EY-5554E-SG-0002

VAXBI Adapters

Student Guide

digital™

Educational Services

Second Edition, February 1987

Printed in U.S.A.

CONTENTS

UNIT 4 CIBCI ADAPTER

FIGURES

TABLES

EXAMPLES

STUDENT GUIDE

INTRODUCTION - THE VAXBI ADAPTERS INFORMATION DATABASE

This database brings together important field service information about specific VAXBI adapters. This information closely follows that presented in a variety of technical documents. With this close correlation, you, a Field Service engineer, should be able to easily relate information presented here to the documentation you use in the field.

The following lists the adapters covered in the database and their categories:

- VAXBI window adapters
  - DWBUA
- Programmed I/O adapters (PIO)
  - DMB32
- Port adapters
  - DEBNT
  - CIBCI
  - KDB50
  - CIBCA

Practice and Tests

The VAXBI adapter information database is designed as a composite information source. The information is not a structured course and, therefore, does not include any practice exercises or tests.

The Menu

While going through this database, you can customize it to meet your needs and help you make the most efficent use of your time. Before entering the database, you are presented with a grid symbolizing the entire course. You may choose those parts of the database that you would like to take.

The leftmost column of the Menu (see Figure SG-1) lists the adapters for which instruction is available. The top row lists the sections available under each adapter. For example, the square in the row labeled DEBNT Adapter under the Installation column would indicate the installation section of the DEBNT adapter.

CHOOSE ALL	OVERVIEW	REGISTERS	INSTALLATION	TROUBLESHOOTING	DOCUMENTATION
DWBUA ADAPTER					
DM832 ADAPTER					
DEBNT ADAPTER					
CIBCI ADAPTER					
KD850 ADAPTER					
CIBCA ADAPTER					

MKV87-0213

Figure SG-1 The Menu

Any of the four arrow keys may be used to move from square to square. If you move left from the leftmost square of a row, you will end up on the rightmost square of that row. If you move right from the rightmost square, you will end up on the leftmost square. If you move up from the top square of a column, you will go to the bottom square, and if you move down from the bottom square, you will go to the top square.

To choose or "check off" a particular section, you must move the highlight to that section and press the RETURN key. You can choose to take all of the instruction for an adapter by moving the highlight to the adapter name and pressing RETURN. And if you would like to take all of the instruction for all of the adapters, you must move the highlight to "CHOOSE ALL" and press RETURN.

To delete a section which has already been chosen, simply move the highlight to that section and press RETURN. The checkmark which was there will disappear.

When you are sure that you have finished choosing the course sections you would like to take, simply press ENTER. Your course choices will be stored, and will be used to determine your path through the course. Please be sure that you have chosen everything you want, because you will not be able to take sections that you have not chosen, and you will not be able to make any changes once you have pressed ENTER.

When you are in the database and choose to see the menu, you are presented with this same grid, but at that point you are only able to use it for moving around in the database. You will only be able to go into sections of the database that have been checked. When you move the highlight to a checked section and press RETURN, you will proceed to that section of the database. If it is not checked, you will stay where you are, and the grid menu will remain active.

TYPES OF ADAPTERS

The various adapters on the VAXBI facilitate the transfer of data and control information between the VAXBI and peripheral buses, communications lines, or peripheral devices. These adapters attach directly to the VAXBI, and can be classified into three general categories:

- Programmed I/O, or PIO adapters
- Window adapters
- Port adapters

The three categories differ in their increasingly distributed intelligence. For example, a port-style adapter performs much more "work" than would a window-style or PIO adapter attached to the same device. The design complexity is evaluated against the desire to off-load as much IO overhead as possible from the host.

This following sections discuss each of the adapter categories.

PIO Adapters

PIO adapters, shown in Figure SG-2, are called programmed I/O adapters because software, rather than hardware, is responsible for managing the transfer of data to and from these devices. Data transfer transactions, for the PIO style adapter, are always issued by the host processor, rather than the adapter itself. Most PIO adapters use interrupts to inform the CPU and the software that they have data to transfer.

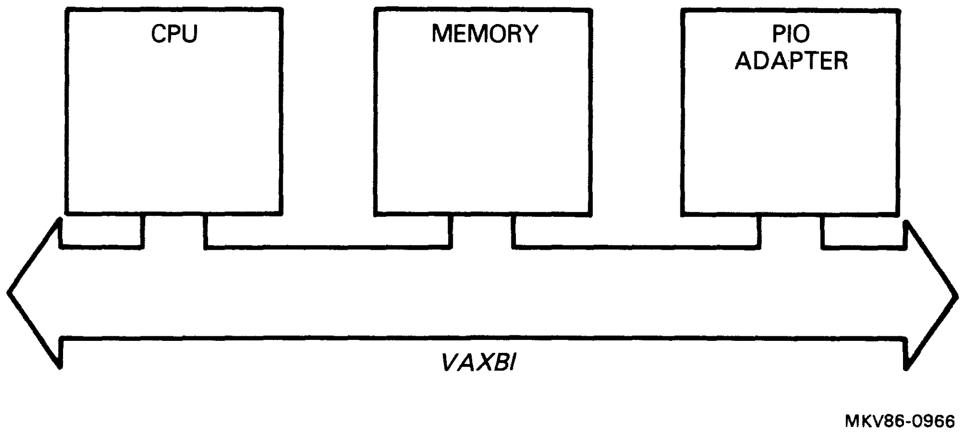


Figure SG-2 PIO Adapter

Window Adapters

A VAXBI window adapter, shown in Figure SG-3, maps all of the addresses in the window space of the VAXBI to the address space used on another bus, and vice versa. This map is used to translate transactions and addresses in a transparent manner from the VAXBI bus and another bus.

The UNIBUS adapter, or DWBUA, is an example of a window adapter. The 256 kilobytes in each of the window spaces on the VAXBI is enough to map the entire address space of the UNIBUS.

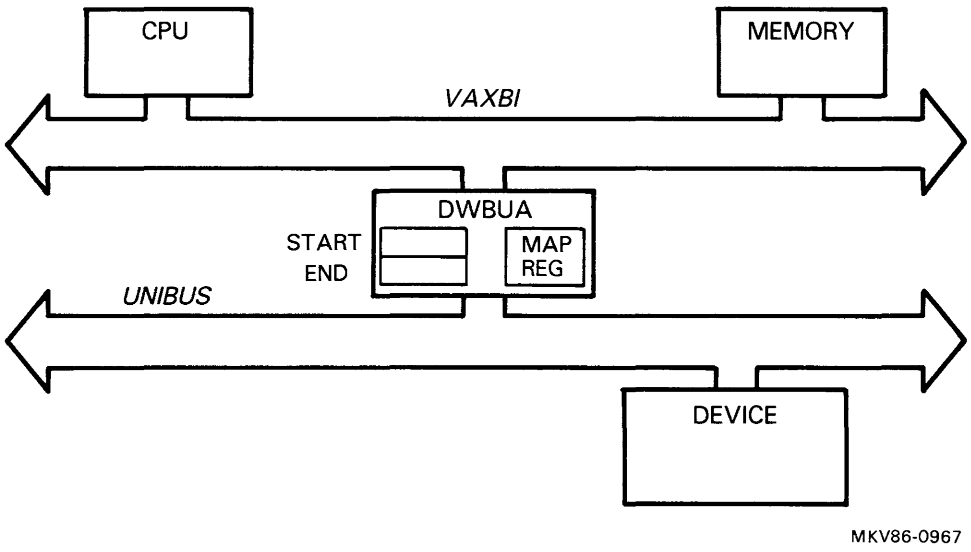


Figure SG-3 Window Adapter

Port Adapters

The last major category of VAXBI adapters is that of port adapters. (See Figure SG-4.) Port adapters interface other interconnects to the VAXBI. Two examples of other interconnects are the cluster interconnect and storage interconnect.

Port adapters can access page tables to translate adapter virtual addresses into physical addresses. This allows the host to specify a data transfer in terms of virtual addresses without having to set up map registers. Port adapters also access main memory queues to fetch command packets and deposit control and status information. This port adapter function allows the processor to issue commands and examine responses independently of the working of the adapters.

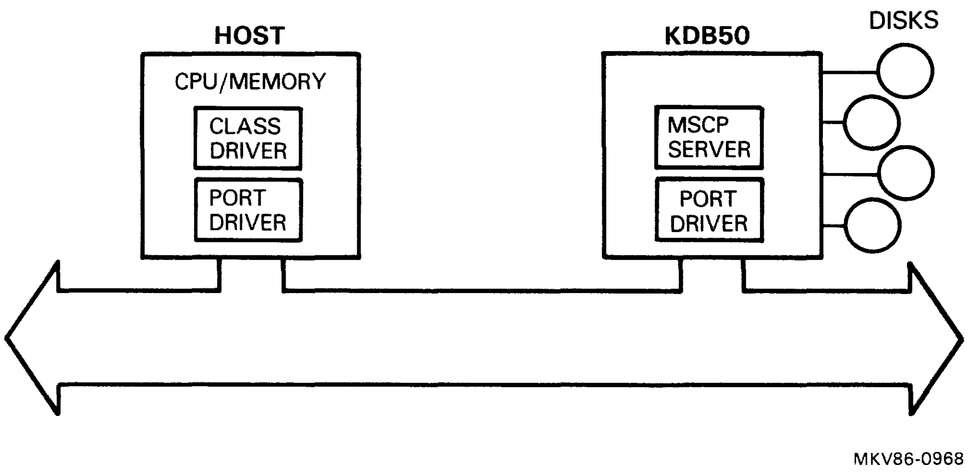


Figure SG-4 Port Adapter

VAXBI ADDRESS SPACE

All VAXBI adapters utilize registers. To understand these registers, it is helpful to know how they fit into the VAXBI address space.

The VAXBI uses 30 bits for addressing; these 30 bits provide 1 gigabyte of physical address space. This space is divided into two equal sections by the most significant bit -- the memory space and the I/O space. All of the memory on the VAXBI is located within the first half of the address space. The I/O space is further divided into two segments as shown in Figure SG-5.

The first of these segments is called NODE ID space. This space is divided into sixteen slots, each node from 0 to F has 8 kilobytes of space reserved for it in this range. The registers in this space handle the necessary VAXBI housekeeping functions for control of the node. These registers also help the node to communicate with the VAXBI and contain the needed diagnostic information for each node.

The space for each node is further divided into three sets of registers:

- VAXBI-required registers
- BIIC-specific device registers
- Device-specific registers

The VAXBI-required registers and the BIIC-specific device registers are found in the BIIC and are the same for all nodes on the VAXBI although they may be used in a slightly different way on each indivual node. The device-specific registers are unique for each of the nodes depending on the needs of the specific adapter. In the presentation on each of the adapters, the most important information about these registers is highlighted.

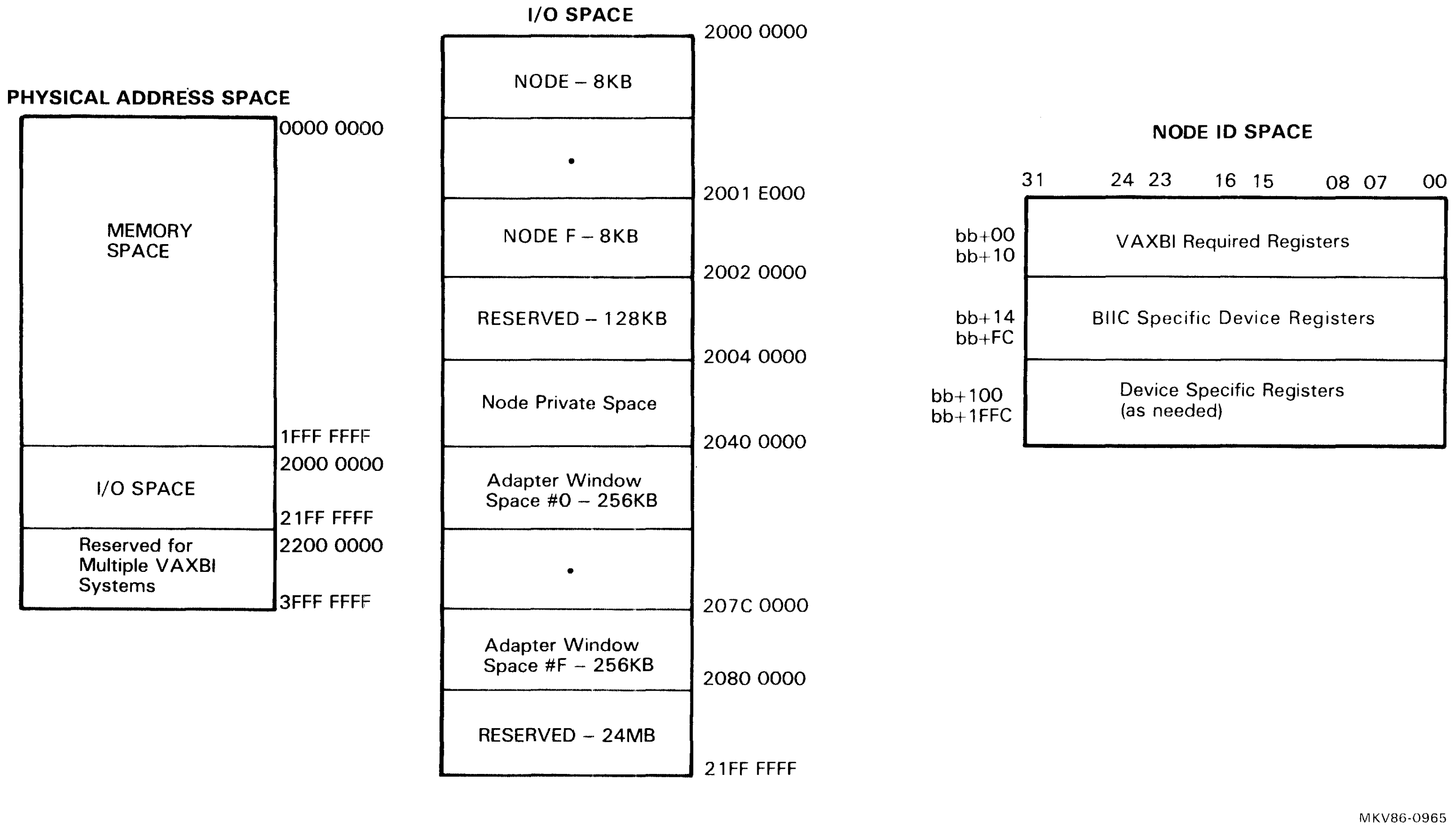


Figure SG-5 VAXBI Address Space

THINGS TO REMEMBER

WARNING

Shut off the system power and disconnect the main system power cord before performing any procedure in this Student Guide.

CAUTION

You must wear an anti-static wrist strap connected to an active ground whenever you work on a system with the covers removed, or handle any of the adapter modules.

CAUTION

The anti-static wrist strap is located within the system cabinet of the host system and is connected to ground. Place this wrist strap on your wrist before performing any of the following procedures.

CAUTION

All adapter modules are supplied in protective anti-static packaging. Do not remove a module from the packaging until you are about to install it.

DWBUA ADAPTER

LESSON 1 - INTRODUCTION

The VAXBI-to-UNIBUS adapter, the DWBUA, enables transfers between the high-speed synchronous VAXBI bus and the asynchronous UNIBUS. (See Figure 1-1.) Through the DWBUA, the VAXBI has access to any UNIBUS address, and the UNIBUS has access to any VAXBI address.

The DWBUA transfers data between the buses in two ways:

1. Through the direct data path, where data is transferred immediately

2. Through a buffered data path, where the DWBUA internally buffers as much as one octaword of data per transfer to maximize the VAXBI bandwidth

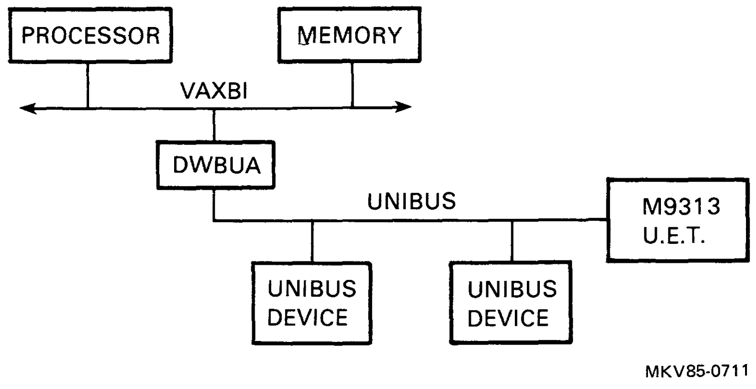


Figure 1-1 Typical DWBUA Configuration

All VAXBI-initiated transactions transfer data through the direct data path. UNIBUS-initiated transactions can transfer data through either the direct data path or a buffered data path.

Other features of the DWBUA are:

- It acts as the arbitrator for all UNIBUS devices.
- UNIBUS devices can use the DWBUA to interrupt on the VAXBI.
- Up to 1.0 Mbytes/second of data is transferred between the DWBUA and the VAXBI.
- It has a self-test to verify data paths and control logic, and to report failures to the VAXBI.

Functional Description

The functional description of the DWBUA is presented in two parts:

1. The first part describes the components on the block diagram.
2. The second part explains how the DWBUA interfaces between the two buses.

Block Diagram - The BIIC is located in the VAXBI corner of the module and is the standard interface for all modules to the VAXBI. (See Figure 1-2.) It is through the BIIC that data is transferred between the DWBUA and the VAXBI. The BIIC handles most VAXBI signals except for BI BAD, which the DWBUA drives until it passes the self-test.

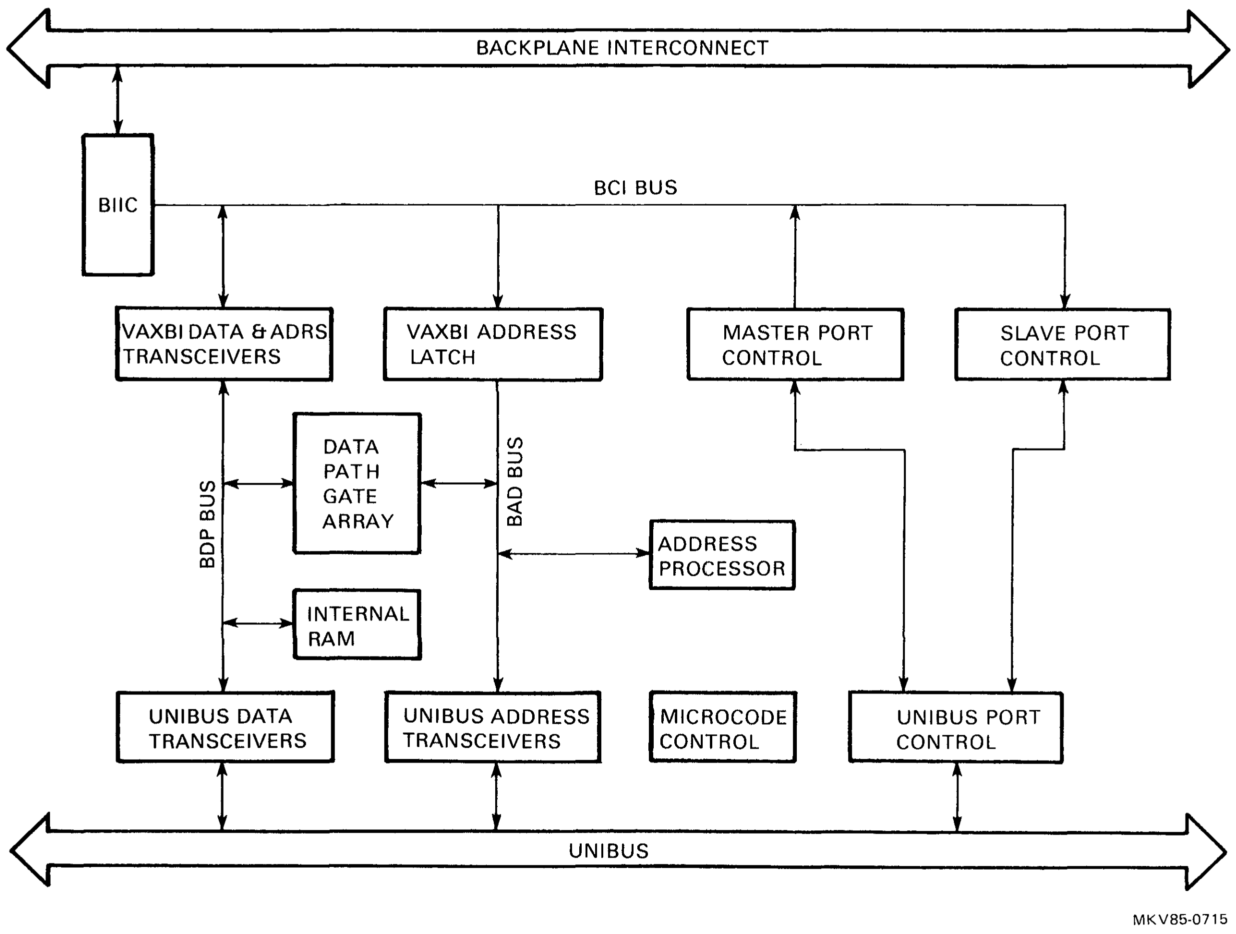


Figure 1-2 DWBUA Block Diagram

To communicate with the BCI, the DWBUA uses the following four functional boxes as the transceivers and control logic:

1. VAXBI data and address transceivers
2. VAXBI address latch
3. Master port control
4. Slave port control

The DWBUA arbitrates which UNIBUS device gets the bus. The following three blocks provide this functionality as well as providing the drivers and receivers for the UNIBUS signals:

1. UNIBUS data transceivers
2. UNIBUS address transceivers
3. UNIBUS port control

The last four functional boxes are used to handle the transition between the two buses and control the self-test:

1. Data path gate array
2. Internal RAM
3. Address processor
4. Microcode control

Transactions - The DWBUA acts as a translator between the VAXBI and the UNIBUS. It interprets commands received from one bus and translates them into a format that the other bus can understand. It provides controls and responses that enable the completion of these commands. These sequences of commands, controls, and responses are called DWBUA transactions. In this section, some typical transactions that are handled by the DWBUA are examined in detail.

DWBUA transactions are divided into three catagories:

1. VAXBI-to-DWBUA transactions
2. VAXBI-to-UNIBUS transactions
3. UNIBUS-to-VAXBI transactions

The block diagram in Figure 1-3 shows four examples. Most of the boxes in this diagram are labeled in the same way as in Figure 1-2. The two blocks with different labels are the buffer data path, which is in the data path gate array, and the map registers, which are in the internal RAM. Although the other boxes on the detailed block diagram play a part in the transaction, they were removed to simplify the illustration of the the flow of data through the DWBUA.

VAXBI commands and addresses are received by the DWBUA. The slave port control determines that the transaction is for the DWBUA. The VAXBI address is latched in the VAXBI address latch. The VAXBI address is the address of the UNIBUS map register that will be written.

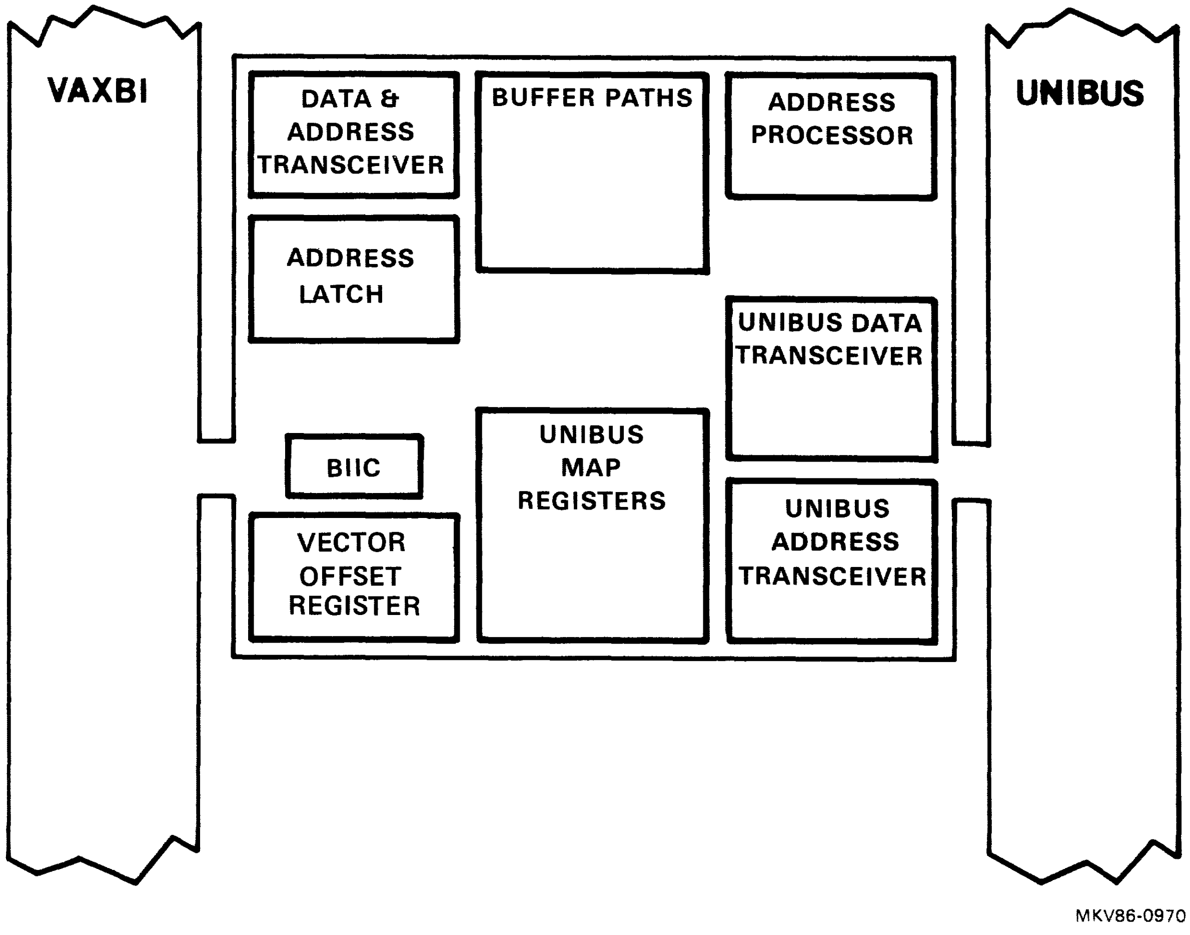


Figure 1-3 Transactions Diagram

LESSON 2 - DWBUA REGISTERS

As shown in Figure 1-4, the DWBUA module's node ID space is divided into three sets of registers:

1. VAXBI required registers
2. BIIC-specific device registers
3. DWBUA registers

The VAXBI-required registers and the BIIC-specific device registers are used in the same way as on other nodes on the VAXBI. The DWBUA registers are unique to this adapter. The most important registers in the node ID space are covered below in greater detail.

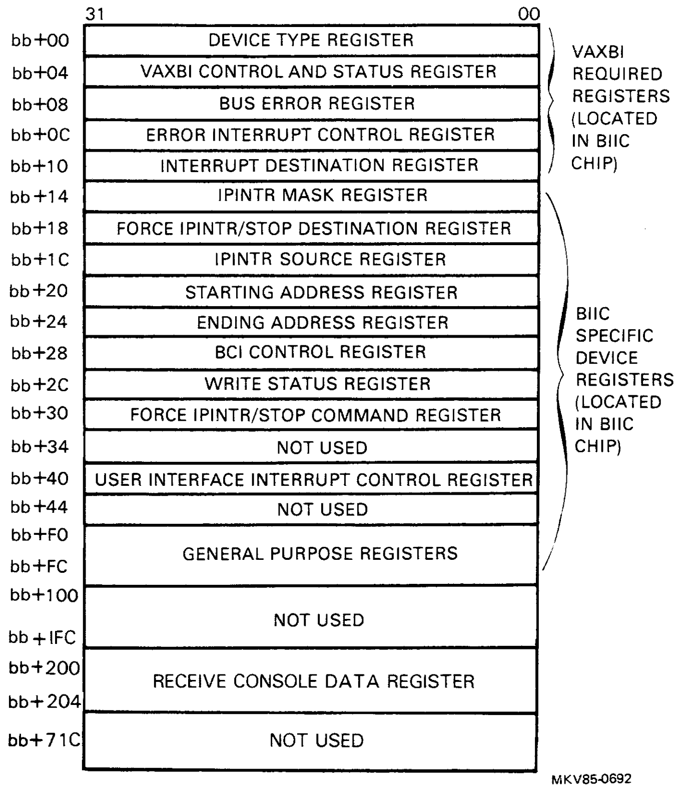

31
00
bb+00
DEVICE TYPE REGISTER
bb+04
VAXBI CONTROL AND STATUS REGISTER
bb+08
BUS ERROR REGISTER
bb+0C
ERROR INTERRUPT CONTROL REGISTER
bb+10
INTERRUPT DESTINATION REGISTER
bb+14
IPINTR MASK REGISTER
bb+18
FORCE IPINTR/STOP DESTINATION REGISTER
bb+1C
IPINTR SOURCE REGISTER
bb+20
STARTING ADDRESS REGISTER
bb+24
ENDING ADDRESS REGISTER
bb+28
BCI CONTROL REGISTER
bb+2C
WRITE STATUS REGISTER
bb+30
FORCE IPINTR/STOP COMMAND REGISTER
bb+34
NOT USED
bb+40
USER INTERFACE INTERRUPT CONTROL REGISTER
bb+44
NOT USED
bb+F0
bb+FC
GENERAL PURPOSE REGISTERS
bb+100
bb+IFC
NOT USED
bb+200
bb+204
RECEIVE CONSOLE DATA REGISTER
bb+71C
NOT USED
VAXBI REQUIRED REGISTERS (LOCATED IN BIIC CHIP)
BIIC SPECIFIC DEVICE REGISTERS (LOCATED IN BIIC CHIP)
MKV85-0692

Figure 1-4 DWBUA Address Space (Sheet 1 of 2)

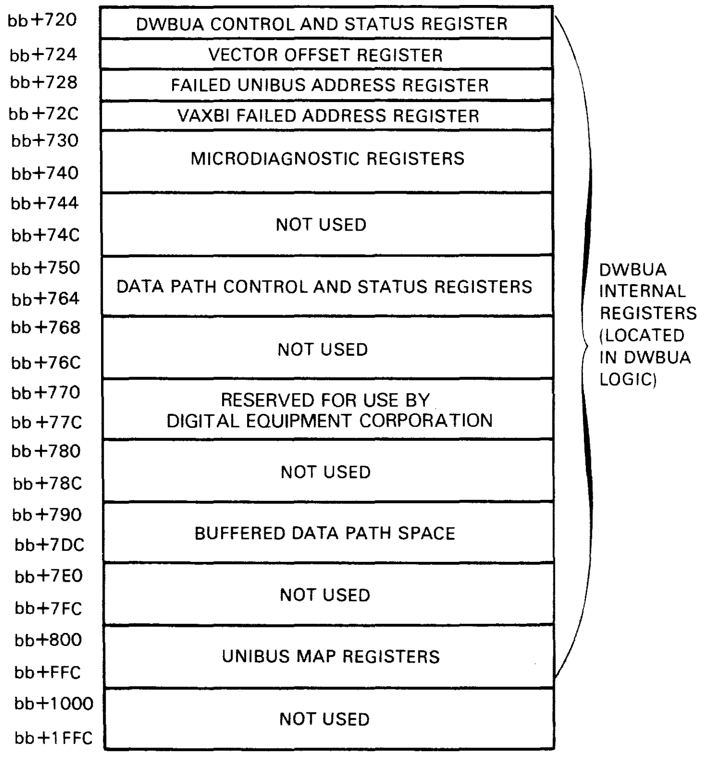


Figure 1-4 DWBUA Address Space (Sheet 2 of 2)

LESSON 3 - INSTALLATION

This lesson covers the installation of the DWBUA into both the VAXBI cage and the UNIBUS cage.

CAUTION
You must wear an anti-static wrist strap connected to an active ground when you install the DWBUA.

