIIC Simulation: Physical Chip Modeling, gives requirements to permit physical chip modeling of the BIIC for simulation of a VAXBI option.

Appendix B, VAXBI Base Layout Package, gives information on the documentation and databases in the VAXBI Base Layout Package.

Appendix C, Transition Header Interconnects, provides information needed to design and build cabling interconnects to the VAXBI card cage transition headers.

Related Documentation

- *VAXBI System Reference Manual* – The specification for the VAXBI bus and the VAXBI primary interface (the BIIC)
- T1999 – Module Control Drawings for the standard VAXBI module
- T1996 – Module Control Drawings for the VAXBI expansion module
- ELEN 626 – Mechanical outline drawing for the standard VAXBI module
- ELEN 633 – VAXBI module layup specification

The following document provides information about VAX architecture.

- *VAX-11 Architecture Reference Manual*

Chapter 4

Migrating Designs to the VAXBI

by VAXBI Development Group

This chapter describes how the VAXBI design philosophy differs from the approaches used for UNIBUS and Q-bus designs. It looks at functions of the UNIBUS-to-VAXBI adapter to show what a VAXBI option must do. From experience with VAXBI designs, DIGITAL alerts designers to problems of designing to the VAXBI bus.

Contents

Reasons for VAXBI Option Design Philosophy

From our design experience with VAXBI options, we learned how design decisions impact the hardware, firmware, and driver software. While it is not possible to specify exactly the impact of design choices, we present overall tradeoffs for specific types of design. Our comments address the BIIC interface and not the user interface portion of designs.

I/O Adapter Interface Model

An I/O adapter can be represented by the simple architectural model shown in Figure 4-1.

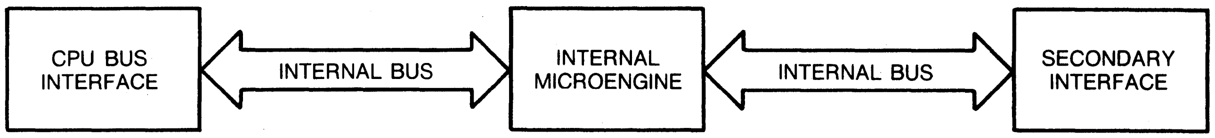


FIGURE 4-1 *Architectural Model of an Adapter*

This diagram represents typical UNIBUS or Q-bus adapters. The secondary interface could be any device that the user wishes to interface to; for example, an RS-232 asynchronous interface for terminals or a parallel port. The internal microengine can vary from standard microprocessors such as a Z80 or 68000 to bit-slice processors such as the 2901 series.

The VAXBI interface model shown in Figure 4-2 is similar to the model shown in Figure 4-1 except that the BIIC and the BCI interface take the place of the CPU bus interface.

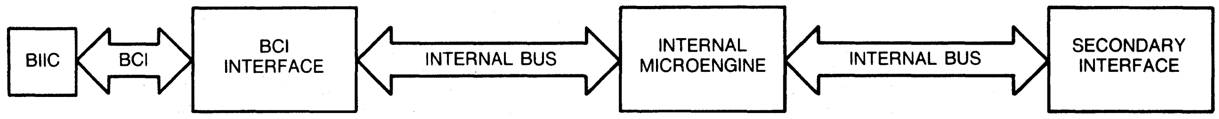


FIGURE 4-2 *Architectural Model of a VAXBI Adapter*

The two designs differ in that the VAXBI designer does not have to implement the VAXBI protocols, since these are implemented by the BIIC. The designer's task is to develop a BCI interface which is much easier to do. In VAXBI designs the BIIC prevents the designer from causing bus problems. This does not mean that system performance will not be affected by a badly designed adapter, but because the BIIC is the primary VAXBI interface, a poor design cannot hang the VAXBI bus.

Migration of Designs from the UNIBUS and Q-bus to the VAXBI Bus

From the two models presented above, one might assume that the migration of UNIBUS and Q-bus designs to the VAXBI bus would be easy. All that seems to be required is to replace the CPU bus interface with a BCI interface.

However, this is not the case, since the BCI interface is not as simple as it appears to be in the diagram.

One complicating factor is that the VAXBI is a synchronous bus, whereas the UNIBUS and Q-bus are asynchronous buses. Moreover, the VAX architecture does not require that pages in memory be physically contiguous. The UNIBUS adapter handles this problem with the use of map registers which permit data transfers to noncontiguous pages in memory. The UNIBUS adapter also contains the necessary functionality to realign data and buffer it into longwords or octawords before transferring the data. When a UNIBUS design is to be implemented on the VAXBI, the designer must consider how to implement these features that the UNIBUS adapter handled before.

Address Translation

With a UNIBUS design on a VAX system, the CPU handled the virtual to physical address translation for DMA controllers by using map registers in a UNIBUS adapter. The use of map registers permitted the 18-bit UNIBUS addresses to be mapped into the larger VAX address space of 30 bits. In addition, map registers overcome the problem of noncontiguous pages in memory. A VAXBI adapter can address the full 30 bits of address space.

There are many ways that the adapter can translate the 32-bit virtual address into a 30-bit physical address. This translation depends on the amount of hardware and firmware support in the adapter. Following are the three major methods for performing this translation:

- Short transfers that do not cross physical page boundaries
- Registers that map contiguous virtual addresses to noncontiguous physical addresses
- Virtual to physical translation by the adapter using the page table entries

Clearly, each of these options requires changes in the VMS driver that supports the device, and the degree of sophistication of the hardware and firmware depends on the choices made. The amount of the translation performed by the adapter reduces the CPU overhead associated with address translation.

Table 4-1 summarizes tradeoffs that should be considered in evaluating how to perform address translation in VAXBI adapter designs. For each adapter type, comments are made on the impact of the basic architecture of the design upon hardware, firmware, and software. Table 4-2 summarizes tradeoffs of how to handle mapping registers, and Table 4-3 presents options of doing address calculations.

Table 4-1 Address Translation Strategies

Adapter Type	Hardware Impact	Firmware Impact	Software Impact
Master/slave with basic DMA	Complex BCI controller; simple DMA controller	Similar to UNIBUS design; DMAs limited to 512 bytes; device must interrupt CPU for next physical address for DMAs greater than 512 bytes	Similar to UNIBUS driver; must segment DMA into physical pages
Master/slave with mapped DMA (see Table 4-2)	Complex BCI controller; more hardware for DMA controller	Firmware has to be aware of map translation	Driver similar to UNIBUS version
Master/slave with virtual to physical translation in adapter	Complex BCI controller; more hardware for DMA controller	Firmware very complex; has to support VAX address translation	Driver is simplified; virtual to physical translation is offloaded to adapter

Table 4-2 Tradeoffs for Map Register Options

Map Register Option	Hardware Impact	Firmware Impact	Software Impact
Hardware map registers in adapter	Requires master/slave design; complex BCI interface	Firmware controls DMA and address translation	Driver uses central routines to generate map register contents
Map registers in main memory	Master-port design is possible; simpler hardware	Firmware fetches map registers from main memory	Driver must allocate area of main memory to hold map registers; can still use central mapping routines to generate map register data*

*For these options it is assumed that DIGITAL's standard BUA map-register format is in use. If not, the user must supply the routines to generate the user's map register format.

Table 4-3 Address Calculation Options

Calculation Technique	Hardware Impact	Firmware Impact
Firmware performs address calculation	Requires minimum hardware; slower than using hardware	Firmware performs address calculation
Hardware ALU performs address calculation	Requires board real estate to implement ALU; highest speed	Less firmware overhead for DMA; firmware is simplified

Data Path Considerations

Data is passed across the VAXBI bus in 32-bit words. If the user interface requires the data in 16-bit words or in 8-bit bytes, the data will need to be multiplexed/demultiplexed depending on transfer direction. However, another problem needs to be overcome: If a block of longwords is to be transferred from main memory to the user interface, the VAX architecture permits data in memory to be aligned on any byte boundary. If the data has to be presented to the user interface in either 16- or 32-bit or greater widths, then the data needs to be fetched and realigned. On VAX systems this process is called byte alignment. To help designers perform byte alignment, a custom VAXBI chip was designed, the BCAI, that contains a byte aligner. Designers whose designs require byte alignment should consider using the BCAI chip.

On UNIBUS systems the odd-byte bit in a map register was the means by which UNIBUS 16-bit words were realigned for transfer to main memory, so a UNIBUS design cannot migrate directly because the VAXBI adapter would need to perform data alignment.

Efficient Use of Bandwidth

DIGITAL's UNIBUS-to-VAXBI adapter takes advantage of the VAXBI bandwidth by performing octaword reads and writes of main memory whenever possible. This operation is transparent to the UNIBUS device which transfers 16-bit words across the UNIBUS, which are then realigned and packed into octawords. A VAXBI adapter design that used only longword transfers or—even worse—longword masked transfers of 16-bit words would be making poor use of available bus bandwidth.

The VAXBI bus protocol has been designed to give optimum bus bandwidth when transactions are performed in octaword quantities. During data transfers, all memory reads should be octawords. When writing to memory, an option should perform a masked octaword write to get the physical address on an octaword boundary and then perform octaword transfers until the last data transfer, which should be octaword masked if necessary.

Physical Implementation

The physical size of VAXBI modules is an important consideration in the design of a VAXBI option. An existing UNIBUS design that fit on a standard hex module may not fit on a single VAXBI module. At least one expansion module will be required if the logic is not repackaged. The tradeoff between multiboard options and single-board options is highly cost sensitive. If the product is to be shipped in volume, the PCB cost is a dominant factor, and one rather than two PCBs is clearly more cost effective. However, to produce a one-board option may require micropackaging logic in gate arrays, a lengthy and costly process. Furthermore, the design tools required for complex, one-board options increase in complexity. If gate arrays are used, simulation is highly recommended. The layout time required to rout a dense PCB is greater than that for a less dense, UNIBUS option. If the option is to be a low-volume, high-cost option, then the PCB costs may not be dominant, and multiboard, less dense, designs may be preferable.

VAXBI I/O Adapter Types

The interface between the user logic and the BIIC is the BCI, which has two ports available to the user. Designs can be classified on the basis of their use of these ports, the slave port and master port.

The slave port consists of the BCI signal lines used by a node in responding to transactions targeted to it. The slave port interface (user interface logic) responds to read and write requests to this node's address space.

The master port consists of the BCI signal lines that a node uses to to generate transactions. The transactions can be directed to other nodes (internode transactions) or to the node that issued the transaction (intranode transactions).

Adapters can therefore be grouped as follows:

- Slave port only
- Master/slave
- Master port only

Slave-port-only options are options that only transmit or receive data across the VAXBI when requested to by a master node. A typical example is a memory node that does not initiate transactions but only executes commands. Slave-port-only designs, because of their limited use, are not discussed further.

Master/slave port adapters implement both the slave and master port features of a VAXBI module. Adapters of this type can be requested by a master node to participate in a data transfer, or they can act as a master and request data transfers to and from main memory. Typical examples of this type of adapter are the UNIBUS-to-VAXBI adapter and the DMB32 communications adapter.

Master-port-only adapters do not implement a slave port. Although they need a certain amount of slave functionality, all that they require is provided by the BIIC—in particular, by the BIIC's four General Purpose Registers. Consequently, master-port-only adapters do not need any user-implemented slave functions. Adapters of this type usually fetch commands in the form of packets in memory and execute the commands by performing a series of DMA transfers to and from main memory.

DIGITAL's experience in designing VAXBI adapters has shown that the BCI interface of a master/slave port adapter is very complex to design. The preferred design architecture for I/O adapters is the master-port-only architecture, which greatly simplifies the hardware design. However, this approach influences the software driver. Instead of writing to physical registers in the hardware, the driver constructs a packet in memory which the hardware fetches from memory and implements the commands in the packet.

Chapter 5
Realchip BIIC Model

by Valid Logic Systems, Inc.

This chapter describes the Realchip BIIC model and its use in simulating designs that use the BIIC chip. Extensive testing by DIGITAL has proved that the model accurately represents the functioning of the BIIC in simulated designs.

Contents

Hardware Modeling for Simulation of VAXBI Designs

As designs become larger and more complex, the use of logic simulation can dramatically shorten product design cycles, thereby allowing systems manufacturers to get their products to market more quickly and at lower cost. Logic simulation helps designers choose the best design approach from the many alternatives available to them. It helps them eliminate hardware design errors before the design is ever committed to prototype hardware, thus eliminating costly, time-consuming, and error-prone breadboarding. And as new complex devices appear on the market, logic simulation can be used to help understand the operation of these devices even before the design is started.

