DEC MicroServer
Manufacturing and Repair Manual

DIGITAL INTERNAL USE ONLY

Digital Equipment Corporation, Maynard, Massachusetts

First printing, January 1989

CONTENTS

CHAPTER 7 Synchronous I/O ports

PART III FIRMWARE

CHAPTER 11 Overview of the Firmware

CHAPTER 12 On-Board Test

PART IV TROUBLESHOOTING

CHAPTER 17 Circuit Board Troubleshooting

INDEX

FIGURES

TABLES

Preface

This book is for engineers and technicians who test, troubleshoot, and repair DEC MicroServer hardware. The book may also be useful to specialists and service engineers who give technical support to customers.

NOTE

This book is a DIGITAL internal document and contains propriatry information. DO NOT use it with, or show it to customers.

Document Structure

The book has five parts:

I **Introduction** -- is an overview of what the DEC MicroServer is, what it's used for, and its major components.

II **Functional and Circuit Description** -- is a detailed description of the hardware.

III **Firmware** -- is an overview of the On-Board Tests (OBT) and other firmware in the DEC MicroServer.

IV **Troubleshooting** -- provides help on how to trace faults on the circuit boards and in the power supply.

V **Appendixes** -- is a collection of reference information backing up the main body of the book. This part also contains a list of other books that have more information on certain topics.

At the back of the book there is an index, and some pages for your own notes.

You can use the notes pages to write any information you may want to add to the book (such as new troubleshooting hints). Since the pages are loose leaf, you can move the notes into the main body of the book.

How To Use the Book

This book is intended as a reference source, not as a tutorial on the DEC MicroServer hardware. You can, however, use the book as a backup to a detailed technical course.

When using this book, you'll find it helpful to have the DEC MicroServer Print Set and the Error Document available. Appendix I gives the order numbers for these documents.

Many of the devices mentioned in this book are multi-purpose. This book concentrates on the features of each device that the DEC MicroServer uses. For further information on any device, refer to the sources listed in Appendix I.

Number Bases

All addresses and offsets are in hexadecimal. All other numbers are in decimal.

PART I

INTRODUCTION

Chapter 1
The DEC MicroServer and its Uses

1.1 What is the DEC MicroServer?

A **server** is a dedicated node on a Local Area Network (or LAN), such as an Ethernet, that provides some form of service for all other LAN nodes. The DEC MicroServer is a **communications server** that gives access to synchronous communications services. Figure 1-1 shows an example of a DEC MicroServer in a LAN.

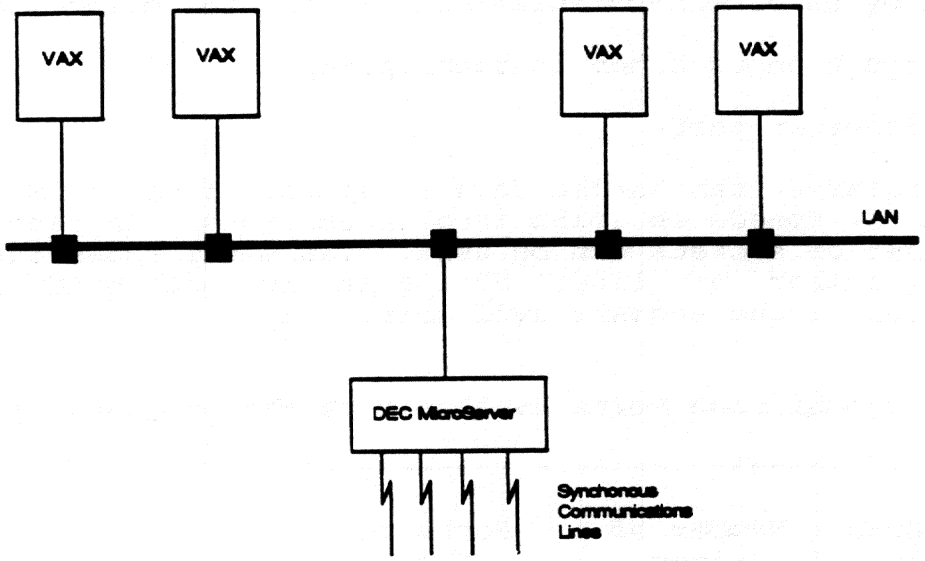


Figure 1-1: Example of a LAN Containing a DEC MicroServer

Advantages of using the DEC MicroServer rather than providing individual links to other nodes are:

- Nodes no longer have the overhead of managing the communications links -- this task is taken over by the DEC MicroServer
- The number of links required can be minimized
- The links can be shared among all the LAN nodes

- The use of the links can be maximized

1.2 General Features

The DEC MicroServer is based on MicroVAX chip technology and has:

- MicroVAX processor
- 1M byte of code/data RAM
- 1M byte of I/O buffer space
- Up to 256K bytes of firmware ROM
- 8K bytes of Non-volatile RAM (NVRAM) for error logging
- 4 synchronous communications ports
- 1 Ethernet port

The DEC MicroServer can handle data at speeds of up to 2M bits/s. This range of speeds is split into three bands. In each band a certain number of sockets can be used. Table 1-1 shows the three bands, the number of lines available in each band, and the identification of the sockets available. [1]

Table 1-1: Synchronous Ports Available at Various Data Speeds

Maximum Data Speed (in bits/s)	Number of Lines Supported	Ports to Use
0 to 64K	4	0, 1, 2, 3
above 64K up to 256K	2	0 and 1
above 256K up to 2M	1	0 only

[1] These are raw data speeds that include protocol and user information. The amount of user data will be less and depends on the protocol overhead.

So, there is a trade-off between the line data speed and the number of lines.

For more detailed specifications of the DEC MicroServer, see Appendix D.

1.3 Applications

The DEC MicroServer can provide access to a wide range of communications services, depending on the software it is running. Typically, though, it is used to give access to a wide area network, as Figure 1-2 shows.

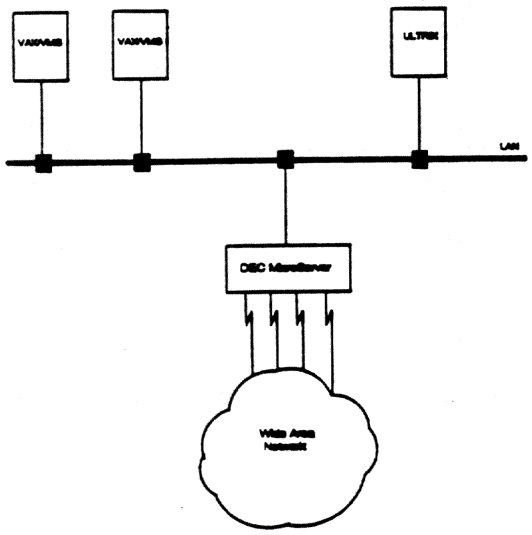


Figure 1-2: The DEC MicroServer Often Provides Access to a Wide Area Network

Some examples of the access that a DEC MicroServer can provide are:

1. **DECnet/SNA Gateway** -- provides communication between nodes in a DECnet network and those in an IBM SNA network.

2. **X.25 Router** -- provides access to a PSI network such as PSS, TELENET, or TYMENET.

3. **DECnet Router** -- provides a number of widely separated LANs to be connected as a single DECnet network. The Router manages the flow of information through the

network.

1.4 Operation

To give the customer as much flexibility as possible, the DEC MicroServer's software runs in RAM rather than being encoded in ROM. This means that the software has to be loaded from somewhere. The DEC MicroServer has no mass storage of its own, and so it depends on a host node elsewhere on the LAN to hold the software.

On power up, the DEC MicroServer broadcasts a message on the LAN asking for its software to be loaded. All other nodes on the LAN examine their DECnet databases to see if they are to act as a load host for the DEC MicroServer. Any that are load hosts can send a reply indicating that they can load the unit.

In many cases, only one host replies (in fact, most installations have only one load host), but if two or more were to reply the DEC MicroServer would choose the first to answer.

Once a host is selected, the DEC MicroServer begins to receive the software from the host. Typically, the host will hold the software on one of its disks and sends it (together with any configuration data) to the DEC MicroServer over the LAN.

Once the load is complete, the DEC MicroServer can start executing the software.

1.5 The Rest of this Book

This chapter has provided some background information on the DEC MicroServer and the tasks it performs. The remainder of this book provides a detailed description of the unit and its circuitry. The next chapter describes the major physical components of the DEC MicroServer. Then there's a detailed functional description followed by a description of the firmware. Appendixes at the back of the book give you reference information.