1. Attach the four UNIBUS cables to the M7166 paddle card. The colored stripe is to the left. Refer to Figure 1-5.

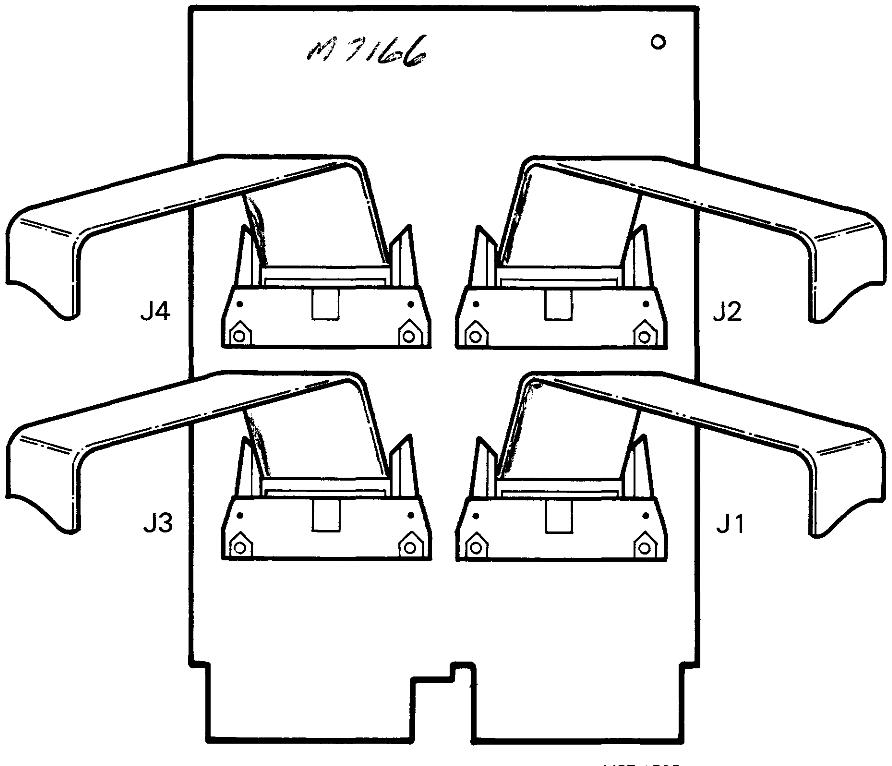


Figure 1-5 Paddle Card with UNIBUS Cables

2. Insert the M7166 paddle card into slot 1, segments A and B, of the UNIBUS backplane, as shown in Figure 1-6.

3. Insert the M9313 UET module into the last slot, segments A and B, of the UNIBUS backplane.

4. Insert grant continuity cards in all unused UNIBUS slots.

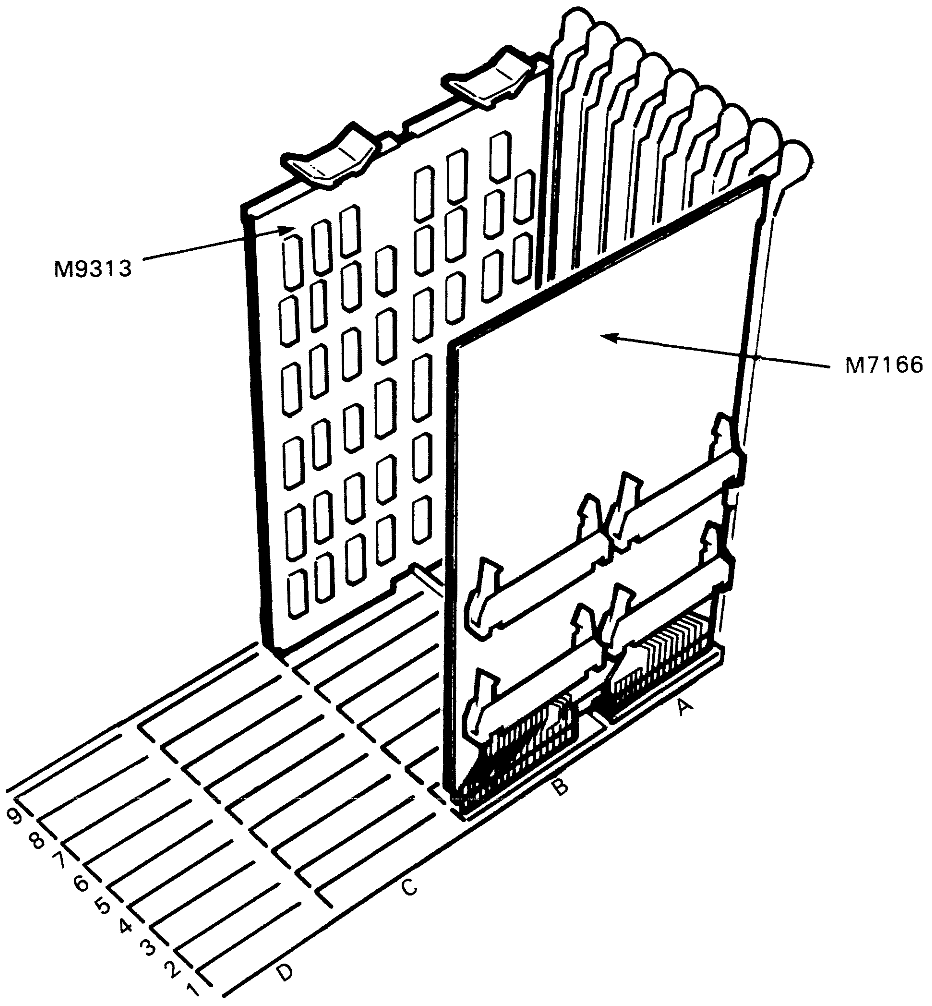


Figure 1-6 UNIBUS Backplane

The UET module enables diagnostic testing of the DWBUA module's capabilities to handle UNIBUS addressing, data transfers, and interrupts. Table 1-1 lists the addresses of registers provided by the UET module for use in diagnostic testing.

Table 1-1 UNIBUS Exerciser Terminator Registers

Register Address (octal)	Register Name/Bits	Notes
772140	Address Register A<15:00>	Word load only. Byte loading causes timeout.
772142	Data Register D<15:00>	Both byte and word loading allowed.
772144	Control Register CR<15:00>	Word load only. Byte loading causes timeout

5. Install the transition header on the backplane of the slot that will hold the T1010 module. See Figure 1-7.

   The T1010 module may be installed in any empty VAXBI slot except slot 1. It is suggested, however, that it be installation in the next empty slot after slot 1.

   When installing the transition header, use only the torque screwdriver provided in the Field Service kit.

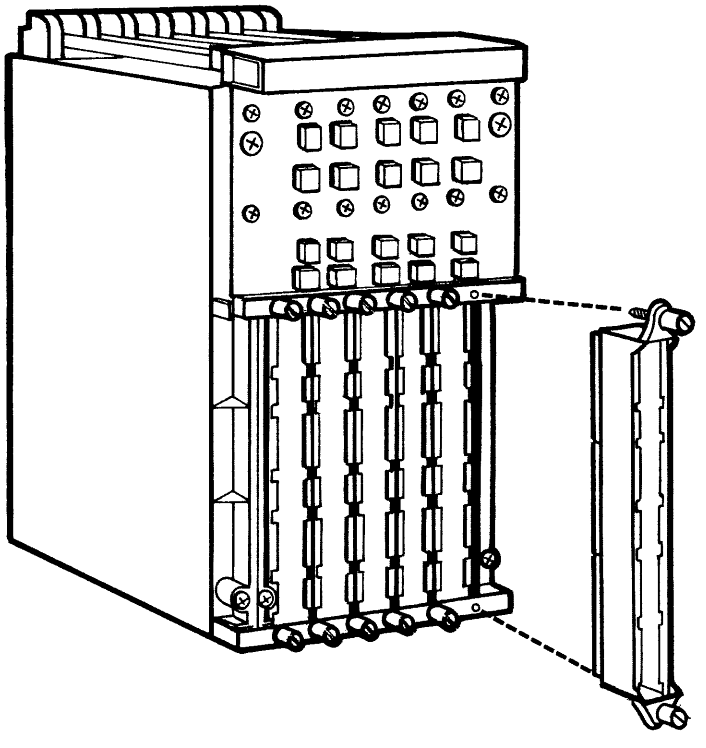


Figure 1-7 VAXBI Transition Header Installation

6. Connect the four UNIBUS cables to the transition header assembly as shown in Figure 1-8. The right and left are relative to the cage in this position.

   - J1 - segment E (left)
   - J2 - segment E (right)
   - J3 - segment D (left)
   - J4 - segment D (right)

   Keep the colored stripe on each cable toward the VAXBI printed circuit card.

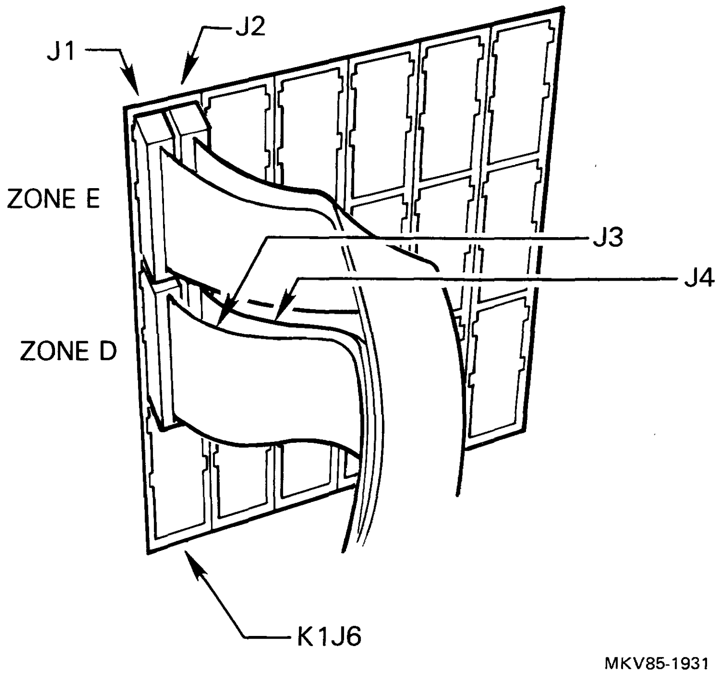


Figure 1-8 UNIBUS Cable Connections

7. Insert the T1010 module into the appropriate slot of the VAXBI cardcage.

8. If the UNIBUS backplane is in an expansion cabinet, the power bus cable can be installed between the processor cabinet and the expansion cabinet, to allow the keyswitch to control the application of power to the UNIBUS devices.

9. Power up the system. The DWBUA self-test runs upon power-up. Ensure that the yellow LED on the T1010 module lights.

   If the yellow LED does not light, there is a problem. Such problems are covered in the next section.

10. Finally, run two full passes of EVCBB, the DWBUA macrodiagnostic program to ensure that the system is working properly.

Self-Test

NOTE

A UNIBUS Exerciser Terminator (UET) module, M9313, must be installed in the last slot, segments A and B, of the UNIBUS backplane. The DWBUA self-test will run only if the UET module is installed.

The DWBUA self-test runs at power-up or when the VAXBI control and status register RESTART bit (BICSR <10>) is set. Successful completion of the self-test is indicated by the lighting of the yellow LED on the T1010 module. The location of the LED is shown in Figure 2-7.

The DWBUA self-test consists of 18 separate tests, which are described in Appendix C of the DWBUA UNIBUS Adapter Technical Manual, EK-DWBUA-TM.

Macrodiagnostic Program

The macrodiagnostic program for the DWBUA is EVCBB. It is a level 3 diagnostic (it runs stand-alone under the VAX diagnostic supervisor), and it isolates failures to the failing function.

Descriptions of the macrodiagnostic tests can be found in Appendix D of the DWBUA UNIBUS Adapter Technical Manual.

To run EVCBB, do the following:

1. Run the VAX diagnostic supervisor.

2. Attach the DWBUA:

   ```
   DS> ATTACH DWBUA HUB DWn node br <RET>
   ```

   DWn is the number of the DWBUA. "n" is a number between 0 and 3. "node" is the VAXBI node ID, expressed as a decimal number (0 to 15).

   NOTE
   Refer to the appropriate system user guide to determine the node ID number.

   "br" is the UNIBUS BR interrupt level, a number between 4 and 7. The recommended value is 7.

3. Run the macrodiagnostic:

   ```
   DS> RUN EVCBB[/SECTION:xxx]<RET>
   ```

Inclusion of the SECTION name ("xxx" in the above command) is optional. If no SECTION name is included, the DEFAULT section is run. The SECTION names and the tests they include are listed in Table 1-2.

Table 1-2 Macrodiagnostic Program Sections

Section	Tests
Default	1 - 30
All	1 - 32
UBE	31, 32

Test 31 and 32 can be run only if a UNIBUS Excerciser (UBE) is attached.

LESSON 4 - TROUBLESHOOTING

This procedure provides the information needed to isolate a DWBUA failure to one of its assemblies:

- T1010 module
- I/O cable
- M7166 paddlecard
- UNIBUS
- M9313 (UET)

Corrective maintenance of the DWBUA consists of replacing faulty subassemblies. This procedure does not attempt to isolate problems caused by the devices attached to the UNIBUS.

The assumption is made in this procedure that system troubleshooting procedures have indicated a problem in the DWBUA subsystem. No system-specific troubleshooting procedures are included here.

The tools and test equipment listed in Table 1-3 are needed to perform the maintenance procedures described in this section.

Table 1-3 Tools and Test Equipment for Maintenance Procedures

Equipment	Manufacturer	Designation	DIGITAL Part Number
Gold Wipes	Texwipe	TX809	49-01603-01
Torque Screwdriver	Utica		29-17381-00
Bus Grant Card			G7273

This section is a step-by-step procedure for isolating faults to the field replaceable unit (FRU). It uses only the tools and test equipment listed in Table 1-3 and the DWBUA module's self-test. By using this procedure, faults in theDWBUA can be isolated when the system is not capable of running diagnostics. (Such a situation can occur if the DWBUA is in the load path for the operating system and diagnostics.)

See Section 2.3.1 and Appendix C of of the DWBUA UNIBUS Adapter Technical Manual, for information on the DWBUA module's self-test.

Follow the steps in the order listed to troubleshoot the DWBUA:

1. Start - Is the DWBUA malfunctioning?

   The DWBUA may be suspect if:

   - You are unable to boot from a UNIBUS device.
   - You are unable to use devices on the UNIBUS.
   - The system console indicates that the node number corresponding to DWBUA device's node ID is malfunctioning.
   - Excessive errors occur when using any UNIBUS device.
   - The system crashes.

2. Power down the system - Wait 30 seconds for the stored power to drain off.

3. Open the cabinet - Open the system cabinet so that you can see yellow lights on the modules.

4. Power up the system - This starts the DWBUA self-test.

5. Check the light on the T1010 - If the yellow light on the T1010 module is lit, then the DWBUA has passed the self-test. The problem is most likely not in the T1010, the UNIBUS cabling, or the terminator card (UET). If the light is off, go to Step 7.

6. Run EVCBB - If the system is operational, run the system-level diagnostic, EVCBB, to further verify that the problem is not in the DWBUA. Refer to the macrodiagnostic printout and documentation to isolate the failing FRU if this diagnostic should fail.

   If one of the symptoms listed in Step 1 exists, but the DWBUA self-test passes, the problem is probably somewhere other than in the DWBUA. Refer to Table 1-4 for suggested areas to troubleshoot.

Table 1-4 Symptoms and Possible Causes

Symptom	Possible Cause
Cannot boot from a UNIBUS device	Boot device
Unable to use devices on the UNIBUS	Bad device or software
Excessive errors when using any devices on the UNIBUS.	Devices on the bus or system wide problems
System crashes	System software

7. The yellow light is off - If the yellow light on the T1010 module is off, the self-test has failed. Look for a fault in one of the items in Figure 1-1. If no fault exists in those items, look for a UNIBUS device that is causing the UNIBUS to malfunction.

8. Determine node number and starting address

   - Halt the system from the console.

   - Type E <address> to examine the contents of the device type register (bb+00) for each node space in succession until you find one with a value of xxxx0102. This is the DWBUA device type. Make a note of the address when you find this value.

If you are working on a system that has more than one VAXBI, bus 0 addresses are as described. See Table 1-5 for the base addresses for bus 1 through bus 3. Refer also to Example 1-1.

Table 1-5 Multiple VAXBI Base Addresses

VAXBI Bus # Number	Base Address
0	2000 0000
1	2200 0000
2	2400 0000
3	2600 0000

NOTE

If nothing is returned from any of the EXAMINES, a system problem exists. See the troubleshooting procedures for your system. THE PROBLEM IS NOT IN THE DWBUA.

```
>>>E 20000000<CR>                          This is the base address
                                           of the first node in I/O
                                           space, node 0

P    20000000          00010001            Address and contents
                                           returned.  Node 0 is not
                                           a DWBUA.

>>>E 20002000                              This is the base address
                                           of node 1

P    20002000          00010101            Node 1 is not a DWBUA.
  .
  .
  .

>>>E 2000C000                              This is the base address
                                           of node 6.

P    2000C000          00010102            Node 6 is a DWBUA.
```

Example 1-1   Determining Node Number and Starting

9. Find GPR0 address - bb + F0 = GPR0 address. Add F0 (hex) to the address you found in the previous step. See Example 1-2.

```
Node 0 GPR  = 20000000 + F0 = 200000F0 * most likely address
Node 1 GPR0 = 20002000 + F0 = 200020F0
Node 2 GPR0 = 20004000 + F0 = 200040F0
Node 3 GPR0 = 20006000 + F0 = 200060F0
Node 4 GPR0 = 20008000 + F0 = 200080F0
```

Example 1-2   Finding the Address of GPR0

10. Examine GPR0 - Use the console to examine GPR0 at the address calculated in the last step. GPR0 bits <31:16> contain the test number that failed in the self-test. Refer to Appendix C of the DWBUA UNIBUS Adapter Technical Manual for a description of the self-test microdiagnostic tests.

NOTE

Failure of the DWBUA self-test can keep you from accessing the DWBUA internal registers. To access these registers to explore the cause of the self-test failure, make sure that bit <08> (UCSREN) is set in the BCICSR (bb+28).

11. Isolate FRU - Use the flowchart in Figure 1-9 to isolate the FRU at fault.

NOTE

The T1010 module is suspect throughout this troubleshooting procedure, since it is the engine running the test.

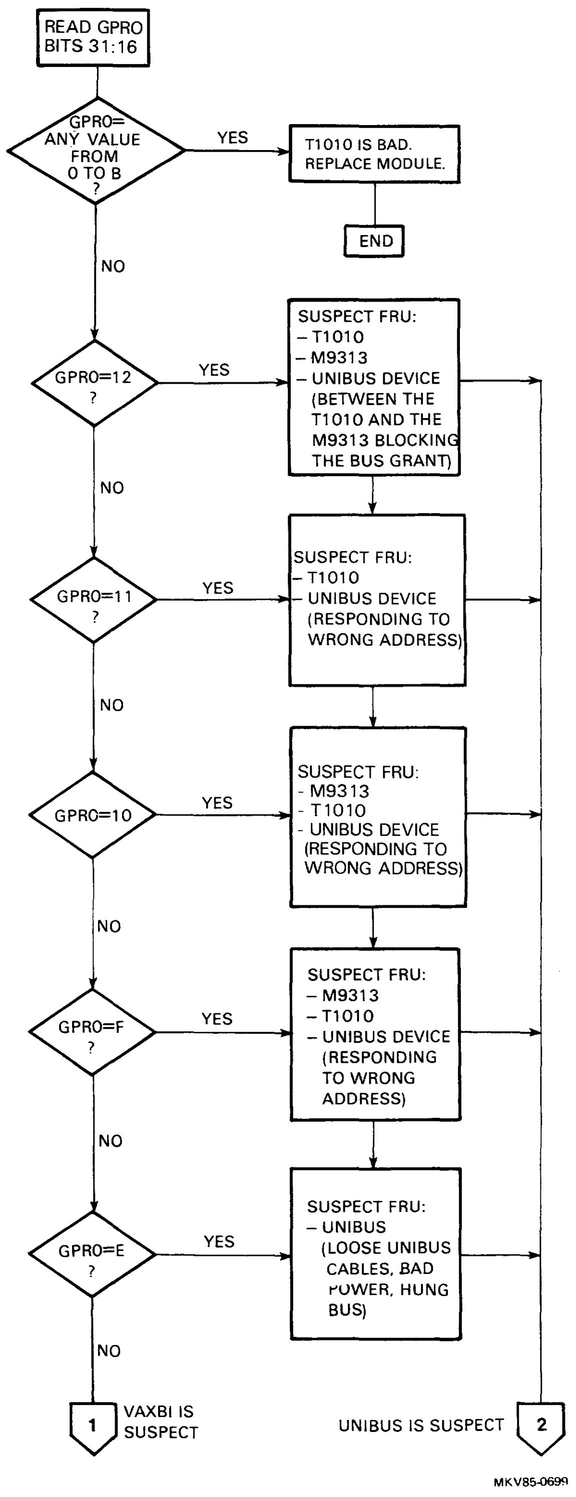

READ GPRO BITS 31:16
GPR0= ANY VALUE FROM 0 TO B ?
YES
T1010 IS BAD. REPLACE MODULE.
END
NO
GPR0=12 ?
YES
SUSPECT FRU: – T1010 – M9313 – UNIBUS DEVICE (BETWEEN THE T1010 AND THE M9313 BLOCKING THE BUS GRANT)
NO
GPR0=11 ?
YES
SUSPECT FRU: – T1010 – UNIBUS DEVICE (RESPONDING TO WRONG ADDRESS)
NO
GPR0=10
YES
SUSPECT FRU: - M9313 - T1010 - UNIBUS DEVICE (RESPONDING TO WRONG ADDRESS)
NO
GPR0=F ?
YES
SUSPECT FRU: – M9313 – T1010 – UNIBUS DEVICE (RESPONDING TO WRONG ADDRESS)
NO
GPR0=E ?
YES
SUSPECT FRU: – UNIBUS (LOOSE UNIBUS CABLES, BAD POWER, HUNG BUS)
NO
1 VAXBI IS SUSPECT
UNIBUS IS SUSPECT 2
MKV85-0699

Figure 1-9 Troubleshooting Flow (Sheet 1 of 3)

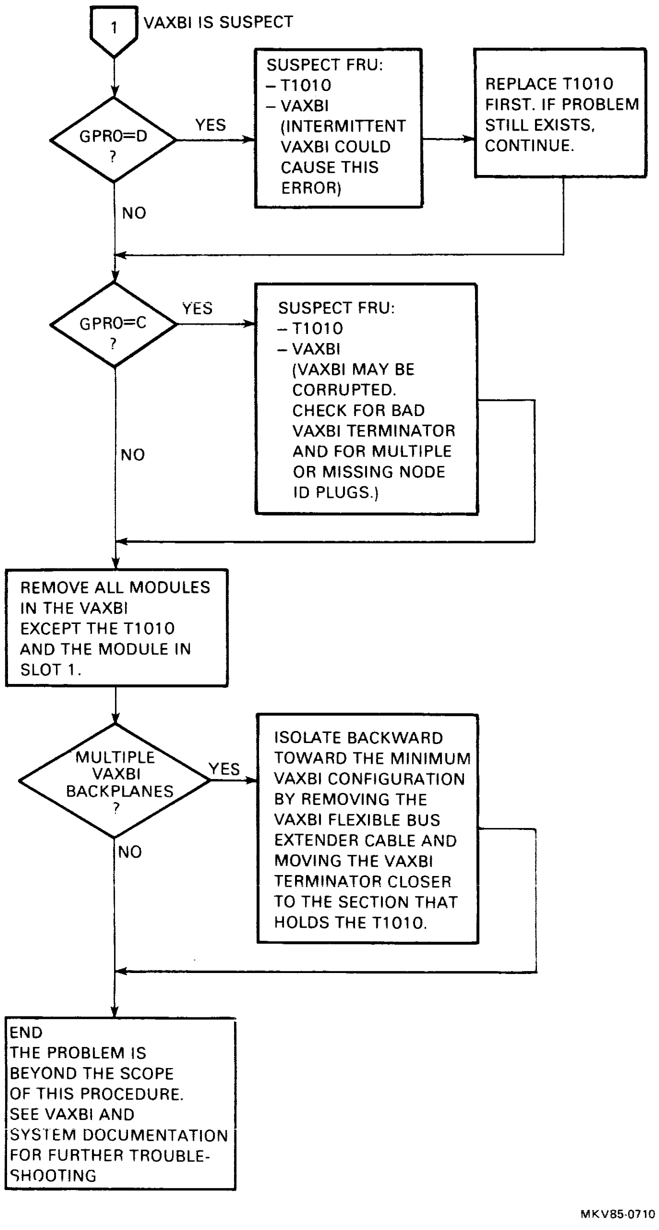

1
VAXBI IS SUSPECT
GPR0=D ?
YES
NO
SUSPECT FRU: - T1010 - VAXBI (INTERMITTENT VAXBI COULD CAUSE THIS ERROR)
REPLACE T1010 FIRST. IF PROBLEM STILL EXISTS, CONTINUE.
GPR0=C ?
YES
NO
SUSPECT FRU: - T1010 - VAXBI (VAXBI MAY BE CORRUPTED. CHECK FOR BAD VAXBI TERMINATOR AND FOR MULTIPLE OR MISSING NODE ID PLUGS.)
REMOVE ALL MODULES IN THE VAXBI EXCEPT THE T1010 AND THE MODULE IN SLOT 1.
MULTIPLE VAXBI BACKPLANES ?
YES
NO
ISOLATE BACKWARD TOWARD THE MINIMUM VAXBI CONFIGURATION BY REMOVING THE VAXBI FLEXIBLE BUS EXTENDER CABLE AND MOVING THE VAXBI TERMINATOR CLOSER TO THE SECTION THAT HOLDS THE T1010.
END THE PROBLEM IS BEYOND THE SCOPE OF THIS PROCEDURE. SEE VAXBI AND SYSTEM DOCUMENTATION FOR FURTHER TROUBLE-SHOOTING
MKV85-0710

Figure 1-9 DWBUA Troubleshooting Flowchart (Sheet 2 of 3)

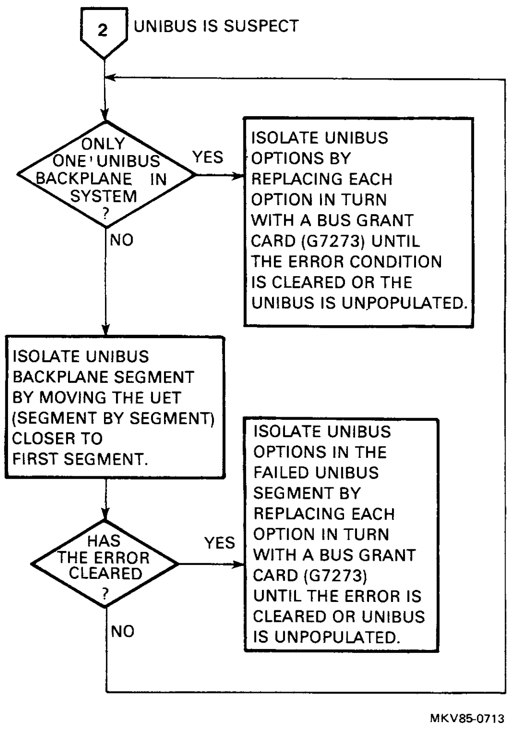

2
UNIBUS IS SUSPECT
ONLY ONE UNIBUS BACKPLANE IN SYSTEM ?
YES
ISOLATE UNIBUS OPTIONS BY REPLACING EACH OPTION IN TURN WITH A BUS GRANT CARD (G7273) UNTIL THE ERROR CONDITION IS CLEARED OR THE UNIBUS IS UNPOPULATED.
NO
ISOLATE UNIBUS BACKPLANE SEGMENT BY MOVING THE UET (SEGMENT BY SEGMENT) CLOSER TO FIRST SEGMENT.
HAS THE ERROR CLEARED ?
YES
ISOLATE UNIBUS OPTIONS IN THE FAILED UNIBUS SEGMENT BY REPLACING EACH OPTION IN TURN WITH A BUS GRANT CARD (G7273) UNTIL THE ERROR IS CLEARED OR UNIBUS IS UNPOPULATED.
NO
MKV85-0713

Figure 1-9 DWBUA Troubleshooting Flowchart (Sheet 3 of 3)

LESSON 5 - RELATED DOCUMENTATION

This is a list of available documentation for the DWBUA adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- DWBUA UNIBUS Adapter Technical Manual, EK-DWBUA-TM
- PDP-11 UNIBUS Processor Handbook, EB-26077-41

DMB32 ADAPTER

LESSON 1 - INTRODUCTION

The DMB32 is an intelligent synchronous/asynchronous multiplexer which provides eight full-duplex asynchronous serial data channels, one synchronous data channel, and one line printer interface on VAXBI systems.

The main features of the DMB32 are:

- Single VAXBI module (T1012) and distribution panel (H3033).
- Eight full-duplex asynchronous (async) data channels.
- One synchronous (sync) data channel.
- One line-printer port. The DMB32 printer port supports the LP32 generic printer specification. This includes the LN01, LN01-B, LN01-S, LP25, LP26, LP27, LXY12, and LXY22 printers.
- Direct Memory Access (DMA) or single-character programmed transfers to and from host memory.
- On-board virtual-to-physical address translation allows the DMB32 to handle data buffers in their virtual address format.
- Large 512-entry first-in first-out (FIFO) buffer on the async channels, for received characters, dataset status changes, and diagnostic information.
- Async ports are electrically and mechanically compliant with RS-232-C, and compatible with V.28/V.24 (the T1012 is also compatible with RS-423-A, V.10, and X.26, but the H3033 is not, due to pin limitations on the 25-pin D-type connectors).
- The sync port is electrically and mechanically compliant with the RS-232-C, RS-422-A/RS-449, V.11, X.27, and V.35, and compatible with RS-423-A/RS-449, V.28/V.24, V.10, and X.26.
- IBM bisync, SDLC/HDLC, and DDCMP protocols are supported on the sync port. NRZI support is also provided.

- General byte (GEN BYTE) protocol provides basic framing and data transfer facilities to implement other byte-oriented protocols on the sync channel.
- Full-duplex point-to-point or auto-answer dial-up operation.
- Programmable split-speed operation.
- Total module throughput is 21000 characters per second.
- Automatic flow-control of transmitted and received data on the async channels.
- Self-test diagnostics.
- Programmable loopback modes and test facilities.
- There are no switches on the module or the distribution panel. Each external cable for the sync channel is electrically coded to select a specific CCITT/EIA standard. Other line characteristics are set under program control.

Enough modem control is provided on all synchronous and asynchronous channels to allow auto-answer dial-up operation over the public switched telephone network (PSTN). The DMB32 can also be used for point-to-point operation over private lines. Modem control is implemented by software in the host.

An integral microprocessor releases the host from many of the data-handling tasks.

The DMB32 has an on-board self-test diagnostic that is executed independently of the host. Online and stand-alone diagnostics are also available.

Functional Description

The DMB32 adapter, shown in Figure 2-1, is a communications adapter of the VAXBI family. It has eight asynchronous channels, one synchronous channel, and a printer channel. The function of the DMB32 is to transfer data between a VAXBI node and the asynchronous, synchronous, and printer ports.

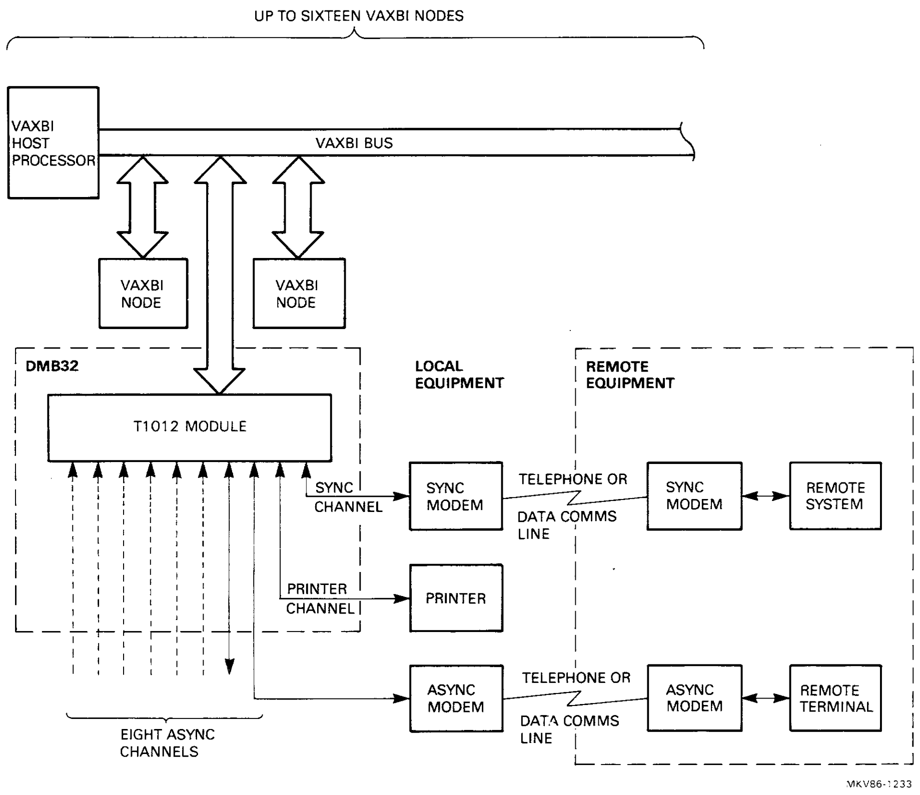


Figure 2-1 Typical DMB32 Configuration

Figure 2-2 is a functional block diagram that shows the data routes through the option. Data is transferred between a VAXBI node on the host and the communications lines via registers and DMA files in the common address store RAM (CASRAM). Also in the CASRAM are control and status registers (CSRs) which configure and program the option. The information that is written to and read from the CSRs is used to support and control the communications functions.

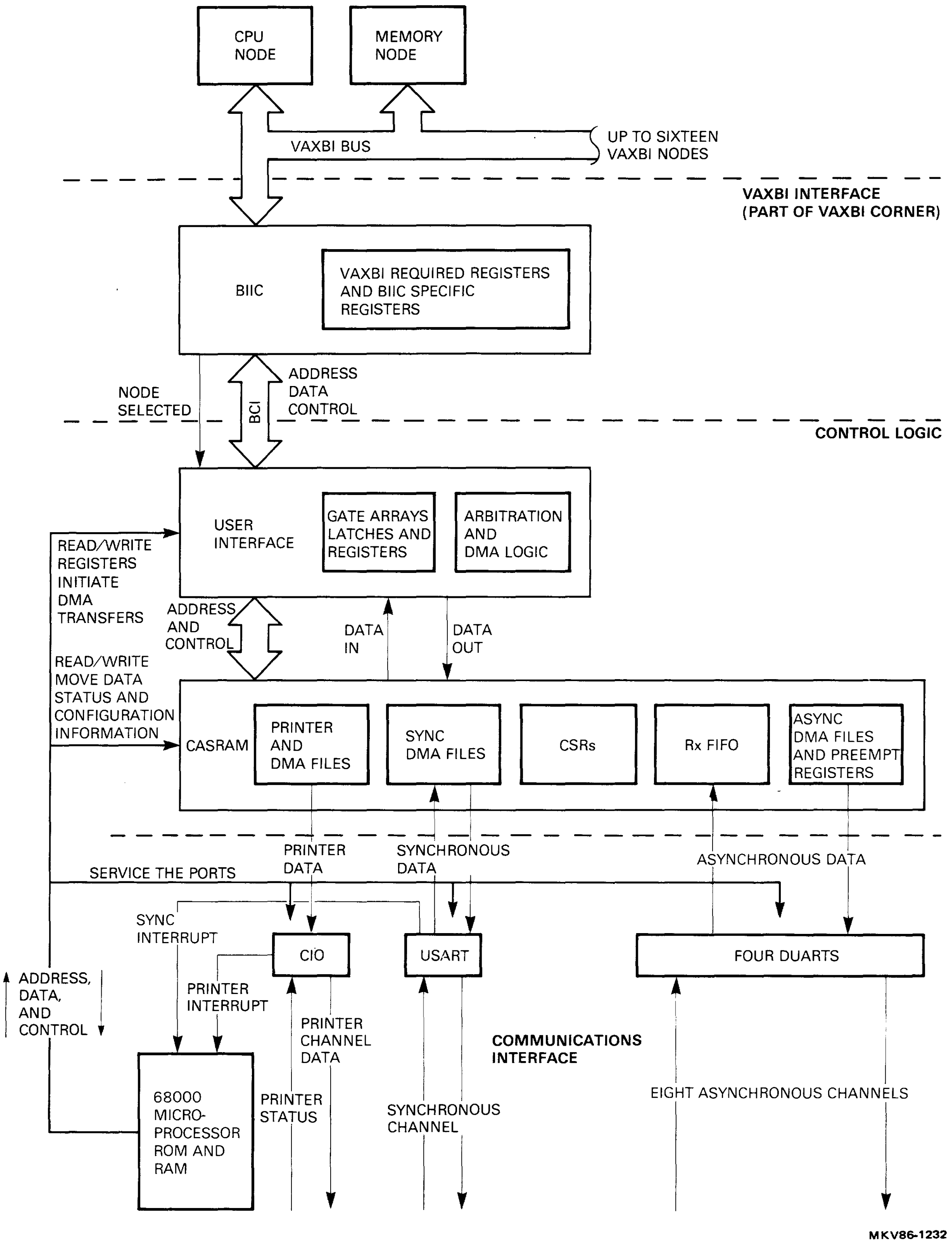

CPU NODE
MEMORY NODE
VAXBI BUS
UP TO SIXTEEN VAXBI NODES
VAXBI INTERFACE (PART OF VAXBI CORNER)
BIIC
VAXBI REQUIRED REGISTERS AND BIIC SPECIFIC REGISTERS
NODE SELECTED
BCI
ADDRESS DATA CONTROL
CONTROL LOGIC
USER INTERFACE
GATE ARRAYS LATCHES AND REGISTERS
ARBITRATION AND DMA LOGIC
READ/WRITE REGISTERS INITIATE DMA TRANSFERS
ADDRESS AND CONTROL
DATA IN
DATA OUT
READ/WRITE MOVE DATA STATUS AND CONFIGURATION INFORMATION
CASRAM
PRINTER AND DMA FILES
SYNC DMA FILES
CSRs
Rx FIFO
ASYNC DMA FILES AND PREEMPT REGISTERS
SERVICE THE PORTS
PRINTER DATA
SYNCHRONOUS DATA
ASYNCHRONOUS DATA
SYNC INTERRUPT
CIO
USART
FOUR DUARTS
ADDRESS, DATA, AND CONTROL
PRINTER INTERRUPT
PRINTER CHANNEL DATA
COMMUNICATIONS INTERFACE
68000 MICRO-PROCESSOR ROM AND RAM
PRINTER STATUS
SYNCHRONOUS CHANNEL
EIGHT ASYNCHRONOUS CHANNELS
MKV86-1232

Figure 2-2 DMB32 Functional Block Diagram

Data can be transferred between the VAXBI and CASRAM by programmed read or write commands, or by the DMB32 adapter's DMA logic. Data is transferred between the communications interface and the CASRAM under the control of a 68000 microprocessor.