Until Valid Logic Systems, Incorporated, introduced Realchip* hardware modeling in March 1984, logic simulation was limited to designs that could be completely modeled in software. Each device in a design had to have a corresponding software model; otherwise, the design could not be simulated. Software models, however, are not practical for complex VLSI devices such as the BIIC chip. Creating such a software model often requires several man-years of effort, with detailed knowledge of the device that often only the chip designers can provide. Furthermore, simulation performance using software models is very slow, because each gate of the software model must be evaluated. In hardware modeling, on the other hand, the device itself is the model, so the simulation operates at hardware speeds instead of software speeds.

A VAXBI option designer attempting to verify an option design that includes a BIIC needs the following items (Valid's products are given in parentheses):

- BIIC model
- Hardware modeling subsystem (Realchip or Realmodel)
- CAE system (Validation Designer)
- Logic simulator compatible with CAE system (ValidSIM)
- Libraries for other components used

* Realchip, Realmodel, Validation Designer, and ValidSIM are trademarks of Valid Logic Systems.

Design Goals for the BIIC Model

Functions of the BIIC influenced the design goals for the BIIC model. Operations involving the BIIC can be short (such as individual word transfers onto and off the VAXBI bus) or very long (such as multiword DMA transfers). Additionally, many VAXBI devices can exchange data with each other because of the VAXBI's multinode operation capability.

A key requirement was to minimize or eliminate any extra work the designer has to do in the simulation process.

With these considerations in mind, Valid set the following objectives in developing the BIIC hardware model:

- Provide a turnkey solution, including software and hardware, that can immediately be used by the designer after installation in a CAE system; also, make the use of hardware modeling transparent to the user.
- Provide the capability to simulate both individual bus transfers and arbitrarily long operations (such as DMA block transfers).
- Maximize performance to allow simulation of these long operations to occur in a reasonable amount of time.
- Facilitate multinode simulation so that designers can simulate the interaction of many different BIIC-based modules in a system simulation environment.
- Provide models compatible with board-level simulation (Realchip) and system-level simulation/virtual software-hardware integration (Realmodel).

BIIC Hardware Model

The Realchip BIIC model consists of three parts, described below.

BIIC Personality Module

The BIIC sits on a 3" X 12" multilayer PC board called the "BIIC personality module" (see

Figure 5-1). The board has a zero insertion force (ZIF) PGA socket for the BIIC, DIP sockets for the VAXBI clock driver and receiver set, and additional support logic. This module defines what the chip does during logic simulation. The personality module is installed within the Realchip or Realmodel modeling system, making it available for design simulation.

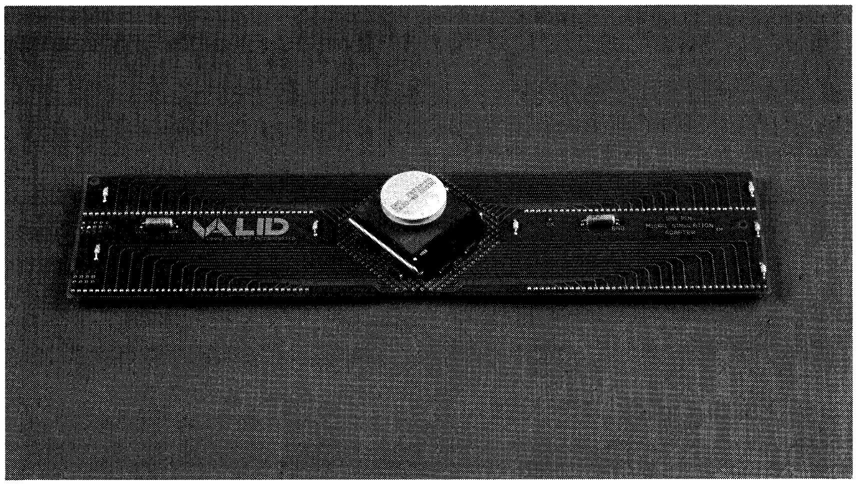


FIGURE 5-1 *Personality Module*

The personality module, consisting of a BIIC and support logic, serves as the functional model of the BIIC. The functional model defines what the BIIC does during logic simulation.

The support circuitry on the personality module performs the following functions:

- Static simulation of the BIIC
- Fast self-test due to hardware clock during node reset
- Multi-BIIC simulation capability due to phase synchronization circuitry
- Visual self-test passed indicator (a green LED)

BIIC Body Drawing

The body drawing is the symbolic representation of the BIIC that is used in schematic diagrams (see Figure 5-2). When the user is creating the schematic for the design (using the schematic editor of a CAE system), the BIIC symbol appears on the screen under the cursor in response to the command "Add BIIC." The user then places the body on the schematic where desired and then connects its signals to other components in the schematic with the "Wire" command.

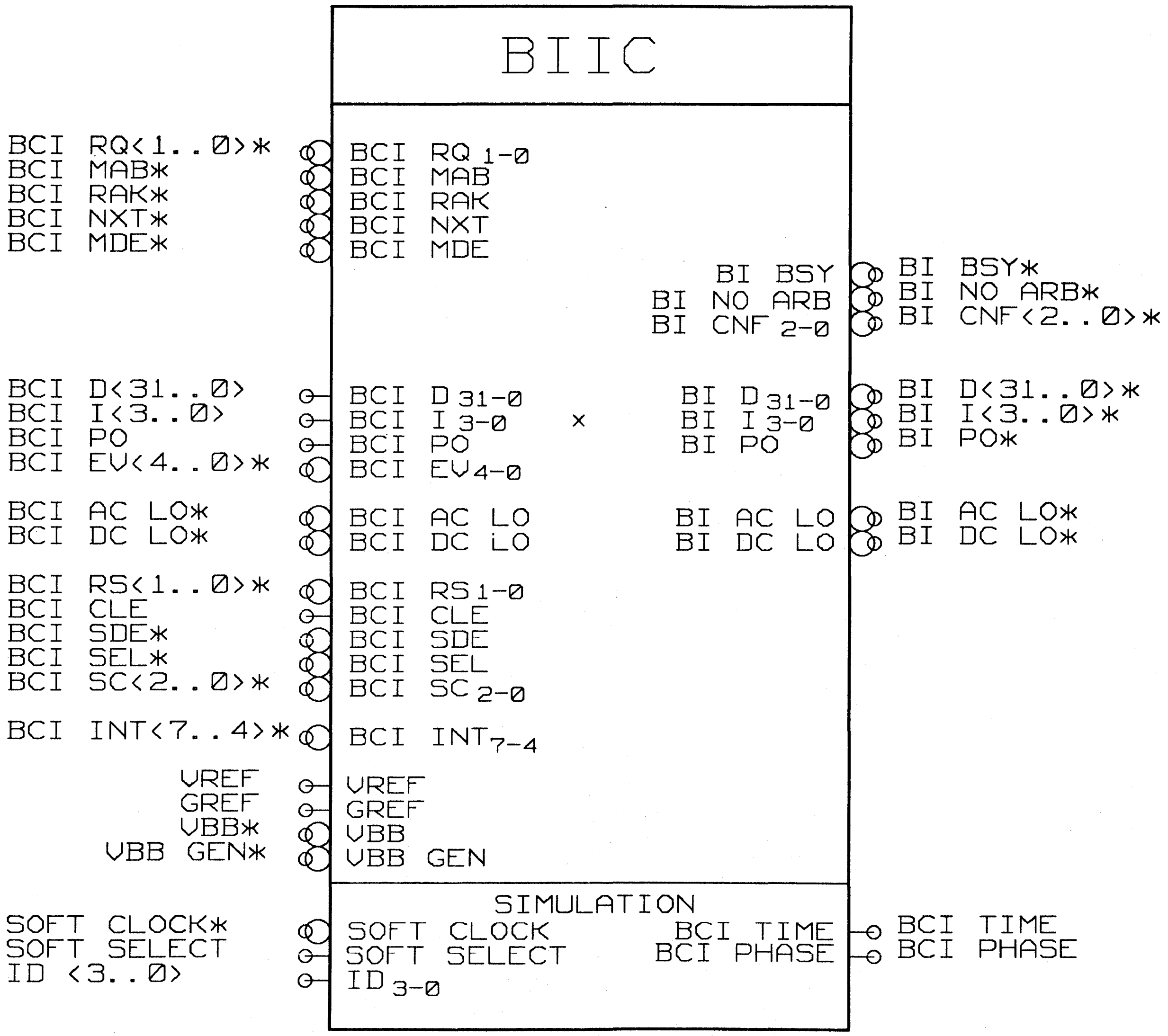


FIGURE 5-2 *BIIC Body Drawing*
The body drawing is the symbolic representation of the BIIC part that is used in schematic diagrams.

Figure 5-3 shows the functions of the support logic. Whereas the BIIC device has two clock pins, BCI TIME* and BCI PHASE*, the simulation model has inputs—SOFT CLOCK, SOFT SELECT, ID<3..0>—and outputs—BCI TIME and BCI PHASE.

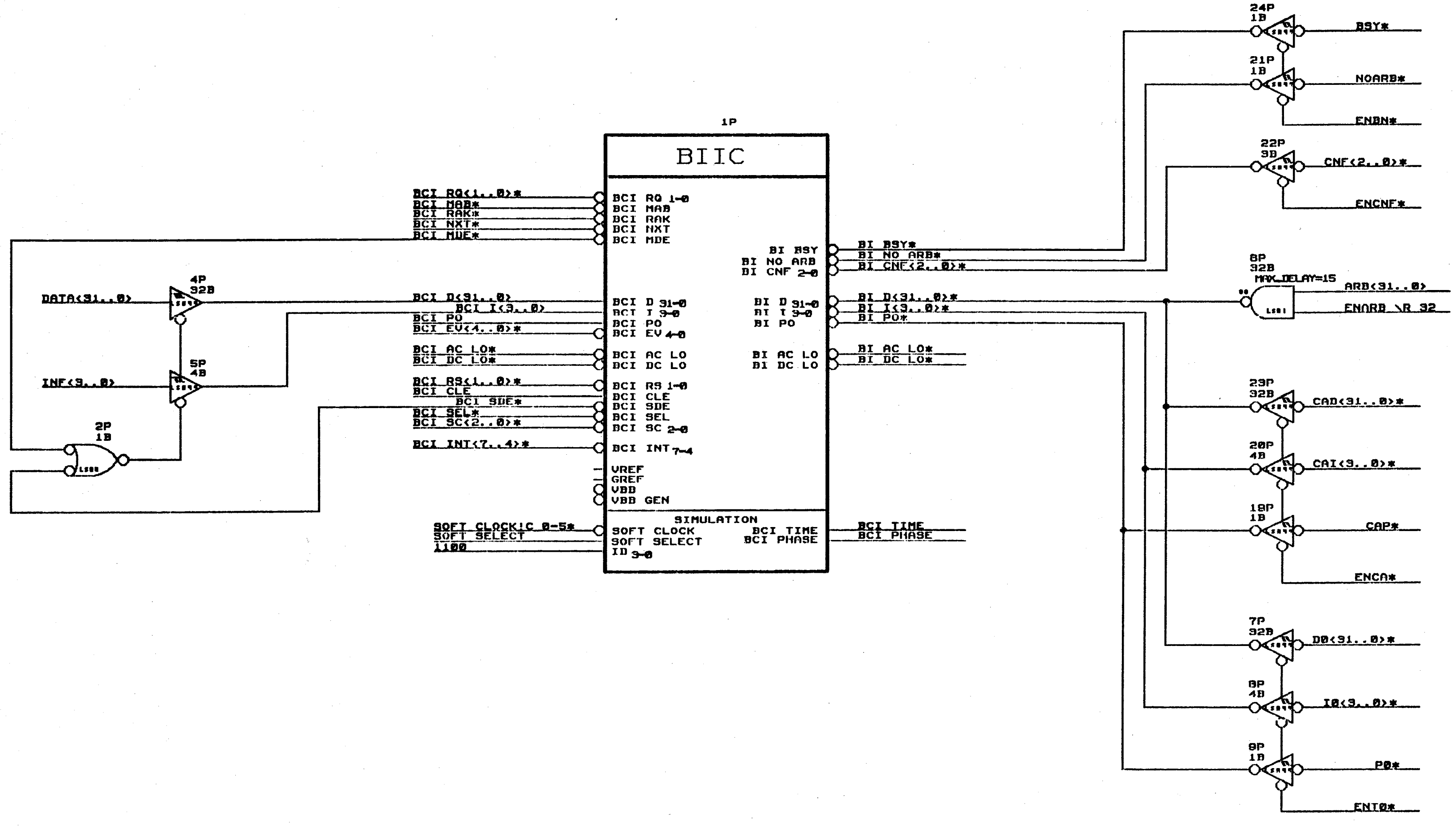


FIGURE 5-3 *Schematic of BIIC Model with Support Logic*
The BIIC hardware model uses support logic to synchronize the BIICs as they complete self-test and to supply the node IDs.