Chapter 2
Major Components and Specifications

2.1 Overview

This chapter covers the general construction of the DEC MicroServer:

- Size, shape, and construction of the box and casing
- Layout of items on the back panel
- Construction of the communications sockets
- Layout of the circuit boards
- General construction of the power supply

This chapter does not give details of how the DEC MicroServer is built from its components. This information is kept by the manufacturing plants.

2.2 Box and Casing

The DEC MicroServer uses the standard corporate box as its housing (Figure 2-2 shows the box dimensions). The box itself is enclosed in plastic casing that allows the unit to be free standing. The DEC MicroServer can also be mounted in a rack, by removing the plastic casing and attaching a pair of rack mounting brackets.

Figure 2-1 shows the construction of the enclosure.

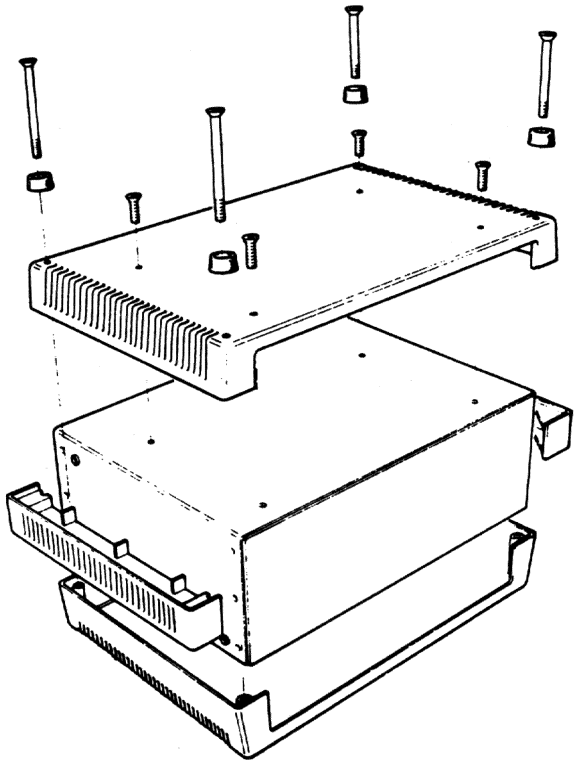


Figure 2-1: The DEC MicroServer Box and Casing

Figure 2-2 shows the dimensions of the box with and without the plastic enclosure.

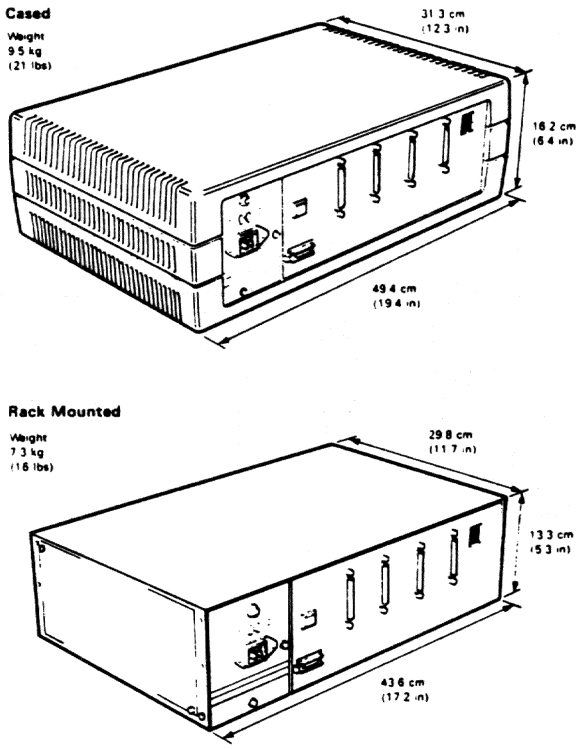


Figure 2-2: The Dimensions of the DEC MicroServer

The box contains the circuit boards, power supply, panel display, and communications sockets. All the sockets and the panel display are on the back panel.

The following sections give more information on the construction of the major elements.

2.3 Back Panel

Figure 2-3 shows the layout of the DEC MicroServer's back panel.

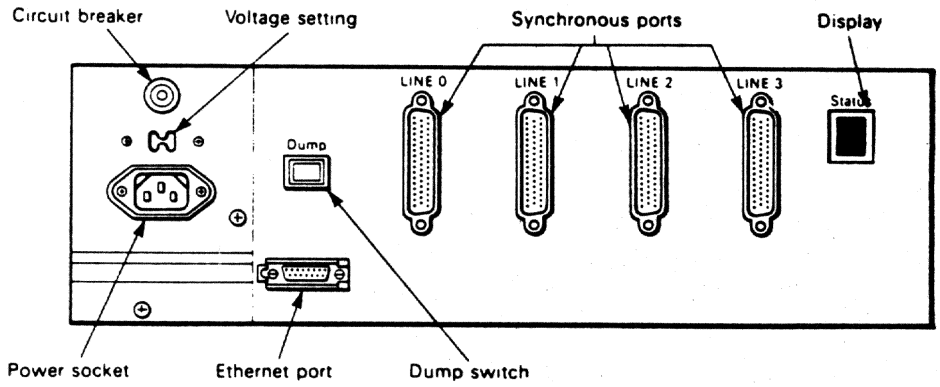


Figure 2-3: Layout of the Back Panel

2.4 Communications Sockets

As Figure 2-3 shows, the DEC MicroServer has 5 communications sockets:

- 1 Ethernet connection
- 4 Synchronous communications connections

The synchronous sockets are mounted on the chassis and attached to a distribution panel. The Ethernet connection is also mounted on the chassis, but does not use a distribution panel.

2.4.1 Ethernet Socket

The Ethernet socket is a 15-way D-type which accepts a transceiver cable. If the unit is to be connected to a LAN conforming to the IEEE 802.3 standard, the transceiver is assumed to handle the conversion between that standard and Ethernet.

2.4.2 Synchronous Connections

Physically, the four synchronous connections are identical. Functionally, the difference is in the speeds that each can operate, as Section 1.2 shows.

Figure 2-4 shows the allocation of the pins on the synchronous sockets.

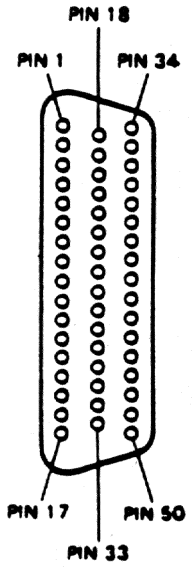


Figure 2-4: Pin allocation of the Synchronous Communications Sockets

2.5 Circuit Boards

The DEC MicroServer contains four circuit boards:

- Main logic module
- Distribution panel
- Synchronous support logic
- Power supply

Sections 2.5.1 to 2.5.3 give more information on the main logic module, the distribution panel, and the synchronous support logic

boards. Section 2.6 provides information on the power supply.

2.5.1 Main Logic Module

The main logic module is a 26.4 cm by 38.6 cm (10.4 in. by 16 in.) board that contains most of the DEC MicroServer's logic. Figure 2-5 shows the layout of this module, and highlights the position of some of the major components.