The maximum data transfer rate on the VAXBI is 13.3 Mbytes per second. Such a high rate makes the design of the VAXBI interface critical, so a standard VAXBI interface (VAXBI corner) has been designed for all VAXBI options. The VAXBI corner is based on a specially designed VAXBI integrated circuit (BIIC), which handles all the VAXBI commands and protocols.

The intelligence of the DMB32 is supplied by the 68000 microprocessor driven by ROM-based firmware. The microprocessor controls and configures the communications interface, and manages both DMA and communications functions. The firmware also includes self-test routines which run automatically at power-up or reset.

Data Transfer - The function of the DMB32 is to transfer data between the VAXBI host and the data lines connected to the communications interface. There are several methods of transfer:

- DMA

  Sync RX and TX data is transferred between host memory buffers and CASRAM files by DMA. The microprocessor keeps track of the buffers, and initiates further DMA transfers as needed. Transfers between CASRAM and the sync port are initiated by an interrupt to the microprocessor.

  Printer data is also transferred across the VAXBI by DMA. Transfers of data from CASRAM to the printer port are done by the microprocessor. Printer status is polled by the microprocessor, which also makes the status information available in the CSRs.

  Async TX data can be transferred by DMA or by programmed transfer. DMA transfers to CASRAM are controlled in the same way as sync data transfers. The microprocessor monitors the async ports to check if they are ready to accept data.

- RX FIFO

  The microprocessor transfers async RX data to a 512-character RX FIFO in CASRAM. The host reads the data, together with error status information, from a single location (RBUF). The RX FIFO is also used to send diagnostic reports to the host.

- Programmed (Preempt) Transfer

  Async TX data may be transferred to CASRAM by DMA, or it can be written to a single-character 'preempt' register (one exists in each async channel). From there the transfer is completed by the microprocessor. A character written to the preempt register of a channel that is transmitting a DMA buffer will be transferred to the async port before any remaining DMA data.

Interrupts - The DMB32 can be programmed to interrupt a VAXBI host under the following conditions:

- When a channel is ready for another preempt character.

- When transfer of a DMA buffer from system memory to an I/O channel has been:

  - Aborted

  - Terminated due to an error

  - Completed successfully

- When a received character has been placed in a previously empty RX FIFO (the DMB32 can be programmed to delay this interrupt so that several characters can be placed in the FIFO before the interrupt is raised).

- When modem status information has been placed in the RX FIFO. This action overrides any programmed interrupt delay.

Physical Description

The DMB32 Option - The DMB32 option is made up of the following major items:

- A single VAXBI module (T1012)
- A distribution panel (H3033)
- Six ribbon cables (17-00740-xx, where xx defines the length)
- A 50-pin unbalanced loopback connector (H3195)
- A 50-pin balanced loopback connector (H3196)
- An async line loopback connector (H3197)

The H3033 Distribution Panel - Figure 2-3 shows the H3033 distribution panel. It identifies the sync, async and printer connectors, and the connectors to which the ribbon cable headers are connected.

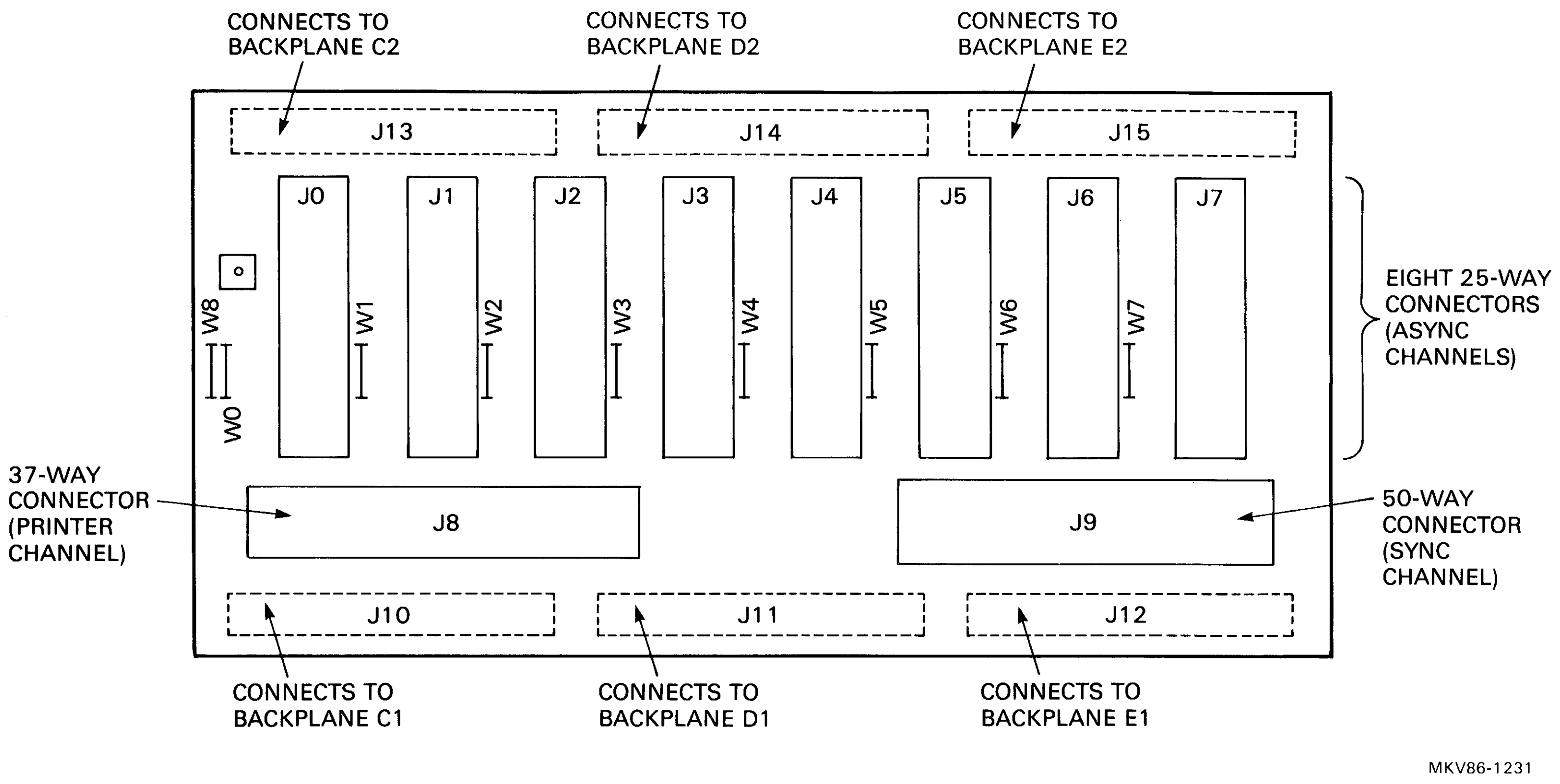


Figure 2-3 H3033 Distribution Panel

LESSON 2 - DMB32 REGISTERS

To help in understanding the registers used by the DMB32 adapter, Table 2-1 shows the most important registers and how they are grouped. For a bit map and a full description of these registers, see the Operation and Programming chapter of the DMB32 User's Guide, EK-DMB32-UG.

Table 2-1 DMB32 Registers

General configuration registers for the device		Sync Port Registers	
		Buffer 1 transmit Control	(TLNCTRL1)
Maintenance Register	(MAINT)	Buffer 1 receive control	(RLNCTRL1)
Async control and status	(ACSR)	Buffer 2 transmit Control	(TLNCTRL2)
Device Configuration	(CONFIG)	Buffer 2 receive Control	(RLNCTRL2)
		Sync buffer control bits	(BUFCTRL)
Address translation registers		Sync line parameters 1	(LPR1)
		Sync line parameters 2	(LPR2)
System page table register	(SPTE)	Sync line parameters 3	(LPR3)
System page table size	(SPTS)	Sync line completion FIFO	(SBUF)
Global page table register	(GPTE)		
Global page table size	(GPTS)	Async Registers	
Printer Registers		Line parameter register	(LPR)
		Line control register	(LNCTRL)
Printer control and status	(PCSR)	Line status register	(LSTAT)
Printer control	(PCTRL)	Flow control characters	(FLOWC)
		Transmit completion FIFO	(TBUF)
		Receive Buffer	(RBUF)

LESSON 3 - INSTALLATION

The procedures for installing and testing the DMB32 option are described in this section.

Installation Task List

Installation of the DMB32 consists of the following tasks:

- Unpacking and inspection
- Installation checks
- VAXBI configuration checks
- Installing the T1012 module
- Installing the transition header assembly on the VAXBI backplane
- Installing the ribbon cables into the transition header
- Installing the H3033 distribution panel and ribbon cables
- Installing the external adapter cables
- Installation testing

DMB32 Installation Kits

The DMB32 option is supplied in a kit that contains the parts needed to install the option, but you may also need the torque wrench (29-17381-00) from the VAXBI installation kit. Check the contents of your DMB32 installation kit against the list given in this section. Examine all parts for physical damage. Report damaged or missing items to the shipper immediately, and inform the local DIGITAL office.

CAUTION

The T1012 module is supplied in protective anti-static packaging. Do not remove the module from its packaging unless you are wearing an anti-static wrist strap.

There are different versions of the installation kit, tailored to the different system configurations. These are:

- DMB32-LJ for the VAX 8800 system
- DMB32-LM for the VAX 8200 and VAX-8300 systems
- DMB32-LN for the VAX 8800 system with an expander cabinet

The only difference between the three versions is the length of the six ribbon cables that go between the VAXBI backplane and the H3033 distribution panel.

Installation Checks

When the DMB32 hardware has been installed, the DMB32 synchronous device driver must be installed before the sync port can be used. The DMB32 synchronous driver is a layered product that must be ordered separately from DIGITAL, it is not part of the DMB32 hardware kit.

The procedure for installing the DMB32 Sync Driver is fully described in the DMB32 Sync Driver Installation Guide supplied in the software installation kit.

NOTE

VAX/VMS requires that an adapter cable is connected to the sync port before it will configure the sync device (SIx0).

Configuration Rules

There are two VAXBI configuration rules which apply to the DMB32:

1. The DMB32 must have a unique node ID (set by the node ID plug). The DWBUA UNIBUS adapter is always assigned a node ID of zero; therefore, if the host system has a DWBUA installed, the DMB32 cannot be node zero.

2. The DMB32 must not be installed in slot 1 of the first backplane. This is also called slot K1J1 and is reserved for the node which supplies the BI TIME and BI PHASE clock signals.

   There may also be system-dependent configuration restrictions, such as those imposed by backplane capacity, and physical mounting limitations within the cabinet. These limitations will be described in the host system documentation.

Mechanical Installation

Figure 2-4 shows how the parts of the DMB32 option fit together. This figure should be used together with the installation instructions given in this section.

T1012 Module Installation

1. Insert the T1012 into a slot in the VAXBI cardcage.

   The module may be installed in any empty slot except slot K1J1 (that is, the first slot in the first backplane). The module is keyed to prevent incorrect installation.

2. Fit an appropriate node ID plug to the reverse side of the backplane at the position corresponding with the slot where the T1012 module is inserted.

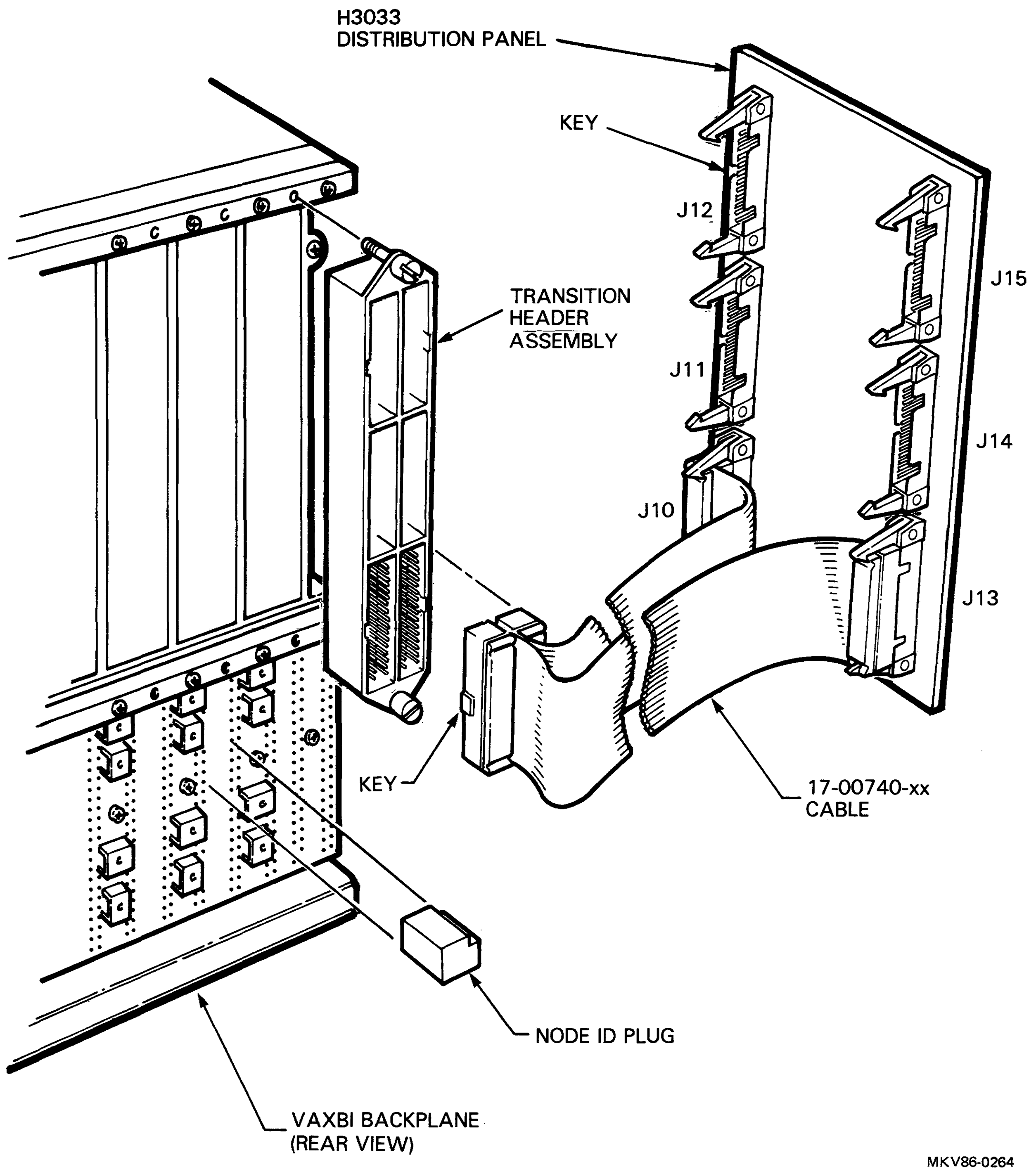


Figure 2-4 DMB32 Installation

Transition Header Assembly Installation

1. Check the VAXBI backplane to see if a transition header assembly (12-22246-01) is already installed on the backplane at the correct position. If it is, you will not need to continue with the procedure described in this section. Continue with the installation of the ribbon cables.

NOTE
You will need the torque wrench (29-17381-00) from the VAXBI installation kit to complete the installation of the transition header assembly.

2. Fit the transition header assembly to the reverse of the VAXBI backplane at the position corresponding with the slot where the T1012 module is inserted.

3. Tighten the screws gradually and alternately to prevent the header assembly from skewing. Use the torque wrench to tighten the screws to a torque of 5 inch-pounds (plus or minus 1 inch-pound).

Ribbon Cable Installation

1. Connect the six ribbon cables (17-00740-xx) into the transition header assembly. Make sure that the ribbed side of the cables faces towards slot 1 (see Figure 2-4).

2. Route the six ribbon cables to the I/O panel where the H3033 distribution panel is to be installed. Refer to the host system documentation for routing details.

Distribution Panel Installation

1. Connect the six ribbon cables to the distribution panel.

   Table 2-2 and Figures 2-3 and 2-4 will help you identify which connector on the transition header connects to which socket on the H3033 distribution panel. The cable header plugs are keyed to prevent incorrect installation.

   An electrical key signal passes through all six ribbon cables. If any cable is incorrectly installed, the path of the key signal will be broken and the DMB32 will fail its self-test.

Table 2-2 Ribbon Cable Connections

Transition Header Connector	H3033 Distribution Panel Socket
C-1	J10
C-2	J13
D-1	J11
D-2	J14
E-1	J12
E-2	J15

2. Install the H3033 distribution panel into the appropriate position on the system cabinet.

3. Fit the adapter cable, line printer cable, and async cables (as appropriate) to the distribution panel.

NOTE

VAX/VMS requires that an adapter cable is connected to the sync port before it will configure the sync device (SIx0). It also requires that a line printer cable is connected to the printer port before it will configure the printer device (LIx0).

Acceptance Testing

When you have installed the DMB32, run the following tests:

1. Self-test (runs on power-up)
2. EVDAK, EVDAJ, EVDAL, and EVAAA diagnostics
3. UETP

LESSON 4 - TROUBLESHOOTING

The sequence used to test the DMB32 is described in the following sections.

Power-Up and Self-Test

1. Power-up the host system.

2. Check that the yellow LED on the T1012 module lights after about four seconds, and stays lit. This indicates a successful self-test.

3. On the VAX 8200 and VAX 8300, check the printout from the system console. Immediately after its own CPU self-test, the host system prints a sequence of numbers and periods, for example:

   ```
   0 . 2 . . . . 7 . . . B C D . .
   ```

   This example indicates that the host has detected modules with node IDs of 0, 2, 7, 11, 12, and 13, and that they have passed their self-test diagnostics. If a module fails its self-test, it is not configured by the host and the host puts a minus sign in front of the node ID, for example:

   ```
   0 . 2 . . . . 7 . . .-B C D . .
   ```

   This example indicates that the option with a node ID of B (hex) has not passed its self-test.

   The host should recognize that a module is present at the node ID with which you have installed the T1012 module.

4. If the DMB32 self-test fails, type the following at the console prompt:

   ```
   >>> E/P/W 200xx000 <RET> (where xx is twice the node ID)
   ```

   The console should respond with 0109 (hex), the valid DMB32 device type.

   If the console reports an error, check that you have the entered the correct address for the node ID.

   If the console responds with FFFF (hex), try reseating the T1012 module.

   If the console gave the correct device type (0109 (hex)), type the following at the console prompt:

   ```
   >>>> E/P/L 200xx0F0 (where xx is twice the node ID)
   ```

   The console will print out the contents of the GPR0 register in the DMB32 which contains bits that indicate the reason for the self-test failing.

Diagnostics

There are five types of diagnostic provided for the DMB32:

1. On-board self-test diagnostic
2. Standalone diagnostic: EVDAK (level 3)
3. Online diagnostics: EVDAJ, EVDAL, EVAAA (level 2R)
4. User Environment Test Program (UETP)
5. Data Communications Link Test (DCLT)

After a successful self-test, use the following diagnostic sequence to make sure that the DMB32 is fully functional. Full details of these diagnostics and how to run them are given in Chapter 4, Maintenance, of the DMB32 User's Guide, EK-DMB32-UG. Run each diagnostic for at least one error-free pass.

1. Load the VAX Diagnostic Supervisor (VDS).
2. Load and run EVDAK, the DMB32 level 3 diagnostic.
3. Boot the VAX/VMS system.
4. Load and run EVDAJ, the DMB32 level 2R async diagnostic.
5. Load and run EVDAL, the DMB32 level 2R sync diagnostic.
6. Load and run EVAAA, the DMB32 level 2R printer diagnostic.
7. Run UETP.

On-Board Self-Test Diagnostic

The self-test diagnostic occupies less than 8K words of the DMB32 module's firmware ROM. It runs automatically:

1. At power-up
2. After a hardware or software reset
3. When the START SELF-TEST bit is set

The self-test performs extensive functional tests of the DMB32 hardware, and then partially initializes the DMB32 before passing control to the functional firmware.

During the tests, the two yellow GO/NOGO LEDs on the T1012 module are turned OFF. They are turned ON again when the option has passed the self-test. If the option does not pass, the LEDs will stay OFF. The GO/NOGO LEDs are driven by the DIAG.FAIL bit (MAINT<11>).

The self-test also reports error and status information to the host through the RX FIFO and some other registers. This information is used by system-based diagnostics such as EVDAK. Diagnostic messages are covered later in this chapter.

Although the self-test is comprehensive, a successful test sequence does not guarantee that all sections of the module are good.

There are three conditions that will cause the self-test sequence to be modified.

- If the SKIP.SELF.TEST bit (MAINT<3>) is set, the self-test will simply initialize the module and pass control immediately to the functional firmware.
- If the FORCE.FAIL bit (MAINT<0>) is set, the self-test will act as if the self-test has failed. It will write an error code into the RX FIFO indicating that it has been forced to fail.

- If the bus signal BI STF L (Self-Test Fast) is asserted a fast self-test will be done. Control will be passed to the functional firmware within 250 ms.

The scope and duration of these variations of the self-test are summarized in Table 2-3.

Table 2-3 Scope and Duration of the Self-Test

BI STF L Asserted?	Skip Self-Test?	Force Fail?	Duration	% of Module Tested
X	YES	X	(very short)	--
X	X	YES	(very short)	--
NO	NO	NO	<10 s	50 to 90
YES	NO	NO	<250 ms	15 to 20

X = don't care

EVDAK Stand-Alone Diagnostic

EVDAK is a suite of functional verification tests. The tests run stand-alone under the VAX Diagnostic Supervisor (VDS) V9.0 or later. The supervisor is documented in the VAX Diagnostic System User's Guide, EK-VX11D-UG.

EVDAK is intended for:

- Users - to identify failing DMB32s and as a confidence check
- Field Service - to isolate faults and to test installations
- Manufacturing - as a final test
- Final Assembly - as part of a system test

EVDAK has four modes of operation:

1.	Internal loopback	In this mode a physical loopback connector is not used. The selected channels are looped back internally. Therefore, the line drivers and receivers will not be tested.
2.	Manufacturing	This mode is used by DIGITAL to test T1012 modules during manufacture, and is for internal use only.
3.	External loopback	In this mode the appropriate loopback connector must be installed on any selected async or sync channel. The connection is made on the H3033 distribution panel. Line drivers and receivers of the looped-back channels are tested.
4.	Modem loopback	In this mode only the data lines are tested. Tests can be run with an external loopback connector or a modem on the channel to be tested. If a modem is used it must be looped back manually.

The printer channel can be tested in all available modes. However, a printer is required, and one of the tests requires operator intervention.

An example of how to run EVDAK is shown in Example 2-1. In the example, TRACE is set to cause all tests to be reported.

```
DIAGNOSTIC SUPERVISOR.  ZZ-EBSAA- 9.0-325   2-OCT-1985 15:36:22

DS> LOAD EVDAK              ; load the program
DS> SET TRACE               ; set trace
DS> ATTACH DMB32 HUB TXA 1  ; HUB = linked to VAXBI
                            ; TXA = Device Name as appropriate
                            ;   1 = VAXBI Node ID (hexadecimal)
DS> SELECT TXA              ; select the device for test
DS> START                   ; start the diagnostic
```

Example 4-1 Starting EVDAK

NOTE

If the machine does not have native VAXBI but uses an adapter, this must be attached before the DMB32. For example, on the VAX-8800 the attach sequence might be:

```
DS> ATTACH NBIA HUB NBIA0 0

DS> ATTACH NBIB NBIA NBIB0 0 2

DS ATTACH DMB32 NBIB0 TXA 4
```

EVDAJ Online Diagnostic (Async)

This level 2R diagnostic runs online under VMS V4.4 or later. The operator interface is via the VAX Diagnostic Supervisor (VDS) V9.0 or later. EVDAJ provides a confidence check of the DMB32 async channels. It consists of five tests:

1. Internal data loopback test on selected channels in sequence.
2. Internal DMA data loopback test on selected channels in sequence.
3. Internal DMA data loopback test on selected channels at the same time.
4. External loopback test (using the H3197 loopback) of modem control signals.
5. External data loopback (using the H3197 loopback, or via a modem). If a modem is used, it must be set up manually.

Before running EVDAJ, the user should be aware of the following points:

1. The tests run online; therefore, it is essential to define the channels to be tested. The test will not run if a channel that is allocated to another process is selected for testing. Channel allocations can be checked by using the VAX/VMS command SHOW DEVICE/FULL.
2. If test 4 is being run, an H3197 loopback connector must be fitted to the selected channel. By setting EVENT flag 20, the operator will cause the program to halt before each channel is tested. This allows the H3197 to be moved to the next channel to be tested. Another event flag (21) causes the program to halt before each complete DMB32 has been tested.

3. If test 5 is to be run, a modem or an H3197 must be fitted to the selected channel. EVENT flags 20 and 21 also function in this test.

4. If tests 4 or 5 are run without some form of loopback installed, error messages will be generated.

See the <u>DMB32 User Guide</u> for an example of how to run EVDAJ.

EVDAL Online Diagnostic (Sync)

This level 2R diagnostic runs online under VMS V4.4 or later. The operator interface is via the VAX Diagnostic Supervisor (VDS) V9.0 or later. EVDAL provides a confidence check of the DMB32 sync channels.

It consists of four tests:

1. DDCMP protocol test
2. BISYNC protocol test
3. HDLC protocol test
4. Modem signal test

In tests 1, 2, and 3:

- Data is transferred in groups of 16, 64, 512, and 1000 bytes, using five different data patterns.
- If the 50-way balanced loopback connector H3196 is fitted and the /EXTERNAL section is being run, the test will automatically run twice. Once with V.35 clear and once with V.35 set.

In test 3:

- SDLC is not specifically tested as it is a subset of HDLC.

In test 4:

- The diagnostic detects which loopback connector/adapter cable is connected. The appropriate modem signals are tested.

See the <u>DMB32 User Guide</u> for an example of how to run EVDAL.

EVAAA Online Diagnostic (Printer)

This level 2R diagnostic runs online under VMS V4.4 or later. The operator interface is via the VAX Diagnostic Supervisor (VDS) V9.0 or later. EVAAA provides a functional test of printers connected to any VAX-11 or VAXBI system (it is not specific to the DMB32).

EVAAA consists of 33 tests for a range of printers. When the diagnostic runs, only those tests relevant to the type of printer connected will be used. A successful test of an attached printer implies that the DMB32 printer port is good.

All EVAAA tests require intervention by the operator or visual inspection of the printout. For details of the tests, refer to the diagnostic listing, EVAAA.LIS.

See the DMB32 User Guide for an example of how to run EVAAA.

User Environment Test Program (UETP)

After the DMB32 has successfully passed the relevant diagnostic tests, the UETP system exerciser should be run to check for interaction problems between the DMB32 and other options.

Data Communications Link Test (DCLT)

If the on-board self-tests detects no faults, DCLT can be used to test the external line.

DCLT is a diagnostic program that tests communications links. It is used to isolate errors to the local communications device, the modem, the physical line, or the remote communications device. Each DCLT diagnostic is written for the specific option on which it runs, but a DCLT diagnostic can communicate with any other DCLT diagnostic via a communications link. For this test, the system at each end of the link must have DCLT loaded and running.

Field-Replaceable Units (FRUs)

There is no preventive maintenance for the DMB32. Corrective maintenance is based on identifying and replacing defective FRUs. The FRUs for the DMB32 are:

- T1012 module
- 17-00740-xx ribbon cable
- H3033 distribution panel
- 12-22246-01 transition header assembly

Figure 2-5 is a flow diagram for troubleshooting the DMB32 module.