Device Definition File

The BIIC definition file, shown in Figure 5-4, maps device signal names to the personality module's pins. The file also defines worst-case timing characteristics for logic simulation waveform generation. It does these tasks automatically, so that the designer can treat the VAXBI BIIC model the same as any other software model. Therefore, the designer does not need to know the operation of the hardware modeling system to use the BIIC in a design simulation.

```
primitive 'BIIC';
{Specs from VAXBI  System Reference Manual : March 1985}

number_dev_pins = 128;

  pin
    input_spec =
        '-BCI RQ'<1>      pin=114,
        '-BCI RQ'<Ø>      pin=1Ø9,
        '-BCI MAB'        pin=1Ø7,
        '-BCI RS'<1>      pin=113,
        '-BCI RS'<Ø>      pin=116,
        '-BCI INT'<7>     pin=1Ø5,
        '-BCI INT'<6>     pin=1Ø6,
        '-BCI INT'<5>     pin=11Ø,
        '-BCI INT'<4>     pin=112,
        '-SOFT CLOCK'     pin=111 : strobe_pin,
        'SOFT SELECT'     pin=117,
        '-BI AC LO'       pin=87,
        '-BI DC LO'       pin=1Ø3,
        'ID'<3>           pin=124,
        'ID'<2>           pin=13Ø,
        'ID'<1>           pin=135,
        'ID'<Ø>           pin=98,
 '-VBB'           pin=152 : nc_pin,
 '-VBB GEN'       pin=7   : nc_pin,
 'GREF'           pin=72  : nc_pin,
 'VREF'           pin=75  : nc_pin;

    io_spec =
          'BCI D'<31>      pin=47 : output_type=(oc,and),
          'BCI D'<3Ø>      pin=51 : output_type=(oc,and),
          'BCI D'<29>      pin=45 : output_type=(oc,and),
          'BCI D'<28>      pin=5Ø : output_type=(oc,and),
          'BCI D'<27>      pin=48 : output_type=(oc,and),
          'BCI D'<26>      pin=43 : output_type=(oc,and),
          'BCI D'<25>      pin=42 : output_type=(oc,and),
          'BCI D'<24>      pin=44 : output_type=(oc,and),
          'BCI D'<23>      pin=15Ø : output_type=(oc,and),
          'BCI D'<22>      pin=148 : output_type=(oc,and),
          'BCI D'<21>      pin=143 : output_type=(oc,and),
          'BCI D'<2Ø>      pin=155 : output_type=(oc,and),
          'BCI D'<19>      pin=153 : output_type=(oc,and),
          'BCI D'<18>      pin=146 : output_type=(oc,and),
          'BCI D'<17>      pin=141 : output_type=(oc,and),
          'BCI D'<16>      pin=151 : output_type=(oc,and),
          'BCI D'<15>      pin=147 : output_type=(oc,and),
          'BCI D'<14>      pin=149 : output_type=(oc,and),
          'BCI D'<13>      pin=139 : output_type=(oc,and),
          'BCI D'<12>      pin=144 : output_type=(oc,and),
          'BCI D'<11>      pin=142 : output_type=(oc,and),
          'BCI D'<1Ø>      pin=14Ø : output_type=(oc,and),
          'BCI D'<9>       pin=138 : output_type=(oc,and),
          'BCI D'<8>       pin=136 : output_type=(oc,and),
          'BCI D'<7>       pin=137 : output_type=(oc,and),
          'BCI D'<6>       pin=131 : output_type=(oc,and),
          'BCI D'<5>       pin=134 : output_type=(oc,and),
          'BCI D'<4>       pin=132 : output_type=(oc,and),
```

FIGURE 5-4 *Device Definition File (Sheet 1 of 3)*

The device definition file relates signal names on the body drawing to pin numbers on the personality module and defines worst-case timing characteristics for logic simulation waveform generation.

```
            'BCI D'<3>        pin=129 : output_type=(oc,and),
            'BCI D'<2>        pin=127 : output_type=(oc,and),
            'BCI D'<1>        pin=128 : output_type=(oc,and),
            'BCI D'<Ø>        pin=125 : output_type=(oc,and),
            'BCI I'<3>        pin=126 : output_type=(oc,and),
            'BCI I'<2>        pin=123 : output_type=(oc,and),
            'BCI I'<1>        pin=122 : output_type=(oc,and),
            'BCI I'<Ø>        pin=91 : output_type=(oc,and),
            'BCI PO'          pin=86 : output_type=(oc,and),
            '-BI BSY'         pin=6 : output_type=(oc,and),
            '-BI NO ARB'      pin=119 : output_type=(oc,and),
            '-BI CNF'<2>      pin=8 : output_type=(oc,and),
            '-BI CNF'<1>      pin=11 : output_type=(oc,and),
            '-BI CNF'<Ø>      pin=15 : output_type=(oc,and),
            '-BI D'<31>       pin=58 : output_type=(oc,and),
            '-BI D'<3Ø>       pin=6Ø : output_type=(oc,and),
            '-BI D'<29>       pin=56 : output_type=(oc,and),
            '-BI D'<28>       pin=62 : output_type=(oc,and),
            '-BI D'<27>       pin=67 : output_type=(oc,and),
            '-BI D'<26>       pin=64 : output_type=(oc,and),
            '-BI D'<25>       pin=69 : output_type=(oc,and),
            '-BI D'<24>       pin=63 : output_type=(oc,and),
            '-BI D'<23>       pin=71 : output_type=(oc,and),
            '-BI D'<22>       pin=68 : output_type=(oc,and),
            '-BI D'<21>       pin=73 : output_type=(oc,and),
            '-BI D'<2Ø>       pin=7Ø : output_type=(oc,and),
            '-BI D'<19>       pin=66 : output_type=(oc,and),
            '-BI D'<18>       pin=74 : output_type=(oc,and),
            '-BI D'<17>       pin=39 : output_type=(oc,and),
            '-BI D'<16>       pin=29 : output_type=(oc,and),
            '-BI D'<15>       pin=37 : output_type=(oc,and),
            '-BI D'<14>       pin=38 : output_type=(oc,and),
            '-BI D'<13>       pin=33 : output_type=(oc,and),
            '-BI D'<12>       pin=36 : output_type=(oc,and),
            '-BI D'<11>       pin=31 : output_type=(oc,and),
            '-BI D'<1Ø>       pin=34 : output_type=(oc,and),
            '-BI D'<9>        pin=26 : output_type=(oc,and),
            '-BI D'<8>        pin=3Ø : output_type=(oc,and),
            '-BI D'<7>        pin=32 : output_type=(oc,and),
            '-BI D'<6>        pin=18 : output_type=(oc,and),
            '-BI D'<5>        pin=27 : output_type=(oc,and),
            '-BI D'<4>        pin=25 : output_type=(oc,and),
            '-BI D'<3>        pin=23 : output_type=(oc,and),
            '-BI D'<2>        pin=21 : output_type=(oc,and),
            '-BI D'<1>        pin=19 : output_type=(oc,and),
            '-BI D'<Ø>        pin=14 : output_type=(oc,and),
            '-BI I'<3>        pin=2Ø : output_type=(oc,and),
            '-BI I'<2>        pin=17 : output_type=(oc,and),
            '-BI I'<1>        pin=1Ø : output_type=(oc,and),
            '-BI I'<Ø>        pin=13 : output_type=(oc,and),
            '-BI PO'          pin=12 : output_type=(oc,and);

    output_spec =

            '-BCI RAK'        pin=95,
            '-BCI NXT'        pin=88,
            '-BCI MDE'        pin=1Ø4,
            '-BCI EV'<4>      pin=97,
            '-BCI EV'<3>      pin=92,
            '-BCI EV'<2>      pin=1Ø2,
            '-BCI EV'<1>      pin=9Ø,
            '-BCI EV'<Ø>      pin=94,
            '-BCI AC LO'      pin=89,
            '-BCI DC LO'      pin=99,
            'BCI CLE'         pin=93,
            '-BCI SDE'        pin=1Ø1,
            '-BCI SEL'        pin=57,
            '-BCI SC'<2>      pin=59,
            '-BCI SC'<1>      pin=54,
            '-BCI SC'<Ø>      pin=49,
```

FIGURE 5-4 *Device Definition File (Sheet 2 of 3)*

The device definition file relates signal names on the body drawing to pin numbers on the personality module and defines worst-case timing characteristics for logic simulation waveform generation.

```
          'BCI TIME'      pin=118,
          'BCI PHASE'     pin=115;
  end_pin;

  jig_id = 8;
  jig_type = static_forever;
  clock_period = 25-2ØØØ;

reset_seq

 '-BI DC LO'    1,1,1,1,1,1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
 'SOFT SELECT'  1,1,1,1,1,1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø;
 '-SOFT CLOCK'  Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø;
        '-BCI RQ'<Ø>    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI RQ'<1>    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI MAB'   .  1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI RS'<Ø>    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI RS'<1>    1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI INT'<4>   1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI INT'<5>   1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI INT'<6>   1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        '-BCI INT'<7>   1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1;
        'BCI I'<3>      Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z;
        'BCI I'<2>      Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z;
        'BCI I'<1>      Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z;
        'BCI I'<Ø>      Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z,Z;

end_reset_seq;

  delay_table

          'BCI PO',
          '-BCI RAK', '-BCI NXT', '-BCI EV'<4>, '-BCI EV'<3>, '-BCI EV'<2>,
          '-BCI EV'<1>, '-BCI EV'<Ø>, '-BCI AC LO', 'BCI CLE',
          '-BCI SEL', '-BCI SC'<2>, '-BCI SC'<1>, '-BCI SC'<Ø>,
          'BCI TIME', 'BCI PHASE':
  '-SOFT CLOCK' = 3Ø;
   '-BCI MDE', '-BCI SDE':
  '-SOFT CLOCK' = 4Ø;
   '-BCI DC LO':
  '-BI DC LO' = 15Ø,
  '-SOFT CLOCK' = 3Ø;
          'BCI D'<31>, 'BCI D'<3Ø>, 'BCI D'<29>, 'BCI D'<28>, 'BCI D'<27>,
          'BCI D'<26>, 'BCI D'<25>, 'BCI D'<24>, 'BCI D'<23>, 'BCI D'<22>,
          'BCI D'<21>, 'BCI D'<2Ø>, 'BCI D'<19>, 'BCI D'<18>, 'BCI D'<17>,
          'BCI D'<16>, 'BCI D'<15>, 'BCI D'<14>, 'BCI D'<13>, 'BCI D'<12>,
          'BCI D'<11>, 'BCI D'<1Ø>, 'BCI D'<9>, 'BCI D'<8>, 'BCI D'<7>, 'BCI D'<6>,
          'BCI D'<5>, 'BCI D'<4>, 'BCI D'<3>, 'BCI D'<2>, 'BCI D'<1>, 'BCI D'<Ø>,
          'BCI I'<3>, 'BCI I'<2>, 'BCI I'<1>, 'BCI I'<Ø>:
  '-SOFT CLOCK' = 4Ø;
          '-BI BSY',
          '-BI NO ARB', '-BI CNF'<2>, '-BI CNF'<1>, '-BI CNF'<Ø>, '-BI D'<31>, '-BI D'<3Ø>,
          '-BI D'<29>, '-BI D'<28>, '-BI D'<27>, '-BI D'<26>, '-BI D'<25>, '-BI D'<24>,
          '-BI D'<23>, '-BI D'<22>, '-BI D'<21>, '-BI D'<2Ø>, '-BI D'<19>, '-BI D'<18>,
          '-BI D'<17>, '-BI D'<16>, '-BI D'<15>, '-BI D'<14>, '-BI D'<13>, '-BI D'<12>,
   '-BI D'<11>, '-BI D'<1Ø>, '-BI D'<9>, '-BI D'<8>, '-BI D'<7>, '-BI D'<6>,
          '-BI D'<5>, '-BI D'<4>, '-BI D'<3>, '-BI D'<2>, '-BI D'<1>, '-BI D'<Ø>,
          '-BI I'<3>, '-BI I'<2>, '-BI I'<1>, '-BI I'<Ø>, '-BI PO':
  '-SOFT CLOCK' = 85;

  end_delay_table;

end_primitive;
```

FIGURE 5-4 *Device Definition File (Sheet 3 of 3)*

The device definition file relates signal names on the body drawing to pin numbers on the personality module and defines worst-case timing characteristics for logic simulation waveform generation.

The BIIC is treated as a static device and can simulate indefinitely. Each instance of the BIIC requires a BIIC personality module, but only one entry is required in the Realchip library file that is specified to the simulator.

Impact of Design Goals on Operation of the Model

The design goals of the BIIC hardware model determined how Realchip and Realmodel operate when used with the BIIC.