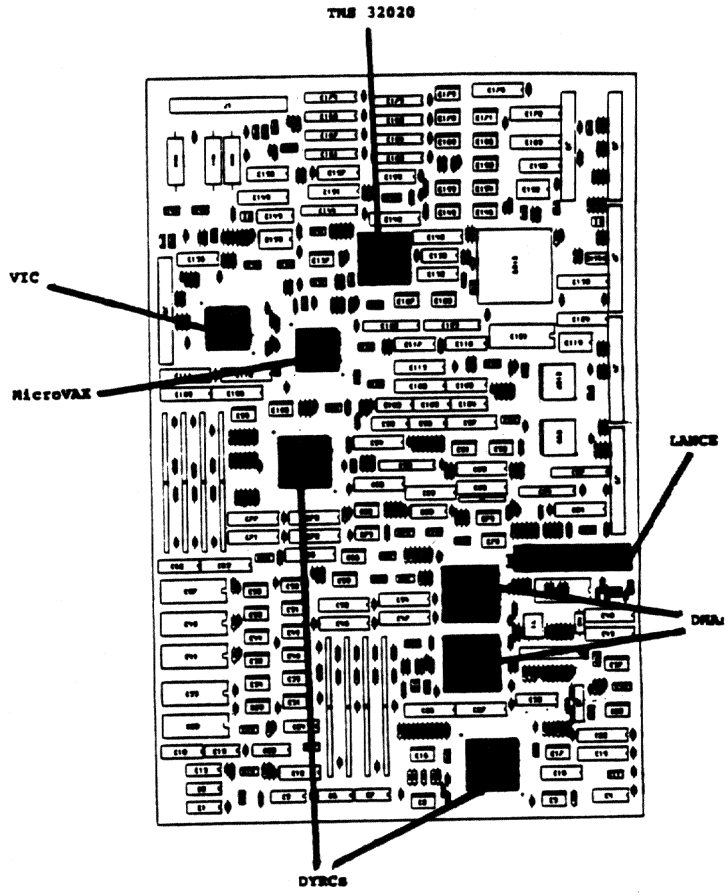


Figure 2-5: Layout of the Main Circuit Board

The board is supported by 8 standoffs (3 on each shorter edge of the board, and 1 on each longer edge) and by 6 rubber bumpers, as Figure 2-6 shows. (Note that the standoffs and bumpers are not shown to scale.)

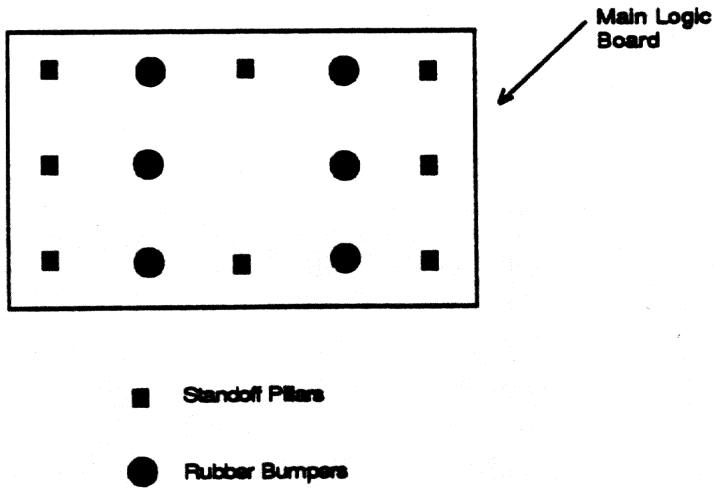


Figure 2-6: Standoffs and Bumpers Supporting the Main Logic Module

2.5.2 Distribution Panel

The distribution panel is a 26.4 cm by 9.7 cm (10.4 in. by 4 in.) board that contains:

- The 50-way synchronous connectors
- The panel display
- The level conversion components for the transmit and receive circuits on all synchronous ports

Figure 2-7 shows the layout of this board.

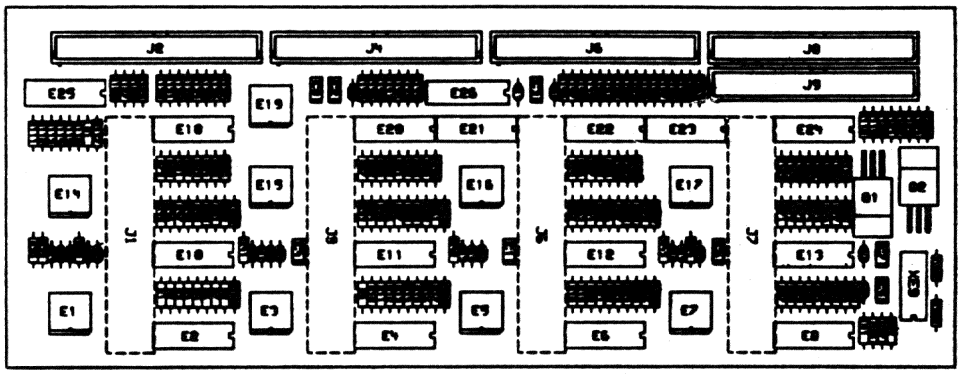


Figure 2-7: Layout of the Distribution Panel

The board is mounted on the chassis behind the back panel, and is held in place by the fixing screws on the 50-way connectors.

2.5.3 Synchronous Support Logic (SSL)

The synchronous support logic is on a 26.4 cm by 19.3 cm (10.4 in. by 8 in.) board and consists of:

- Bit rate generators
- Receive FIFOs
- Transmit synchronizers
- Multiplexing and loopback logic

Figure 2-8 shows the layout of this board.

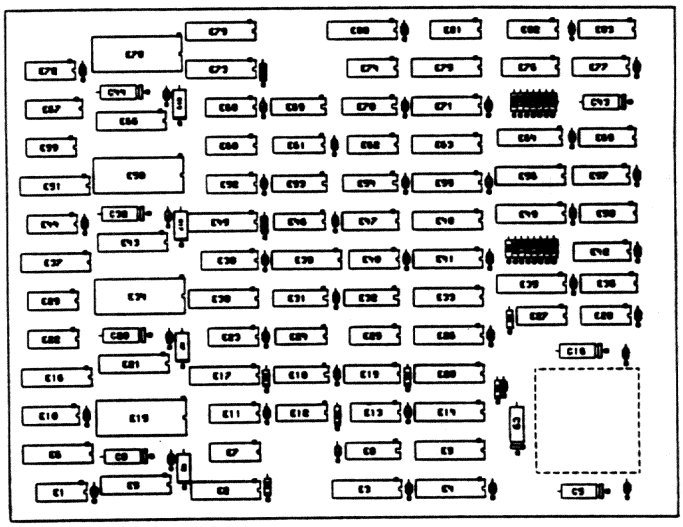


Figure 2-8: Layout of the Synchronous Support Logic

The board is mounted above the main logic module. Five pillars keep the board separate from the main logic module, as Figure 2-9 shows (note that the standoffs are not to scale). Electrical connection is through a 145-pin socket, with an extra pin

providing orientation.

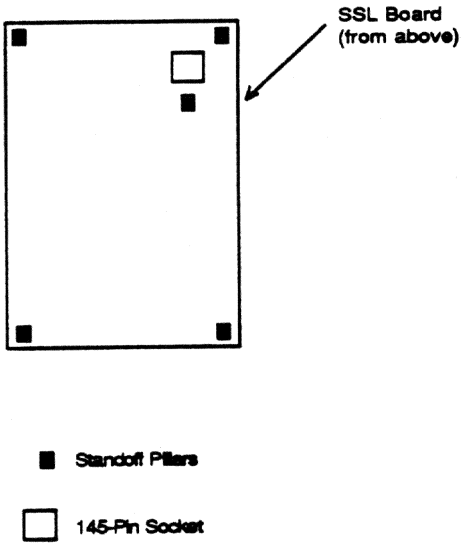


Figure 2-9: Standoffs and 145-Pin Socket Supporting the Synchronous Support Logic Board

2.6 Power Supply

The DEC MicroServer draws power from the local AC mains. An internal power transformer (part number: 54-15086-02) provides the DC voltages listed in Table 2-1.

Table 2-1: DC Voltages

Voltage*	Typical Current	Maximum Current
+5 V	11 A	16 A
+12 V	1.9 A	3 A
-12 V	0.5 A	1 A

*All DC voltages have a tolerance of +/- 5%

The table also shows the current supplied on each voltage rail. The middle column lists the current typically drawn by the DEC MicroServer when in use. This figure includes 1 A at 12 V for the Ethernet connector. The right hand column lists the maximum current the power supply can deliver for each voltage.

The power supply is a 208 W unit that can operate from 120 V or 240 V AC mains at either 50 Hz or 60 Hz. Table 2-2 lists the amount of current used at these AC voltages.

Table 2-2: AC Power Consumption

AC Voltage	Typical Current
120 V	1.25 A
240 V	0.60 A

The DC power is passed to the main logic board through a 15-way wiring harness.

PART II

FUNCTIONAL AND CIRCUIT DESCRIPTION

Chapter 3
System Architecture

3.1 Overview

The following sections explain the architecture of the DEC MicroServer, showing:

- The functional blocks that make up the system
- The way those blocks are connected together
- How the blocks communicate with one another
- How the system operates as a whole

Succeeding chapters give more details of the content and function of each block.

3.2 General Architecture

At the most basic level, the DEC MicroServer has two parts, connected as shown in Figure 3-1.