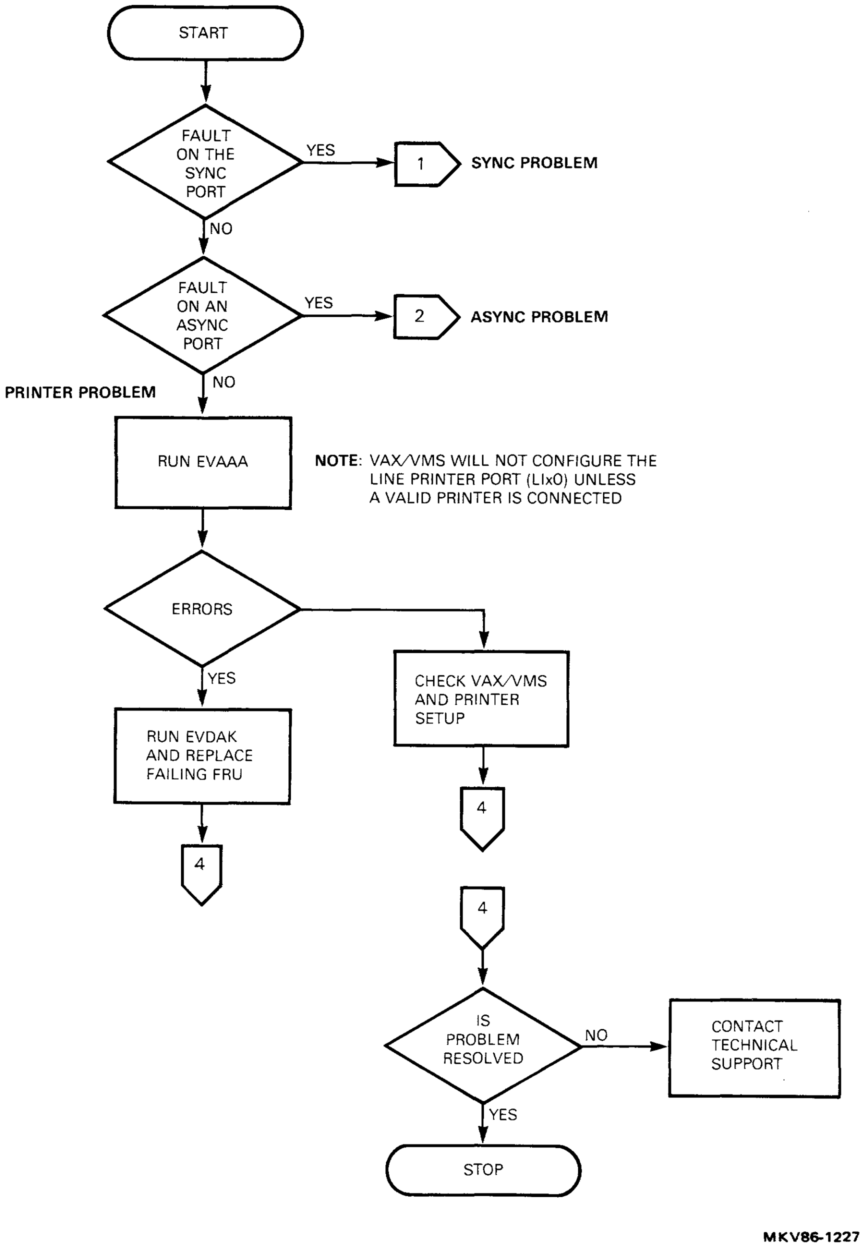

START
FAULT ON THE SYNC PORT
YES
1
SYNC PROBLEM
NO
FAULT ON AN ASYNC PORT
YES
2
ASYNC PROBLEM
NO
PRINTER PROBLEM
RUN EVAAA
NOTE: VAX/VMS WILL NOT CONFIGURE THE LINE PRINTER PORT (LIx0) UNLESS A VALID PRINTER IS CONNECTED
ERRORS
YES
RUN EVDAK AND REPLACE FAILING FRU
4
CHECK VAX/VMS AND PRINTER SETUP
4
4
IS PROBLEM RESOLVED
NO
CONTACT TECHNICAL SUPPORT
YES
STOP
MKV86-1227

Figure 2-5 DMB32 Troubleshooting Flowchart (Sheet 1 of 4)

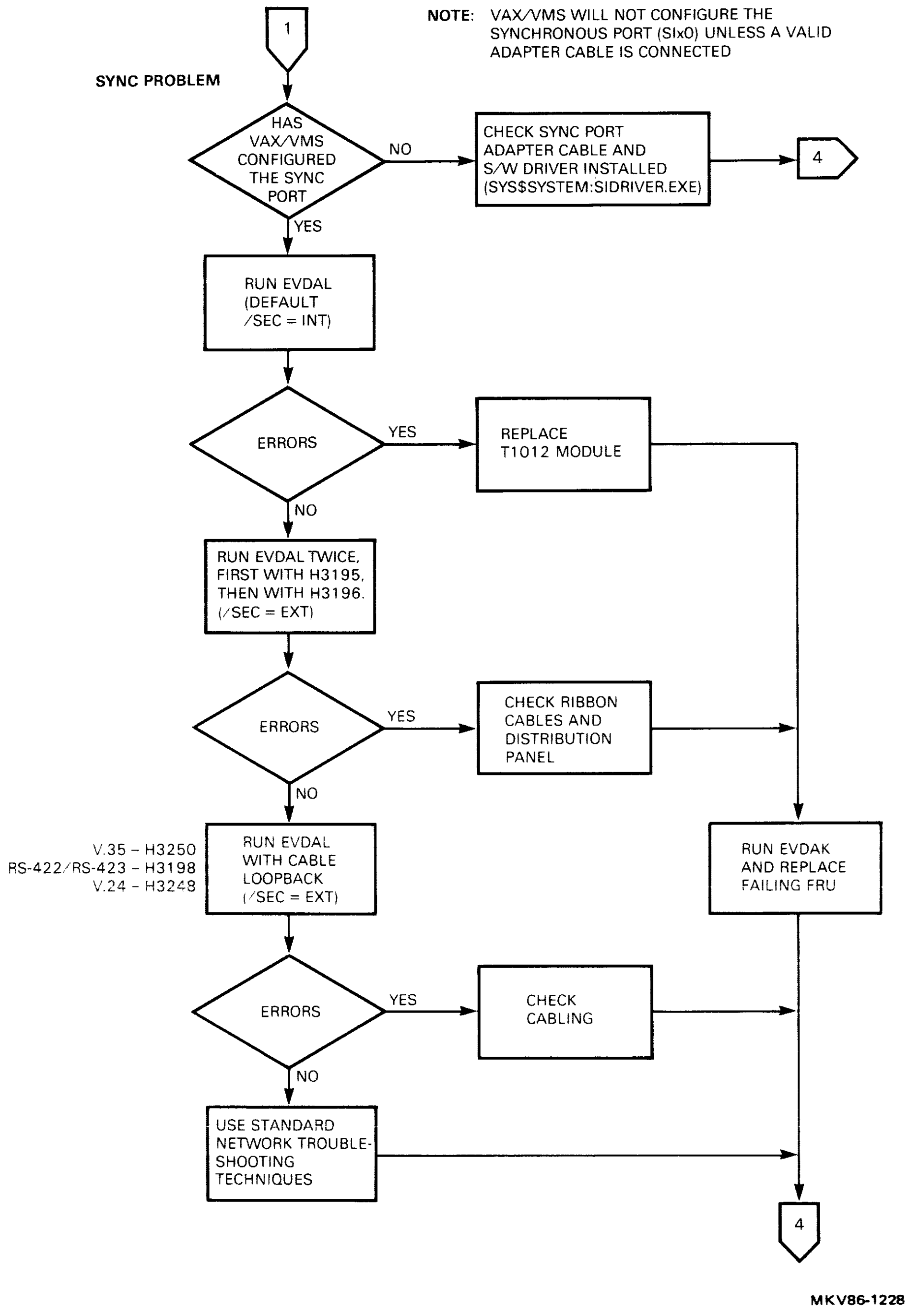

NOTE: VAX/VMS WILL NOT CONFIGURE THE SYNCHRONOUS PORT (SIx0) UNLESS A VALID ADAPTER CABLE IS CONNECTED
1
SYNC PROBLEM
HAS VAX/VMS CONFIGURED THE SYNC PORT
NO
CHECK SYNC PORT ADAPTER CABLE AND S/W DRIVER INSTALLED (SYS$SYSTEM:SIDRIVER.EXE)
4
YES
RUN EVDAL (DEFAULT /SEC = INT)
ERRORS
YES
REPLACE T1012 MODULE
NO
RUN EVDAL TWICE, FIRST WITH H3195, THEN WITH H3196. (/SEC = EXT)
ERRORS
YES
CHECK RIBBON CABLES AND DISTRIBUTION PANEL
NO
V.35 - H3250
RS-422/RS-423 - H3198
V.24 - H3248
RUN EVDAL WITH CABLE LOOPBACK (/SEC = EXT)
RUN EVDAK AND REPLACE FAILING FRU
ERRORS
YES
CHECK CABLING
NO
USE STANDARD NETWORK TROUBLE-SHOOTING TECHNIQUES
4
MKV86-1228

Figure 2-5 DMB32 Troubleshooting Flowchart (Sheet 2 of 4)

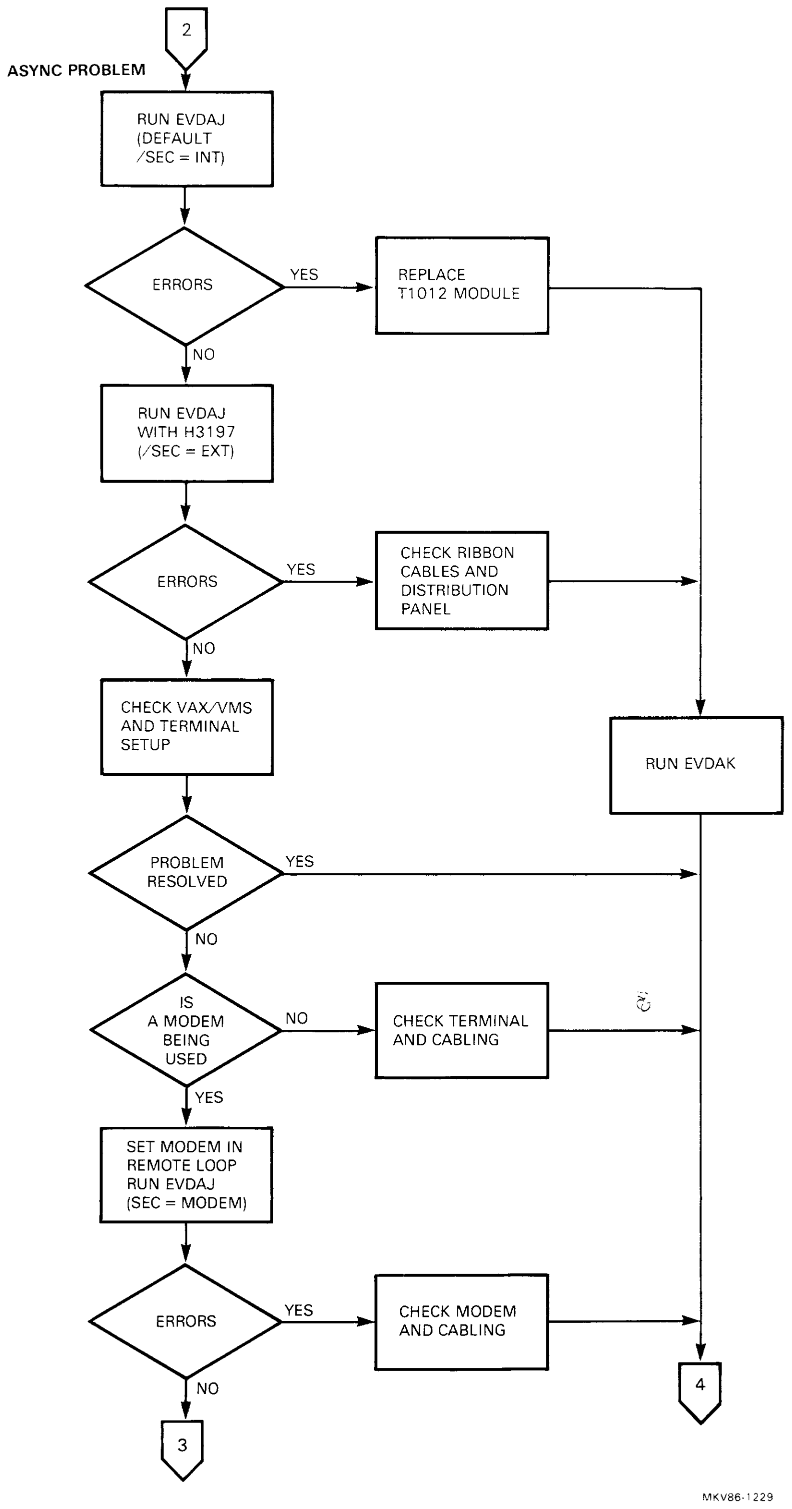

2
ASYNC PROBLEM
RUN EVDAJ (DEFAULT /SEC = INT)
ERRORS
YES
REPLACE T1012 MODULE
NO
RUN EVDAJ WITH H3197 (/SEC = EXT)
ERRORS
YES
CHECK RIBBON CABLES AND DISTRIBUTION PANEL
NO
CHECK VAX/VMS AND TERMINAL SETUP
RUN EVDAK
PROBLEM RESOLVED
YES
NO
IS A MODEM BEING USED
NO
CHECK TERMINAL AND CABLING
YES
SET MODEM IN REMOTE LOOP RUN EVDAJ (SEC = MODEM)
ERRORS
YES
CHECK MODEM AND CABLING
NO
3
4
MKV86-1229

Figure 2-5 Troubleshooting Flowchart (Sheet 3 of 4)

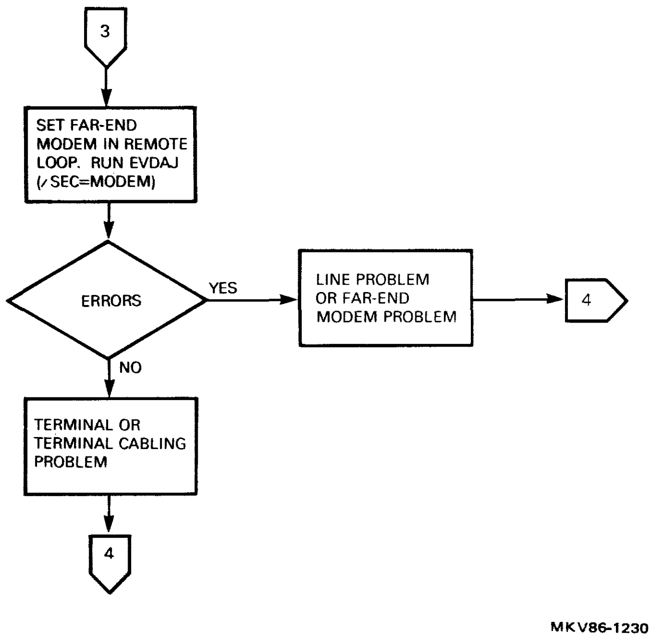

3
SET FAR-END MODEM IN REMOTE LOOP. RUN EVDAJ (/SEC=MODEM)
ERRORS
YES
LINE PROBLEM OR FAR-END MODEM PROBLEM
4
NO
TERMINAL OR TERMINAL CABLING PROBLEM
4
MKV86-1230

Figure 2-5 DMB32 Troubleshooting Flowchart
(Sheet 4 of 4)

LESSON 5 - RELATED DOCUMENTATION

This is a list of available documentation for the DMB32 adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- DMB32 User Guide, EK-DMB32-UG
- DMB32 Technical Description, EK-DMB32-TD
- DMB32 Field Maintenance Print Set, MP-01797-01

DEBNT ADAPTER

3

LESSON 1 - INTRODUCTION

The DEBNT Ethernet/tape controller is an intelligent I/O controller designed for use with the VAXBI bus, as shown in Figure 3-1. The DEBNT interfaces a single VAXBI host with an Ethernet local area network and a TK50 streaming tape drive. Since the DEBNT has its own compute engine, a dedicated MicroVAX processor, it is able to control I/O transfers independently of the host. The details of Ethernet and tape transactions, including error correction and data transfer over the VAXBI bus, are thus transparent to the host.

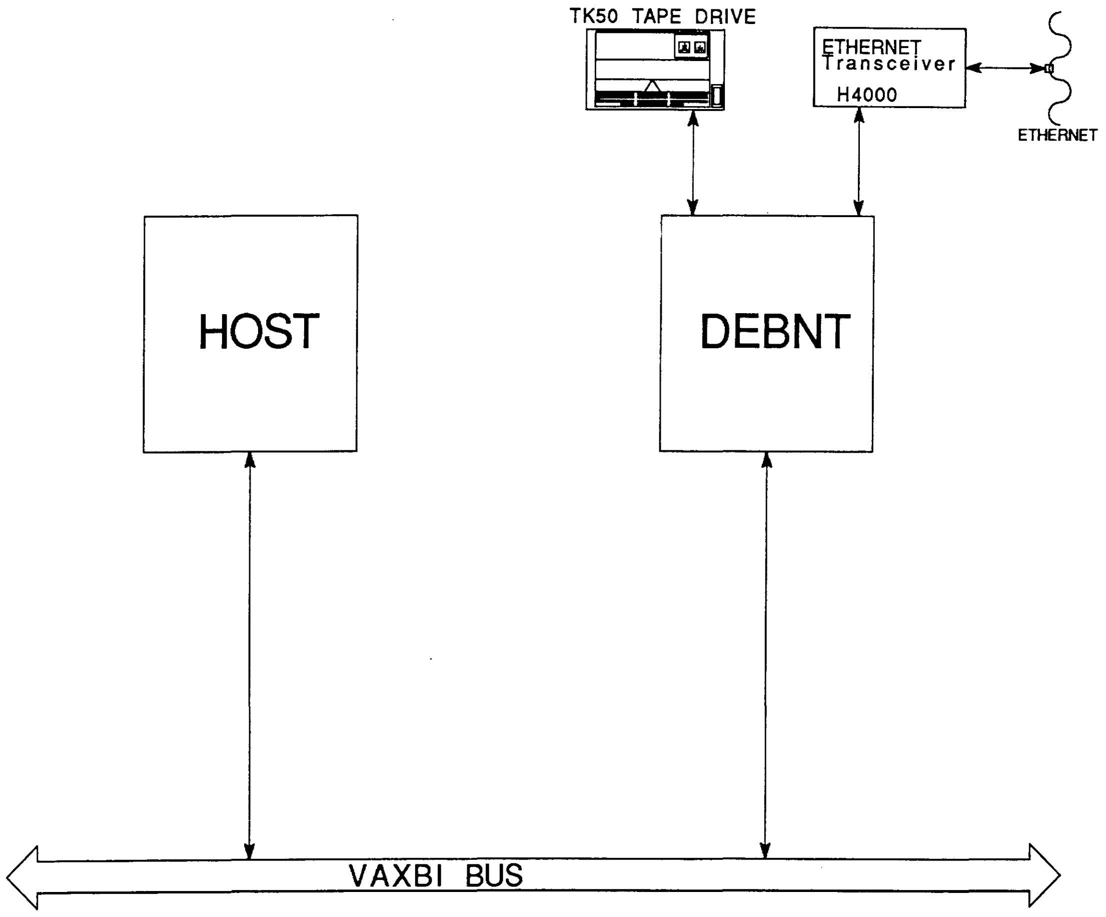


Figure 3-1 The DEBNT in a VAXBI System

The DEBNT supports one Ethernet port which provides physical and data-link communications. It also supports the TK50 tape drive which is a 5 1/4-inch unit that drives a 1/2-inch cartridge tape at 75 inches per second.

The tape controller in the DEBNT supervises tape operations such as read, write, position, and rewind and buffers data to and from the tape drive. In addition, the tape controller performs CRC error checking and ECC error checking and correction.

The Ethernet controller in the DEBNT enables the host to communicate with other processors in an Ethernet local area network. The controller implements the complete Ethernet protocol, and provides error detection and correction at the network management level.

The DEBNT executes all data transfers between itself and the host. To better utilize the VAXBI bus's large bandwidth, the DEBNT transfers data over the bus in octaword packets. For tape operations, The peak transfer rates are 62.5 Kbytes per second total and 45 Kbytes per second user data. For Ethernet transfers, the average sustained data rate is approximately 1 Mbyte per second.

The DEBNT has extensive on-board diagnostics. On power-up or reset, the DEBNT tests itself and makes its status available over the VAXBI bus. A set of LEDs on the DEBNT controller's printed circuit board also indicates the test results. In addition, a user may invoke other on-board diagnostics from the system console to test the DEBNT controller's logic and functionality more extensively. Other on-board diagnostics can be invoked to test Ethernet transceivers in loop-back mode and the TK50 functionality.

As illustrated in Figure 3-2, the DEBNT has the following component groups:

- MicroVAX processor and associated control logic
- II32 bus
- II32 memory and patch hardware
- VAXBI interface logic
- Ethernet controller
- Tape controller

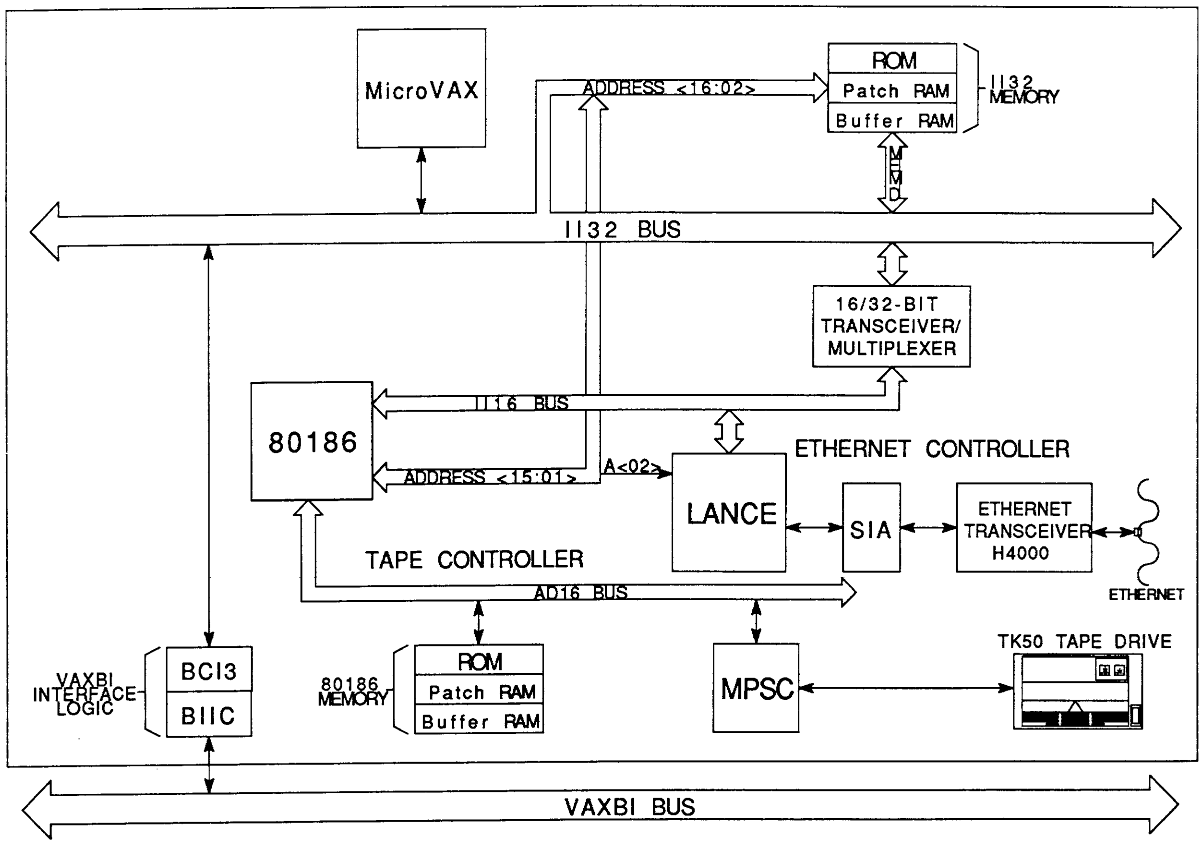


Figure 3-2 DEBNT Summary Block Diagram

MicroVAX and Associated Control Logic

The MicroVAX is a 32-bit, single-chip processor. In the DEBNT, the MicroVAX is dedicated to running firmware; it cannot be used directly by application programs running on the host processor or by a user at the system console. The MicroVAX is aided by external logic in controlling various high-level functions in the DEBNT.

II32 Bus

The interchip interconnect or II32 bus is a 32-bit bus that connects the MicroVAX with on-board memory and on-board devices. The II32 is time-multiplexed for addresses and data. The MicroVAX acts as the default bus master and central arbiter for the II32, granting the bus to other devices wishing to use it.

Memory on the II32 bus consists of 8K longwords of RAM and 32K of ROM. II32 RAM is logically divided into buffer RAM and patch RAM. II32 buffer RAM buffers data transfers between the MicroVAX and the following:

- Ethernet controller
- Tape controller
- VAXBI bus

II32 buffer RAM also stores most of the temporary data structures used by firmware running on the MicroVAX. II32 patch RAM, which consists of the upper 1K longword of II32 RAM, is reserved for patches to control firmware that runs on the MicroVAX.

II32 ROM stores initialization, control, and diagnostic firmware that runs on the MicroVAX. The control firmware, which controls the normal operation of the DEBNT, can be patched with substitute code loaded into II32 patch RAM. When patched ROM locations are addressed, patch hardware causes patch RAM to be referenced for instructions rather than ROM. The II32 patch function allows the DEBNT adapter's control firmware to be modified without replacing the II32 ROM.

A 32-bit data path, called the memory data or MEMD bus, interfaces the data ports of II32 memory with the II32 bus. A 32-bit transceiver connects the two buses.

Ethernet Controller

The Ethernet controller consists of the following chip set:

1. Local Area Network Controller for Ethernet (LANCE)
2. Serial Interface Adapter (SIA)

These chips are located on the II16 bus, a 16-bit, time-multiplexed bus. The II16 and II32 buses interface through a 16/32-bit transceiver/multiplexer.

The LANCE chip implements the data link layer of the Ethernet interface. The LANCE provides channel access, on-board DMA or direct memory access, packet handling, and error reporting.

The SIA chip implements the physical layer of the Ethernet interface. The SIA performs the following functions:

- Manchester encoding and decoding of the serial bit stream between the LANCE chip and Ethernet bus
- TTL/differential conversion between the LANCE chip and the Ethernet bus
- Collision detection on the Ethernet bus

Tape Controller

The tape controller consists of the following components:

- An 80186 microprocessor
- A multiple protocol serial chip (MPSC)
- An 80186 microprocessor memory and patch hardware

The 80186 is a 16-bit microprocessor that runs the TK50 drive firmware. During normal operation, the 80186 functions as a slave to the MicroVAX. The two processors communicate through shared data structures resident in the tape controller's memory. The details of tape operation are transparent to the MicroVAX.

The 80186 is located on the II16 bus. The 80801's peripheral devices are located on the 80186 microprocessor's local bus, the AD16. The AD16 is a 16-bit, time-multiplexed bus that interfaces with the II16 bus through a 16-bit transceiver.

The MPSC chip interfaces the 80186 processor to the tape drive. The MPSC performs parallel-to-serial and serial-to-parallel conversion, CRC generation, and CRC checking.

The 80186 has 16 Kbytes of RAM logically divided into buffer RAM and patch RAM. The 81086 buffer RAM data transfers between the following:

1. The 80186 and the MicroVAX
2. The 80186 and the MPSC chip

The 80186 buffer RAM also stores temporary data structures required by firmware running on the 80186. The upper 1 Kbyte of 80186 RAM is reserved for patches to 80186 ROM. This reserved space is called 80186 patch RAM.

80186 ROM stores the control and diagnostic firmware that runs on the 80186. All of this firmware can be patched with substitute code loaded into 80186 patch RAM. When patched firmware is addressed, 80186 patch hardware causes 80186 patch RAM to output substitute code instead. This ROM patch function allows the normal operation of the tape controller, as well the tape controller's diagnostics, to be modified without replacing the 80186 ROM.

Basic Operations

The DEBNT performs two basic sets of operations:

- Ethernet transmissions and receptions
- Tape reads and writes

This section will outline all these two sets of transactions to provide an overview of the DEBNT adapter's operation.

Ethernet Transmissions

In an Ethernet transmission, shown in Figure 3-3, the DEBNT receives packets from the host, buffers and processes the packets internally, and then transmits the packets over the Ethernet bus.

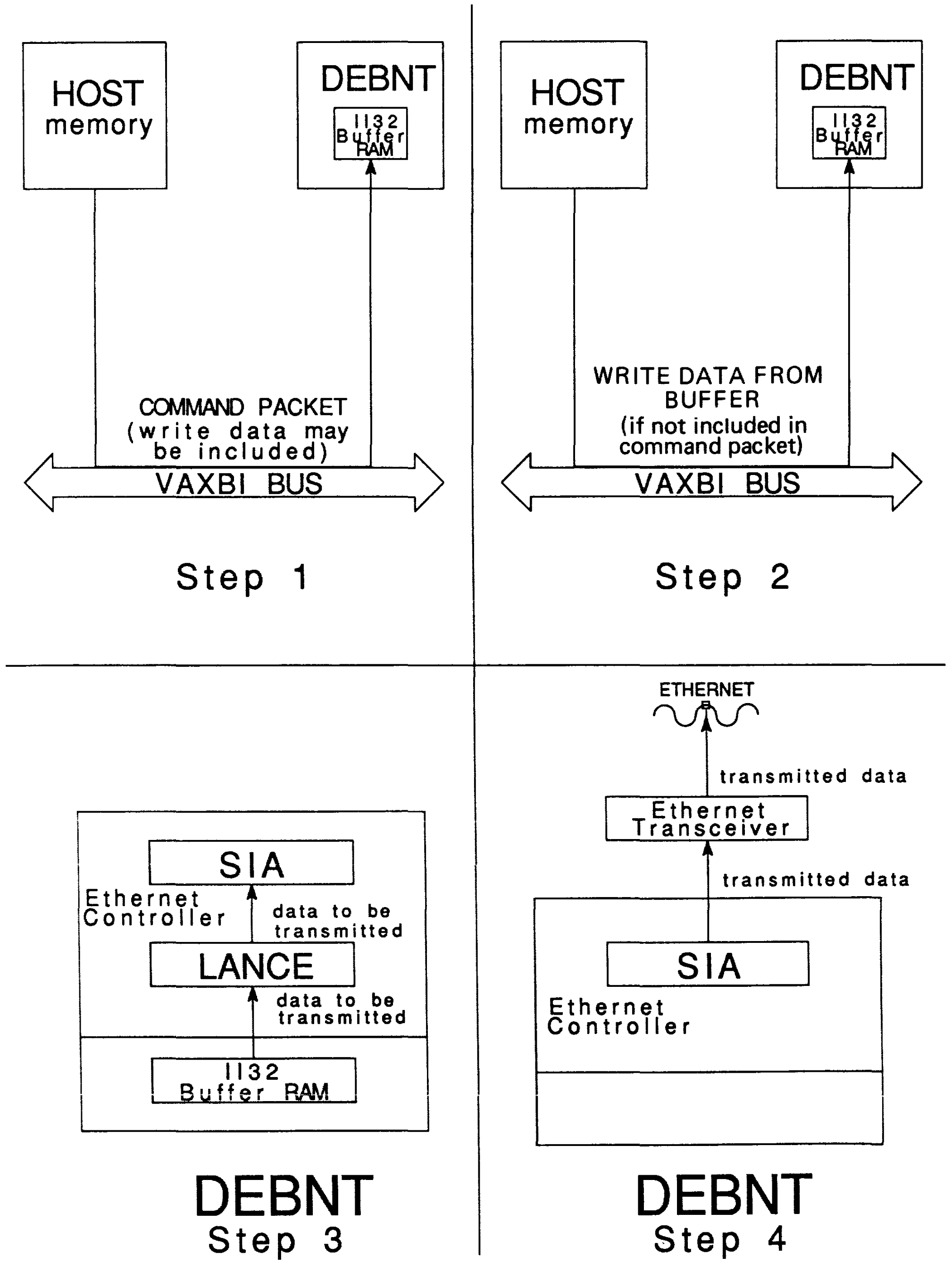


Figure 3-3 Ethernet Transmission

An Ethernet transmission has the following major steps:

1. The host links a command packet to the end of a command queue in host memory. If the command queue was empty, the host signals the DEBNT by means of a write to the port control register, to indicate that a command is pending. Otherwise, the DEBNT will process via port control register, the new command after it has processed other command packets in the queue.

2. The DEBNT delinks the command packet and caches it in II32 buffer RAM. The command packet describes the Ethernet packet to be transmitted. The command packet may contain the data to be transmitted, or may describe data buffers in host memory that contain the data to be transmitted.

3. The DEBNT transfers the Ethernet packet and any additional data from host memory to II32 buffer RAM.

4. The LANCE chip in the Ethernet controller transfers the data from II32 buffer RAM to an internal first-in-first-out or FIFO memory.

   This transfer is pipelined with data transfer from the host to II32 memory. The LANCE processes the data into an Ethernet packet by packaging it with preamble and synchronization information and CRC information. This packet building occurs while data is passing through the LANCE. Data input to the LANCE is pipelined with serial data output from the LANCE to the SIA chip.

5. The SIA chip converts the TTL non-return-to-zero or NRZ data from the LANCE into differential Manchester signals for transmission over the Ethernet. If the Ethernet is available, the SIA transmits the packets serial to the Ethernet transceiver.

Ethernet Reception

In an Ethernet reception, shown in Figure 3-4, the DEBNT adapter's Ethernet controller receives packet from the Ethernet bus through a transceiver; filters the packets according to protocol type, destination address, and source address; buffers the packets internally; and then transfers accepted Ethernet packets to the host.

1. The SIA chip in the Ethernet controller receives packets from the Ethernet bus in bit serial form. The SIA converts the differential Machester data from Ethernet into TTL NRZ data for input to the LANCE chip.

2. The LANCE inputs packet(s) from the SIA chip and transfers the data one word at a time to II32 buffered RAM. While receiving the packets from the SIA chip, the LANCE continuously calculates a CRC checksum for each packet. After receiving an entire packet, the LANCE compares the calculated checksum with the received checksum. If the two CRC checksums are not equivalent, the LANCE sets the CRC Error bit in the transmit descriptor ring which is a data structure in II32 memory shared by the LANCE and the MicroVAX. The host will discard any corrupted packets.

3. The network interconnect or NI firmware filters the received packets according to protocol type, destination address, and source address. The NI firmware will forward acceptable packets and determine programability to the host.

   The LANCE signals the MicroVAX to process the received packet.

4. On command from the DEBNT adapter's MicroVAX, the BCI3 chip in the DEBNT transfers accepted packets from II32 buffer RAM to an NI response queue in host memory. If the response queue is empty, the DEBNT interrupts the NI port driver to indicate that data has been received over the Ethernet.

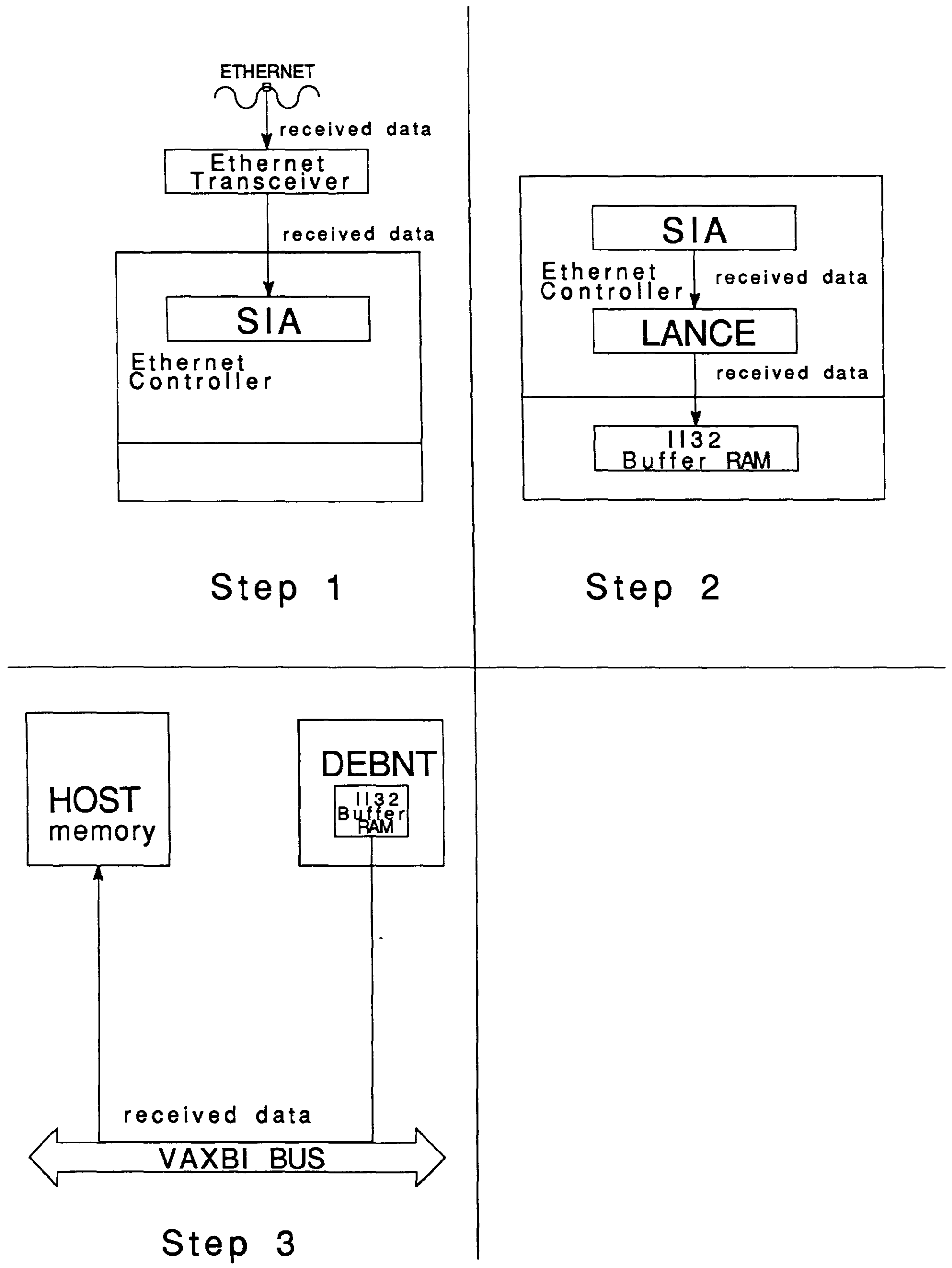


Figure 3-4 Ethernet Reception

Tape Read

In a tape read transaction, shown in Figure 3-5, the DEBNT reads data from the tape, buffers the data internally, and then transfers the data over the VAXBI bus to host memory. The host always sees a perfect media; that is, the host need only report errors to higher-level user software, since the DEBNT performs all error recovery and handles all media defects.