- **Turnkey Solution.** Valid's creation of the personality module, the body drawing, and the device definition file, coupled with the actual BIIC device, means that the BIIC model can be used as soon as the BIIC is inserted into the Realchip or Realmodel modeling system.
- **Lengthy Bus Operations.** Because the BIIC operates in static mode, any number of VAXBI bus cycles can be simulated.
- **Maximize Simulation Performance.** Operation in static mode assures highest simulation performance of the device. Also, additional logic on the personality module permits the use of the Realchip hardware clock, which speeds the operation of the long self-test sequence of the BIIC.
- **Multinode Operation.** Because a VAXBI system can have up to 16 BIICs, the hardware model of the BIIC had to provide for multiple BIICs. Upon initialization each BIIC performs a self-test and each node that uses a BIIC performs a self-test of its user interface logic. The BIIC model causes the nodes to be synchronized after the performance of fast self-test, so that the simulation can begin quickly.

  The support logic on the personality module decouples Realchip's hardware clock from the BIIC when it has completed its self-test sequence and assures that all the nodes are phase synchronized when switched to the simulator's clock for operation.
- **Compatible Models.** Care was taken to make sure that the software and hardware developed for the BIIC model could be used in both board design and system-level or software/hardware integration applications.

How the BIIC Model Operates

BIIC Reset/Self-Test Sequence

During reset, certain BIIC lines must be valid or asserted/deasserted:

- On the last cycle in which BCI DC LO* is asserted, the node ID, the Device Register, and the BIIC parity mode are loaded from the BCI I, BCI D, and BCI P0 lines, respectively.
- BCI RQ<1..0>*, BCI MAB, BCI RS<1..>, BCI INT<7..4> must be deasserted.

The second requirement posed no problem since these signals are defined as inputs in the definition file and are continuously driven during the reset/self-test period to the state specified by the simulator during simulation and by the definition file before simulation time 0 ns (reset_sequence).

The signals in the first requirement, however, are bidirectional, and as such, are not continuously driven during node reset. Instead, tristate buffers on the personality module are used to drive the BCI I<3..0> lines from ID<3..0> when BCI DC LO* is asserted. The Device Register can be modified through a write-type transaction. Only BIIC-generated parity is permitted.

Multinode Simulation

The final simulation requirement is:

- The BI data lines BCI D, BCI I, and BCI P0 can be wire-tied (AND function) with other VAXBI nodes.

This is a necessary constraint for multinode simulation and is accomplished by specifying the pin property OUTPUT_TYPE= (OC,AND) in the definition file. The simulator will then evaluate the net as a wire-AND function. Also, 270-ohm pullup resistors are attached to all the bidirectional BI lines.

Because of functions required of the BIIC hardware model, the BIIC personality module had certain electrical constraints:

- Carrier had to have large power and ground planes
- Many decoupling capacitors required on the +5V supply

The BIIC personality module was designed with these requirements in mind. Decoupling capacitors are positioned close to the VCC pins of the BIIC and its support components; also, larger tracks and wires are used for routing power and ground.

Simulation Using the BIIC Hardware Model

After the user creates the schematic using the CAE system's schematic editor, the design is prepared automatically for simulation. The user then invokes the simulator (ValidSIM). The simulator's capability to accept input from a file (called a script file) is used to reset and self-test the BIIC (Figure 5-5 shows such a file). (Note: At the end of the self-test sequence, the green LED on the BIIC personality module should be ON.) All other BIIC operations require no extra user commands to initiate other than to advance the simulation clock. Input for the BIIC can be provided from a file containing test vectors (called a stimulus file) or from a program that drives a microprocessor when such devices are included in the design.

The SOFT CLOCK signal on the BIIC body provides the input clock on the personality module to the clock driver, which drives the clock receiver that in turn generates BCI TIME*, BCI PHASE*, BCI TIME, and BCI PHASE. Typically, in multinode simulation, the SOFT CLOCK signals are identical. SOFT SELECT on the BIIC body drawing should be attached to signal name SOFT SELECT in order to be referenced by the initialization script ACDCLO. BI DC LO* should be labeled similarly. ID<3..0> should be set to the desired node ID.

```
history 1ØØØØØ
wave Ø 1ØØØ
o SOFT CLOCK!C Ø-5*
o SOFT SELECT
o BI DC LO*
o BCI TIME
o BCI PHASE
o BCI RQ<1..Ø>*
o BCI MAB*
o BCI RS<1..Ø>*
o BCI INT<7..4>*
o BCI RAK*
o BCI NXT*
o BCI MDE*
o BCI D<31..Ø>
o BCI I<3..Ø>
o BCI PO
o BCI EV<4..Ø>*
o BCI AC LO*
o BCI DC LO*
o BCI CLE
o BCI SDE*
o BCI SEL*
o BCI SC<2..Ø>*
o BI BSY*
o BI NO ARB*
o BI CNF<2..Ø>*
o BI D<31..Ø>*
o BI I<3..Ø>*
o BI PO*
o BI AC LO*
o DATA
o INF
logic_init -*
o BI DC LO*
d Ø
sim 75
d 1
sim 2ØØ
o SOFT SELECT
d 1
sim 125
sim 2ØØ
wave Ø 4ØØ
pause
```

FIGURE 5-5 *ACDCLO Script File*
This file causes a reset operation, which causes the BIIC to self-test.

BIIC Simulation Example: A Multinode Simulation

```
This document briefly explains what is happening on the simulator display
during each of the simulation pauses.
This demo illustrates several BIIC transactions requested through the
BIIC user interface and the VAXBI bus.
INITIALIZATION
   Pause #1
       This display shows the BIIC undergoing reset and self-test.
       Notice that BCI TIME and BCI PHASE are random since the signal
       SOFT SELECT is selecting the independent hardware clock on the
       personality module. While BI DC LO* is asserted, BCI D<31..0>
       and BCI PO are asserted as specified in the VAXBI SRM, but
       BCI I<3..0> is physically driven on the adapter board to value C
       in order to load the node ID. The value of 1B on the BCI EV<4..0>*
       lines indicates that the chip has passed self-test. Also note that
       the LED is now lit. Type "resume" to get to the next display.
LOOPBACK READ OF VAXBICSR REGISTER
   Pause #2
       In this waveform, the BIIC is requested to perform an intranode
       transaction to read the VAXBICSR register in the BIIC CSR node
       space. The value 1 is deposited on BCI RQ<1..0>* and is removed
       when the master port request is acknowledged. (The BIIC asserts
       BCI RAK*.) The command on BCI I and the address on BCI D are
       supplied when BCI MDE* is asserted.
   Pause #3
       Having accepted the command, the BIIC outputs the contents of the
       VAXBICSR. The data 0401380C, valid on the asserted edge of BCI NXT*,
       verifies that the BIIC has been properly loaded with the node ID C.
       Also, self-test has succeeded, and the VAXBI Interface Revision field
       indicates that this chip is a Pass 4 part. The EV codes MCP and AKRSD
       (1E and 1D) concludes that this transaction is successful and
       terminated.
LOOPBACK READ OF DEVICE REGISTER
   Pause #4
       This waveform reveals that the Device Register has been loaded with
       all ones and has not been initialized.
WRITING TO REG GPR0 THROUGH INTERNODE TRANSACTION
   Pause #5
       This waveform depicts the access of GPR0 through the VAXBI side of
       the BIIC by mimicking an internode transaction. Observe that imaginary
       node 0 first arbitrates for the bus. Then command and address
       information is issued on BI I<3..0>* and BI D<31..0>* in inverted form.
       This information also appears on the BCI side of node C.
   Pause #6
       This waveform shows the data cycle of the internode write transaction.
       The event code of IRW (19) indicates that the internal register has
       been written. In addition, the BI CNF<2..0>* lines indicate a
       successful write.
LOOPBACK READ OF GPR0 REGISTER
   Pause #7
       This final pause shows that indeed the GPR0 has been correctly written
       to.
       This demo program is completed. Type "exit" to return to the
       graphics editor and/or "quit" to exit back to "UNIX."
```

The authors are Michael S. Glenn, Ron Jew, and Bill Harding of Valid Logic Systems, Inc., 2820 Orchard Parkway, San Jose, CA 95134.

Chapter 6

VAXBI Debug Tool: The DAS91VB

by Tektronix, Inc.

This chapter describes the debug tool available from Tektronix that can be used to verify VAXBI boards during prototype design and manufacturing test and to diagnose VAXBI systems in the field.

Contents

Complex Bus Demands Unique Test Tools

With the BIIC, VAXBI bus node designers can avoid the complex circuitry for a bus interface and devote more time to their specific application. But even with the BIIC, the developer can't use conventional logic analysis debug tools to examine the bus. Any attempts to attach a logic analyzer directly to the bus would violate its electrical design specifications. Even if this were not the case, it would require an extremely intelligent processor to emulate a BIIC and decipher the lines into VAXBI transactions and arbitrations.

DEC needed a hardware debug tool for the VAXBI bus, not only to help their own engineers in designing modules for the bus, but also to provide third-party vendors with a means of debugging their new VAXBI designs. TEK's solution to DEC's VAXBI testing needs was to take its flagship logic analyzer, the DAS9100, and build a custom interface between it and the VAXBI. The resultant product, the DAS91VB, is essentially a fully loaded DAS9100 logic analyzer having extensive stimulation, acquisition, timing, and triggering capabilities (see Figure 6-1).

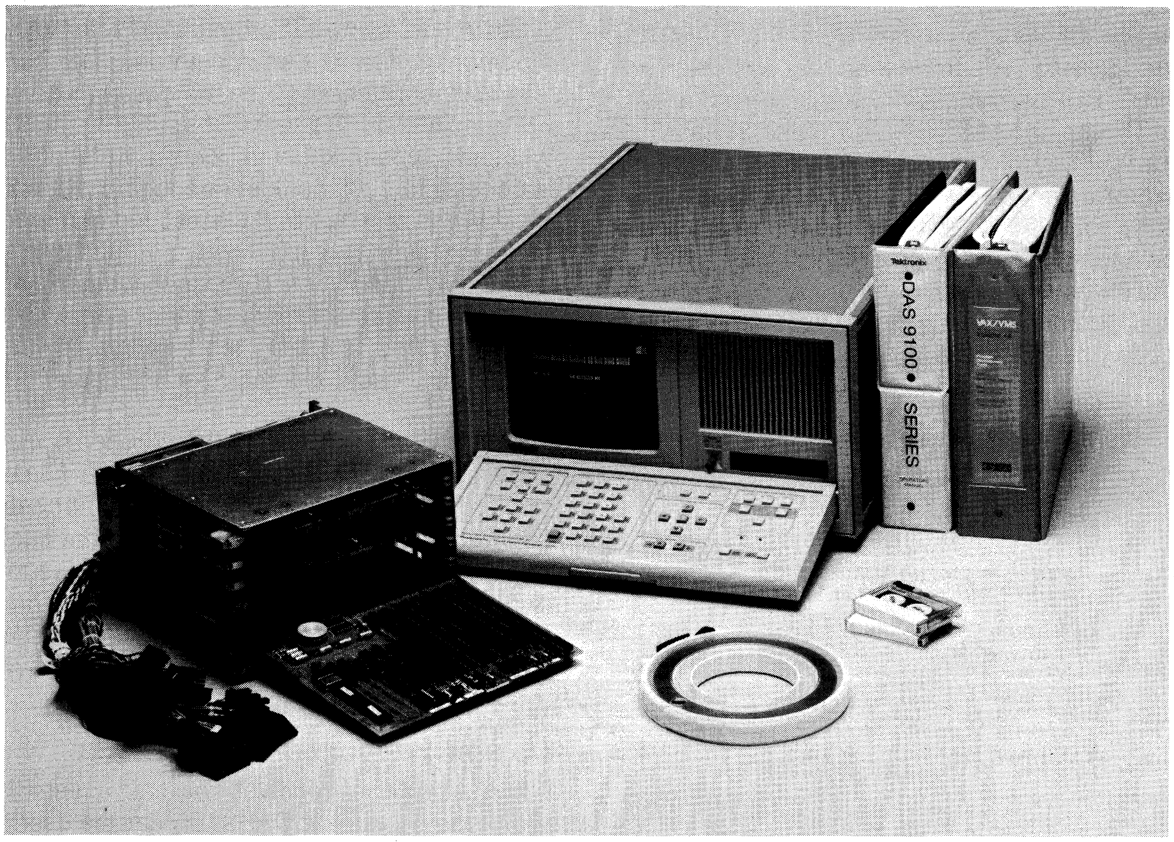


FIGURE 6-1 *The DAS91VB*

Product Overview

The DAS91VB has two custom VAXBI modules that plug into a VAXBI card cage. While the modules are identical, they are switch selectable so that one can be used for sending transactions onto the VAXBI bus and the other can be used for data acquisition from the bus.

Each of these VAXBI modules contains a BIIC that handles all bus transactions, address decoding and recognition, and user interface signals for data/control lines. The product also includes all necessary probes, custom cable sets, mnemonic decode tables, and control software that runs under VAX/VMS to allow remote operation.