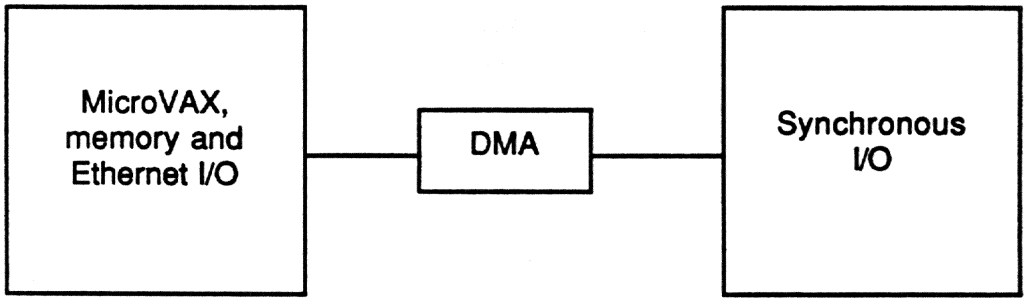


Figure 3-1: The Major Divisions of the DEC MicroServer Architecture

The DMA devices connect the parts together allowing them to communicate and work in parallel. In addition, they allow the MicroVAX processor to control the synchronous ports.

Sections 3.2.1 and 3.2.2 show each side in more detail. Section 3.2.3 gives some information on the DMA devices, and Section 3.2.4 gives a complete picture of the architecture.

3.2.1 MicroVAX Side

Figure 3-2 shows the major blocks on the MicroVAX side of the architecture. The figure also shows how the blocks are connected together.

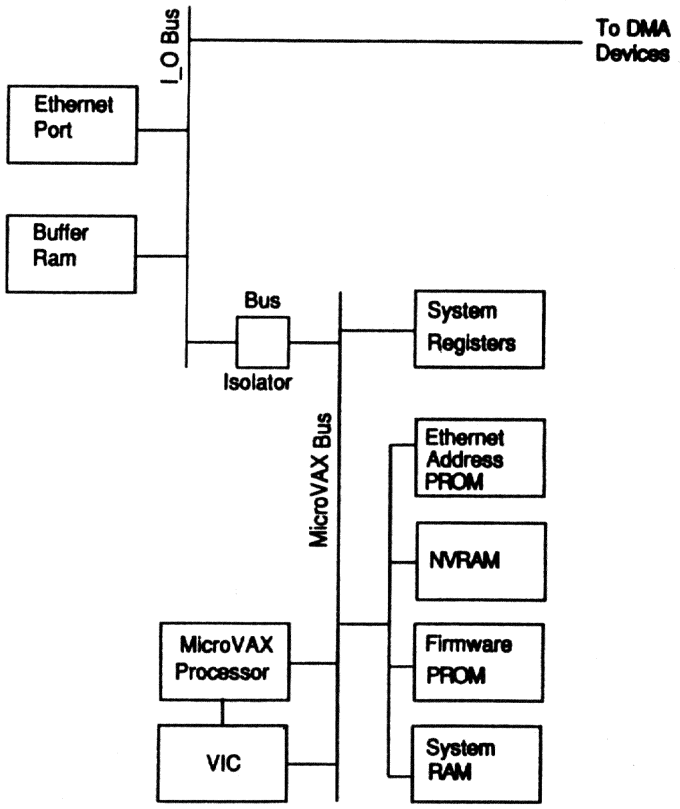


Figure 3-2: Architecture of the MicroVAX Processor, Memory and Ethernet Interface

Notice that there are two major buses:

- MicroVAX bus
- I_O bus

Both are 32-bit DAL buses, and the following sections show what's in each.

3.2.1.1 **MicroVAX Bus** - The major blocks on this bus are:

- **MicroVAX Processor** and **Vectored Interrupt Controller (VIC)** -- providing overall control of the system
- **System RAM** -- 1M byte of storage holding the system's operational software
- **Firmware PROM** -- holding the system's firmware and on-board test
- **Non-Volatile RAM (NVRAM)** -- holding permanent system parameters and error information
- **System Registers** -- providing overall control/status information (includes the seven-segment display on the unit's rear panel).

3.2.1.2 **I_O Bus** - The major blocks on this bus are:

- **Ethernet Interface** -- providing access to the Ethernet and the host computers
- **Buffer RAM** -- 1M byte of storage that holds data received from, or about to be sent to, one of the 5 communication ports

3.2.1.3 **Why Two Buses?** - Both are basically MOS buses, and so have limited drive capabilities. This makes it difficult (if not impossible) to connect all the major blocks to a single bus. However, there are also performance benefits in using two buses.

The two buses can work independantly, which maximizes bus bandwidth. For example, the MicroVAX can execute code at the same time as data from a synchronous port is being put in the Buffer RAM through the DMA devices.

The only time that the buses need to synchronize (and act as a single bus) is when the MicroVAX processor needs to access devices on the I_O bus (to control the Ethernet Interface, for example).

3.2.2 Synchronous I/O Side

Figure 3-3 shows the major blocks on the Synchronous I/O side of the architecture. The figure also shows how the blocks are connected together.

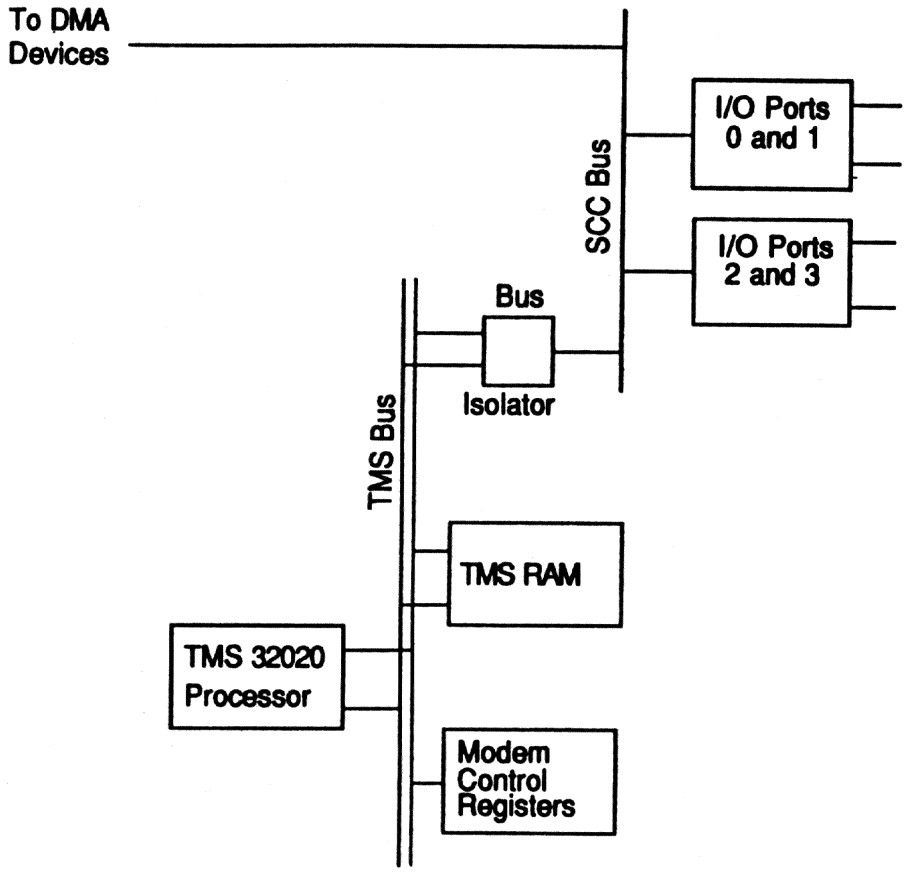


Figure 3-3: Architecture of the Synchronous I/O Ports

Again, there are two major buses:

- SCC bus

- TMS bus

The SCC bus is a multiplexed 17-bit address, 16-bit data bus (similar to the MicroVAX and I_O buses) using one set of lines for both data and addresses. The 16-bit TMS bus, on the other hand, has separate address and data lines to match the requirements of the TMS processor.

The following sections show what's connected to each bus.

3.2.2.1 SCC Bus - The SCC bus has the **four synchronous ports** that provide access to the wide area communications network. The devices used mean that the ports are grouped in two pairs.