A tape read operation has five major steps:

1. The host links a command packet to a command queue in host memory. If the queue was empty, the host also signals the DEBNT via the port control register to indicate that a command is pending. Otherwise, the DEBNT will process the new command after it has processed other command packets in the queue. The DEBNT eventually delinks the command packet from the front of the command queue and caches the packet. The packet informs the tape controller of three things:

   - That the tape controller should perform a tape read
   - The number of bytes to be transferred
   - The address of a host data buffer in which to deposit the data

2. The tape controller starts the tape drive, reads the tape, and transfers the required tape data into local memory. While reading data, the tape controller performs CRC and ECC error checking and data correction. If the tape controller discovers an unrecoverable ECC error, it flages the data as bad in buffer RAM. The host will decide what to do with the corrupted data after the read operation is finished.

3. The tape controller transfers the tape data from the 80186 buffer RAM to the MicroVAX memory. This transfer can occur while the tape controller is reading more data into the 80186 buffer RAM.

4. On command from the DEBNT adapter's MicroVAX, the BCI3 in the DEBNT transfers the read data from the II32 buffer RAM to host memory.

5. When the read is finished, the DEBNT links the original command packet, as an end packet, to the end of a response queue. The end packet signals that the read operation is finished, and indicates the completion status of the operation.

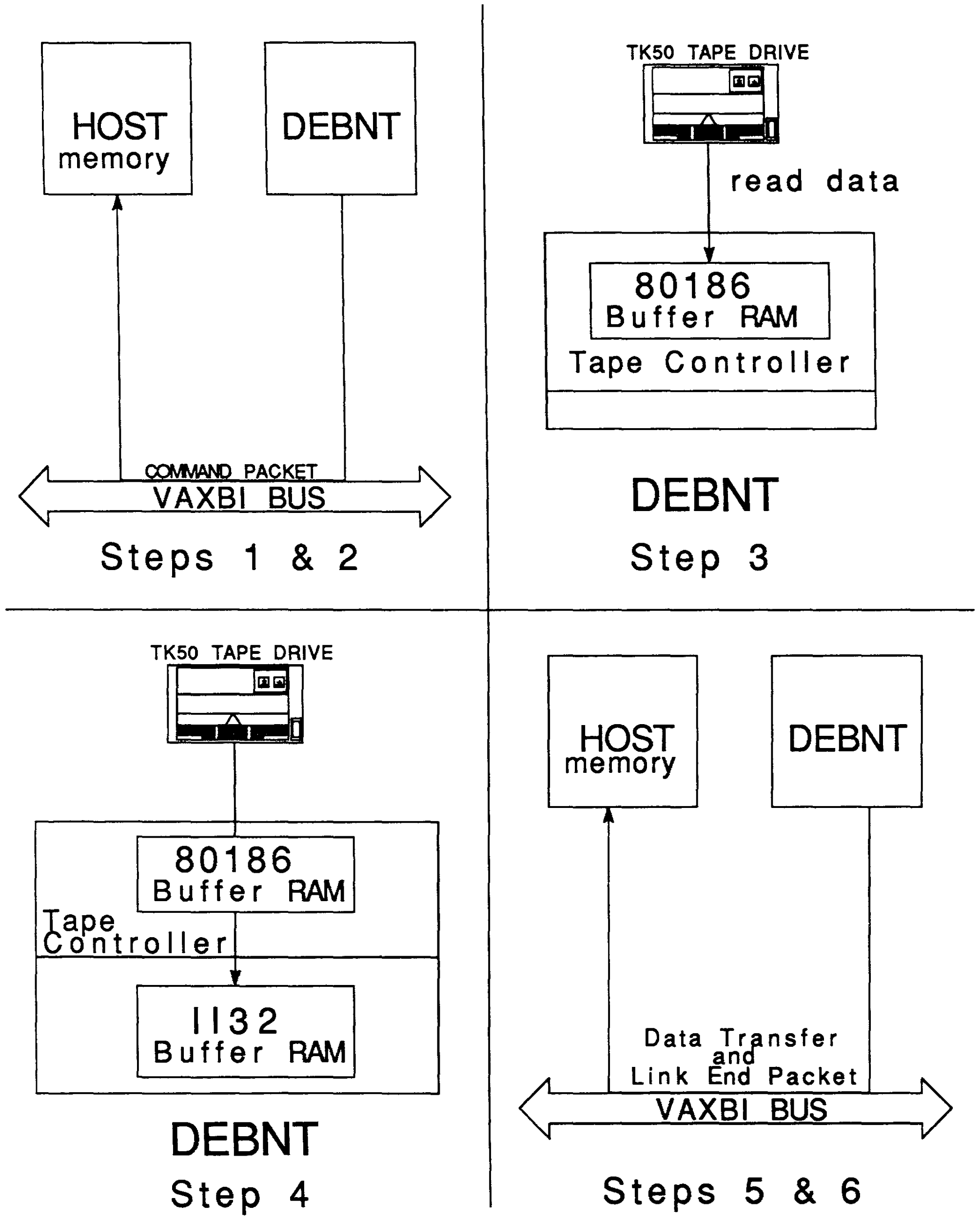


Figure 3-5 Tape Read Transaction

Tape Write

In a tape write transaction, the DEBNT retrieves data from host memory, buffers the data internally, and then writes the data to the tape. A write operation has four major steps.

1. The host links a command packet to the end of a command queue in host memory. If the queue was empty, the host also signals the DEBNT to indicate that a command is pending. Otherwise, the DEBNT will discover the new command after it has processed other commands packets in the queue. The DEBNT eventually delinks the command packet from the front of the command queue and caches the packet. The packet informs the tape controller of three things:

   - That the tape controller should perform a tape write
   - The number of bytes to be transferred
   - The address of a host data buffer that contains the data to be written

   The tape firmware transfers the write data from host memory to the II32 buffer RAM and signals the tape controller to transfer the data to tape.

2. The tape controller transfers the write data from the II32 buffer RAM to its own memory. This transfer can occur while the DEBNT is reading more write data from the host into the II32 buffered RAM.

3. The tape controller starts the tape drive, and transfers the data in the 80186 buffer RAM to the tape. The tape controller calculates a CRC checksum for each block and double ECC code for each six blocks.

4. When the write is finished, the DEBNT links the original command packet, as an end packet, to the end of a response queue. The end packet indicates that the write operation is finished and the completion status of the operation.

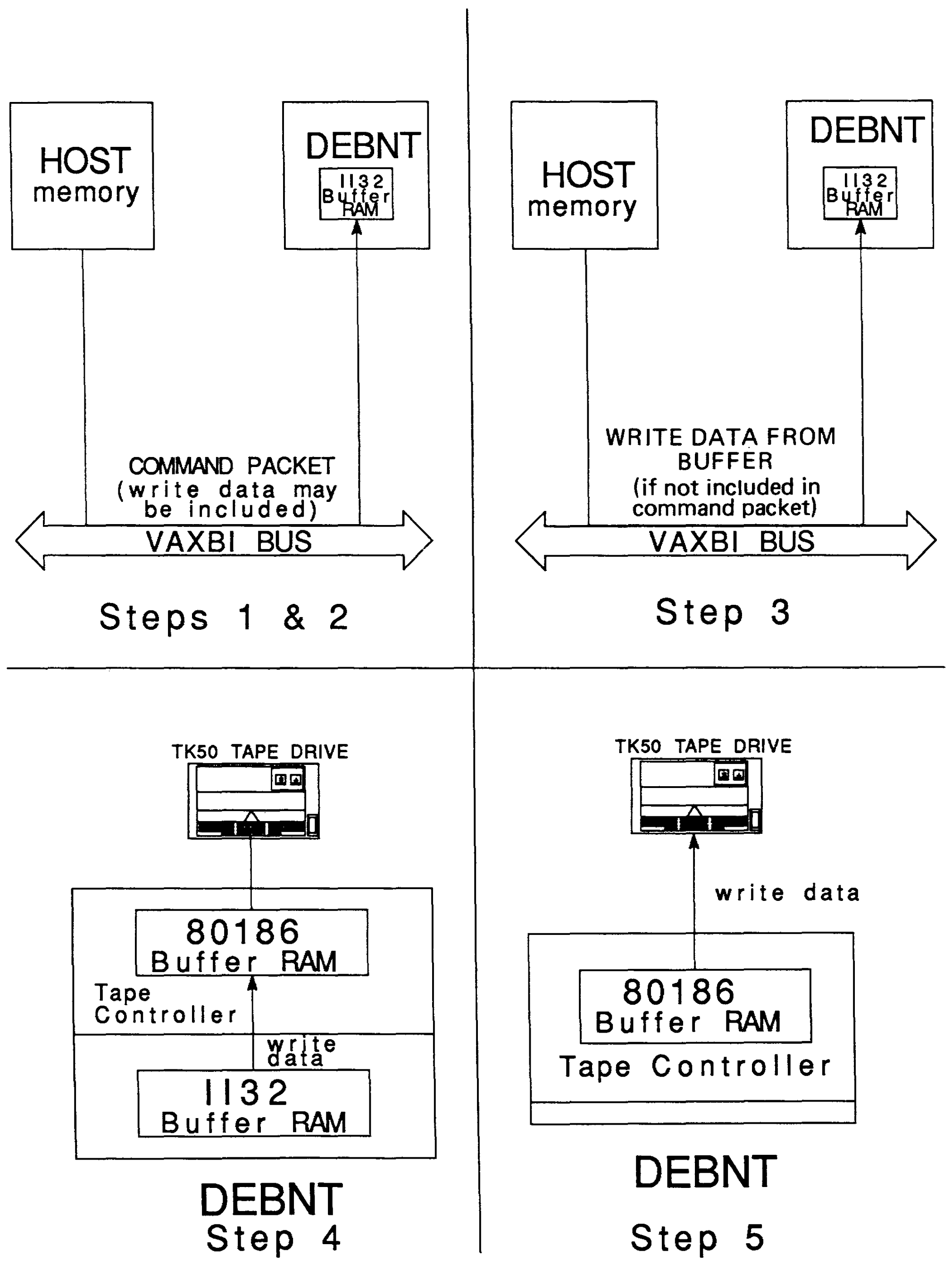
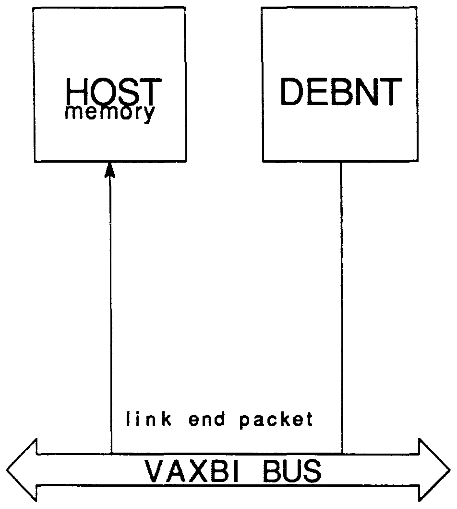


Figure 3-6 Tape Write Transaction (Sheet 1 of 2)



Step 6

Figure 3-6 Tape Write Transaction (Sheet 2 of 2)

Control Firmware Overview

The DEBNT is a multi-port adapter; it implements one port for the Ethernet controller and a separate port for the tape controller. The two ports are logically distinct: they are controlled by logically distinct port drivers running on the host, and operate in parallel. Furthermore, each port is individually defined and initialized, and has its own dedicated data structures. State changes that occur for one port normally do not affect the other port, except in the following situations:

- A hardware failure or several logical failures occur in the adapter.
- The DEBNT receives a VAXBI STOP command, which forces both ports to enter the STOPPED state.
- A BI BVP VAX Port RESTART command forces both ports to enter the UNDEFINED state.
- A node reset forces both ports to enter the UNDEFINED state.

The DEBNT implements one I/O port for the tape drive and one I/O port for Ethernet. The tape port is called the tape adapter or TA port. The Ethernet port is called the network interconnect or NI port.

Network Interconnect (NI) Port

The NI port handles all communication between the host and the Ethernet controller.

The NI is implemented by two firmware layers. (See Figure 3-7.) The upper layer is the NI layer, which is logically connected to the NI driver in the host. The lower layer is a BI VAX Port or BVP that is logically connected to the NI BVP port driver in the host.

When the host has data to transmit over the Ethernet the NI layer buffers the data in the II32 buffer RAM, and starts up the Ethernet controller. When the Ethernet controller receives packets over Ethernet, the NI layer does second-level filtering for multicast addresses, filters for protocol types, monitor the LANCE's CRC checking, reports any CRC errors to the host, and buffers the received data in the II32 buffer RAM.

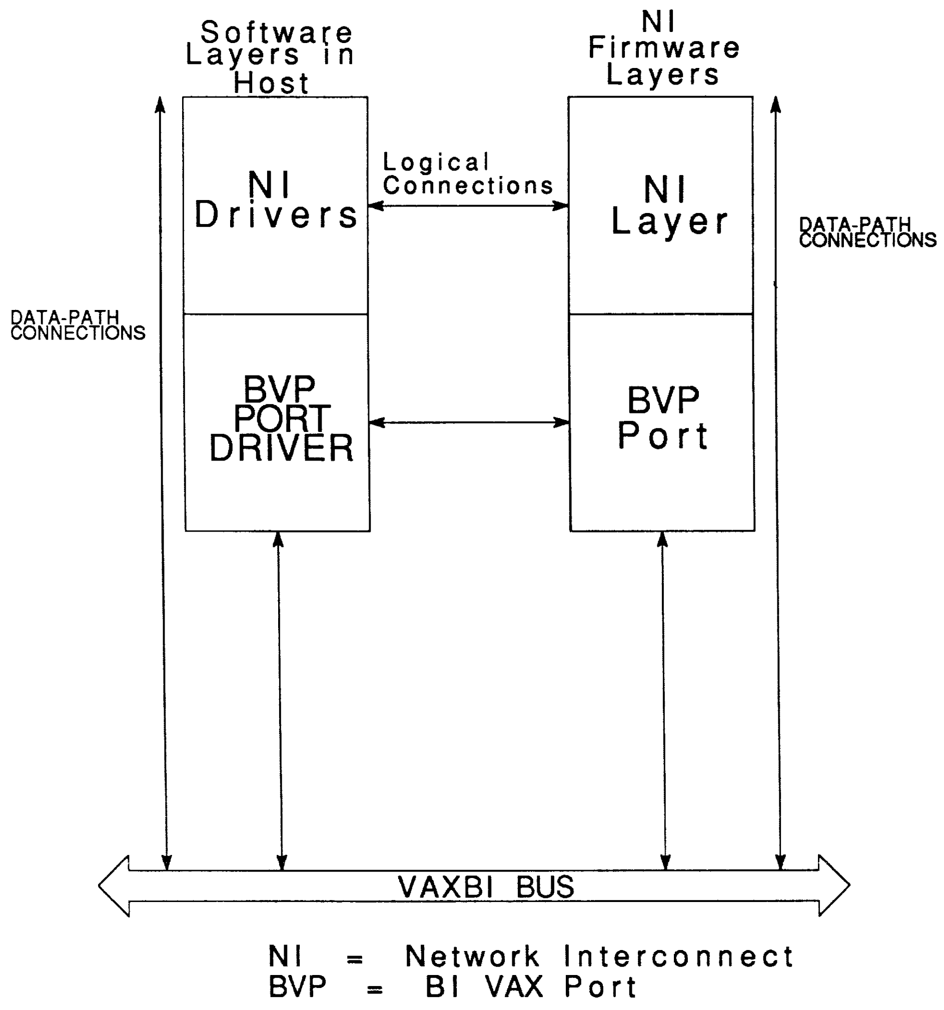


Figure 3-7 NI Firmware and Corresponding Software Layers in Host

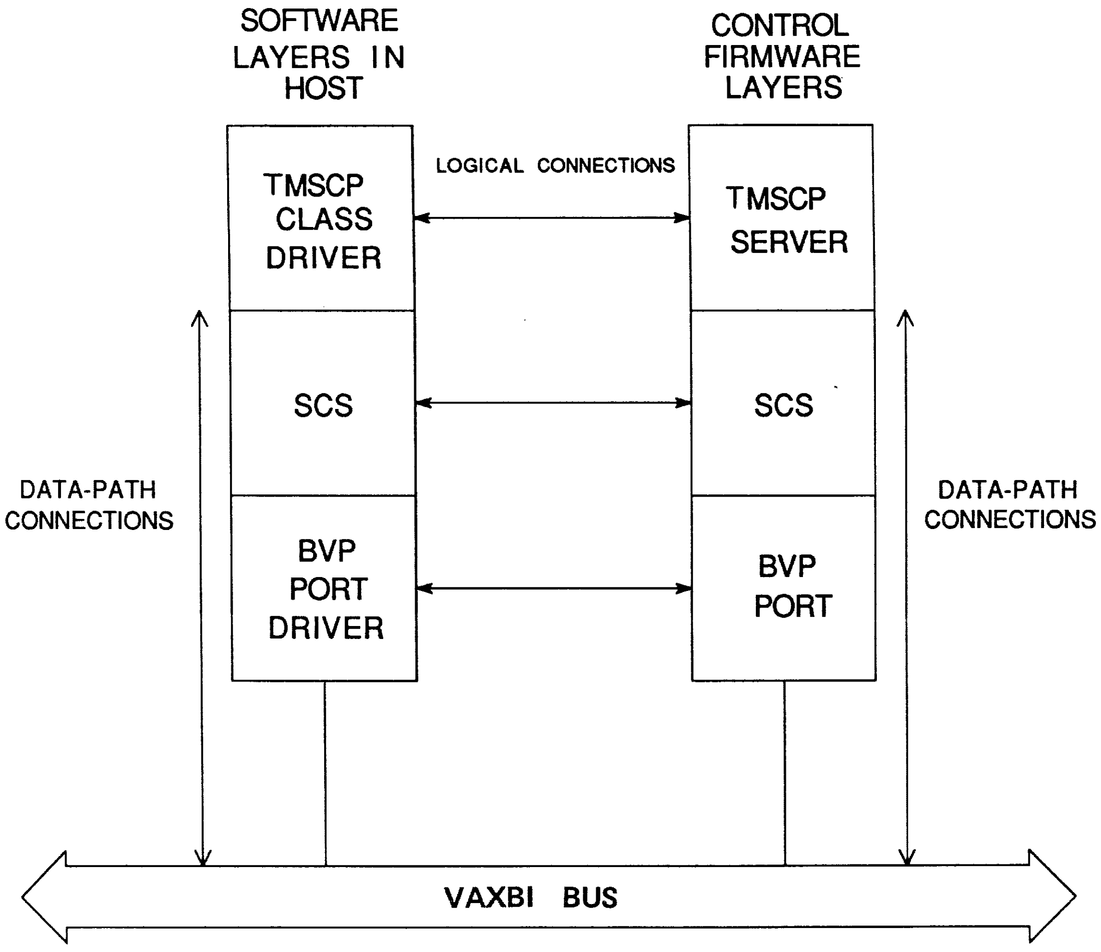
Tape-Port Firmware in the II32 ROM

The BVP port is the DEBNT option's logical interface to the host port driver.

The control firmware in the II32 ROM consists of the following layers:

- A tape mass storage control protocol or TMSCP server
- A systems communication services or SCS layer
- A VAXBI VAX port or BVP layer

In terms of the SCA architecture the TMSCP server is the systems applications layer, SCS is the communications layer, and the BVP port is the PPD (port/port driver) layer. Figure 3-8 shows the logical and data-path connections between the tape-control firmware layers in the II32 ROM and the software layers resident in the host.



MSCP	=	MASS STORAGE CONTROL PROTOCOL
SCS	=	SYSTEMS COMMUNICATION SERVICES
BVP	=	BI VAX PORT

Figure 3-8 Control Firmware Layers in II32 ROM

LESSON 2 - DEBNT REGISTERS

In this Lesson you will look at the registers that are important to the DEBNT adapter. All registers associated with the DEBNT are located in the VAXBI nodespace.

First, you will look at the control logic associated with the MicroVAX; then you will look at the VAXBI registers.

The control logic associated with the MicroVAX consists of the following:

- DEBNT control and status registers
- Status register
- Address PAL
- DMA arbitration PAL
- Bus control

NOTE
These registers are located in the VAXBI node private address space and are not directly visible on the VAXBI.

Of the above registers, the first two are most useful for Field Service engineers.

The most useful VAXBI registers are listed below:

- Port register
- Bus error register
- Device register
- Starting and ending registers

NOTE
These registers are located in the VAXBI node space and are visible using most console commands. These registers also appear in the error log entries.

LESSON 3 - INSTALLATION

To install the DEBNT:

1. Gain access to the BI cage.

2. Insert the DEBNT module into any unused slot in the BI cage. (Refer to Figure 3-9.)

3. Secure the modules in the BI cage by closing the remote actuator handle for the DEBNT module slot. Ensure that the handle closes all the way, signifying that the modules are properly inserted. The ZIF connectors provide the electrical connection between the DEBNT blackplane I/O connectors and the DEBNT module.

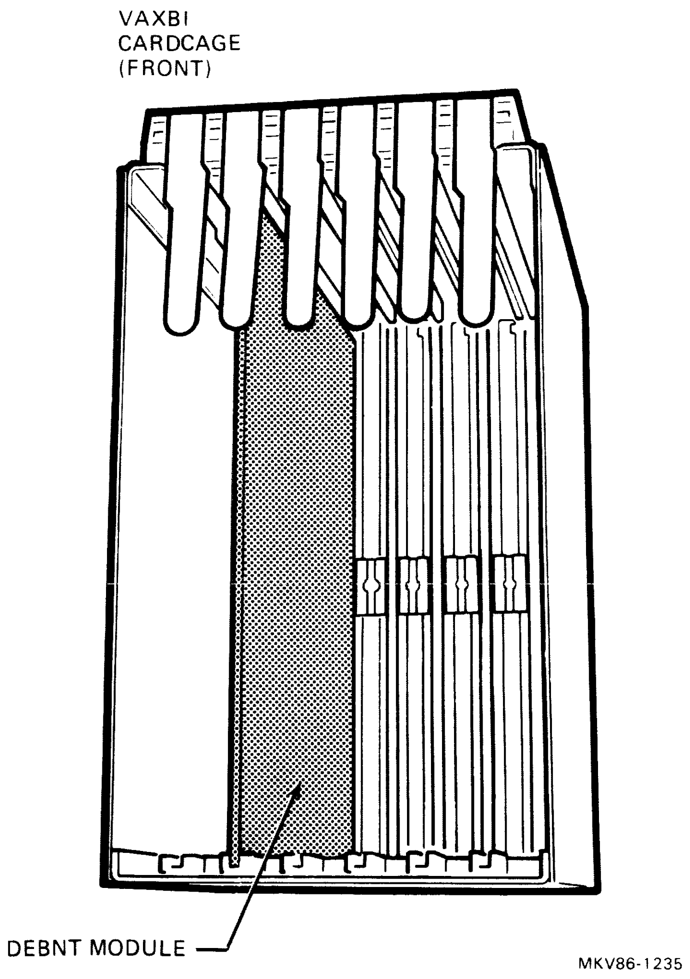


Figure 3-9 Inserting the DEBNT Module

4. Attach the DEBNT backplane I/O cables to the backplane of the BI cage using the following procedure.

   a. Gain access to the BI backplane.

   b. Remove the dust covers from the BI backplane.

   c. Install the transition header on the backplane of the slot that will hold the DEBNT module.

      NOTE

      When installing the transition header, use only the torque screwdriver provided in the Field Service kit.

   d. Connect the cables to the transition header assembly as shown in Figure 3-10.

   e. Return the VAXBI cage to its original position.

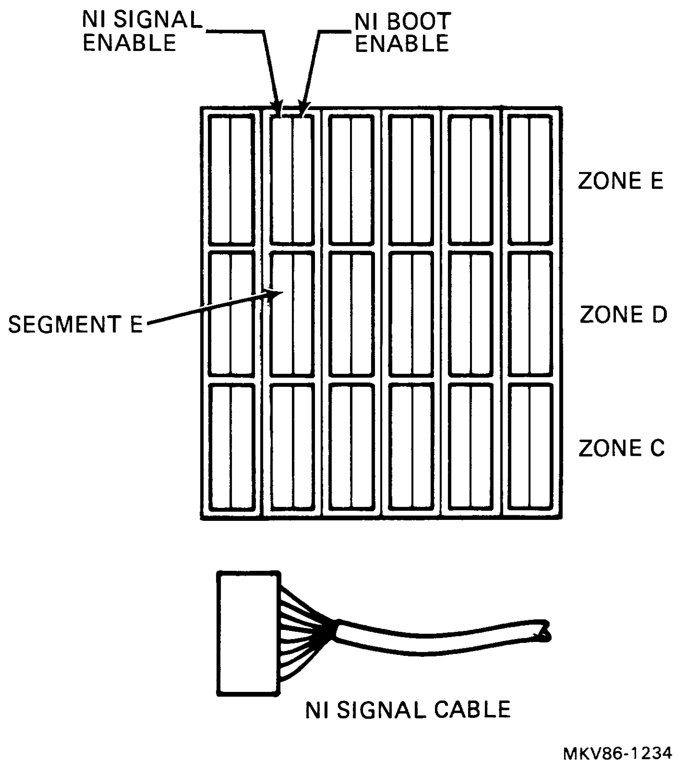


Figure 3-10 VAXBI Backplane Cable Connections

1. Attach the I/O bulkhead on the VAXBI-based system cabinet using the following procedure (refer to Figure 3-11):

   a. Bring the internal cables through the opening of one of the available option bulkhead panels of the VAXBI-based system backframe.

   b. Screw the internal cables into the small holes on on the I/O bulkhead. These connections are keyed so they will only attach one way.

   c. Mount the I/O bulkhead on the option bulkhead panel.

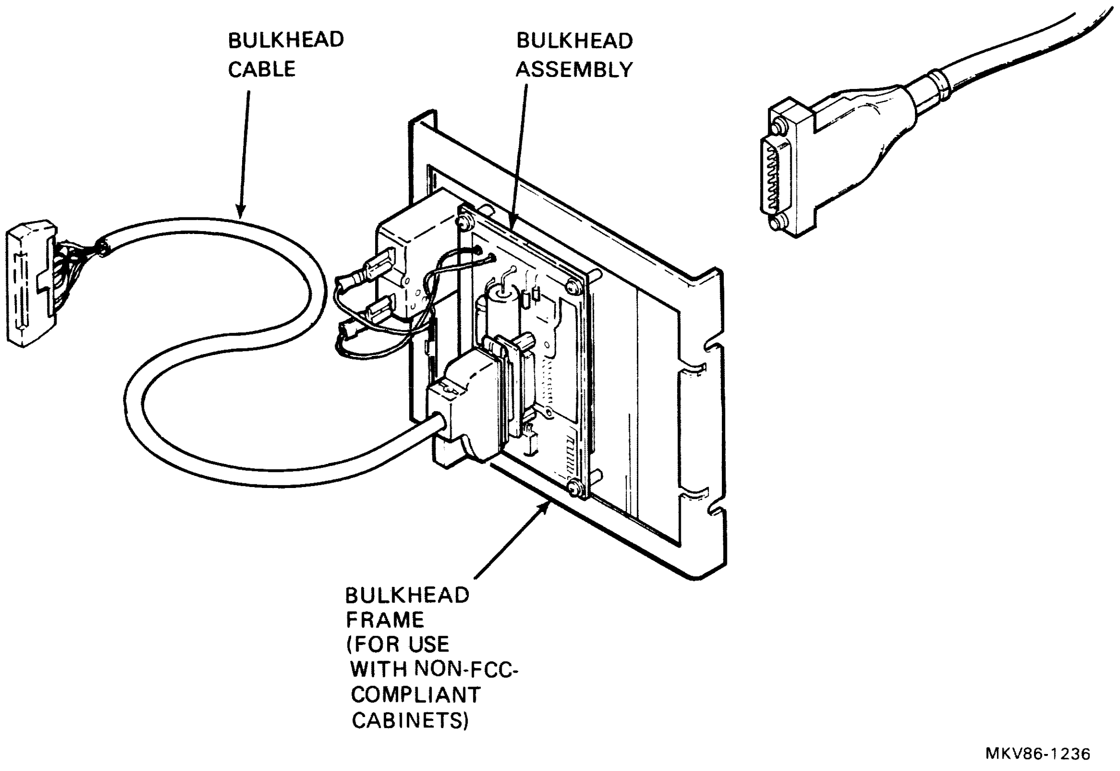


Figure 3-11 Attaching the I/O Bulkhead

2. Attach the keyed, external cables to the VAXBI cabinet I/O bulkhead.

3. Power up the system. The self-test on the KDB50 nodule set is activated upon power-up. At this time, two single yellow LED indicators show the current state of the KDB50.

   Check that the yellow LEDs on each KDB50 module turn on after about ten seconds.

4. If the yellow LEDs do not turn on there is a problem. Dealing with such problems will be covered in the next section.

LESSON 4 - TROUBLESHOOTING

For the field acceptance test procedure, refer to the documentation for the system you are working on and to the documention for the Ethernet.

Figure 3-12 is a flowchart to be used in troubleshooting this adapter.

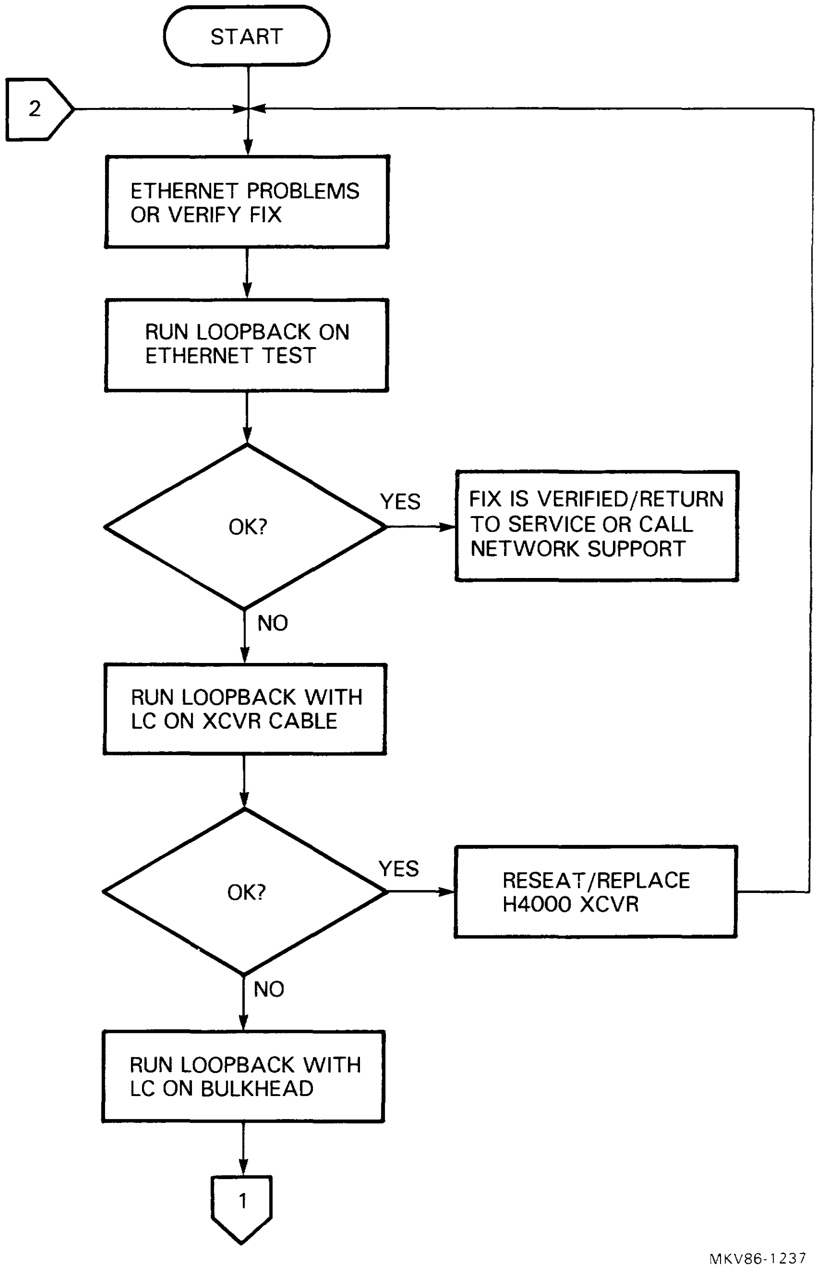

START
2
ETHERNET PROBLEMS OR VERIFY FIX
RUN LOOPBACK ON ETHERNET TEST
OK?
YES
FIX IS VERIFIED/RETURN TO SERVICE OR CALL NETWORK SUPPORT
NO
RUN LOOPBACK WITH LC ON XCVR CABLE
OK?
YES
RESEAT/REPLACE H4000 XCVR
NO
RUN LOOPBACK WITH LC ON BULKHEAD
1
MKV86-1237

Figure 3-12 DEBNT Troubleshooting Flowchart (Sheet 1 of 3)

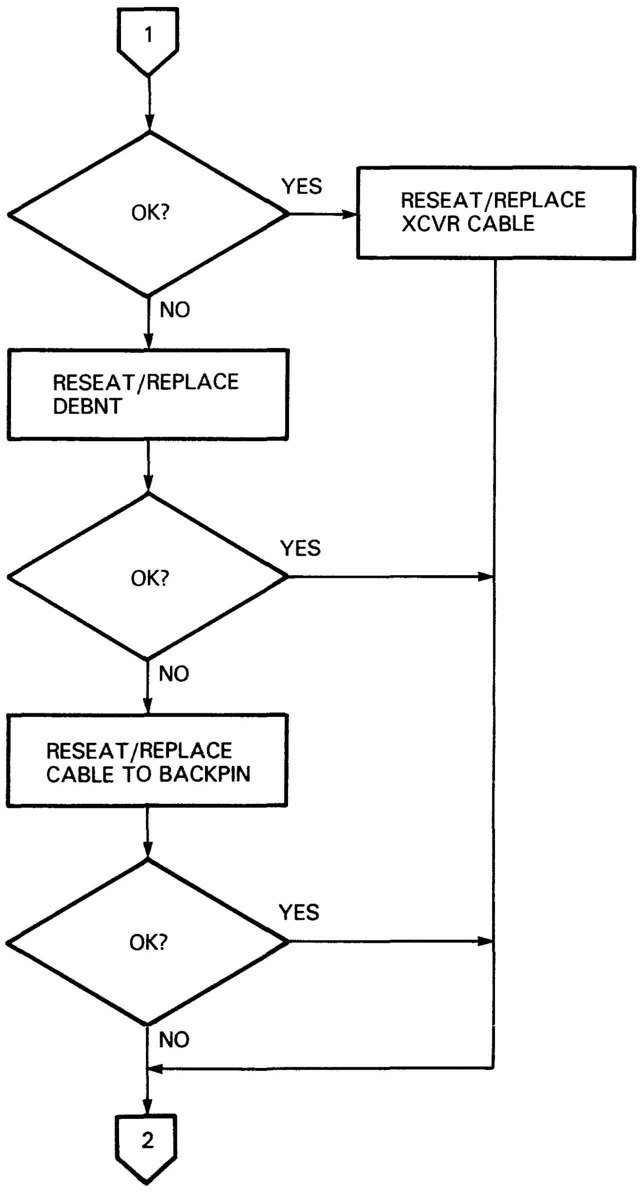


Figure 3-12 DEBNT Troubleshooting Flowchart (Sheet 2 of 3)