User Interface

The DAS91VB provides two modes of user interface as diagramed in Figure 6-2. First, it can be directly operated from the DAS9100 front panel as a stand-alone test system. Second, it can be remotely controlled from a VAX mainframe terminal over an RS-232 link. The latter is useful for engineers who are more comfortable with a VAX terminal keyboard than with a logic analyzer front panel and are more conversant in VAXBI mnemonics than in state data (ones and zeros). This mode also provides an opportunity to use the DAS91VB in a production board-test environment.

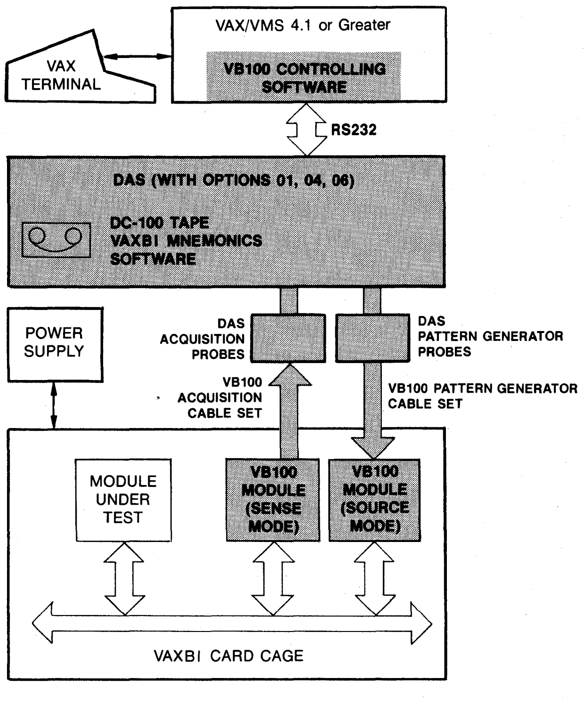

VAX TERMINAL
VAX/VMS 4.1 or Greater
VB100 CONTROLLING SOFTWARE
RS232
DAS (WITH OPTIONS 01, 04, 06)
DC-100 TAPE
VAXBI MNEMONICS SOFTWARE
POWER SUPPLY
DAS ACQUISITION PROBES
DAS PATTERN GENERATOR PROBES
VB100 ACQUISITION CABLE SET
VB100 PATTERN GENERATOR CABLE SET
MODULE UNDER TEST
VB100 MODULE (SENSE MODE)
VB100 MODULE (SOURCE MODE)
VAXBI CARD CAGE

FIGURE 6-2 *Diagram of VAX-DAS-VAXBI Bus Path*

When controlled from a VAX, TEK's menu-driven control software performs all VAX-DAS91VB translations and presents acquired VAXBI transactions on the engineer's terminal in easy-to-read VAXBI mnemonics. A typical test sequence might be as follows: The user first creates an ASCII file on the VAX using any editor. This file would contain VAXBI command, address, and data information in mnemonic form. The file is read by the TEK-supplied control software. The VAXBI bus mnemonics are automatically translated and sent to the DAS where they drive the pattern generator. Patterns are output through the probes and cable set to the VAXBI module and onto the bus. A second module acquires bus transactions and passes them to the DAS, where they are displayed in mnemonic form on its screen as well as passed on to the VAX terminal for display.

Since the system communicates in VAXBI mnemonics, the test equipment learning curve is greatly shortened. Once the designer comes up to speed on the VAXBI bus itself, much of what he needs to know to operate his test tools is understood. All the user needs to provide the VAX with is command, address, and data information. Once created, these test files can be saved for future testing or linked together to build complex test suites. As far as hardware at the VAXBI end, all that is required is a powered VAXBI card cage and the user's module to be tested.

Tektronix also simplified the necessary hardware connections through the design of custom probe cables. The solution called for cable sets with multiple 30-pin connectors that attach directly to the user interface header on the rear of the VAXBI card cage. These cable sets replace the cumbersome individual lead sets (almost 180 connections) that were previously required.

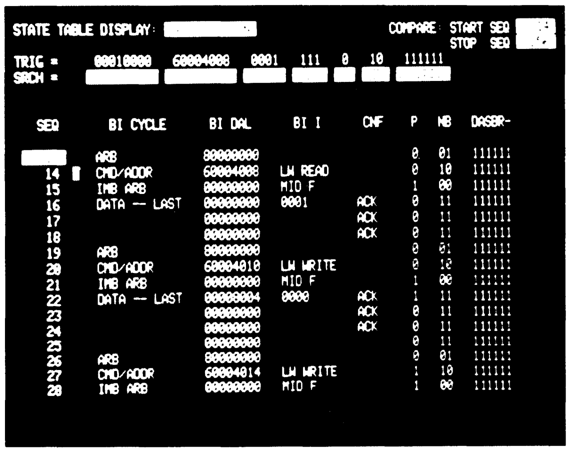
Design Verification

When a designer receives the first assembled prototype board, the DAS91VB can be used to verify that the hardware meets both the designer's specifications and all of DEC's VAXBI specifications. During this phase, the DAS91VB's extensive triggering combinations and fault-testing capabilities are extremely valuable.

In the *VAXBI System Reference Manual,* DEC details a number of requirements that every VAXBI module must meet. These constraints are meant to ensure that the module will function in any system configuration and that it will not hamper the maximum throughput of the bus. By performing stimulation, monitoring, and extensive fault-testing, the DAS91VB helps the designer verify that the module under development meets these requirements. One such requirement is that the fast self-test complete in a specified amount of time. The DAS91VB, with its built-in counter/timers, provides an easy method to measure this design requirement.

Another requirement is that each class of node (processor, memory, and adapter) must respond to a certain subset of legal bus commands based on whether the node is a master or a slave. Using its word recognizers to detect when a module has been addressed, the DAS91VB can be used to monitor all the transactions of a selected node and determine if the node violates any of its class requirements.

The custom VAXBI modules can operate in two modes: pass-all and filter. In pass-all mode, every VAXBI cycle (that is, every 200 ns) is sent to the DAS—even if the bus is idle and no commands are being sent. This mode (shown in Figure 6-3) is used to debug the VAXBI at the bit level. In this mode imbedded arbitrations and slave command acknowledgments (ACKs) are visible.



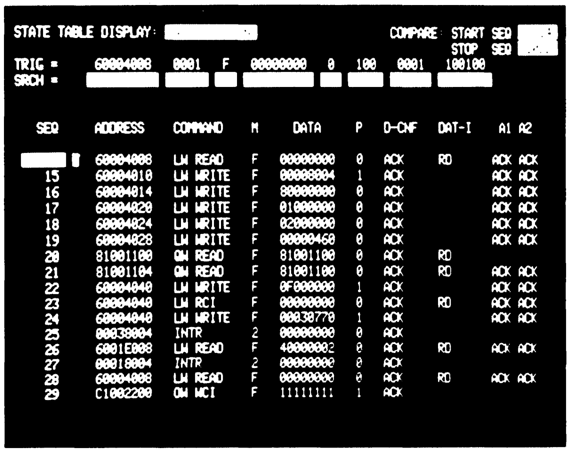
```
STATE TABLE DISPLAY:                          COMPARE: START SEQ
                                                       STOP  SEQ
TRIG =   00010000  60004008  0001  111  0  10  111111
SRCH =

  SEQ     BI CYCLE      BI DAL     BI I        CNF   P  NB  DASBR-

          ARB           00000000                     0  01  111111
   14     CMD/ADDR      60004008   LW READ           0  10  111111
   15     IMB ARB       00000000   MID F             1  00  111111
   16     DATA -- LAST  00000000   0001        ACK   0  11  111111
   17                   00000000               ACK   0  11  111111
   18                   00000000               ACK   0  11  111111
   19     ARB           00000000                     0  01  111111
   20     CMD/ADDR      60004010   LW WRITE          0  10  111111
   21     IMB ARB       00000000   MID F             1  00  111111
   22     DATA -- LAST  00000004   0000        ACK   1  11  111111
   23                   00000000               ACK   0  11  111111
   24                   00000000               ACK   0  11  111111
   25                   00000000                     0  11  111111
   26     ARB           00000000                     0  01  111111
   27     CMD/ADDR      60004014   LW WRITE          1  10  111111
   28     IMB ARB       00000000   MID F             1  00  111111
```

FIGURE 6-3 *Pass-all Mode State Display*

In filter mode, each DAS acquisition is a VAXBI transaction. This mode lets the user debug at the VAXBI command level and conserves memory when the bus has long idle periods. Figure 6-4 is an example of a DAS screen display showing data acquired in filter mode.



```
STATE TABLE DISPLAY:                              COMPARE: START SEQ
                                                           STOP  SEQ
TRIG =   60004008  0001  F  00000000  0  100  0001  100100
SRCH =

  SEQ    ADDRESS   COMMAND   M  DATA      P  D-CNF  DAT-I   A1  A2

         60004008  LW READ   F  00000000  0  ACK    RD      ACK ACK
   15    60004010  LW WRITE  F  00000004  1  ACK            ACK ACK
   16    60004014  LW WRITE  F  80000000  0  ACK            ACK ACK
   17    60004020  LW WRITE  F  01000000  0  ACK            ACK ACK
   18    60004024  LW WRITE  F  02000000  0  ACK            ACK ACK
   19    60004028  LW WRITE  F  00000460  0  ACK            ACK ACK
   20    81001100  OW READ   F  81001100  0  ACK    RD
   21    81001104  OW READ   F  81001100  0  ACK    RD      ACK ACK
   22    60004040  LW WRITE  F  0F000000  1  ACK            ACK ACK
   23    60004040  LW RCI    F  00000000  0  ACK    RD      ACK ACK
   24    60004040  LW WRITE  F  00030770  1  ACK            ACK ACK
   25    00038004  INTR      2  00000000  0  ACK
   26    6001E008  LW READ   F  40000002  0  ACK    RD      ACK ACK
   27    00018004  INTR      2  00000000  0  ACK
   28    60004008  LW READ   F  00000000  0  ACK    RD      ACK ACK
   29    C1002200  OW WCI    F  11111111  1  ACK
```

FIGURE 6-4 *Filter Mode State Display*

The DAS91VB can also display timing diagrams such as the one shown in Figure 6-5. The relationship of BSY and NO ARB is shown, while the two power signals and some status lines are monitored for glitches.

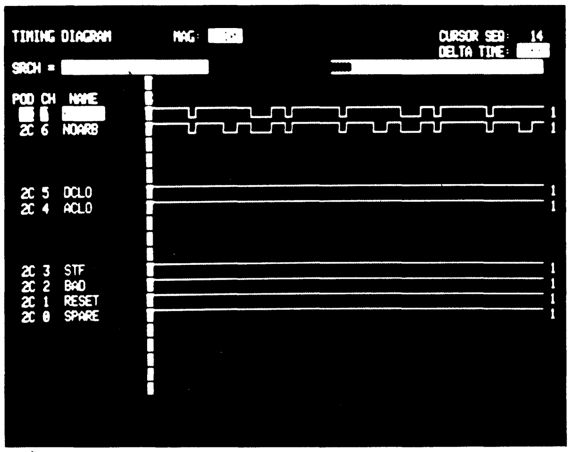


FIGURE 6-5 *Timing Display*

Manufacturing Module Test

After a verified board design goes into production, the DAS91VB can be used to ascertain that no functional design specifications or VAXBI requirements were compromised during the manufacturing process. With its remote programming capabilities, the DAS91VB can be used to perform automatic board testing in volume. A VAXBI card cage can be loaded with boards and the DAS used to run a set of VAX mainframe generated vectors on each board over the VAXBI bus. Since the test vectors are user defined, the test procedure can be tailored to balance coverage versus time to test. While it would be desirable to have an exhaustive set of test vectors that guarantee VAXBI compatibility, the complexity of the VAXBI bus would make such a test prohibitively long. DEC has, however, spent months and months of CPU time in extensive simulation of the VAXBI bus protocol with its unique dual round-robin arbitration scheme.

VAXBI Bus Performance Tuning

Once a product is in the field, there is always a need for system diagnosis. Here the DAS91VB can assume the role of an analysis aid. The DAS91VB modules can be inserted into a VAXBI system to simulate additional VAXBI modules. Using the DAS91VB modules allows the diagnostician to perform operations, with actual hardware, that wouldn't be possible using standard system components.

Because it is independent of the system CPU, the DAS91VB can produce any bit pattern as a transaction as well as any sequence of transactions, legal or illegal. Thus, it is possible to implement such operations as forcing bad parity, issuing nonexistent commands, accessing bad memory locations, testing interlocked memory, forcing a bad interrupt vector, and issuing multiple STALLs.