3.2.2.2 TMS Bus - This bus has the following blocks:

- **TMS processor** -- providing overall control of the synchronous ports
- **TMS RAM** -- memory that holds the code and data for the TMS processor
- **Modem control registers** -- allowing the TMS processor to set up (and read) the modem control signals

3.2.2.3 Why Two Buses? - The TMS processor has a different architecture to all other parts of the system, in particular its use of a 16-bit bus. Therefore, the DEC MicroServer needs two buses. However, the DEC MicroServer gains from this arrangement because of the benefits of parallel operation.

The two buses can work independently so that the TMS can execute code while data passes through the synchronous ports. The two buses need to synchronize when the TMS processor wants to access the ports or the Buffer RAM on the I_O bus. Synchronization is also necessary if the MicroVAX needs to access the TMS bus. This, however, can happen only under certain circumstances.

3.2.3 Position of the DMA Devices

The main function of the DEC MicroServer is to transfer data between its 5 communications ports. All data must pass through the Buffer RAM, so the best place to put the DMA devices is as

the link between the I_O and SCC buses. This provides the shortest path for data between the synchronous ports and the Buffer RAM.

The MicroVAX and TMS processors also use the Buffer RAM to communicate with each other. To access this information, the TMS processor uses one of the DMA devices. However, this sort of access is relatively infrequent and so the overhead of synchronizing buses is not critical.

Similarly, the MicroVAX processor occasionally needs to access the Synchronous I/O side of the DEC MicroServer. In most cases, the only time the MicroVAX needs to do this is to load the TMS processor's code into the TMS RAM. The code is loaded only when the system is booted and so the overhead of synchronizing all four major buses is not important.

Positioning the DMA devices in this way is a little unconventional. Usually, the devices are closely associated with (and connected to the same bus as) a MicroVAX processor. In this sort of configuration they provide the link to an entire I/O subsystem. The literature for the DMA devices reflects this convention by referring to the **processor bus** and to the **I/O bus**.

However, note that the DEC MicroServer uses two buses on each side of the DMA devices. In this case, the DMA devices are not on the same bus as any controlling processor hence DMA literature references to the processor bus can be misleading.

When the DMA literature refers to processor or MicroVAX buses, this means the MicroVAX or I_O buses in the DEC MicroServer. When it refers to the I/O bus, this means the SCC or TMS buses in the DEC MicroServer.

If you are ever in doubt, refer to the DEC MicroServer architecture diagram (Figure 3-4 in Section 3.2.4. The MicroVAX side of the architecture is what the DMA literature also calls the MicroVAX bus (or side). The Synchronous I/O side of the architecture is what the DMA literature calls the I/O bus (or side).

3.2.4 Complete Architecture

Figure 3-4 shows the complete architecture of the DEC MicroServer. Basically, this is a combination of Figures 3-2 and 3-3, but including the DMA devices. The MicroVAX and the Synchronous I/O parts of the architecture are shown by the dotted line boxes.

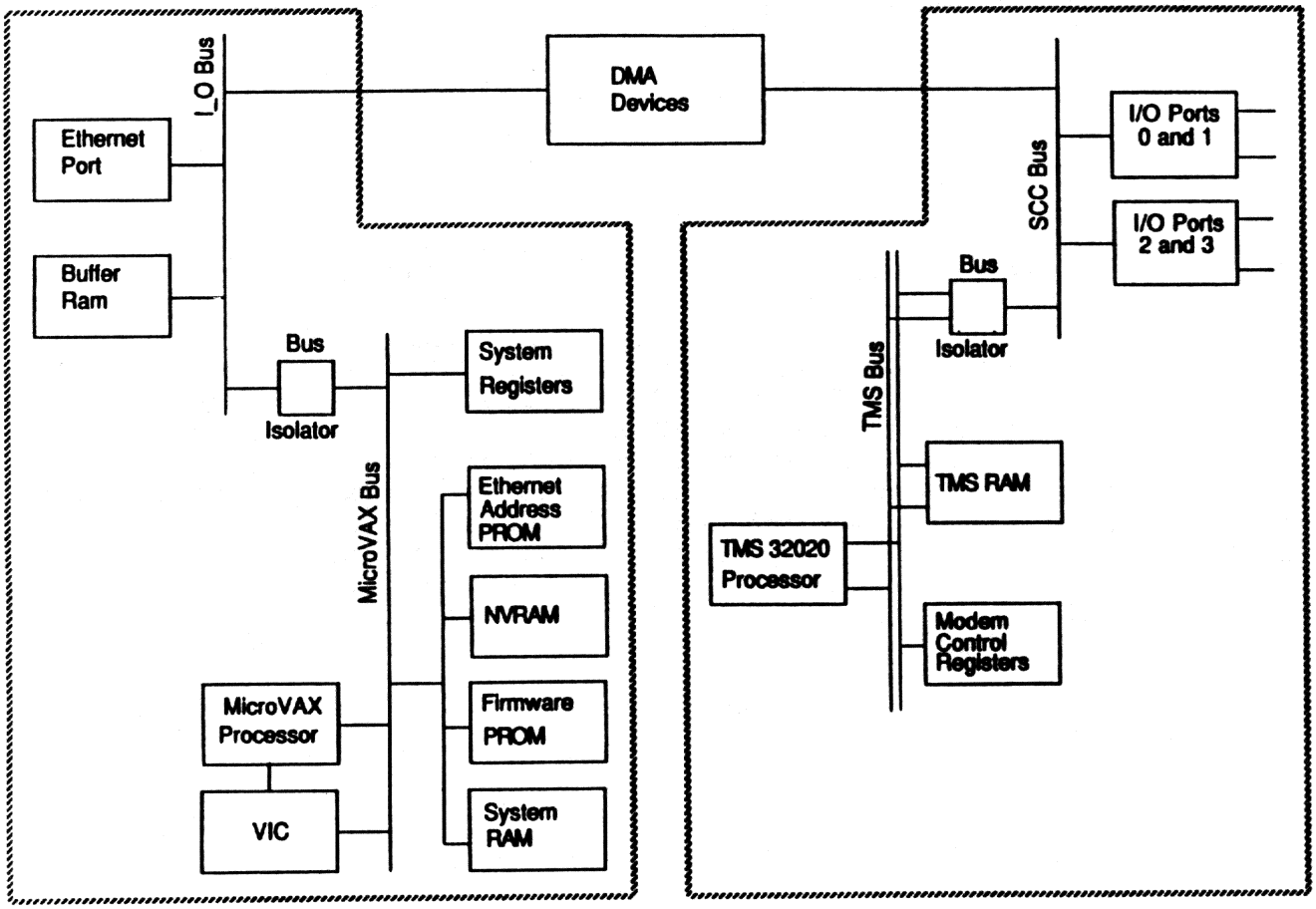


Figure 3-4: Architecture of the DEC MicroServer

3.3 System Control

Although parts of the system can run in parallel, certain system operations require coordination and central control.

The most important areas are:

1. How the MicroVAX processor accesses and controls the devices on the SCC bus.

2. How the Buffer RAM is accessed.

3. How the interrupts generated in the system are handled.

3.3.1 MicroVAX Access to the SCC Bus

The MicroVAX processor can access any part of the system. However, to access devices on the SCC or TMS buses, the MicroVAX processor uses one of the DMA devices in **IO_Access** mode. IO_Access mode allows the MicroVAX to access the Synchronous I/O side's address space, though the TMS RAM space and the SCC bus should only be accessed when the TMS is not running (see Chapter 6 for full details). The DMA in IO_Access mode is the only way that the MicroVAX can access the SCC and TMS buses. In this way, the DMA provides a 'window' through which the MicroVAX accesses the SCC amd TMS buses. For ease of reference, this manual often uses the term 'DMA window' to refer to this use of the DMA.

Appendix B contains the address maps for the system, including the DMA window.

3.3.2 I/O Processor Access to the Buffer RAM

The MicroVAX processor controls the synchronous side of the system through a control block and through structures called ring buffers. In all, there are eight sets of ring buffers (two for each synchronous port). Each port has two rings: one for input operations, and the other for output.

Every buffer has a descriptor giving its position and size as well as some other control information that the MicroVAX and the TMS processor need. The complete set of buffers and descriptors is controlled through a control block.

The Buffer RAM holds all this information, and the TMS Processor uses one of the DMA devices to access it. During such an operation, the DMA controller becomes master of the I_O bus.

So until the operation is complete, the MicroVAX cannot use this bus. In addition, the TMS processor cannot access the I_O bus until the DMA device can be bus master. This exclusive access ensures that both processors do not try to access the buffers and their descriptors at the same time.