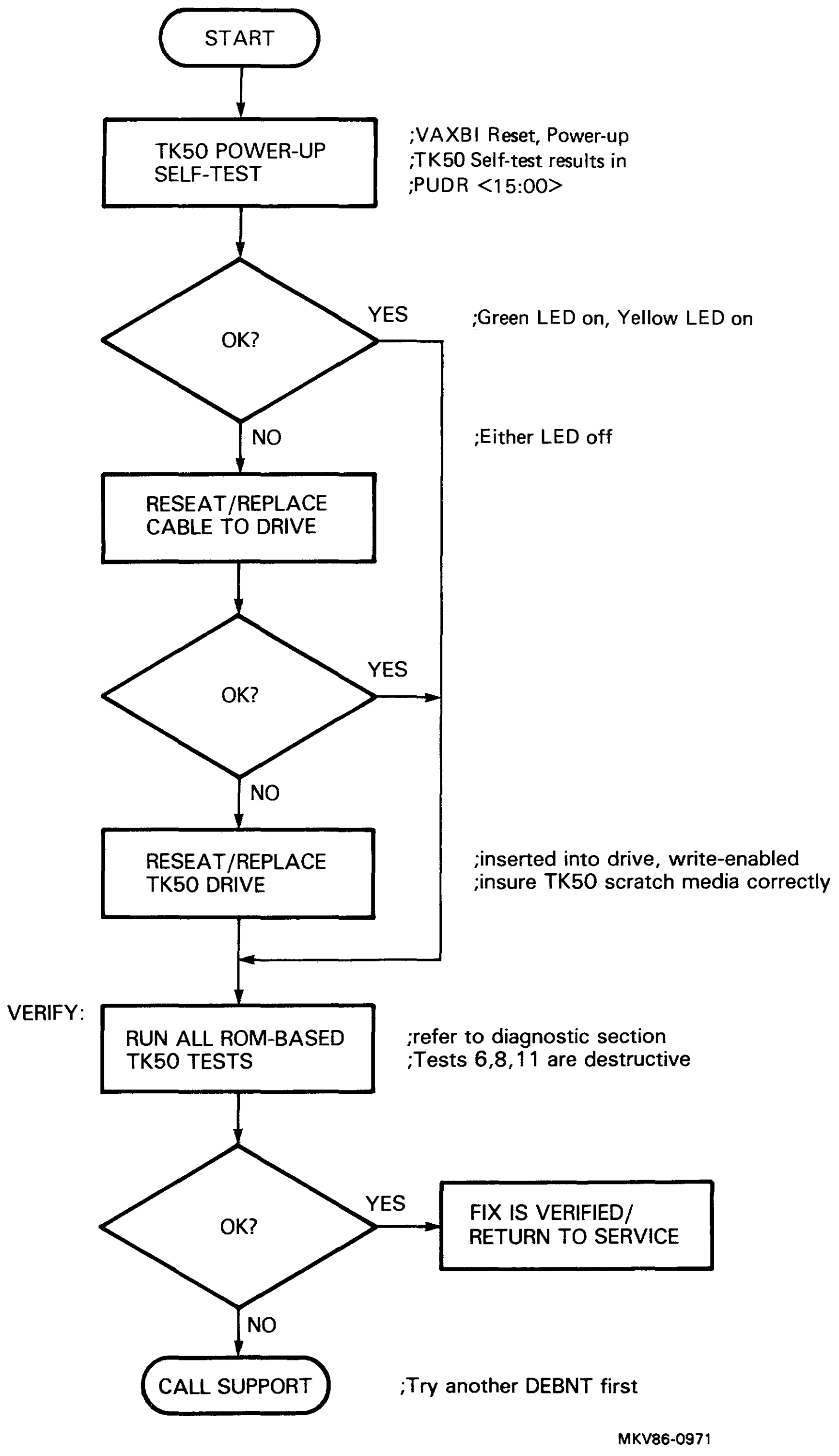

START
TK50 POWER-UP SELF-TEST
;VAXBI Reset, Power-up
;TK50 Self-test results in
;PUDR <15:00>
OK?
YES
;Green LED on, Yellow LED on
NO
;Either LED off
RESEAT/REPLACE CABLE TO DRIVE
OK?
YES
NO
RESEAT/REPLACE TK50 DRIVE
;inserted into drive, write-enabled
;insure TK50 scratch media correctly
VERIFY:
RUN ALL ROM-BASED TK50 TESTS
;refer to diagnostic section
;Tests 6,8,11 are destructive
OK?
YES
FIX IS VERIFIED/ RETURN TO SERVICE
NO
CALL SUPPORT
;Try another DEBNT first
MKV86-0971

Figure 3-12 DEBNT Troubleshooting Flowchart
(Sheet 3 of 3)

LESSON 5 - RELATED DOCUMENTATION

This is a list of available documentation for the DEBNT adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- DEBNT Ethernet/Tape Controller Technical Manual, EK-DEBNT-TM
- MicroVAX Handbook, EB-25126-47
- VAX Architecture Handbook, EB-19580
- Ethernet Specification, AA-K759B-TK

CIBCI ADAPTER

4

LESSON 1 - INTRODUCTION

The computer interconnect port adapter or the CIBCI is the adapter used to connect a host system into a VAXcluster. The CIBCI is centrally controlled by a single on-board data processor to provide buffered parallel-to-serial communications between two corporate interconnect bus architecture protocols; the VAXBI bus of the host processors, and the dual-path computer interconnect or CI bus. Figure 4-1 is a simplified diagram of the CIBCI port adapter connection.

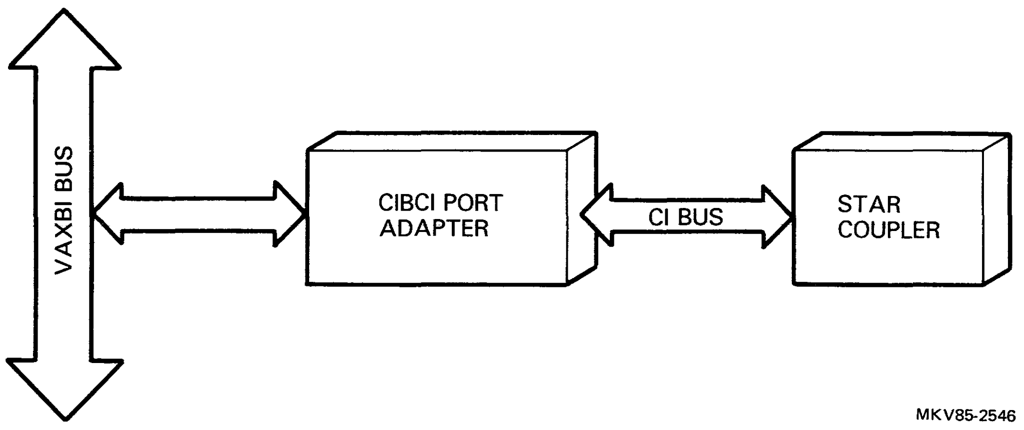


Figure 4-1 Simplified CIBCI Port Adapter Connection

As a buffered comunications port, the CIBCI adapter completes high-level computer communications, thereby reducing software processing overhead. This is accomplished with hardware that provides all the necessary data buffering, address translation, and serial data encoding/decoding. The CIBCI utilizes queue structures provided under the VAX/VMS operating system to transfer packet messages and to initiate the transfer of blocks of data between the host memory system and/or other nodes within the VAXcluster.

The CIBCI is partitioned into two separate hardware interfaces: one host processor interface and one computer interconnect interface. These interfaces consist of the following major components:

1. Two T-series modules
   a. Adapter data module
   b. Adapter control module
2. Three extended hex L-series modules
   a. Link interface modules
   b. Packet buffer module
   c. Data path module

The main features of the CIBCI are:

- VAX backplane interconnect design
- 2-3 Mbytes/s performance
- Diagnostic data loopback (internal/external) capability
- Data integrity via cyclic redundancy checking
- Round-robin arbitation at light loading

- Contention arbitration at heavy loading
- Packet-oriented data transmission
- Power-up self-test
- Operational modes:
  - Disable
  - Enable
  - Uninitialized
  - Disabled/maintenance
  - Enabled/maintenance
  - Uninitialized/maintenance

See the CIBCI Adapter User/Installation Guide, EK-CIBCI-UG, for a list of the model variations available with the CIBCI. The model variations are specified according to the host system type, and the major hardware components.

It is important to note that the CI bus cables and SC008 Star Coupler are separately ordered options. They are not included as part of the bill of materials on all CIBCI adapter models.

General Specifications

Table 4-1 shows the general specifications for the CIBCI adapter.

Table 4-1 General Specifications

Priority Arbitration	
Light loading	Round-Robin
Heavy loading	Contention
Parity	Cyclic Redundancy Check
Data format	Manchester-encoded serial packet
Data transfer rate	5 Mbytes/s maximum
Data throughput	2 - 3 Mbytes/s (typical sustained)
Operational modes	Disabled Disabled/Maintenance Enabled Enabled/Maintenance Uninitialized Uninitialized/Maintenance

Host Processor Interface Hardware

The host processor interface consists of two T-series type modules. The T-series modules are housed in two of six available options slots within the VAXBI cardcage of the host system. These modules are used to interface the host system's VAXBI bus to the CIPA bus located within the CI mounting box. See Figure 4-2.

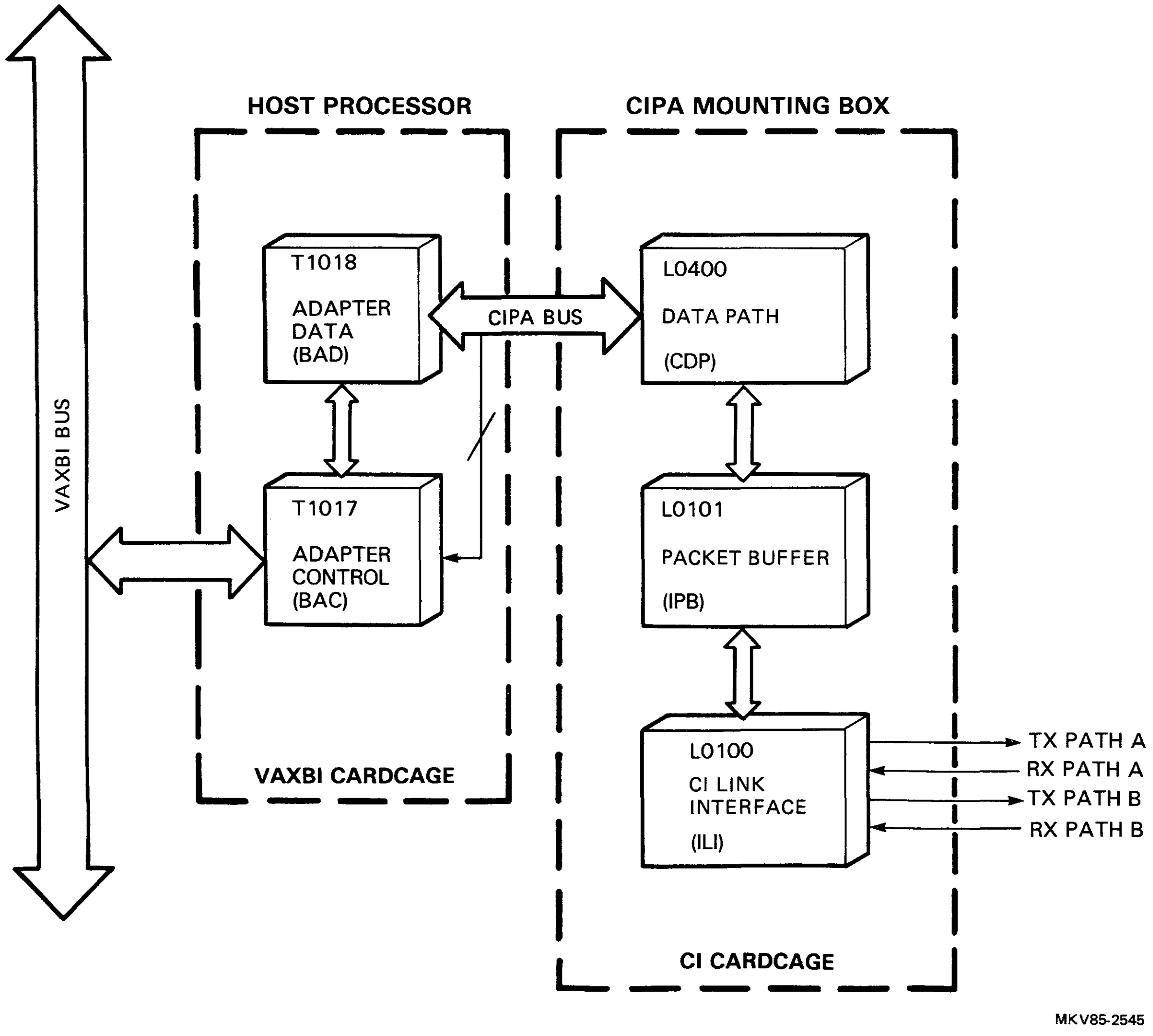


Figure 4-2 Simplified CIBCI Interface Block Diagram

The Adapter Control Module (BAC) - The adapter control module, part number T1017, contains the VAXBI protocol, and the VAXBI control logic.

The Adapter Data Module (BAD) - The adapter data module, part number T1018, is the major interface to the CI port adapter (CIPA) bus and consists of transactions buffers between the VAXBI and the CIPA buses.

BCI Cables - The four BCI cables, part numbers 17-01029-01 and 17-01029-2, are used to electrically interconnect the T0117 and T1018 modules together.

Computer Interconnect Interface Hardware

CI Box Assembly - The computer interconnect or CI interface hardware is housed in a dedicated, but universal, mounting box and designated the CI box assembly. This CI box assembly is also referred to as the CIPA mounting box.

CIPA-Backplane - The CIPA-backplane, housed within the CI box assembly, is a five-slot backplane used to house the three L-series modules: the data path, packet buffer, and the CI link.

Data path module (CDP)--- The data path module, part number L0400, provides the necessary arithmetic and logical processing of general port functions, as well as, local storage for the port. It also provides transceivers and buffer registers as the interface for the CIPA bus.

Packet buffer module (IPB)--- The packet buffer module, part number L0101, contains the port control store microcode and dual transmit and receive CI packet buffers. Each CI packet buffer has a storage capacity of 1K bytes.

CI link interface module (ILI)--- The CI link interface module, part number L0100, is the actual interface to the CI bus and is capable of servicing dual CI paths. The module provides the necessary serialization/deserialization of data, data validity, CI bus protocol handling, and distributed priority arbitration.

The Port Adapter (CIPA) Bus - The CI port adapter or CIPA bus, cable part number 17-01027-01, is a control and data bus used for backplane-to-backplane communication between the VAXBI cardcage and the CI cardcage. It is a cable consisting of two 15-foot flat ribbons, separated by a foam dielectric.

LESSON 2 - CIBCI REGISTERS

This section covers the most important registers used by the CIBCI adapter. For a bit discription of each register see the CIBCI Adapter User/Installation Guide.

Figure 4-3 shows the VAXBI interface and adapter registers and Figure 4-4 shows the VAXBI-required registers.

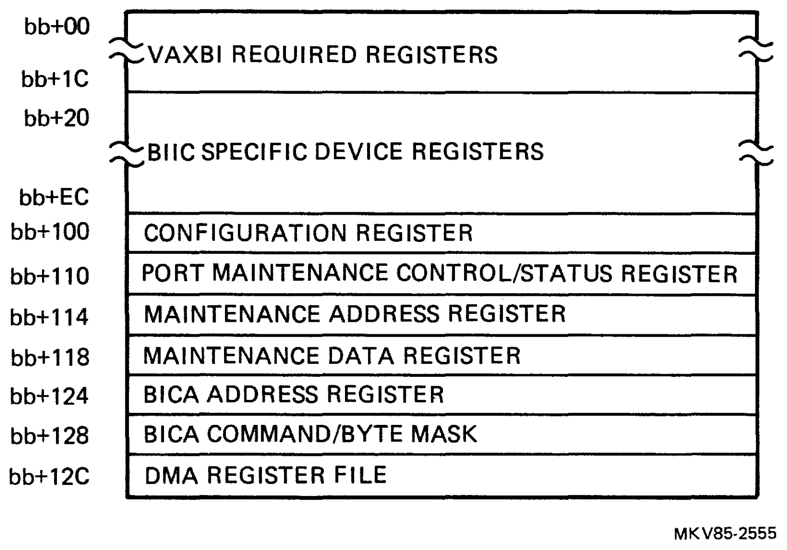


Figure 4-3 VAXBI Interface Registers and Adapter Registers

bb+00	DEVICE TYPE REGISTER
bb+04	VAXBI CONTROL/STATUS REGISTER
bb+08	BUS ERROR REGISTER
bb+0C	ERROR INTERRUPT CONTROL REGISTER
bb+10	INTERRUPT DESTINATION REGISTER
bb+14	INTER-PROCESSOR INTERRUPT MASK REGISTER

MKV85-2556

Figure 4-4 VAXBI Required Registers

LESSON 3 - INSTALLATION

This lesson covers the installation of the CIBCI in the VAXBI cage and overviews what is needed in the CIPA mounting box. For more detailed information on the installation of the CIPA mounting box, see the CIBCI User/Installation Guide.

1. Insert the T1017 and T1018 modules into any two adjacent module slots within the VAXBI cardcage. The T1017 must be positioned to the left of the T1018 module. Refer to Figure 4-5.

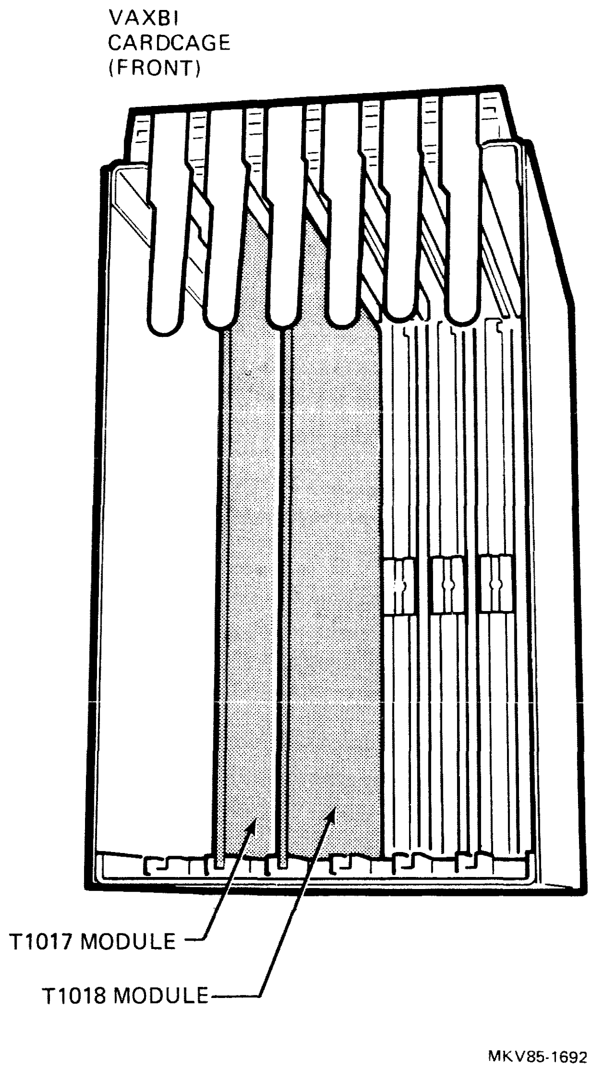


Figure 4-5 VAXBI Backplane - Module Installation

2. Connect the CIBCI cables to the CIBCI. There are four cables that must be attached. Refer to Figure 4-6.

   a. Attach a 3-inch cable at zone C between the innermost connectors of the T1017 and T1018 modules.

   b. Attach a 3-inch cable at zone D between the innermost connectors of the T1017 and T1018 modules.

   c. Attach a 3.75-inch cable at zone C between the outermost connectors of the T1017 and T1018 modules.

   d. Attach a 3.75-inch cable at zone D between the outermost connectors of the T1017 and T1018 modules.

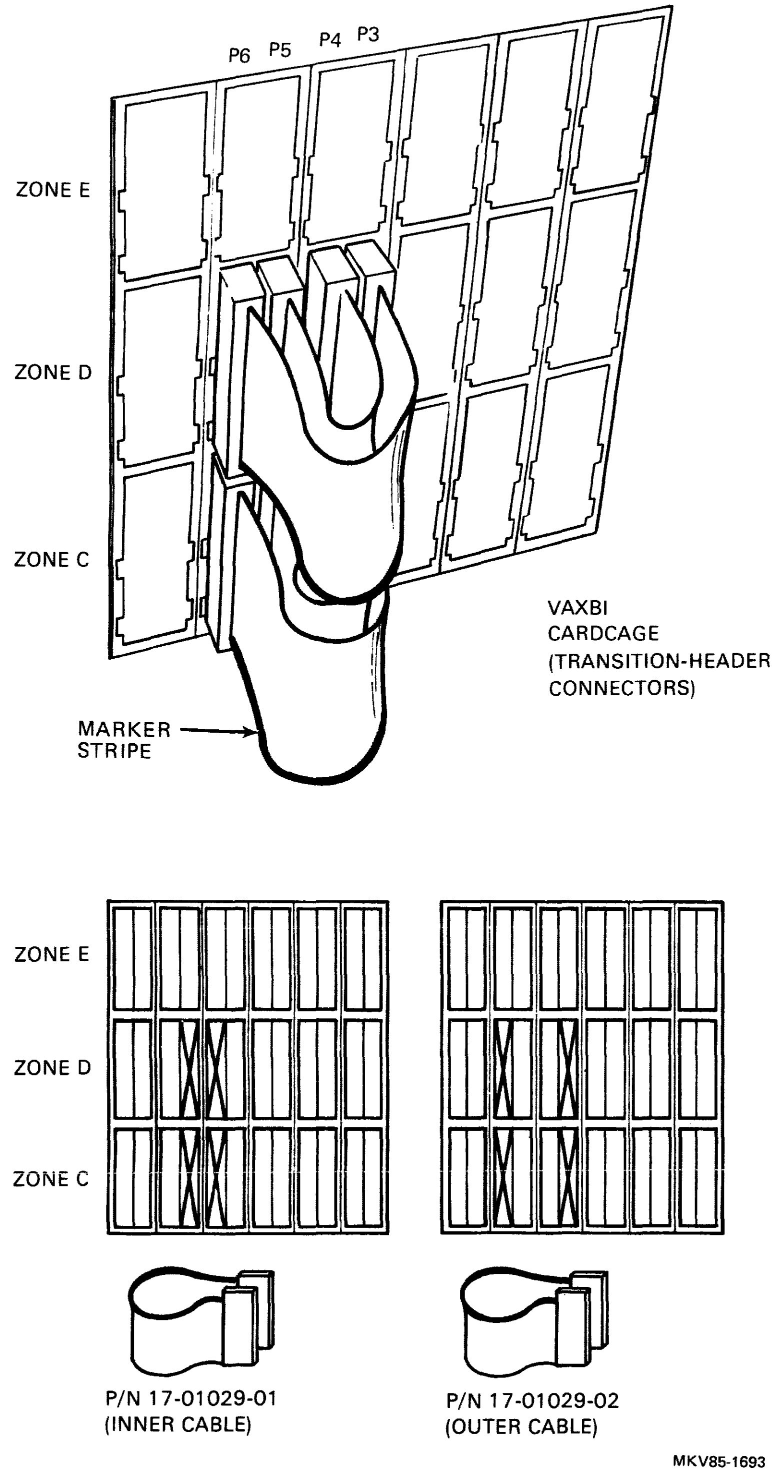


Figure 4-6 VAXBI Backplane - CIBCI Cable Connections

3. Connect the CIPA bus cables to the VAXBI cardcage connectors. There are four cables that must be attached. Refer to Figure 4-7.

   a. Attach connector P6 of the CIPA bus cable to the side 2 connector of the T1017 module at zone E.

   b. Attach connector P5 of the CIPA bus cable to the side 1 connector of the T1017 module at zone E.

   c. Attach connector P4 of the CIPA bus cable to the side 2 connector of the T1018 module at zone E.

   d. Attach connector P3 of the CIPA bus cable to the side 1 connector of the T1018 module at zone E.

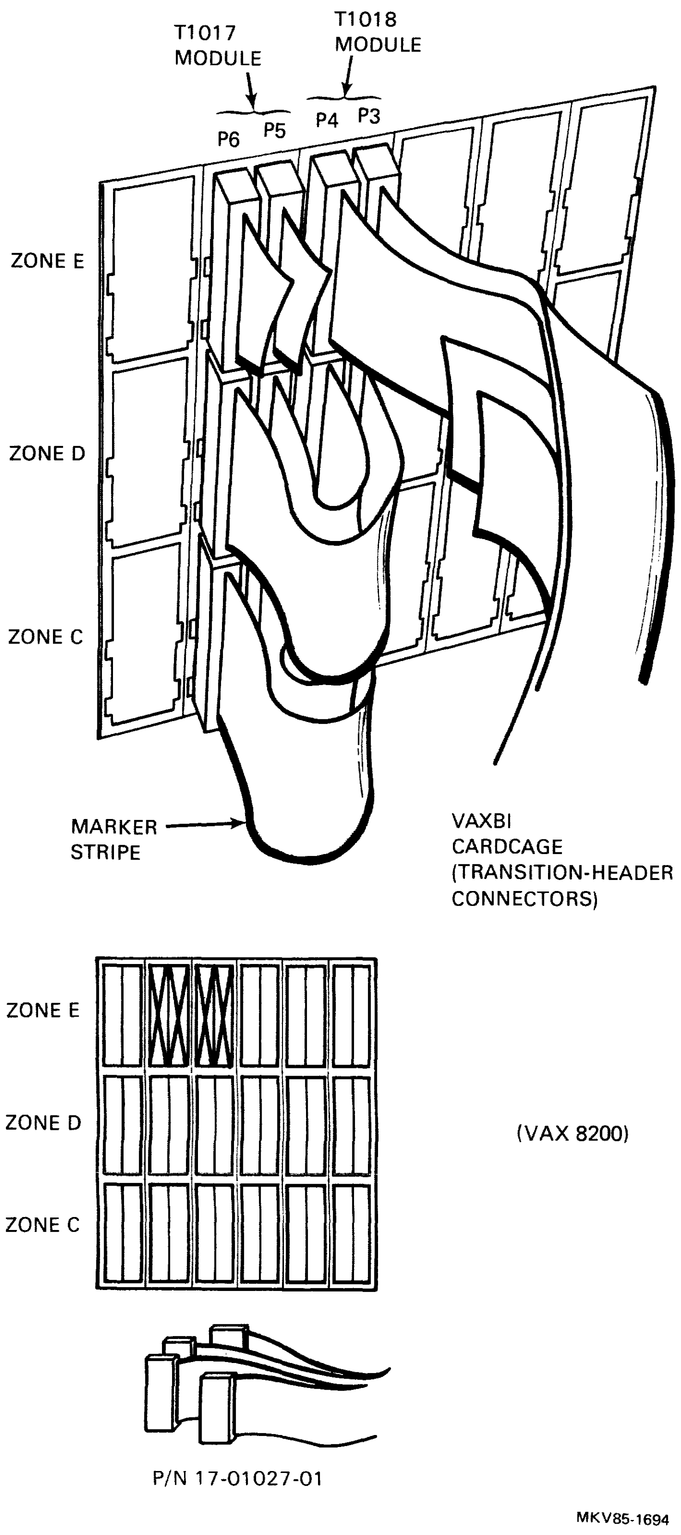


Figure 4-7 VAXBI Backplane - CIPA Bus Connections

LESSON 4 - TROUBLESHOOTING

To determine if the CIBCI adapter hardware is functioning properly, a series of level 3 diagnostic programs must be executed. See Table 4-2 for a list of the CIBCI diagnostic programs. Six of these diagnostic programs are executed with the system operating in a stand-alone environment. This is referred to as repair-level testing. Two additional diagnostic programs are then executed in order to test the functionality of the hardware. This is refered to as functional-level testing. These diagnostic programs, along with their appropriate diagnostic supervisor program, are contained on separate RX50 floppy diskettes.

A minimum of five successful passes of each diagnostic program must be completed to satisfy acceptance testing requirements.

HELP files are available under the diagnostic supervisor for all the diagnostic programs including the supervisor program itself.

Table 4-2 Level 3 CIBCI Diagnostic Programs

Program Designation	Program Title
EVCKA	CIBCI Repair Level Diagnostic 1
EVCKB	CIBCI Repair Level Diagnostic 2
EVCKC	CIBCI Repair Level Diagnostic 3
EVCKD	CIBCI Repair Level Diagnostic 4
EVCKE	CIBCI Repair Level Diagnostic 5
EVCKF	CIBCI Repair Level Diagnostic 6
EVGAA	CI Functional Diagnostic 1
EVGAB	CI Functional Diagnostic 2

Table 4-3 summarizes the diagnostic testing hierarchy used on the CIBCI.

Table 4-3 Summary of the Diagnostic Testing Hierarchy

Diagnostic Category	Diagnostic Program Level	Testing Function
Repair	Level 3	Tests the detailed hardware operation of the CIBCI adapter
Functional	Level 3	Tests the functional hardware operation of the CIBCI adapter
Exerciser	Level 2R	Tests the communications between CI nodes
		Detects a failing CI node
		Verifies repair of a failing CI node

Diagnostic Verification

After running the diagnostic supervisor, attach the CIBCI using the commands as shown below.

This command is used to attach the CIBCI adapter to the VAXBI as node number 6 and as CI node number 0:

```
DS> ATTACH CIBCI HUB PAA0 6 4 0
```

NOTE

This command is used to attach the CIBCI adapter to the VAXBI as node number 6 and as CI node number 0.

The SELECT command is used to select the CIBCI adapter as the unit under test:

```
DS> SELECT PAA0
```

NOTE
This command is used to select the CIBCI adapter as the unit under test.

See the CIBCI Adapter User/Installation Guide for a description of the diagnostic programs for the CIBCI.

LESSON 5 - RELATED DOCUMENTATION

This is a list of available documentation for the CIBCI adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- CIBCI Print Set, MP-01784-01
- CIBCI Hardware Technical Description, EK-CIBCI-TD
- CIBCI Adapter User/Installation Guide, EK-CIBCI-UG
- SC008 Star Coupler User's Guide, EK-SC008-UG
- H7202D Power Supply Specification, SP-H7202-D
- H7202B Power System Technical-Description, EK-PS730-TD

KDB50 ADAPTER

LESSON 1 - INTRODUCTION

The KDB50 is part of the Digital Storage Architecture (DSA) family of intelligent disk controllers. It interfaces any combination of up to four DSA disk drives to a host CPU operating on a VAXBI bus. Since the KDB50 is a DSA controller, the KDB50:

1. Handles most of the I/O management traditionally performed by the host

2. Allows the host to view the disk subsystem as one contiguous string of error-free sectors known as logical blocks

3. Takes over the traditional host concern of disk geometry (cylinder, track, and block)

4. Communicates with the host via the host bus (VAXBI) and mass storage control protocol (MSCP)

5. Communicates with the disk drives via the SDI bus and SDI protocol

The KDB50 communicates with the disk drives using the SDI bus and SDI protocol. It communicates with the host using the VAXBI bus and MSCP. Figure 5-1 shows the basic KDB50 subsystem configuration.

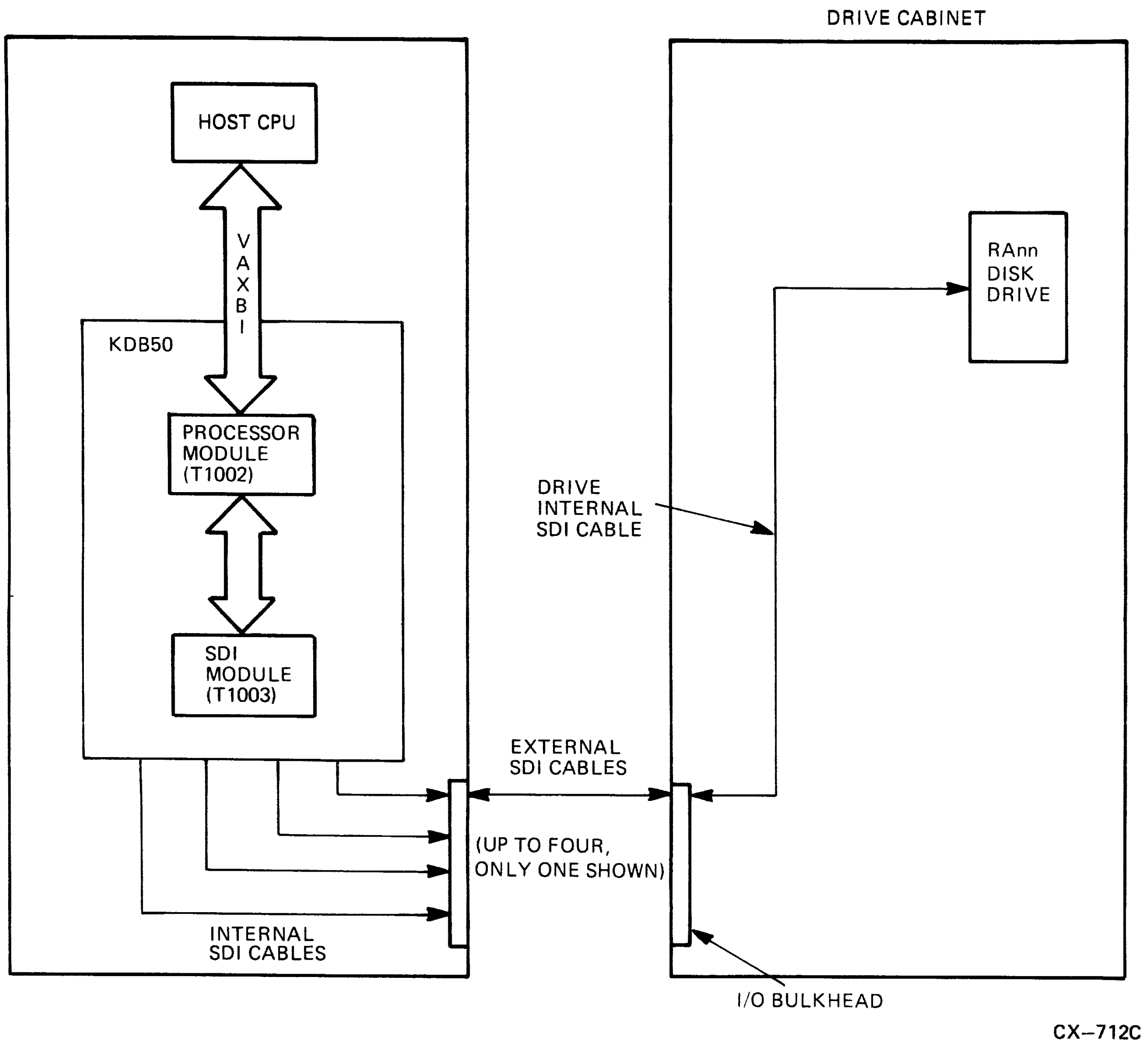


Figure 5-1 KDB50 Subsystem Configuration

Mass Storage Control Protocol

MSCP is a set of rules that defines how DSA hosts and controllers communicate. MSCP communications are packet oriented: that is, the host sends a command packet to the controller which performs the command and then sends a response packet to the host.