The use of the STALL response is critical to VAXBI system performance. The VAXBI bus allows a slave node to issue a STALL in response to a command when the node is not ready to service a request. This can result in a RETRY, which is generated by the BIIC until a specified number of STALLs occurs. It is often difficult to perform this type of "stress" testing in a "normal" system environment. Yet if left undiscovered, the subtle side effects of incorrect STALL responses or error handling can destroy system performance. The DAS91VB can easily accomplish this type of testing, and because of its flexibility, the DAS91VB can also be used for general-purpose logic analysis tasks when not being used for VAXBI testing.

* * *

With the use of Tektronix's debug tool, the DAS91VB, DEC was able to accelerate its internal development of the VAXBI bus. The creation of the DAS91VB also provided an easy and immediately available tool for the numerous third-party vendors hoping to reserve a seat on the VAXBI bus for themselves.

The author, Allen Cicrich, is Computer Resources Manager, Logic Analyzer Division, of Tektronix, Inc. Tektronix is located in Beaverton, Oregon, and specializes in test and measurement equipment.

Appendix C Transition Header Interconnects

This appendix provides information needed to design and build cabling interconnects for use in VAXBI systems. All cables attach to the transition header assembly that screws on to the I/O connector segments of the backplane.

A variety of cabling can be used in VAXBI systems. Modules are connected either by rigid interconnects or by flexible cables. Like the module interconnects, cables to the bulkhead also attach to the transition headers. For bulkhead cables, DIGITAL uses cables of four lengths: 2', 5', 8', and 12'. Table C-1 shows the length of cables used in DIGITAL's 8000 series computer systems. The 2' length is not yet used in any product.

Table C-1 VAXBI Systems Bulkhead Cable Lengths

System	Length
8200, 8300	8'
8500, 8550, 8700, 8800	5' and 12'

The transition header assembly (shown in Figure C-1) consists of three 60-pin segments, to which two 30-pin cables can be attached. Table C-2 gives part numbers for DIGITAL's transition header and for ribbon connectors that can be used with this transition header.

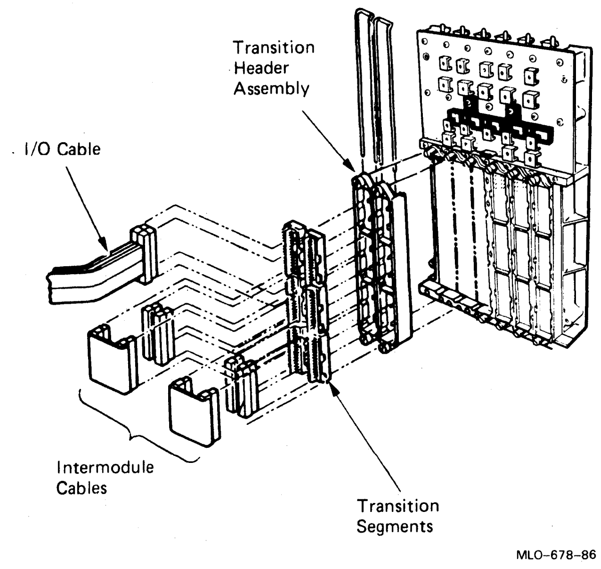


FIGURE C-1 *Cables and the Transition Header Assembly*

Table C-2 VAXBI Transition Header and Ribbon Connectors

Part No.	Vendor	Part Description
BIT3S-1	Burndy	5-segment transition header assembly
FRS30BS-8P	Burndy	CONN (IDC) 30POS (2X15) .100CC
746680-7	AMP	CONN (IDC) 30POS (2X15) .100CC

Figure C-2 shows two nested cables that connect the inner and outer halves of two adjacent segments. Figure C-3 shows how rigid interconnects rather than cables can be used with the transition header. Figure C-4 shows a closeup of two adjacent segments and gives the center to center spacing. Figure C-5 shows the dimensions for an interconnect for half of a 60-pin segment. Figure C-6 shows the contact area of the connector body. Figure C-7 shows the dimensions for a rigid intermodule interconnect.

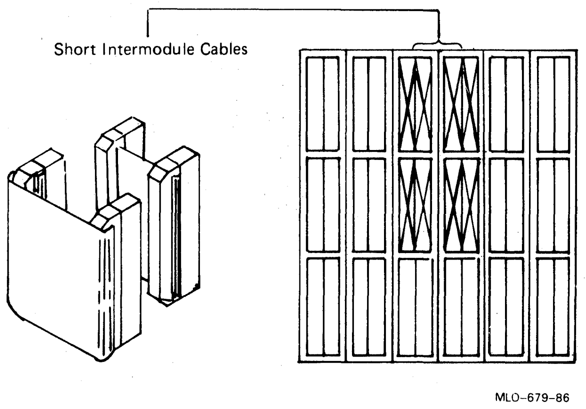


FIGURE C-2 *Nested Cables Between Two Adjacent Segments*

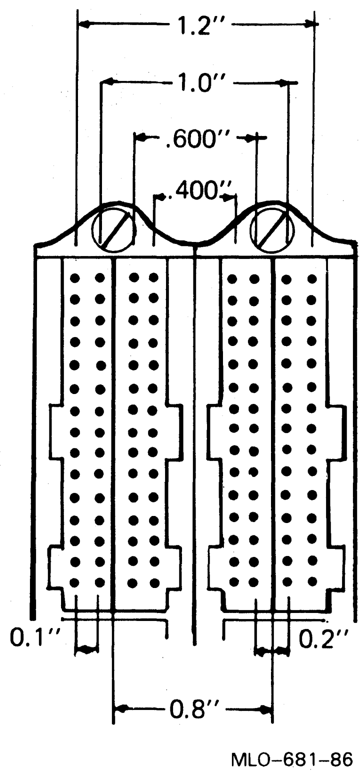


FIGURE C-4 *Closeup of Two Adjacent Segments*

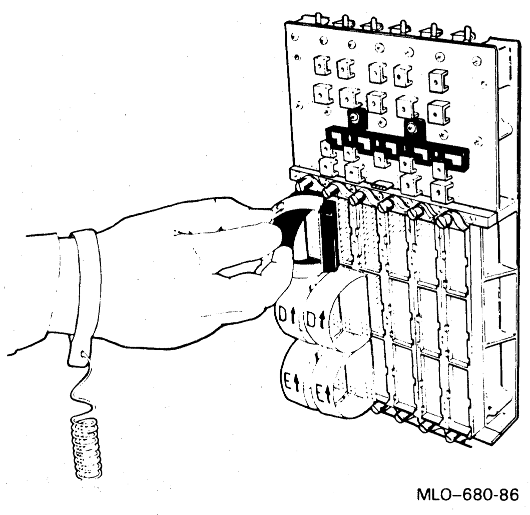


FIGURE C-3 *Rigid Intermodule Interconnect Between Two Adjacent Modules*

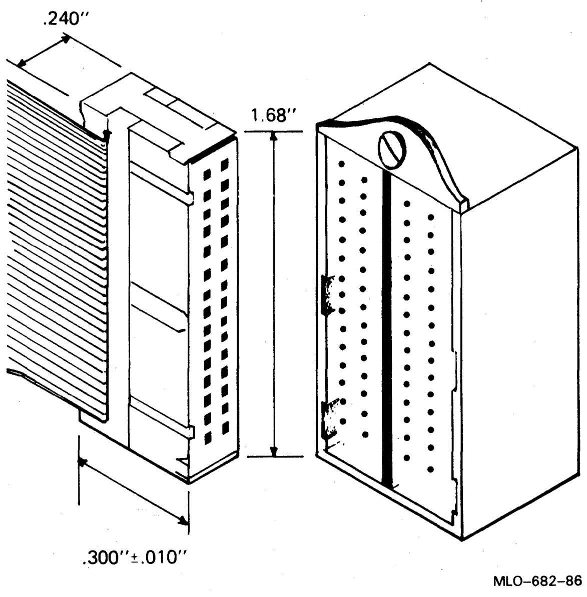


FIGURE C-5 *Dimensions for 30-Pin Connector*

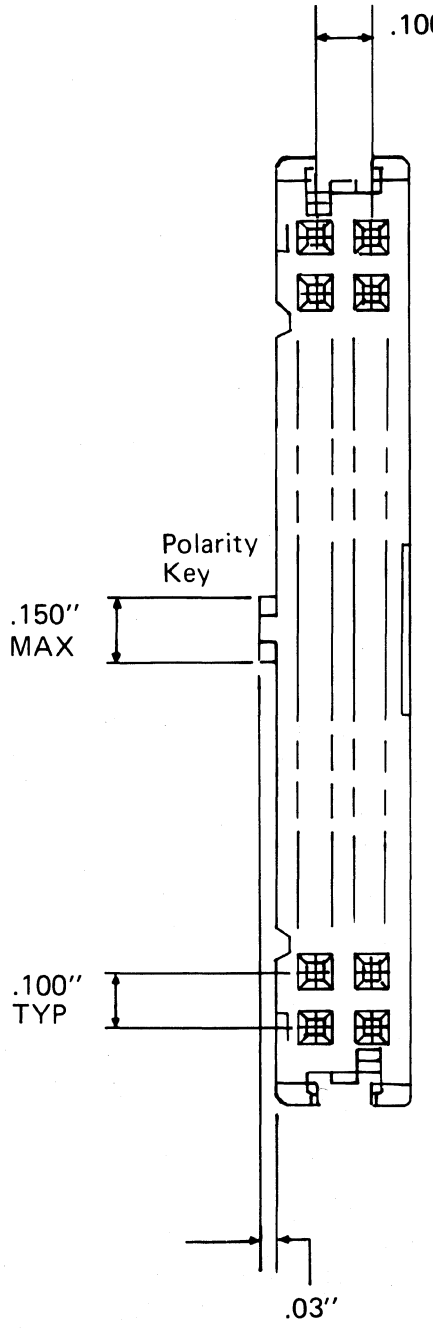


FIGURE C-6 *Contact Area of Connector Body*

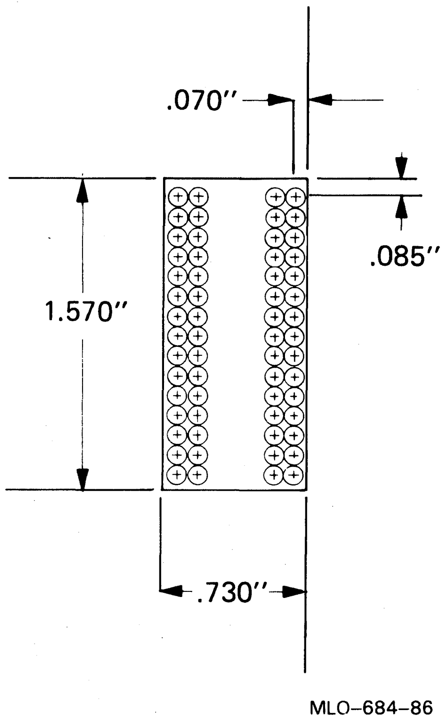


FIGURE C-7 *Dimensions for Rigid Intermodule Interconnect*

Index

UPDATE NOTICE NO. 2

VAXBI Designer's Notebook

EK-VBIDS-R2-001

This update contains revised pages and one new appendix. The pages enclosed in this package are listed below:

Title/copyright page
iii/iv
v/vi
B-1 through B-4
D-1 and D-2

Digital Equipment Corporation • Maynard, Massachusetts

VAXBI DESIGNER'S NOTEBOOK

EK-VBIDS-RM-001

Digital Equipment Corporation • Maynard, Massachusetts

First Printing, January 1986
Updated, January 1987
Updated, June 1988

The manuscript for this book was created using generic coding and, via a translation program, was automatically typeset. Book production was done by Educational Services Media Communications Group in Bedford, MA.

BASEWAY	DRB32	TOPS-10
DEC	MASSBUS	TOPS-20
DECconnect	MicroVAX/VMS	ULTRIX
DECdirect	PDP	UNIBUS
DECmate	P/OS	VAX
DECnet	Professional	VAXBI
DECtalk	Rainbow	VAXELN
DECUS	RSTS	VAX/VMS
DECwriter	RSTS/E	VMS
DIBOL	RSX	VT
digital	RT	Work Processor

Contents

Preface

The purpose of this notebook is to provide a focus for those who are involved with designing options for the VAXBI bus. This book attempts to put the *VAXBI System Reference Manual* into perspective, explaining which portions of the VAXBI specification option designers need to be most concerned about. Some portions deal with requirements that are implemented by DIGITAL-supplied hardware. The option designer need only use the specified hardware to comply with many of the requirements that are detailed in the comprehensive specification.

With the proper perspective, designers can understand more quickly what they need to do. To help them understand *how* to go about their task, we will provide design examples. In addition, as design tools become available we will include commentary on their use.