See Chapter 15 for details of how the system uses ring buffers. Chapter 6 has details of the control block.

3.3.3 Interrupt Control

Most of the interrupt processing is done through the MicroVAX Vectored Interrupt Controller (VIC) and the MicroVAX processor. The VIC accepts interrupts from the synchronous ports (through the TMS processor), the Ethernet interface, and the DMA devices. From these it produces an interrupt priority value and an interrupt vector, both of which are fed to the MicroVAX processor.

Other events (such as the watchdog timer and the power fail indicator) feed directly into the MicroVAX processor. These cause the processor to Halt or Reset, which in turn makes the processor start executing from a specific location in the firmware. That code determines the cause of the event and takes appropriate action.

Chapter 10 has more information on interrupt and error processing. Chapter 13 has more information on the firmware's actions when the processor restarts after a Halt or Reset.

3.4 System Operation

Before describing the hardware in detail, an overview of how all the blocks work together is useful. This shows how they work in relation to each other and how the buses are used at various times.

The following sections show how:

1. The system is loaded and initialized
2. Synchronous messages are sent and received
3. Ethernet messages are sent and received

4. The system dumps its memory and registers if a fatal error occurs

5. The Buffer RAM is used

3.4.1 System Startup

There are 4 steps in the system startup process:

1. Reset and test
2. Software load
3. System initialization
4. TMS processor load and start

3.4.1.1 Reset and Test - At power up, most of the components on the board are in their Reset state, including the TMS processor. The reset causes the MicroVAX to start running code from the Firmware PROM, and it runs the on-board test.

This test checks that the hardware works and can run the operational software. It uses the seven-segment display to show the progress of the testing and to report any errors. It also records any errors in the NVRAM.

3.4.1.2 Software Load - When the self test has completed successfully, control is passed to the DEC MicroServer firmware. This sends a message on the Ethernet asking for a load host to provide software. When a load host replies, the software image is sent to the DEC MicroServer as a series of Ethernet messages. As each is received, it is copied into the Buffer RAM. The firmware copies the code segments from there to the System RAM. When the load is complete, the firmware passes control to the software.

3.4.1.3 System Initialization - The software first sets up its own data structures, and then:

1. Sets up and initializes the buffers and the control blocks in the Buffer RAM

2. Reads an initialization file from the load host and uses this to set up the synchronous I/O side of the system

3.4.1.4 TMS Processor Load and Start - The software also loads the TMS RAM with code that's normally called the TMS firmware. This code is loaded into the System RAM as part of the software image. To transfer the code to the TMS RAM, the system software:

1. Checks the System CSR to see how much TMS RAM the system has
2. Sets up a DMA window that provides access to the TMS RAM
3. Copies the code from the System RAM to the TMS RAM
4. Releases the TMS processor

Once released, the TMS processor starts to run. It first uses information in the RAM to initialize itself and then waits for instructions from the MicroVAX.

3.4.2 Sending and Receiving Synchronous Messages

The MicroVAX is in overall control of the system, and it uses commands to set up and start synchronous communications. It uses the control block in the Buffer RAM to pass these commands to the TMS processor.

On a timer, the TMS processor regularly looks for commands and for messages to send in the Buffer RAM, acting on any that it finds.

The general procedure for sending a message is:

1. The MicroVAX finds a buffer for the message and sets a flag in the buffer descriptor to indicate the message is ready to be sent.
2. The next time the TMS processor scans the buffer descriptors, it notices the flag has been set.
3. The TMS processor moves the data from the Buffer RAM to the appropriate I/O port (using the DUSCC and DMA).
4. When the transmit completes, the TMS processor puts its return status in the buffer descriptor, resets the flag, and interrupts the MicroVAX.

5. The MicroVAX interrupt handler checks the flag change, examines the status, and decides what action to take. If the transfer was successful, no action is needed and the buffer can be reused. If there was an error, the MicroVAX can take a number of paths, such as resending the message.

The process for message reception is similar:

1. The TMS processor detects the incoming message through changes in the modem signals, and looks for a free buffer in the Buffer RAM.

2. The TMS processor then sets up a DMA transfer between the appropriate synchronous port and the buffer.

3. When the transfer completes, the TMS processor does some error checking (such as checking CRC values).

4. The TMS processor puts some status information in the buffer header, sets a flag to indicate the buffer contains a message, and interrupts the MicroVAX.

5. The MicroVAX detects the change in the flag and processes the buffer, based on the status information.

3.4.3 Sending and Receiving Ethernet Messages

Sending and receiving Ethernet messages follows a similar pattern to the synchronous messages. However, the I/O operations are driven by the Ethernet Interface instead of the TMS processor.

The interface has its own set of buffers and it scans them regularly. In addition, the MicroVAX can give commands directly to the interface. For example, it can tell the Ethernet Interface to start or stop data communications.

The general process for sending a message is:

1. The MicroVAX looks for a free buffer and puts the message in it.

2. The MicroVAX sets a flag to indicate that the buffer contains a message waiting to be sent.

3. The Ethernet Interface detects the change in the flag and prepares for the transfer. It then uses its own DMA capabilities to read the data in the buffer.

4. The Ethernet Interface puts the status of the transfer in the buffer's header, resets the flag, and interrupts the MicroVAX.

5. The MicroVAX reads the status and reacts accordingly. For example, it may try to resend the message.

The general procedure for receiving a message is:

1. The Ethernet Interface detects the incoming message by recognizing the Ethernet address, and looks for a free buffer.

2. On finding one, the Ethernet Interface uses its own DMA capabilities to transfer the incoming data to the buffer.

3. At the end of the message, the Ethernet Interface writes status information in the buffer's descriptor, sets a flag to indicate the buffer contains information to be processed, and interrupts the MicroVAX.

4. The MicroVAX detects the change in the flag, reads the status information, and reacts accordingly.

3.4.4 System Dumping

Sometimes the system may detect an irrecoverable error. In such a case, the system has to be reloaded, but before doing so the DEC MicroServer tries to dump the erroneous system's memory and registers.

A system dump is basically a record of the contents of the entire system as it was when the error occurred. System dumps are used by specialists to find the cause of an error.

The DEC MicroServer relies on a host computer on the Ethernet to process the dump. In many cases, this involves receiving the information and putting it in a file.

Transferring the information to the host computer is the same for all dumps, though there are slight differences in the procedure prior to the transfer depending on whether the MicroVAX or the TMS processor detected the error.

3.4.4.1 MicroVAX Detected Error - If the MicroVAX detects the error, it starts the dump procedure. First it saves its own

context (register values and so on). Then the MicroVAX tries to include as much of the TMS RAM (and the TMS processor) context as possible.

To do this the MicroVAX generates a special interrupt to the TMS processor. On receiving this, the TMS saves its context in the TMS RAM and then sets a flag in the control block in Buffer RAM. The MicroVAX detects that the flag has been set and starts the dump.

3.4.4.2 TMS Processor Detected Error - If the TMS processor detects the irrecoverable error, the preparation is slightly different.

When it detects the error, the TMS processor saves as much of its context as possible in the TMS RAM. Then it sets the flag in the control block and generates an interrupt to the MicroVAX.

The MicroVAX detects the change in the flag and starts the dump procedure. The MicroVAX saves its own context and then executes the dump routines in the Firmware ROM.

3.4.4.3 Dumping the System - Once the status of both processors has been saved, the dump code saves the context of the Ethernet interface. Then, it sends a MOP dump request on the Ethernet asking for a host to receive the dump.

On receiving a reply from a host, the dump is sent in a series of messages on the Ethernet. This information includes the:

- The context of both processors
- The contents of the System, Buffer, and TMS RAMs
- As many other device registers (from the Ethernet interface, the DMA devices, and so on) as can be retrieved.

When the dump is complete, the DEC MicroServer resets itself, runs the On-Board Test (OBT), and reloads the system.

3.4.5 Use of Buffer RAM

The Buffer RAM is the part of the system through which all data passes, and holds the control information for the communication devices. The RAM holds:

1. Buffers for the Ethernet interface (one set for transmit and one for receive)
2. Buffers for the synchronous ports (four sets for transmit and four for receive)
3. Initialization information for the Ethernet Interface
4. Control block for the TMS processor and the MicroVAX to communicate with each other
5. Work space for the VAX firmware

The system software sets up the first four of these and determines where they are, how many buffers there are, and how large each is. This information is passed to the TMS processor through the control block. The MicroVAX tells the TMS the location of the control block by passing it a parameter when loading the TMS firmware.