One significant advantage of MSCP is that it hides device-dependent requirements, such as disk geometry and error-recovery strategies, from the host. Because these requirements are invisible to the host, one class driver can replace several different device drivers. For example, instead of supplying a device driver for each type of disk drive in a subsystem, MSCP supplies a disk class driver that is able to communicate with all the possible SDI disk drives.

The following description further illustrates the flow of an MSCP command/response sequence:

- To request an I/O operation, the host constructs an MSCP message and sends it to the controller. The MSCP message contains the drive address, the function to be performed, the starting logical block number, and the amount of data requested.
- When the controller receives the request, it independently performs all drive management and data movement, as well as any necessary error recovery. Upon command completion, the subsystem gives status information by sending the host an MSCP response message.

SDI Bus Interface

The SDI bus interface consists of:

1. Cable(s)
2. Communications protocol

A separate SDI cable is supplied for each disk drive, up to a maximum of four disk drives. The cable contains eight wires that carry the four SDI bus signals: two wires for each bus signal. Together, these four signals make up the SDI bus.

KDB50 VAXBI Modules

Two VAXBI modules make up the KDB50: the standard disk interface or SDI module and the processor module. Using circuitry on these two modules as well as microprogrammed control, the KDB50 interfaces with the VAXBI bus and the SDI bus. The following sections describe the hardware on both the SDI and processor modules.

The SDI module is the communication interface between the KDB50 processor module and the disk drives. Some of the characteristics of the SDI module are that it:

- Contains a 16-Kword, high-speed buffer known as the RAM data buffer
- Generates parity for the RAM data buffer
- Detects RAM data buffer parity errors
- Converts data between the parallel format of the KDB50 internal bus DBUS and the serial format of the SDI bus
- Generates the Reed-Solomon error correction code (ECC) for disk write data
- Checks the ECC for disk read data
- Provides the electrical interface to the SDI bus
- Detects pulse errors on the SDI bus

The 16-Kword, high-speed RAM data buffer increases system performance by supplying temporary data storage during data transfers between the disk drive, the KDB50, and the host. Because this buffer can store up to 41 blocks of data, the possibility of a buffer-full condition is reduced. This, in turn, lessens the possibility that data will be missed due to insufficient temporary storage. By reducing the possibility of missed data due to a buffer-full condition, the 16-Kword, high-speed buffer increases system performance. When data is missed, the disk must make another full revolution to retrieve it; this decreases system performance.

The SDI module contains circuitry that generates and checks parity for the RAM data buffer. This ensures that the parity of the data read from the RAM buffer has not changed since it was written. In turn, this ensures that single bit and massive data errors are detected, thus providing data integrity.

Data is converted between DBUS and the SDI bus in the following two stages:

1. Between parallel format and serial non-return-to-zero or NRZ format

2. Between serial NRZ format and serial SDI format

Both the parallel format and the serial NRZ format use TTL levels to represent data. However, parallel format transfers data 16 bits at a time, while serial format transfers data one bit at a time.

The serial SDI format is designed so that any NRZ data pattern can be represented as alternating positive and negative pulses. Because any data on the SDI must consist of alternating pulses, the receiving device (disk drive on one end, KDB50 on the other) detects transmission errors when it observes two adjacent pulses with the same polarity. Figure 5-2 shows the various stages of data conversion between the SDI bus and the DBus. Notice that DBus is common to both the KDB50 SDI module and the KDB50 processor module. For simplicity, it shows only one port to the SDI bus. However, there may be up to four ports.

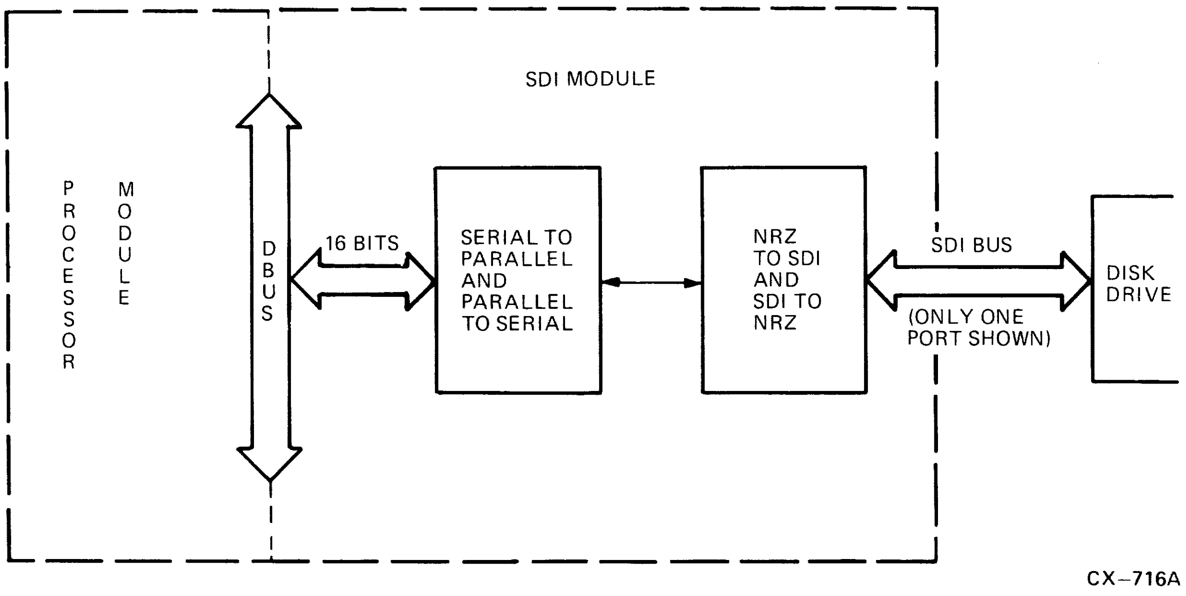


Figure 5-2 SDI-to-DBus <15:00> Datapath

Error correction codes are used to correct errors in data. By performing certain mathematical calculations, they make it possible to correct portions of a data block that have been altered. Altered data occurs when media problems or circuit problems are experienced. The ECC consists of 17 ten-bit symbols representing the 512 data bytes as well as the EDC word contained within the block.

The ECC code is generated and checked by a custom DIGITAL component known as the R-S GEN. This is done for each sector (block) of data sent and received from the disk drive. For example, before data goes to the disk drive, it passes through the R-S GEN. The R-S GEN generates a unique ECC for that block of data and stores it with the block of data. When the data is subsequently read from the disk, it again passes through the R-S GEN. However, this time, the R-S GEN recalculates the ECC and compares it to the ECC read from the disk. The R-S GEN detects an error if the recalculated ECC does not compare with the actual ECC read from the disk. Figure 5-3 illustrates ECC generation and Figure 5-4 illustrates ECC checking.

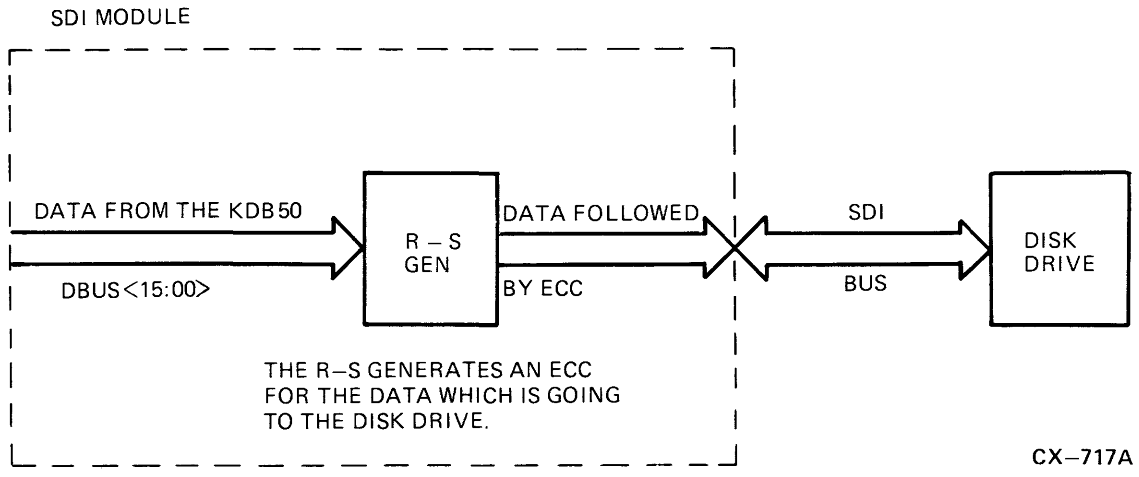


Figure 5-3 ECC Generation

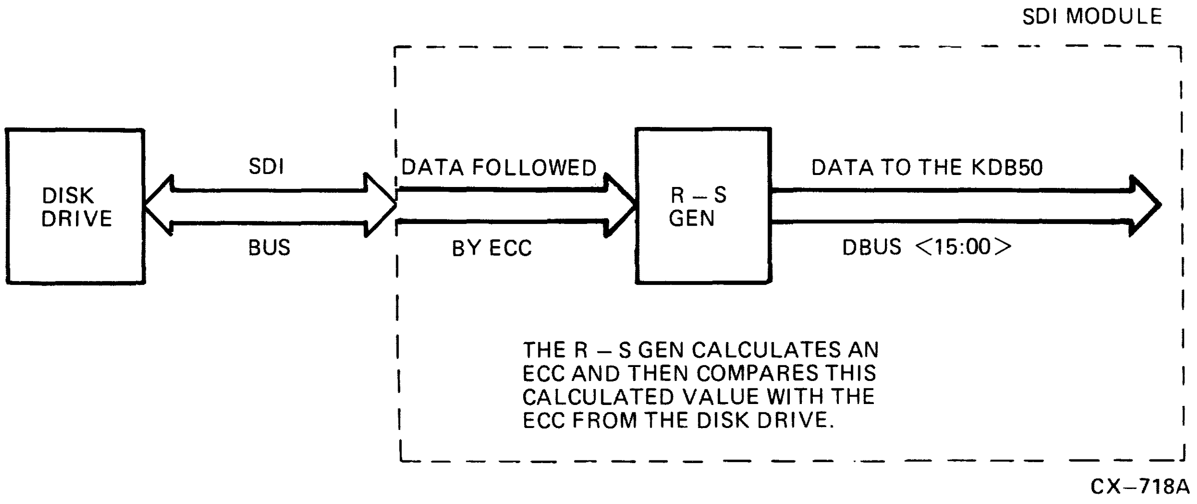

SDI MODULE
DISK DRIVE
SDI
BUS
DATA FOLLOWED
BY ECC
R – S GEN
DATA TO THE KDB50
DBUS <15:00>
THE R – S GEN CALCULATES AN ECC AND THEN COMPARES THIS CALCULATED VALUE WITH THE ECC FROM THE DISK DRIVE.
CX–718A

Figure 5-4 ECC Checking

If the R-S GEN detects an ECC error, it notifies the KDB50 processor module of the error. Using a microcode algorithm, the processor module corrects the data, if possible, before sending it to the host. If the error cannot be corrected, the data is not sent to the host, and the KDB50 notifies the host of an uncorrectable ECC error. The correction capability of this algorithm is up to eight error bursts, each with up to ten bits in error.

Processor Module

The processor module (T1002) is the control portion of the KDB50:

- It performs all KDB50 interaction with the VAXBI bus via two custom DIGITAL chips: the backplane interconnect interface chip (BIIC) and the VAXBI chip interface adapter interface (BCAI).
- It contains a dual microprocessor consisting of two 12-bit sequencers and a shared 16-bit ALU.
- It contains the control store read only memory (CROM) for the microprocessor.
- It provides parity checking for the CROM.
- It preserves parity on data from the RAM to the VAXBI bus and from the VAXBI bus back to the RAM.

The BIIC and the BCAI provide an interface between the VAXBI bus, the BCI bus, and the KDB50 DBUS and control lines. The BIIC provides the interface between the VAXBI bus and the BCI bus, while the BCAI provides the interface between the BCI bus, the KDB50 DBUS and control lines. Figure 5-5 shows the BIIC and BCAI configuration on the KDB50 processor.

The BCAI is a DIGITAL custom IC containing a dual-ported register file that acts as a buffer file between the BIIC and the KDB50 internal data bus. In addition, the BCAI contains a DMA master port that the KDB50 uses when performing DMA operations. The DMA master port consists of two octaword DMA data buffers, a command/address register with increment capability, and a next-page frame register.

The dual 16-bit microprocessor consists of several bit-slice components. By itself, each bit-slice component has a limited function and a narrow data path of four bits. However, when connected in parallel, they create two 12-bit sequencers and a 16-bit ALU. Each sequencer provides a program counter, stack, and stack pointer, while the ALU provides the arithmetic and logic capabilities.

The two sequencers, known as the VAXBI processor and the drive processor, operate alternately and execute individual microprograms from CROM. For example, while the VAXBI processor is using the ALU to perform an operation, the drive processor is fetching its next instruction from the CROM. Similarly, while the drive processor is using the ALU to perform an operation, the VAXBI processor is fetching its next instruction from CROM.

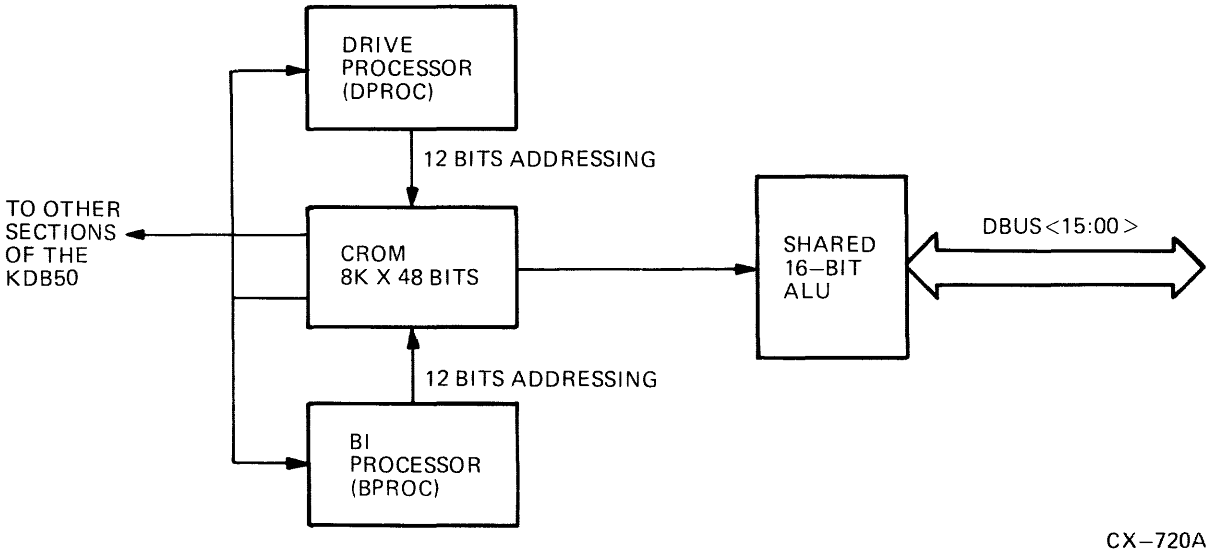


Figure 5-5 KDB50 Dual Microprocessor Scheme

The CROM is an 8K x 48-bit read only memory (ROM) containing microcode for both the VAXBI processor and the drive processor. One half of the CROM is dedicated to the BI processor, and the other half is dedicated to the driver processor.

A CROM parity circuit constantly monitors the CROM data for odd parity. If parity is not odd, the parity circuit assumes a corrupted microword has been accessed and halts all KDB50 operation. The KDB50 then flags the host of an error and stays in a wait state until re-initialized by the host. Before initialization can complete, the KDB50 will examine the entire contents on the CROM for parity errors.

LESSON 2 - KDB50 REGISTERS

The KDB50 uses nine BIIC registers which are divided into three groups: five VAXBI-required registers, two BIIC control registers, and two KDB50- specific registers. The following sections describe these registers as implemented on the KDB50. Refer to your system manual for more detailed information.

The VAXBI-required registers are the minimum set of registers that any node needs to operate on the VAXBI bus. The required registers are:

1. The device register
2. The control and status register
3. The bus error register
4. The error interrupt control register
5. The interrupt control register

The two BIIC control registers give the KDB50 microcode additional control over the BIIC and the ability to issue interrupts without using BCI INT. They are:

1. The BCI control register
2. The user interface interrupt control register

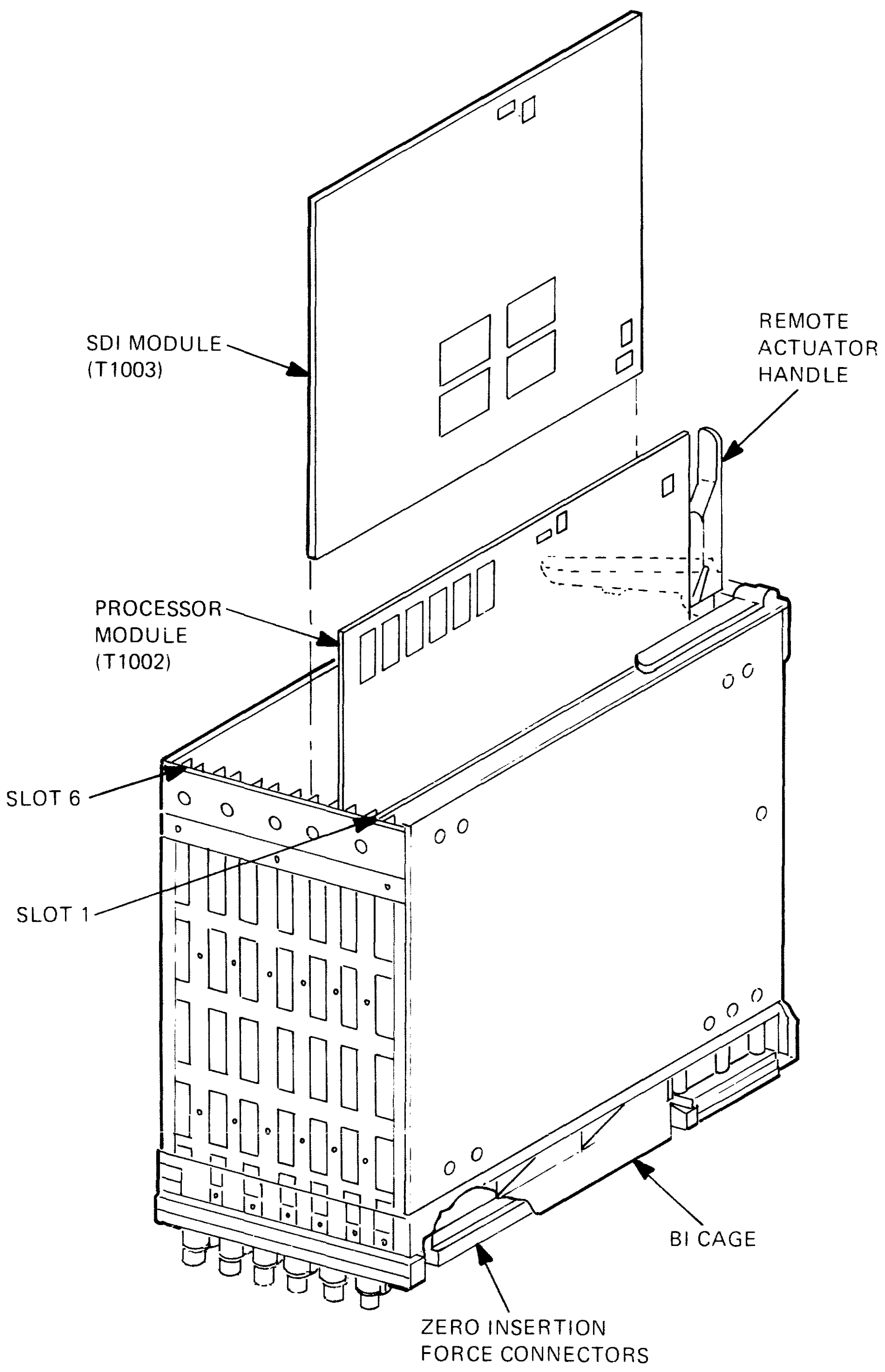
The KDB50-specific registers are two BIIC general-purpose registers that are used to make the KDB50 IP and SA registers. The KDB50-specific registers are:

1. Initializing and polling (IP) register (GPR0)
2. Status and address (SA) read register (lower work of GPR1)
3. Status and address (SA) write register (upper word of GPR1)

LESSON 3 - INSTALLATION

To install the KDB50:

1. Gain access to the BI cage.
2. Insert the two KDB50 modules into adjacent slots in the same BI cage. The KDB50 SDI module should be placed in the higher number slot because of the configuration of the KDB50 backplane I/O assembly. Refer to Figure 5-6.



NOTE: MODULE POSITIONS ARE FOR DEMONSTRATION PURPOSES ONLY.

CX-722A

Figure 5-6 Inserting KDB50 Modules

3. Secure the modules in the BI cage by closing the remote actuator handle for each KDB50 module slot. Ensure that the handle closes all the way, signifying that the modules are properly inserted. The ZIF connectors provide the electrical connection between the KDB50 blackplane I/O assembly and the KDB50 modules.

4. Attach the KDB50 backplane I/O assembly to the backplane of the BI cage using the following procedure. Refer to Figure 5-7.

   a. Gain access to the BI backplane.

   b. Remove the dust covers from the BI backplane.

   c. Attach the keyed KDB50 backplane assembly to the backplane sections that correspond to the KDB50 module slots.

   d. Return the VAXBI cage to its original position.

   e. Refer to your CPU user guide or installation manual to determine the proper internal SDI routing.

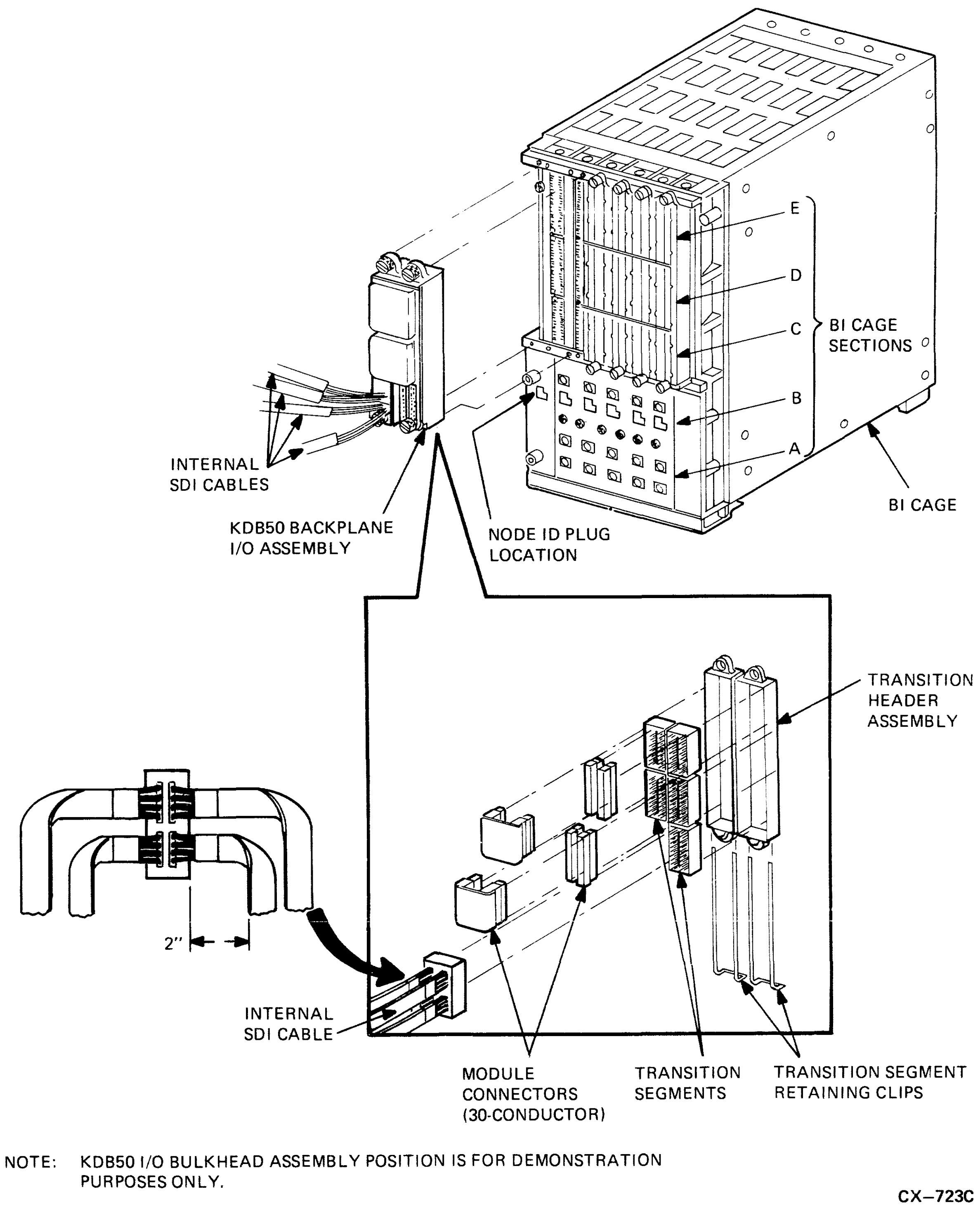

E
D
C
B
A
BI CAGE SECTIONS
BI CAGE
INTERNAL SDI CABLES
KDB50 BACKPLANE I/O ASSEMBLY
NODE ID PLUG LOCATION
TRANSITION HEADER ASSEMBLY
2"
INTERNAL SDI CABLE
MODULE CONNECTORS (30-CONDUCTOR)
TRANSITION SEGMENTS
TRANSITION SEGMENT RETAINING CLIPS
NOTE: KDB50 I/O BULKHEAD ASSEMBLY POSITION IS FOR DEMONSTRATION PURPOSES ONLY.
CX–723C

Figure 5-7 Attaching the KDB50 Backplane I/O Assembly

5. Attach the I/O bulkhead on the VAXBI-based system cabinet using the following procedure:

   a. Bring the internal SDI cables through the opening of one of the available option bulkhead panels of the VAXBI-based system backframe. Refer to Figure 5-8.

   b. Screw the internal SDI cables J1 through J4 into the small holes on either side of each port on the I/O bulkhead. These connections are keyed so they only attach one way.

   c. Mount the I/O bulkhead on the option bulkhead panel.

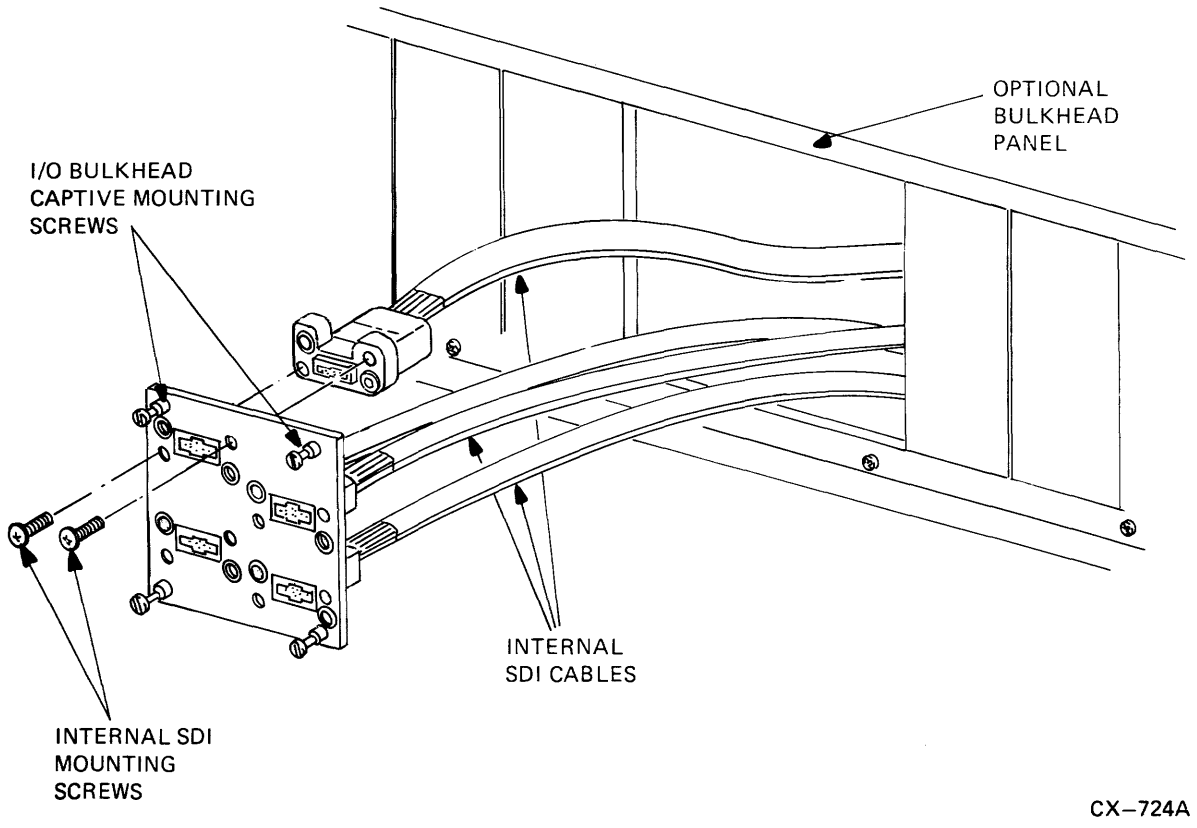

OPTIONAL BULKHEAD PANEL
I/O BULKHEAD CAPTIVE MOUNTING SCREWS
INTERNAL SDI CABLES
INTERNAL SDI MOUNTING SCREWS
CX-724A

Figure 5-8 Internal SDI Cable Installation

6. Attach the keyed, external SDI cables to the VAXBI cabinet I/O bulkhead. Refer to Figure 5-9.

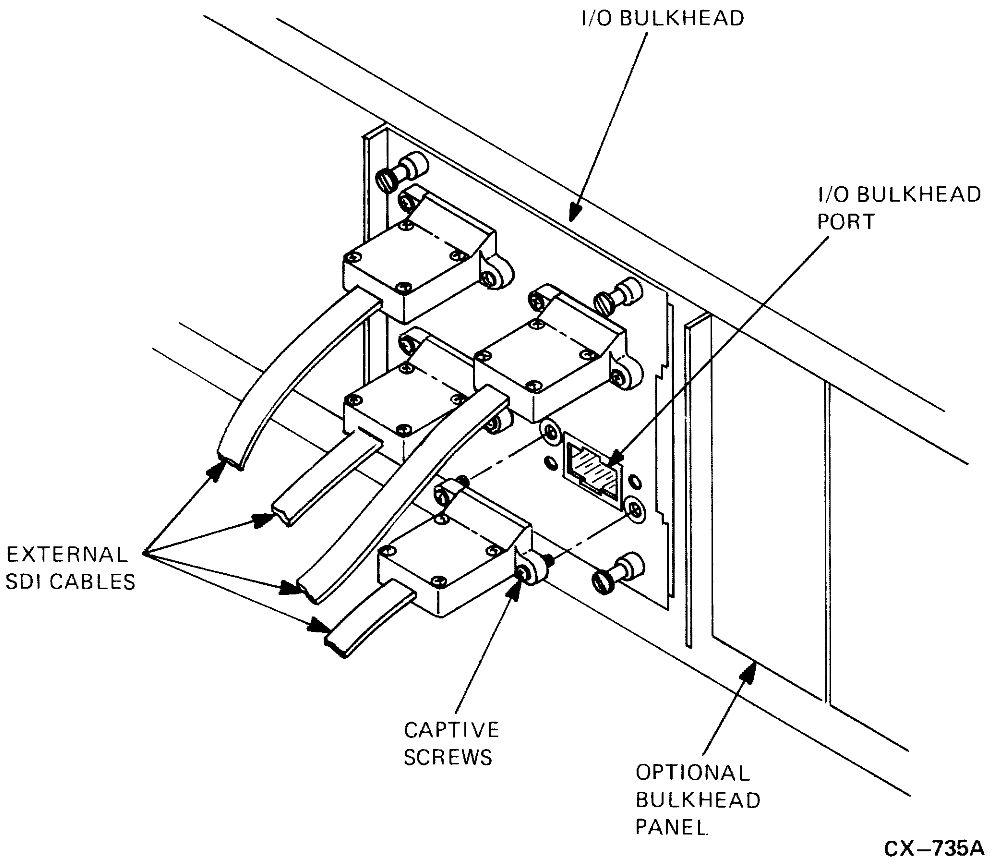


Figure 5-9 External Cable Installation

7. Power up the system. The self-test on the KDB50 nodule set is activated upon power-up. At this time, two single yellow LED indicators show the current state of the KDB50.

   Check that the yellow LEDs on each KDB50 module turn on after about ten seconds.

LESSON 4 - TROUBLESHOOTING

The field acceptance and test procedure for the KDB50 disk subsystem has two parts:

1. The KDB50 disk controller self-test

2. The KDB50 subsystem diagnostics

ROM-based diagnostics run upon power-up or anytime the KDB50 is re-started by the host. These diagnostics test the various circuits on both KDB50 modules to ensure they operate correctly. After the tests are complete, the diagnostics wait for the host to begin initialization which establishes a connection between the host software and the KDB50 microcode.