Intended Audience

This notebook is primarily for engineers who design options for VAXBI systems. System architects and others who want to understand DIGITAL's design philosophy for the bus will also find this information useful. Chapter 3, which is a guide to module layout, should be read and studied by module layout designers; this chapter also includes information pertinent to manufacturing.

Structure of This Manual

Chapter 1, Introduction to VAXBI Option Design, provides an overview of the VAXBI bus and of the documentation from an option designer's point of view. The chapter also poses design issues and gives hints for designing options.

Chapter 2, The Instrument Control Adapter, is an example of a VAXBI option design. The design requirements and the resulting design decisions are described. The option is a master-port-only design.

Chapter 3, VAXBI Module Layout Guide, serves as a guide for engineers and module layout designers as they design options and build VAXBI modules. It points out problem areas and reports some of DIGITAL's experiences. References are made to the appropriate module control drawings.

Chapter 4, Migrating Designs to the VAXBI, describes how the VAXBI design philosophy differs from the approaches used for UNIBUS and Q-bus designs. It looks at the functions of the UNIBUS-to-VAXBI adapter to show what a VAXBI option must do.

Chapter 5, Realchip BIIC Model, is a description of Valid Logic Systems' hardware modeling product that is used in simulating designs that use the BIIC chip.

Chapter 6, VAXBI Debug Tool: The DAS91VB, describes a debug tool developed by Tektronix. The DAS91VB is a logic analyzer with two custom VAXBI modules that can be used to verify VAXBI boards during prototype design and manufacturing test and to diagnose VAXBI systems in the field.

Appendix A, VAXBI BIIC Simulation: Physical Chip Modeling, gives requirements to permit physical chip modeling of the BIIC for simulation of a VAXBI option.

Appendix B, VAXBI Base Layout Package, gives information on the documentation and databases in the VAXBI Base Layout Package.

Appendix C, Transition Header Interconnects, provides information needed to design and build cabling interconnects to the VAXBI card cage transition headers.

Related Documentation

- *VAXBI System Reference Manual* – The specification for the VAXBI bus and the VAXBI primary interface (the BIIC)
- T1999 – Module Control Drawings for the standard VAXBI module
- T1996 – Module Control Drawings for the VAXBI expansion module
- ELEN 626 – Mechanical outline drawing for the standard VAXBI module
- ELEN 633 – VAXBI module layup specification

Other Useful Documentation

- *VAX-11 Architecture Reference Manual* – Information about VAX architecture; order part no. EY-3459E-DP
- *Writing a Device Driver for VAX/VMS* – Reference source for device driver development and support; order part no. AA-Y511B-TE

- *VAX DRB32 Driver V1.0 Documentation Kit* – Reference source for device driver development and support; order part no. QLZ95-GZ-1.0/RZ
- *DECdirect Plus Catalog* – Information relating to DIGITAL's computer products and services; send request to DECdirect Plus, MKO1/W83, Continental Blvd., Merrimack, NH 03054-9987

Appendix B VAXBI Base Layout Package

The following information is from the cover letter that is distributed with the VAXBI Base Layout Package.

This document describes the contents of the documents and magnetic tapes in this package. This package provides the necessary information to design and build either a Standard VAXBI Module or an Expansion VAXBI Module.

The following documents are supplied:

T1999	Standard Module Control Drawing
T1996	Expansion Module Control Drawing
ELEN 633	Layup Specification
ELEN 626	Mechanical Outline Specification

(References within these documents to DIGITAL internal documents are for the use of DIGITAL manufacturing. These documents are not needed by customers.)

The following magnetic tapes are supplied:

T1999A	Gerber Tape (10 artwork layers)
T1999D	Drill Tape
T1996A	Gerber Tape (10 artwork layers)
T1996D	Drill Tape

This package contains information for two types of VAXBI modules. T1999 and its associated databases are for the standard VAXBI module with a VAXBI Corner; T1996 and its associated databases are for the VAXBI expansion module. ELEN 626, ELEN 633, and the Gerber wheel description apply to both types of VAXBI modules. The Gerber tapes are for the 10 layers of the module and, when plotted, will produce the artwork for the DIGITAL-defined portions of a VAXBI module. The drill tapes contain the necessary data for drilling the holes in the DIGITAL-defined portions of the module. The precise format of the data on these tapes is described below.

The artwork for the module has added certain special features to assist PCB manufacturing. The connector area gold plating feature that DIGITAL manufacturing uses has been added; it can be removed and replaced if desired. Beware that if you supply your own plating feature and it does not work, you must not use the module in a VAXBI backplane because the module will damage the connector if it is not properly plated. Each layer has two targets on it to assist in alignment; DIGITAL's tooling features have not been added.

The intent is that a designer can produce boards either by reading the Gerber data into a CAD system as it is provided to produce a layout or by plotting the artwork and then digitizing it back into the CAD system. (If you intend to do the latter, you should plot the artwork at 2:1 at least. Feedback from users has indicated that 4:1 is preferable.)

Note: A manufacturing problem can exist when removing the finished PCB from the panel in which it was etched. Because some shearing equipment cannot hold the necessary tolerances on the module outline, we recommend routing the PCB out of the panel in these cases.

Gerber Tape Format

The decision to use Gerber tape format for the artwork was made to give the best coverage in terms of photo-plotting systems in use. The format on tape can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999A, T1996A)

ASCII format, incremental coordinates

Coordinate format 3.3 (no leading zeros)

All coordinates are positive X and positive Y. The coordinates are referenced to the Gerber plotter origin, but the artwork is offset by 1.1" in the X direction and 1.7" in the Y direction (DIGITAL manufacturing's standard offset for VAXBI modules). The artwork produced is data as viewed from side 2. The first five layers are wrong reading; the bottom five layers are right reading. It is recommended that you verify that the data does not require a coordinate translation for the plotter system in use. (Note that all pads and traces will be plotted at finished line width; look in T1999/T1996 for line widths.)

The Gerber tape format used works on many Gerber photo-plotters; however, many different plotters exist and several versions of software run on the plotter controllers.

Two problems can occur when attempting to photo-plot the layers.

- The file format is ANSI standard X3.27-1969, which means that most Gerber controllers see ANSI file format information as plot files and consequently give errors. The rule for determining the file which contains a given layer plot file is: F = 3L − 2, where F is the file number to plot and L is the layer to be plotted. For example, to plot layer 6, request the Gerber controller to plot file 16.
- The D-codes for changing the aperture in use are not preceded by a G54 tool select code. If your Gerber photo-plotter controller requires a G54 select code, you have two alternatives:
  - Read the photo-plot data into your VAX and write a program to detect all D-codes greater than or equal to 10 and precede these D-codes with a G54 select code.
  - Request a version of software from Gerber that assumes a G54 whenever a D-code is encountered.

Drill Tape Format

The drill tape format was chosen to give drill information in a format that can be translated into any of the other standard formats in use. The tape format can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999D, T1996D)

Excellon 0 reference, absolute ASCII with ASCII format leaders

The finished hole size is given in the ASCII format leaders. If your manufacturing process requires compensation on drill size, you must change the size in the ASCII format leaders. The data is in the form of absolute coordinates from the first hole. If you wish to read this data into your CAD system, then you need to shift the coordinates by the coordinate of the first hole from the origin that was used for the artwork data. The coordinates are as follows:

For T1999 drill data:

X offset = 1300
Y offset = 1900

For T1996 drill data:

X offset = 1300
Y offset = 1900

Gerber Wheel Data

The following table gives details of the Gerber wheel required to photo-plot the Gerber tapes.

Wheel No. 8, Fine Line Wheel, Pads

Design Outer Diam	Size Inner Diam	Shape Pad	App Pos	'D' Com	Plot Outer Diam	Size Inner Diam	Mif Size	Comments
187	0	C	1	10	187	0	187	
125	0	C	2	11	125	0	125	
100	0	C	3	12	100	0	100	
80	0	C	5	14	80	0	80	
70	0	C	6	15	70	0	70	
80	0	S	7	16	80	0	80	Square
40	0	C	8	17	40	0	40	
60	0	S	9	18	60	0	60	Square
61	0	C	20	27	61	0	61	
96	16	H	24	73	96	16	101	Heat Relief
5	0	C	22	29	5	0	5	

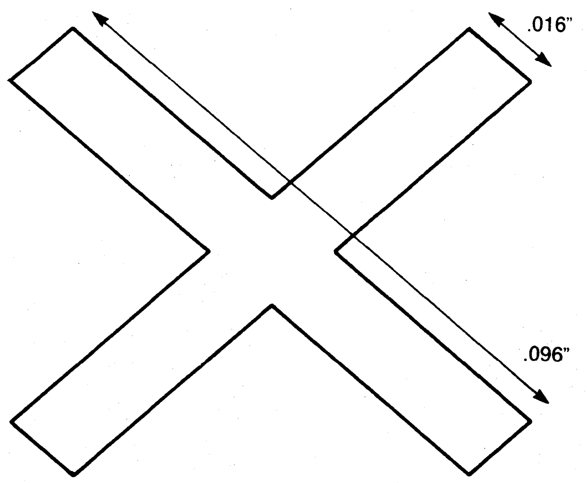
Shape Key: C - Circular, H - Heat Relief, X - Heat Relief, S - Square, R - Rectangular, D - Donut, L - Line, M - Slit Aperture

Wheel No. 8, Fine Line Wheel, Lines

Design Outer Diam	Size Inner Diam	Shape Pad	App Pos	'D' Com	Plot Outer Diam	Size Inner Diam	Mif Size	Comments
18	0	L	4	13	18	0	18	
100	0	L	10	19	100	0	100	
30	0	L	13	20	30	0	30	
15	0	L	14	21	15	0	15	
12	0	L	15	22	12	0	12	
12	0	L	16	23	12	0	12	
60	0	L	17	24	60	0	60	
10	0	L	18	25	10	0	10	
8	0	L	19	26	8	0	8	
40	0	L	21	28	40	0	40	
5	0	L	22	29	5	0	5	
50	0	L	11	70	50	0	50	
25	0	L	12	71	25	0	25	
6	0	L	23	72	6	0	6	

Shape Key: C - Circular, H - Heat Relief, X - Heat Relief, S - Square, R - Rectangular, D - Donut, L - Line, M - Slit Aperture

The following figure shows the shape of the heat relief pad.



Summary Information for T1999A

Tape Format: ANSI Standard (1600 BPI)
Volume Label: ARTT01
Character Set: ASCII
Coordinate Mode: Incremental
Coordinate Format: 2.4 (Tenths of mils, no leading zeros)

Artwork Tools Summary

Plot No.	Filename	No. of Blks.	Wheel No.	File No.	Filemark	Layer(s)	Type of Plot
1	T1999X.R01	51	W8	1	1	1	Regular Artwork
2	T1999X.R02	32	W8	2	4	2	Regular Artwork
3	T1999X.R03	25	W8	3	7	3	Regular Artwork
4	T1999X.R04	27	W8	4	10	4	Regular Artwork
5	T1999X.R05	26	W8	5	13	5	Regular Artwork
6	T1999X.R06	27	W8	6	16	6	Regular Artwork
7	T1999X.R07	26	W8	7	19	7	Regular Artwork
8	T1999X.R08	24	W8	8	22	8	Regular Artwork
9	T1999X.R09	28	W8	9	25	9	Regular Artwork
10	T1999X.R10	48	W8	10	28	10	Regular Artwork

X - Revision

Summary Information for T1996A

Tape Format: ANSI Standard (1600 BPI)
Volume Label: ARTT01
Character Set: ASCII
Coordinate Mode: Incremental
Coordinate Format: 2.4 (Tenths of mils, no leading zeros)

Artwork Tools Summary

Plot No.	Filename	No. of Blks.	Wheel No.	File No.	Filemark	Layer(s)	Type of Plot
1	T1996X.R01	42	W8	1	1	1	Regular Artwork
2	T1996X.R02	20	W8	2	4	2	Regular Artwork
3	T1996X.R03	19	W8	3	7	3	Regular Artwork
4	T1996X.R04	21	W8	4	10	4	Regular Artwork
5	T1996X.R05	19	W8	5	13	5	Regular Artwork
6	T1996X.R06	21	W8	6	16	6	Regular Artwork
7	T1996X.R07	20	W8	7	19	7	Regular Artwork
8	T1996X.R08	19	W8	8	22	8	Regular Artwork
9	T1996X.R09	22	W8	9	25	9	Regular Artwork
10	T1996X.R10	37	W8	10	28	10	Regular Artwork

X - Revision

Appendix D Software Services Directory

DECdirect

DECdirect add-ons, upgrades, accessories, and supplies for all Digital systems:

- Mail orders to:
  Digital Equipment Corporation
  P.O. Box CS2008
  Nashua, NH 03061
- Call toll free:
  800-DIGITAL
- For technical presales assistance:
  call 800-343-4040
- Electronic ordering (at 1200/2400 baud):
  800-332-3366

Digest Training Update and Schedule

Digital Educational Services Today is published quarterly by the Educational Services Department of Digital Equipment Corporation.