The work space for the VAX firmware is an initialization block and set of buffers for the Ethernet Interface. This enables the firmware to use the interface without disturbing other values in the Buffer RAM. For example, the firmware uses these structures when dumping the system so that the contents of the real buffers are not disturbed.

Chapter 4
The MicroVAX and I_O Buses

4.1 Overview

This chapter, and Chapter 5, contain a more detailed description of the MicroVAX side of the DEC MicroServer's architecture. This chapter deals with:

- The structure of the MicroVAX and I_O buses
- The MicroVAX processor
- System clocks
- The blocks of memory (System RAM, Firmware ROM, NVRAM, and Buffer RAM)
- System Registers
- Address decoding
- Error reporting
- Bus arbitration

Chapter 5 concentrates on the Ethernet Interface.

4.2 Bus Structures

Both the MicroVAX and I_O buses are multiplexed using 32 address/data lines. A pair of signals (Address Strobe and Data Strobe) indicate whether the bus is carrying an address or data. The Field Maintenance Print Set has details of the bus structures and their signals.

The buses have limited drive capability; the current drive in a high state is 2 milliamps, and in the low state 400 microamps. This limits the number of units that can be directly connected to each bus, and so some buffering is used.

Figure 4-1 is a diagram of the MicroVAX bus and the items attached to it. Figure 4-2 is a similar diagram for the I_O bus.

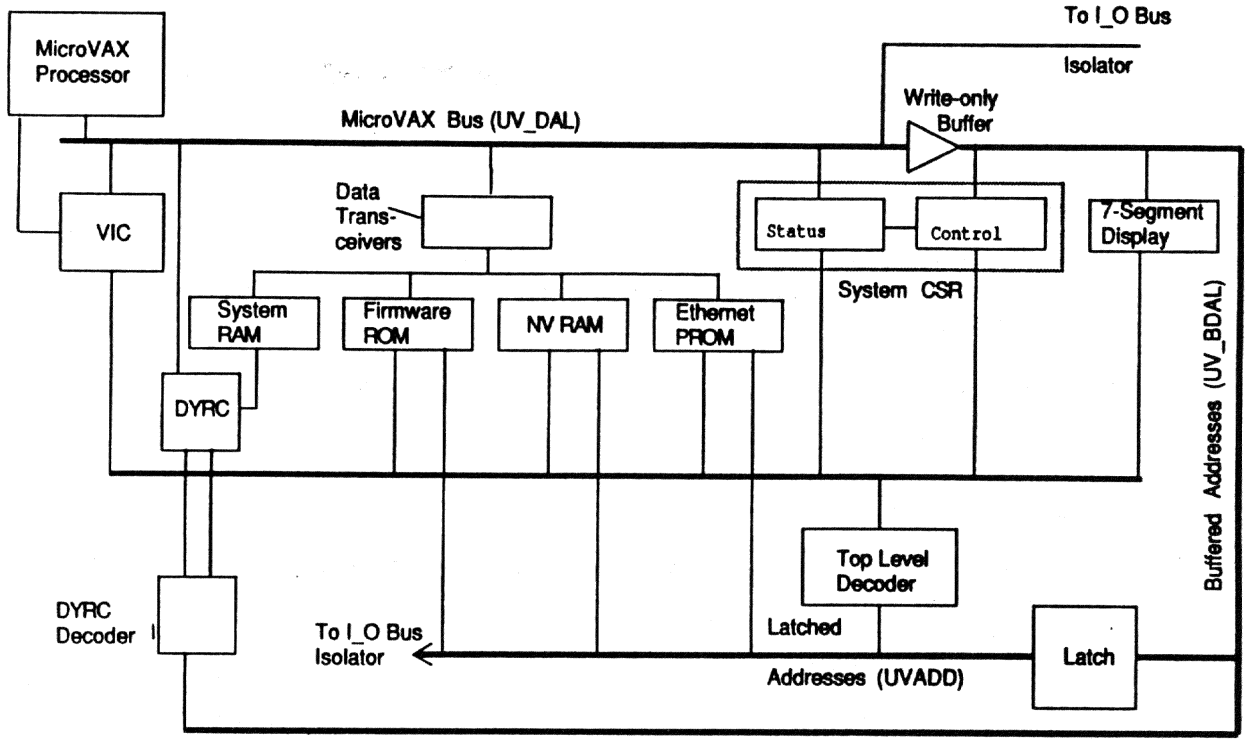


Figure 4-1: Block Diagram of the MicroVAX Bus

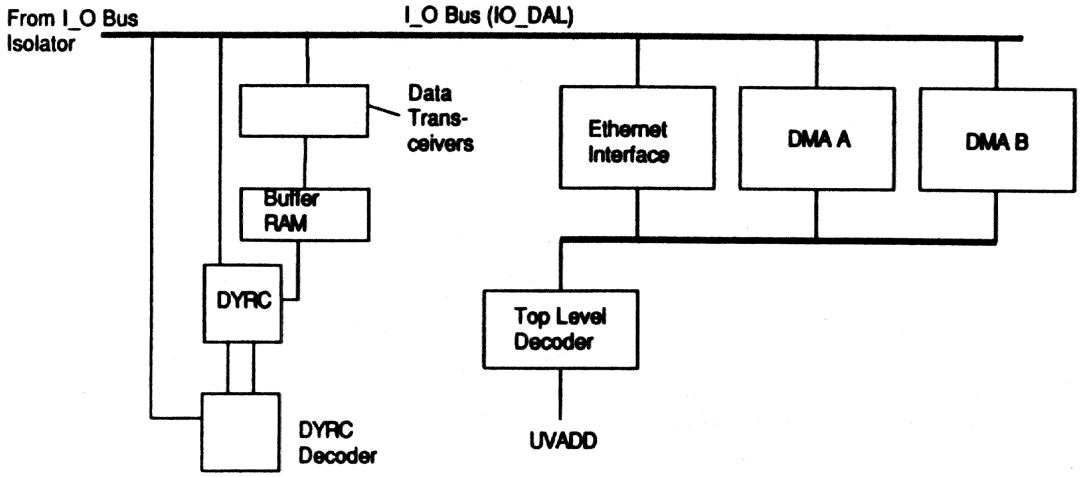


Figure 4-2: Block Diagram of the I_O Bus

4.3 MicroVAX Processor

The MicroVAX II processor is the overall controller of the entire system. The software it runs determines the type of communications service to users, and controls other parts of the system (either directly or indirectly).

Figure 4-3 shows the processor and its associated components.

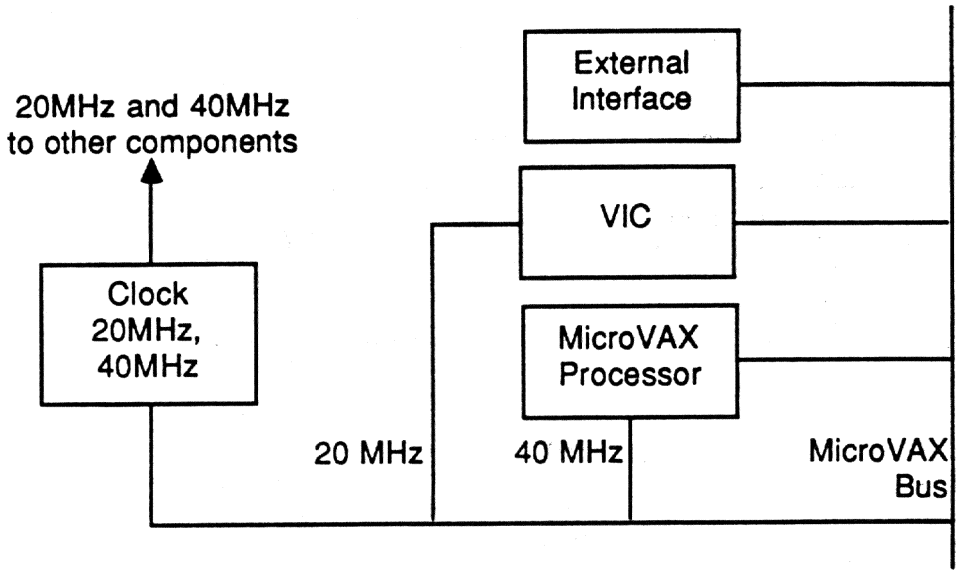


Figure 4-3: The MicroVAX II Processor and its Surrounding Circuitry

The processor is directly connected to the MicroVAX bus, and is always its bus master. It becomes master of the I_O bus only when accessing that bus.