During the self-test portion of the diagnostic, the LEDs report either an error code, if there is a failure or a cycling pattern. The cycling pattern indicates the self-test portion of the diagnostics has been successfully completed and the KDB50 is waiting for the initialization process to begin.

During the cycling pattern, LEDs begin flashing on the processor module and then progress to the SDI module. The LEDs flash one at a time, starting at the least significant bit and progressing through the most significant bit. The flash goes on and off for approximately one quarter-second and then repeats at about a seven-second rate. However, the pattern is executed so fast that it looks like all LEDs are flashing at the same time. Refer to Table 5-1 for a description of the LED error codes.

Table 5-1 LED Error and Symptom Codes

T1002 LEDs 8 4 2 1	T1003 LEDs 8 4 2 1	Error and Symptoms	Most Likely Failure
		NOTE 1 = LED on, 0 = LED OFF, x = LED may be ON or OFF	
0 0 0 1	x x x x	Hex 1; undefined	Undefined
0 0 1 0	0 0 0 0	Hex 2; microcode stuck in init step 2	T1002 or software
0 0 1 1	0 0 0 0	Hex 3; microcode stuck in init step 3	T1002 of software
0 1 0 0	0 0 0 0	Hex 4; microcode stuck in init step 4 or VAXBI bus timeout error	T1002 or host inactive
0 1 0 BLINK	0 0 0 0	Hex 4/5; test complete Normal display for operating KDB50	No problem
0 1 1 0 x x x x	x x x x 0 1 1 0	Hex 6; undefined	Undefined
0 1 1 1 x x x x	x x x x 0 1 1 1	Hex 7; undefined	Undefined
1 0 0 0	0 0 0 0	Hex 8; wrap bit 14 set in SA register	T1002 or software
1 0 0 1 0 0 0 0	0 0 0 0 1 0 0 1	Hex 9; board error	T1002
1 0 1 0 1 0 1 0	0 0 0 0 1 0 1 0	Hex A; board two error	T1003
1 0 1 1 x x x x	x x x x 1 0 1 1	Hex B; undefined	Undefined
x x x x 1 1 0 0	1 1 0 0 x x x x	Hex C; Timeout error check error code in SA register. Refer to KDB50 Service Manual	Many causes
1 1 0 1 x x x x	x x x x 1 1 0 1	Hex D; RAM parity error	T1003

Table 5-1 LED Error and Symptom Codes (Cont)

T1002 LEDs 8 4 2 1	T1003 LEDs 8 4 2 1	Error and Symptoms	Most Likely Failure
1 1 1 0 x x x x	x x x x 1 1 1 0	Hex E; ROM parity error	T1002
1 1 1 1	1 1 1 1	Hex F; sequencer error	T1002
Cycling pattern	Cycling pattern	None	No problem
		If the cycling pattern continues beyond the start of the initialization process. The KDB50 is not responding to the host CPU	T1002

KDB50 Subsystem Diagnostics

The KDB50 subsystem diagnostics consist of several different tests designed to check out different portions of the KDB50 subsystem. If the diagnostic program reports errors, refer to the KDB50 Disk Controller Service Manual, EK-KDB50-SU.

Because these diagnostics are subject to future revisions, always follow the program directions that are displayed on your terminal. The following section provides a brief description of each diagnostic.

When EVRLB is run, it erases customer data on the disk. EVRLB should only be used after it has been established that a hardware problem has created false entries in the replacement control table or RCT.

EVRLB when run will:

1. Clear the RCT
2. Move the factory bad block control table or FCT to the RCT
3. Write and verify each sector and replace, or revector, as necessary

EVRLF consists of the following three tests:

1. Test number 1 is the VAXBI bus addressing test.

   This test checks to make sure that each addressing line can be driven to both one and zero (that the KDB50 addresses unique locations as each line is toggled). The test then does large transfers to and from memory with known data patterns.

2. Test number 2 is the disk-resident diagnostic test.

   This test issues a DIAGNOSE command to each disk drive selected for testing. Each drive should accept the DIAGNOSE command and respond. Failure status and all data from the disk drive will be sent to the host for error reporting. An interactive dialogue allows an operator to manually select any other diagnostic option available in the disk drive.

3. Test number 3 is the disk function test.

   This test checks out all disk functions as defined by the SDI. The test performs such functions as DRIVE CLEAR, READ, WRITE, SEEK, FORMAT, and RECALIBRATE. All writing and formatting is performed on the diagnostic cylinders of the disk drives. An extensive test of the positioning capability is performed before attempting any format operation. Positional verification is implemented before any formatting is allowed.

The EVRLG KDB50 disk exerciser test, tests the disk drives selected by issuing random seeks and reads/write transfers. Parameters can be specified to use diagnostic or host cylinders, restrict testing to specific areas of the disk, control length to test, and do error logging.

EVRLG duplicates all the firmware error recovery mechanisms, thereby thoroughly checking out the KDB50 firmware.

Before running the KDB50 subsystem diagnostics, the diagnostic supervisor must be booted.

A sample procedure is shown in Example 5-1. Refer also to Table 5-2.

```
>>>B CSA1
```

If you need assistance in this task, refer to the appropriate system manual.

The following procedure describes how to load the KDB50 subsystem diagnostics:

```
DS>LOAD EVRNN
DS>AT
DEVICE TYPE?  KDB50
DEVICE LINK?  HUB
DEVICE NAME?  DUA (or DUB)
```

Example 5-1 Loading KDB50 Susbsystem Diagnostics

NOTE

The presence of B at the third character position in this string indicates the existence of a second UDA50/KDB50-Q/KDB50 controller.

NODE ID?--- The node ID is the value on the node ID plug for the processor module. The response range is 0 to 15.

BR LEVEL?--- For the BR LEVEL type in 5, this is the VAXBI interrupt level assigned to the KDB50.

Table 5-2 Sample KDB50 Subsystem Diagnostic Sequences

Sequence for any fixed media disk drive

```
DE>AT
DEVICE TYPE? RANN
DEVICE LINK? DUA (or DUB)
DEVICE NAME? DUAN (or DUBN)
DS>
```

Sequence for an RA60

```
DS>AT
DEVICE TYPE? RA60
DEVICE LINK? DUA (or DUB)
DEVICE NAME? DJA0 (or DJBO)
```

Sequence for an RA81

```
 DS>AT
 DEVICE TYPE? RA81
 DEVICE LINK? DUA (or DUB)
 DEVICE NAME? DUA1 (or DUB1)
DS>
```

NOTE

Notice the J instead of the U. This is how VMS identifies the drive as removable.

```
DS>SELECT DUA,DUA0,DJA0
DS>RUN (or START) EVRNN
```

The KDB50 has the ability to return information to the operating system for inclusion into an error log. These entries may include specific information on the operation of the KDB50, its attached drives or other parts of the system which may be important in diagnosing problem sources.

Some reports contained in the error log, however may represent changes in the configuration or operation of your system that ore only informational in nature and do not represent the occurrence of an actual error condition. Examples of this may be:

1. Completion of the initialization sequence between the port driver and the KDB50

2. Attention messages pertaining to the availability of a disk drive which may be the result of changing a drive unit number

For further directions, refer to the appropriate system documentation.

Drive Numbering

DSA/SDI drives that can be connected to the KDB50 can usually be given a unit number ranging from 0 to 254. Consult operating system documentation for accepted drive numbers. Consult drive documentation for unit number programming instructions.

The unit numbers assigned to drives attached to a KDB50 do not imply any priority or other special property: all drives are treated equally by the KDB50. The only requirement is to avoid duplicating drive unit numbers. If these numbers are duplicated, the KDB50 will not permit a drive to be accessed. This situation may be corrected by assigning unique drive numbers.

Unit numbers can be easily changed at the drive, although it is recommended that the drive be taken off line to the KDB50.

Figure 5-10 is a flow diagram for troubleshooting the KDB50.

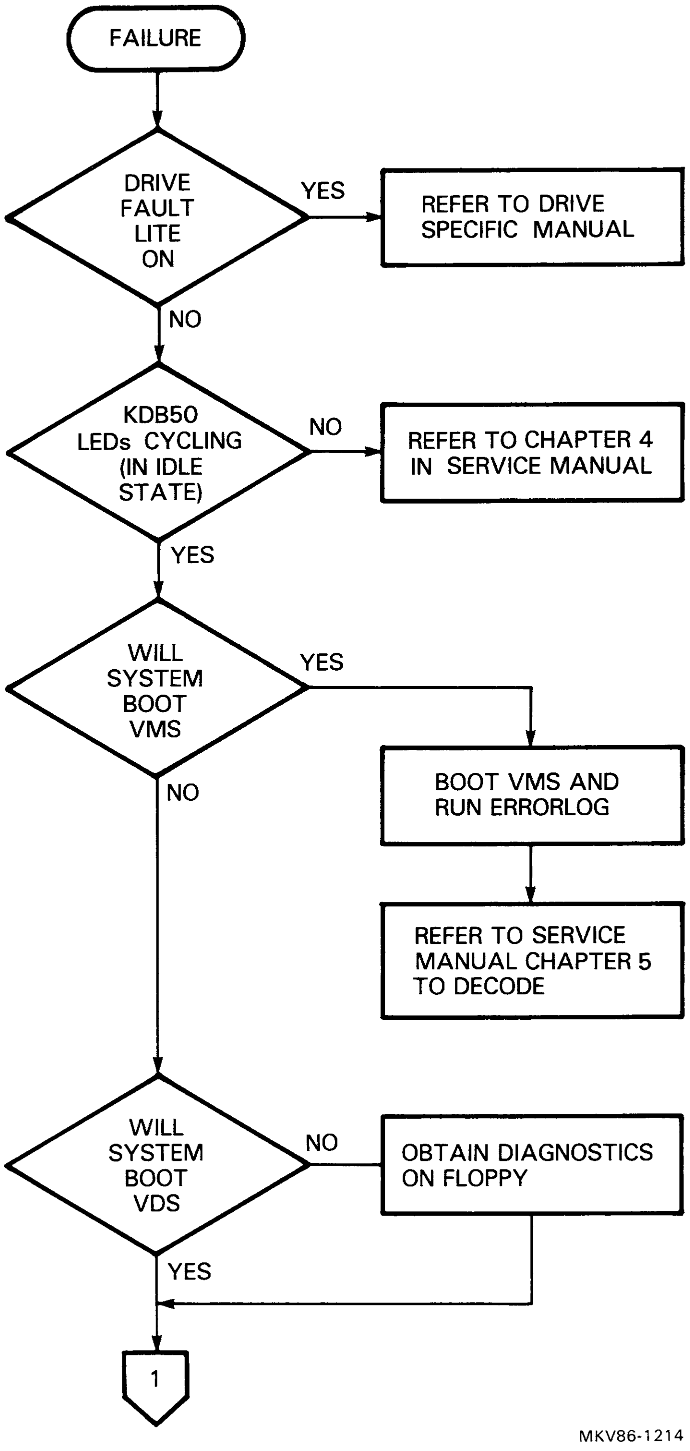

FAILURE
DRIVE FAULT LITE ON
YES
REFER TO DRIVE SPECIFIC MANUAL
NO
KDB50 LEDs CYCLING (IN IDLE STATE)
NO
REFER TO CHAPTER 4 IN SERVICE MANUAL
YES
WILL SYSTEM BOOT VMS
YES
BOOT VMS AND RUN ERRORLOG
REFER TO SERVICE MANUAL CHAPTER 5 TO DECODE
NO
WILL SYSTEM BOOT VDS
NO
OBTAIN DIAGNOSTICS ON FLOPPY
YES
1
MKV86-1214

Figure 5-10 Troubleshooting Flowchart (Sheet 1 of 2)

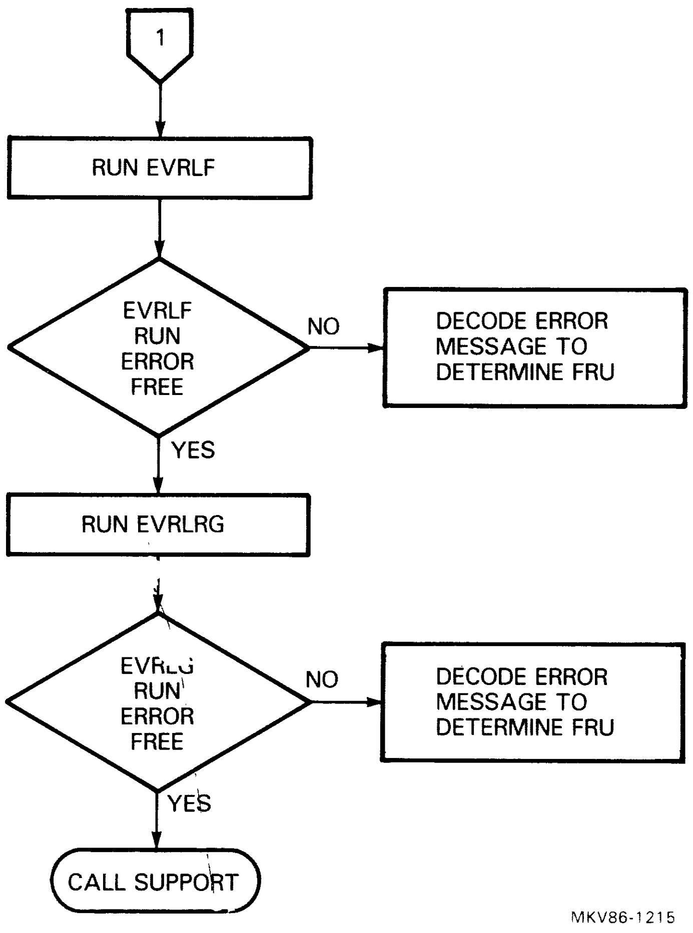


Figure 5-10 Troubleshooting Flowchart (Sheet 2 of 2)

LESSON 5 - RELATED DOCUMENTATION

This is a list of available documentation for the KDB50 adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- KDB50 Disk Controller User Guide, EK-KDB50-UG
- KDB50 Disk Controller Service Manual, EK-KDB50-SV
- KDB50 Disk Controller Technical Description, DK-KDB50-TD

CIBCA ADAPTER

LESSON 1 - INTRODUCTION

The computer interconnect port adapter, or CIBCA, connects a host system to a VAXcluster. The CIBCA is centrally controlled by a single, on-board data processor. This processor provides buffered parallel-to-serial communication between two corporate interconnect bus architecture protocols: the host processor's VAXBI bus and the dual path Computer Interconnect, or CI bus. Figure 6-1 illustrates how the CIBCA fits into a system.

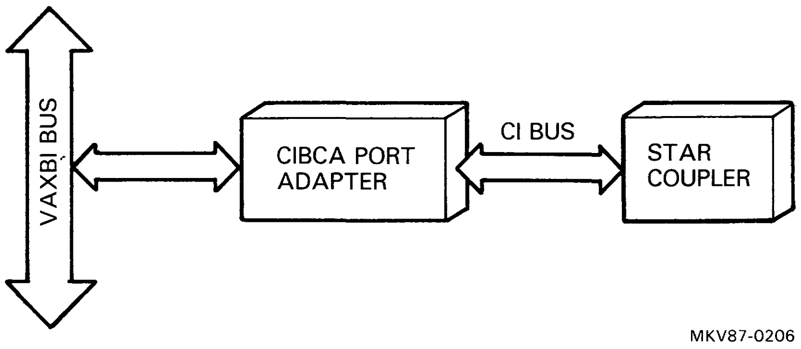


Figure 6-1 Simplified CIBCA Port Adapter Connection

As a buffered communications port, the CIBCA adapter completes high level computer communications. This reduces software processing overhead. The hardware provides all the necessary data buffering, address translation, and serial data encoding/decoding. The CIBCA uses queue structures provided by the VAX/VMS operating system to transfer packet messages and initiate the transfer of blocks of data between the host memory system and other nodes in the VAXcluster.

The CIBCA is partitioned into two separate hardware interfaces: a host processor interface and a computer interconnect interface.

The port controller module provides the host processor interface, and contains the VAXBI provides, the VAXBI control logic, a microprocessor, and the control store.

The link/packet buffer module provides the computer interconnect interface by means of the CI bus protocol logic and the CI packet buffer memory.

The main features of the CIBCA are:

- VAX backplane interconnect design
- Diagnostic data loopback (internal/external) capability
- Data integrity by means of cyclic redundancy checking
- Round-robin arbitration on the VAXBI
- Round-robin arbitration on the CI during light loading
- Contention arbitration on the CI during heavy loading
- Packet-oriented data transmission
- Power-up self-test
- Operational modes:
  - Disable
  - Enable
  - Uninitialized

General Specifications

The following list shows the general specifications for the CIBCA.

CI arbitration	
Light loading	Round-robin
Heavy loading	Contention
Parity	Cyclic redundancy check
Data format	Manchester-encoded serial packet
Operational modes	Disabled Enabled Uninitialized

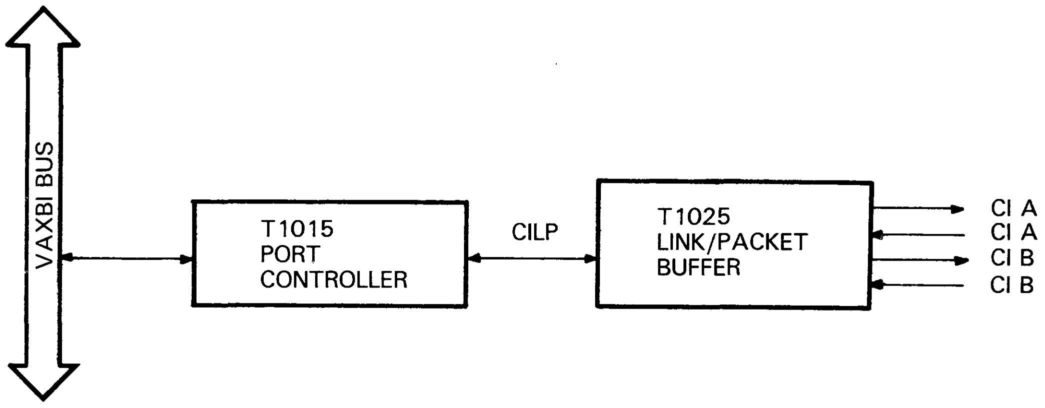


Figure 6-2 Simplified CIBCA Block Diagram

Figure 6-2 shows a simplified block diagram of the CIBCA. Notice that the CIBCA consists of two T-series modules. The two modules are housed in two option slots within the VAXBI cardcage of the host system. These modules are used to interface the host system's VAXBI bus to the CI bus.

The first of these two modules is the port controller. The port controller monitors the VAXBI for operations addressed to the CIBCA, and generates needed VAXBI transactions.

The second module is the link/packet buffer module. This module uses the packet buffers to interface with the CI bus and contains the protocol logic to control the flow of information through the buffers.

Two cables (part numbers PN 17-01504-01 and PN 17-01504-02) are used to electrically interconnect the two modules.

LESSON 2 — REGISTERS

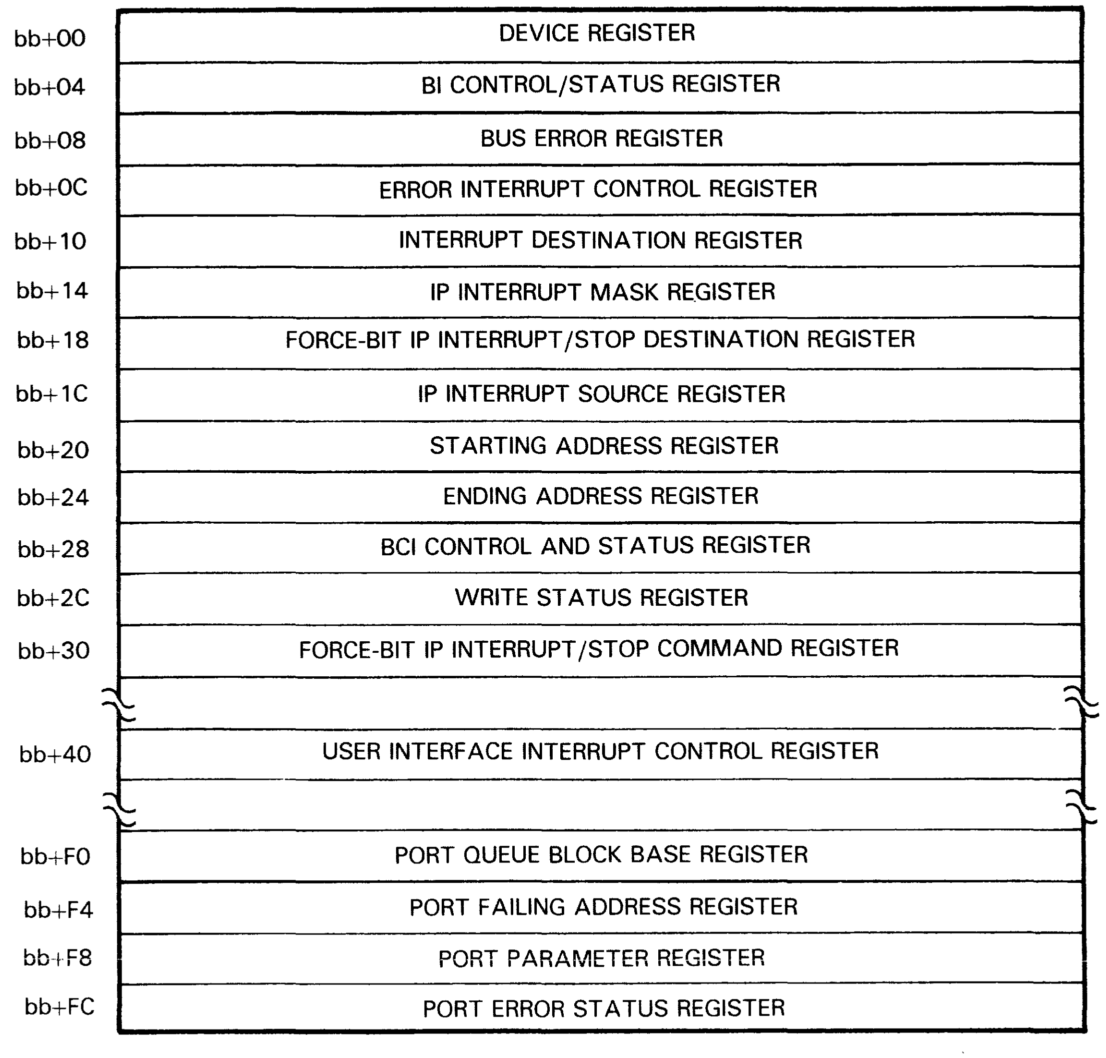

bb+00 DEVICE REGISTER
bb+04 BI CONTROL/STATUS REGISTER
bb+08 BUS ERROR REGISTER
bb+0C ERROR INTERRUPT CONTROL REGISTER
bb+10 INTERRUPT DESTINATION REGISTER
bb+14 IP INTERRUPT MASK REGISTER
bb+18 FORCE-BIT IP INTERRUPT/STOP DESTINATION REGISTER
bb+1C IP INTERRUPT SOURCE REGISTER
bb+20 STARTING ADDRESS REGISTER
bb+24 ENDING ADDRESS REGISTER
bb+28 BCI CONTROL AND STATUS REGISTER
bb+2C WRITE STATUS REGISTER
bb+30 FORCE-BIT IP INTERRUPT/STOP COMMAND REGISTER
bb+40 USER INTERFACE INTERRUPT CONTROL REGISTER
bb+F0 PORT QUEUE BLOCK BASE REGISTER
bb+F4 PORT FAILING ADDRESS REGISTER
bb+F8 PORT PARAMETER REGISTER
bb+FC PORT ERROR STATUS REGISTER
MKV87-0203

Figure 6-3 VAXBI Node ID Address Space from 0 to FC

The node ID space for the CIBCA is shown in Figures 6-3 and 6-4. Notice that the general purpose registers (address offsets F0 to FC) for this adapter have been renamed.

bb+1000	PORT STATUS REGISTER
bb+1004	PORT MAINTENANCE CONTROL/STATUS REGISTER
bb+1008	MAINTENANCE ADDRESS REGISTER
bb+100C	MAINTENANCE DATA REGISTER
bb+1010	PORT COMMAND QUEUE 0 CONTROL REGISTER
bb+1014	PORT COMMAND QUEUE 1 CONTROL REGISTER
bb+1018	PORT COMMAND QUEUE 2 CONTROL REGISTER
bb+101C	PORT COMMAND QUEUE 3 CONTROL REGISTER
bb+1020	PORT STATUS RELEASE CONTROL REGISTER
bb+1024	PORT ENABLE CONTROL REGISTER
bb+1028	PORT DISABLE CONTROL REGISTER
bb+102C	PORT INITIALIZE CONTROL REGISTER
bb+1030	PORT DATAGRAM FREE QUEUE CONTROL REGISTER
bb+1034	PORT MESSAGE FREE QUEUE CONTROL REGISTER
bb+1038	PORT MAINTENANCE TIMER CONTROL REGISTER
bb+103C	PORT MAINTENANCE TIMER EXPIRATION CONTROL REGISTER
bb+1040	RESERVED REGISTER 3
bb+1044	RESERVED REGISTER 4
bb+1048	RESERVED REGISTER 1
bb+104C	RESERVED REGISTER 2
bb+1050 bb+1BFC	LOCAL STORE
bb+1C00 bb+1FFC	VIRTUAL CIRCUIT DESCRIPTOR TABLE

MKV87-0204

Figure 6-4 VAXBI Node ID Address Space from 1000 to 1FFC

For a complete description of the bits in each of these registers, consult the CIBCA Adapter User/Installation Guide.

LESSON 3 — INSTALLATION

Insert the T1015 and the T1025 modules into any two adjacent module slots in the VAXBI cardcage, as shown in Figure 6-5. The T1015 must be placed in the lower numbered slot.

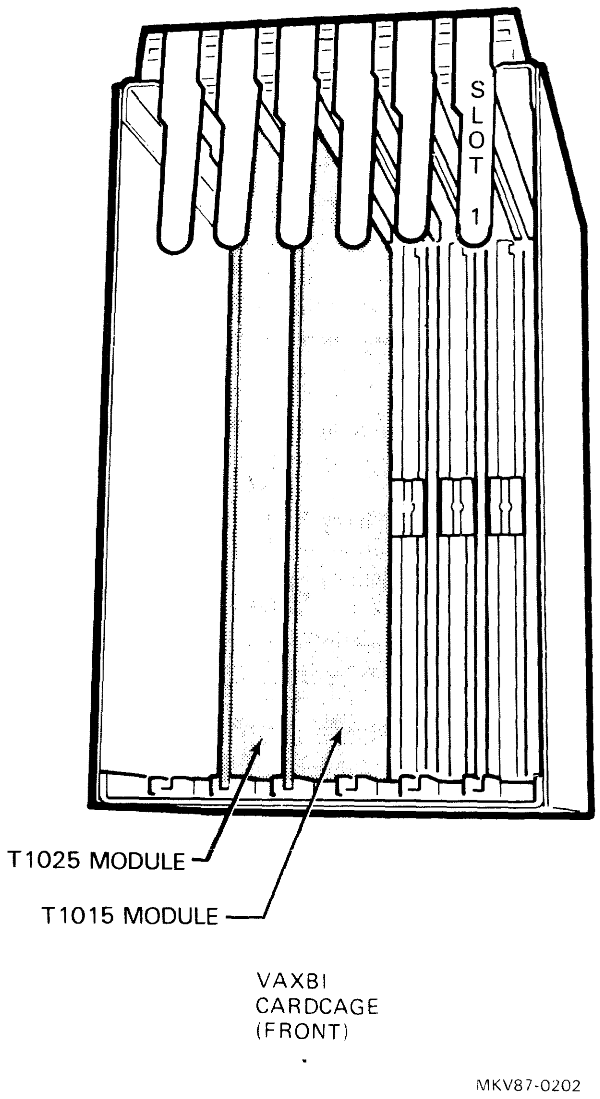


Figure 6-5 VAXBI Backplane - Module Installation

Connect the modules with the two BCI cables. (See Figure 6-6.)

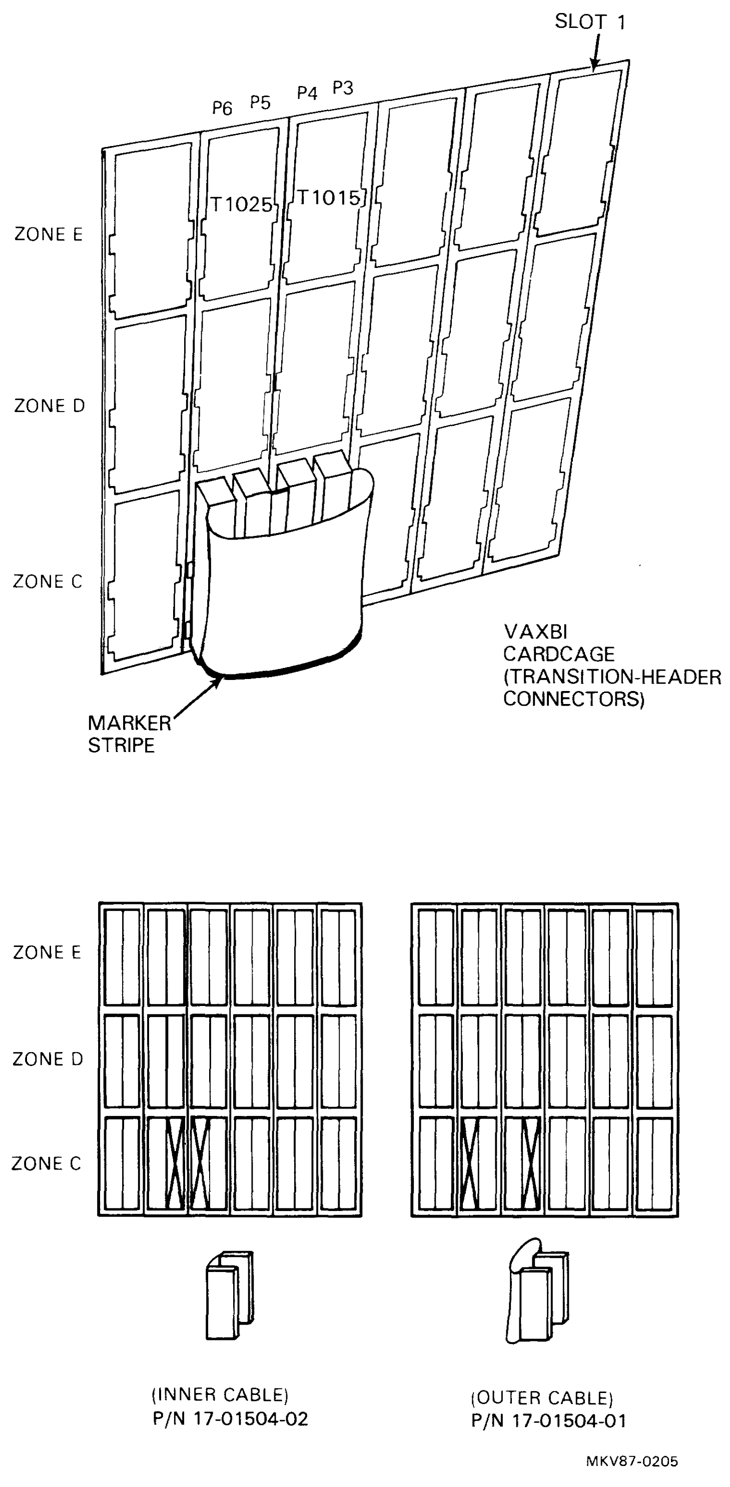


Figure 6-6 VAXBI Backplane - BCI Cable Connection

Install a transmit dummy connector in zone D of the slot containing the T1025 module and a receive dummy connector in zone E of the same slot.

Install the transmit CI cables on the other side of zone D of the slot with the T1025 module and the receive CI cables on the other side of zone E of the same slot.

LESSON 4 — TROUBLESHOOTING

To determine whether the CIBCA adapter hardware is functioning properly, a series of seven Level 3 diagnostic programs must be executed. Level 3 diagnostics are executed in a standalone environment, that is, without VMS. Of the seven diagnostics, five are executed with the system not connected to a VAX cluster. The other two are executed with the system connected to the coupler. This is done to test the functional CIBCA hardware operation within the system. Such testing is referred to as "functional level" testing. These diagnostic programs, along with their appropriate diagnostic supervisor program, are contained on separate RX50 diskettes.

At least five successful passes of each diagnostic program must be completed to satisfy acceptance testing requirements.

Diagnostics

HELP files are available under the diagnostic supervisor for all of the diagnostic programs, including the supervisor program itself.

Table 6-1 CIBCA Level 3 Diagnostic Programs

Program Designation	Program Title
EVGCA	CIBCA T1015 Repair Level Diagnostic 1
EVGCB	CIBCA T1015 Repair Level Diagnostic 2
EVGCC	CIBCA T1015 Repair Level Diagnostic 3
EVGCD	CIBCA T1015 Repair Level Diagnostic 4
EVGCE	CIBCA T1025 Repair Level Diagnostic 5
EVGAA	CI Functional Diagnostic 1
EVGAB	CI Functional Diagnostic 2

Table 6-2 shows a summary of the diagnostic testing hierarchy used on the CIBCA.

Table 6-2 Summary of the Diagnostic Testing Hierarchy

Diagnostic Category	Diagnostic Program Level	Testing Function
Repair	Level 3	Tests the detailed hardware operation of the CIBCA adapter
Functional	Level 3	Tests the functional hardware operation of the CIBCA adapter
Exerciser	Level 2R	Tests the communications between CI nodes
		Detects a failing CI node
		Verifies repair of a failing CI node

After running the diagnostic supervisor, attach the CIBCA by using the command shown below.

```
DS> ATTACH CIBCA HUB PAA0 6 4 0
```

NOTE

This command is used to attach the CIBCA adapter to the VAXBI as node number 6 and as CI node number 0.

The SELECT command is used to select the CIBCA adapter as the unit under test.

```
DS> SELECT PAA0
```

NOTE

This command is used to select the CIBCA adapter as the unit under test.

For a description of the diagnostic programs for the CIBCA, consult the CIBCA Adapter User/Installation Guide.

LESSON 5 - RELATED DOCUMENTATION

Below is a list of available documentation for the CIBCA adapter. You may find it helpful to refer to these sources as questions arise in working with this adapter.

- CIBCA Print Set, MP-01836-01
- CIBCA Adapter User/Installation Guide, EK-CIBCA-UG
- CIBCA Hardware Technical Description, EK-CIBCA-TD
- SC008 Star Coupler User's Guide, EK-SC008-UG
- H7202D Power Supply Specification, SP-H7202-D
- H7202B Power System Technical Description, EK-PS730-TD

Digital Equipment Corporation • Bedford, MA 01730