For further information on Educational Services' products and services, contact your account representative or your nearest Digital Equipment Corporation sales office.

The Digest publishes a six-month schedule of software courses held in U.S. training centers; a nine-month schedule is included for all hardware courses held in the U.S. Schedules are listed by operating systems for software training and by specific categories for hardware maintenance training. Each schedule is then listed by location to allow you to plan training at the most convenient site.

Several courses are also available in a self-paced instruction format for purchase and use at your work site. For more information, write to Digital Equipment Corporation, Educational Services BUO/E55-46, 12 Crosby Drive, Bedford, MA 01730; or contact Customer Support Telephone: 1-800-332-5656 X20 or Seminar Programs: (617) 276-4949.

Customer Course Catalog from Educational Services

The *Customer Course Catalog* is a comprehensive reference of Digital's quality training programs.

The catalog contains information on language training, VAX/VMS training, MicroVAX/VMS training, VAX information architecture training, VMS programmer productivity tools, network training, office administration training, artificial intelligence training, RSTS/E training, RT-11 training, TOPS-10/20 training, hardware maintenance training, (VAX, PDP-11, data communications) and computer integrated manufacturing (BASEWAY) training.

For additional information, call your Digital Sales Representative or the training center nearest you.

The Boston area training center is located in Bedford, MA. The telephone number is (617) 276-4380.

Software Documentation Products Directory

The *Software Documentation Products Directory* includes all current products that are available for ordering through DECdirect. This directory also contains information on unreleased products. It is a complete source of product information and supersedes all other directories.

Products that become archived will not be offered for sale. Digital's customers will be given the right to copy, at no charge any Digital Archival Software Documentation Publication (excluding restricted or third-party products) that we no longer offer. However, the copyright is retained as the exclusive property of Digital Equipment Corporation.

Documentation within the major hardware families is arranged by operating systems and then layered products. Documentation kits list the order number followed by the description and then the price.

All current documentation will be shipped within 30 days.

To order by phone in the U.S.A., call toll free 1-800-258-1710.

To order by direct mail in the U.S.A., write to Digital Equipment Corporation, P.O. Box CS2008, Nashua, NH 03061

Device Drivers and Handlers

Writing a Device Driver

Writing a Device Driver for VAX/VMS is a standard manual available separately for VAX/VMS systems. It is an excellent reference source for device driver development support. Order Part Number: AA-Y511B-TE.

DRB32 VMS Driver

DRB32 VMS Drivers, Version 1.0, supports the DRB32 options (DRB32-M, DRB32-E and DRB32-W). It provides device drivers and other programs that support the DRB32 under VAX/VMS on VAXBI systems.

Included in the kit are two device drivers: one for the DRB32-M/DRB32-E, and one for the DRB32-W DR11-W-emulation.

DRB32-M/DRB32-E driver features include a simple interface for user programs.

The DRB32-W driver is a modified version of the VMS DR11-W driver, with a nearly identical user QIO interface.

- Refer to SPD number 27.69
- CPU/Operating System: VAX/VMS
- Price: Available upon request
- Contact SDC Hotline: (617) 874-3383
- Digital Equipment Corporation
  146 Main Street
  Maynard, MA 01754-2752

UPDATE NOTICE NO. 3

VAXBI Designer's Notebook

EK-VBIDS-R3-001

This update contains revised pages. The pages enclosed in this package are listed below:

B-1 through B-4

Digital Equipment Corporation • Maynard, Massachusetts

Appendix B VAXBI Base Layout Package

The following information is from the cover letter that is distributed with the VAXBI Base Layout Package.

This document describes the contents of the documents and magnetic tapes in this package. This package provides the necessary information to design and build either a Standard VAXBI Module or an Expansion VAXBI Module.

The following documents are supplied:

T1999	Standard Module Control Drawing
T1996	Expansion Module Control Drawing
ELEN 633	Layup Specification
ELEN 626	Mechanical Outline Specification

(References within these documents to DIGITAL internal documents are for the use of DIGITAL manufacturing. These documents are not needed by customers.)

The following magnetic tapes are supplied:

T1999A	Gerber Tape (10 artwork layers)
T1999D	Drill Tape
T1996A	Gerber Tape (10 artwork layers)
T1996D	Drill Tape

This package contains information for two types of VAXBI modules. T1999 and its associated databases are for the standard VAXBI module with a VAXBI Corner; T1996 and its associated databases are for the VAXBI expansion module. ELEN 626, ELEN 633, and the Gerber wheel description apply to both types of VAXBI modules. The Gerber tapes are for the 10 layers of the module and, when plotted, will produce the artwork for the DIGITAL-defined portions of a VAXBI module. The drill tapes contain the necessary data for drilling the holes in the DIGITAL-defined portions of the module. The precise format of the data on these tapes is described below.

The artwork for the module has added certain special features to assist PCB manufacturing. The connector area gold plating feature that DIGITAL manufacturing uses has been added; it can be removed and replaced if desired. Beware that if you supply your own plating feature and it does not work, you must not use the module in a VAXBI backplane because the module will damage the connector if it is not properly plated. Each layer has two targets on it to assist in alignment; DIGITAL's tooling features have not been added.

The intent is that a designer can produce boards either by reading the Gerber data into a CAD system as it is provided to produce a layout or by plotting the artwork and then digitizing it back into the CAD system. (If you intend to do the latter, you should plot the artwork at 2:1 at least. Feedback from users has indicated that 4:1 is preferable.)

Note: A manufacturing problem can exist when removing the finished PCB from the panel in which it was etched. Because some shearing equipment cannot hold the necessary tolerances on the module outline, we recommend routing the PCB out of the panel in these cases.

Gerber Tape Format

The decision to use Gerber tape format for the artwork was made to give the best coverage in terms of photo-plotting systems in use. The format on tape can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999A, T1996A)

ASCII format, incremental coordinates

Coordinate format 2.4 (no leading zeros)

All coordinates are positive X and positive Y. The coordinates are referenced to the Gerber plotter origin, but the artwork is offset by 1.1" in the X direction and 1.7" in the Y direction (DIGITAL manufacturing's standard offset for VAXBI modules). The artwork produced is data as viewed from side 2. The first five layers are wrong reading; the bottom five layers are right reading. It is recommended that you verify that the data does not require a coordinate translation for the plotter system in use. (Note that all pads and traces will be plotted at finished line width; look in T1999/T1996 for line widths.)

The Gerber tape format used works on many Gerber photo-plotters; however, many different plotters exist and several versions of software run on the plotter controllers.

Two problems can occur when attempting to photo-plot the layers.

- The file format is ANSI standard X3.27-1969, which means that most Gerber controllers see ANSI file format information as plot files and consequently give errors. The rule for determining the file which contains a given layer plot file is: F = 3L − 2, where F is the file number to plot and L is the layer to be plotted. For example, to plot layer 6, request the Gerber controller to plot file 16.
- The D-codes for changing the aperture in use are not preceded by a G54 tool select code. If your Gerber photo-plotter controller requires a G54 select code, you have two alternatives:
  - Read the photo-plot data into your VAX and write a program to detect all D-codes greater than or equal to 10 and precede these D-codes with a G54 select code.
  - Request a version of software from Gerber that assumes a G54 whenever a D-code is encountered.

Drill Tape Format

The drill tape format was chosen to give drill information in a format that can be translated into any of the other standard formats in use. The tape format can be summarized as follows:

1600 BPI ANSI format magnetic tape (volume labels T1999D, T1996D)

Excellon 0 reference, absolute ASCII with ASCII format leaders

The finished hole size is given in the ASCII format leaders. If your manufacturing process requires compensation on drill size, you must change the size in the ASCII format leaders. The data is in the form of absolute coordinates from the first hole. If you wish to read this data into your CAD system, then you need to shift the coordinates by the coordinate of the first hole from the origin that was used for the artwork data. The coordinates are as follows:

For T1999 drill data:

X offset = 1300
Y offset = 1900

For T1996 drill data:

X offset = 1300
Y offset = 1900

Gerber Wheel Data

The following table gives details of the Gerber wheel required to photo-plot the Gerber tapes.

Wheel No. 8, Fine Line Wheel, Pads

Total Pad Apertures: 11

Dimension_1	Dimension_2	Shape	D_Command
187.00	0.00	C	10
125.00	0.00	C	11
100.00	0.00	C	12
80.00	0.00	C	14
80.00	0.00	S	16
70.00	0.00	C	15
40.00	0.00	C	17
61.00	0.00	C	27
60.00	0.00	C	27
96.00	16.00	X	73
60.00	0.00	S	18

Shape Key: C - Circular, H - Heat Relief, X - Heat Relief, S - Square, R - Rectangular, D - Donut, L - Line, M - Slit Aperture

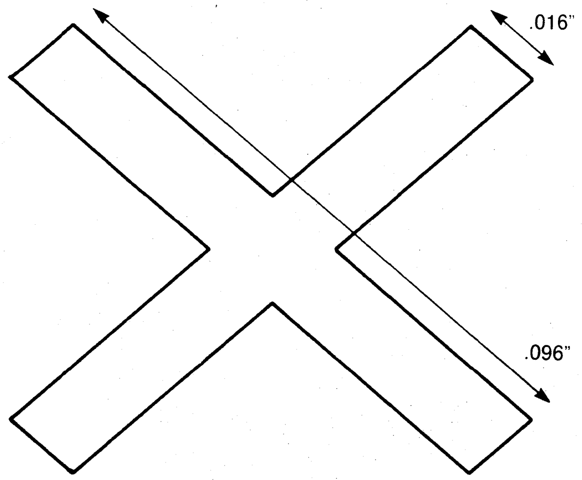
Wheel No. 8, Fine Line Wheel, Lines

Total Line Apertures: 13

Dimension_1	Dimension_2	Shape	D_Command
100.00	0.00	L	19
50.00	0.00	L	70
25.00	0.00	L	71
30.00	0.00	L	20
15.00	0.00	L	21
18.00	0.00	L	13
12.00	0.00	L	23
60.00	0.00	L	24
10.00	0.00	L	25
8.00	0.00	L	26
40.00	0.00	L	28
5.00	0.00	L	29
6.00	0.00	L	72

Shape Key: C - Circular, H - Heat Relief, X - Heat Relief, S - Square, R - Rectangular, D - Donut, L - Line, M - Slit Aperture

The following figure shows the shape of the heat relief pad.



Summary Information for T1999A
Tape Format: ANSI Standard (1600 BPI)
Volume Label: ARTT01
Character Set: ASCII
Coordinate Mode: Incremental
Coordinate Format: 2.4 (Tenths of mils, no leading zeros)

Artwork Tools Summary

Plot No.	Filename	No. of Blks.	Wheel No.	File No.	Filemark	Layer(s)	Type of Plot
1	T1999X.R01	50	W8	1	1	1	Regular Artwork
2	T1999X.R02	32	W8	2	4	2	Regular Artwork
3	T1999X.R03	25	W8	3	7	3	Regular Artwork
4	T1999X.R04	27	W8	4	10	4	Regular Artwork
5	T1999X.R05	26	W8	5	13	5	Regular Artwork
6	T1999X.R06	27	W8	6	16	6	Regular Artwork
7	T1999X.R07	26	W8	7	19	7	Regular Artwork
8	T1999X.R08	24	W8	8	22	8	Regular Artwork
9	T1999X.R09	29	W8	9	25	9	Regular Artwork
10	T1999X.R10	48	W8	10	28	10	Regular Artwork

X - Revision

Summary Information for T1996A
Tape Format: ANSI Standard (1600 BPI)
Volume Label: ARTT01
Character Set: ASCII
Coordinate Mode: Incremental
Coordinate Format: 2.4 (Tenths of mils, no leading zeros)

Artwork Tools Summary

Plot No.	Filename	No. of Blks.	Wheel No.	File No.	Filemark	Layer(s)	Type of Plot
1	T1996X.R01	42	W8	1	1	1	Regular Artwork
2	T1996X.R02	20	W8	2	4	2	Regular Artwork
3	T1996X.R03	19	W8	3	7	3	Regular Artwork
4	T1996X.R04	21	W8	4	10	4	Regular Artwork
5	T1996X.R05	19	W8	5	13	5	Regular Artwork
6	T1996X.R06	21	W8	6	16	6	Regular Artwork
7	T1996X.R07	20	W8	7	19	7	Regular Artwork
8	T1996X.R08	19	W8	8	22	8	Regular Artwork
9	T1996X.R09	22	W8	9	25	9	Regular Artwork
10	T1996X.R10	37	W8	10	28	10	Regular Artwork

X - Revision

digital™

Digital Equipment Corporation