For a detailed description of the device, its architecture, and instruction set, refer to the books listed in Appendix I. This section summarizes the way the processor is used in the DEC MicroServer.

4.3.1 Address Space

To the processor, the two buses appear as a continuous address space containing both code and data. Only the addresses used distinguish one bus from another. Addresses below 30000000 (hex) access devices on the MicroVAX bus. Addresses 30000000 and above access devices on the I_O bus. See Appendix B for a detailed address map of the MicroVAX processor.

To access memory, the processor puts longword addresses on the bus, and uses the Byte Mask (BM) signals to indicate whether a byte, word, or the complete longword is being accessed.

4.3.2 Access to the Synchronous I/O Side

The processor can only access devices on the MicroVAX and I_O buses in this direct way. Once the system is operational, this is usually sufficient. However, from time to time it needs to access devices on the synchronous I/O side of the architecture. The processor particularly needs to do this when loading the TMS processor or when dumping the system contents.

To access devices on the synchronous side, the MicroVAX uses one of the DMA channels in IO_Access mode. The specific address used by the processor depends on the DMA channel being used. Appendix B shows the MicroVAX address space and also shows the offsets of devices, including the location of the DMA channels.

The MicroVAX cannot access the synchronous I/O side when the TMS processor is running. This prevents potential lock up. You can find more details on the DMA devices and their use in Chapter 8.

4.3.3 Interrupts and Exceptions

Together, the VIC and the MicroVAX handle most of the interrupts and exceptions in the system. Many of these come from the synchronous side of the architecture to indicate the completion of data transfers and to report changes in the synchronous control block.

The VIC handles most of the interrupts, and Chapter 10 describes them in some detail. The MicroVAX, however, does handle some interrupts directly:

- **100 Hz interval timer** -- this is used in the software for timeout counters and so on. The timer provides an interrupt every 10 ms at the INTTIM pin on the MicroVAX. The timer is provided by the DYRC for the System RAM.

- **Power fail** -- this is generated if the POK signal drops, and allows the system to save its context before the power is lost.

- **Dump switch** -- pressing this switch causes the processor to restart and tells the ROM code to dump the system before running the OBT and reloading the software. This

interrupt is connected to the MicroVAX's HALT input.

- **Watchdog timer** -- this timer runs out if the software falls into an unknown state. The watchdog timer interrupt causes the system to dump and restart. This interrupt is also wired to the MicroVAX's HALT pin.

- **Bus errors** -- errors on the MicroVAX or I_O buses are reported as interrupts to the MicroVAX. Errors on the MicroVAX bus are reported through the ERR pin on the processor, while errors on the I_O bus are reported through the PWRFL pin.

Chapter 10 also contains details of these interrupts. Section 4.12 has an overview of error handling, with Chapter 10 having more detailed information.

4.4 System Clocks

All processors in the system (including the TMS processor) use a 40 MHz or a 20 MHz timing signal. There are three clocks of each frequency in the system, all derived from a single 40 Mhz crystal.

The 40 MHz signals are taken directly from the crystal and buffered before distribution to the rest of the circuitry. The 20 MHz signals are derived by dividing the 40 MHz signal and then buffering the resultant signal.

The Ethernet Interface and the Synchronous ports require clocks isolated from the rest of the system. So, these devices use their own crystal based clock sources.

4.5 System RAM

The System RAM is organized as 256K longwords with 4 parity bits for each longword (one for each byte). The memory uses dynamic RAM devices, managed by a MicroVAX Dynamic RAM Controller (DYRC). This handles functions such as address decoding, object decoding, and memory refresh. Figure 4-4 is a simplified diagram of the System RAM block (in particular, the buffering between the RAM devices and the DYRC is not shown).

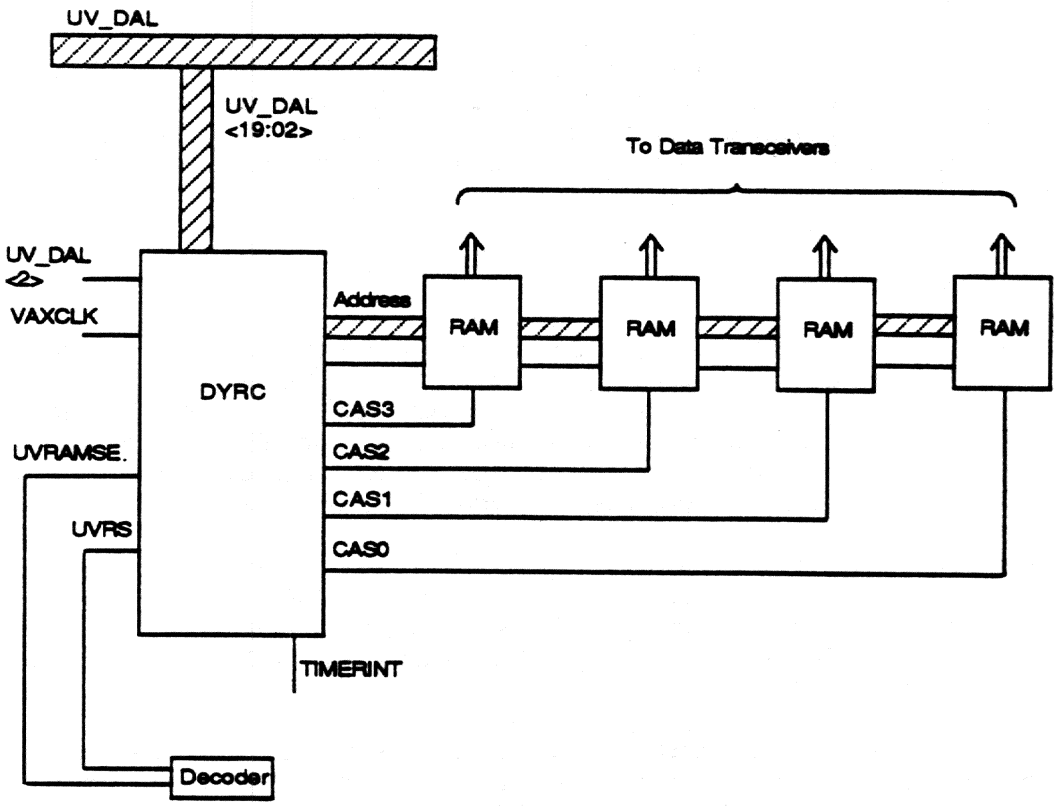


Figure 4-4: Simplified View of the System RAM Block

4.5.1 Memory Element Addressing

The DYRC accepts an 18-bit longword address from the MicroVAX bus (UV_DAL<19:02>) and uses this to generate the column and row addresses for the four RAM devices.

Column and row addresses are multiplexed; strobe signals indicate which is being generated at a particular time. All the memory devices receive the row address and they read this when the row address strobe (UVRAS) is asserted.

The DYRC has four column address strobes, one for each byte in a longword. Which of the four strobes is asserted depends on what sort of access is in progress:

- If a byte is being accessed, only the appropriate (single) strobe signal is asserted.
- If a complete longword is being accessed, all four strobes signals are asserted.

The DYRC uses the Byte Mask signals from the MicroVAX to determine which column address strobes to generate.

When the correct memory element has been addressed, the UVWR signal is used to determine whether the element is to be read or written. All data passes through the data transceivers that provide the bus interface for all the memory on the MicroVAX bus.

4.5.2 DYRC Registers

The DYRC has its own set of registers that MicroVAX software can use to manage the unit's operation. These registers appear in locations 24000000 and 24000004 (hex) of the MicroVAX address space. However, the DYRC has a separate pin for selecting the registers instead of memory. So, to provide the correct signals from the address lines, a decoder (type 74F138) is used.

The decoder asserts UVRS when the address is in the range 24000000 to 25FFFFFF. This tells the DYRC that this is a register access, and the address line 2 determines which register is selected: the Command and Status Register (CSR) or the Fault Address Register (FAR).

The 16-bit CSR gives information on the status of the memory and allows the MicroVAX to control features such as parity checking. Figure 4-5 shows the layout of the CSR, and Table 4-1 shows the meaning of each bit in the register.

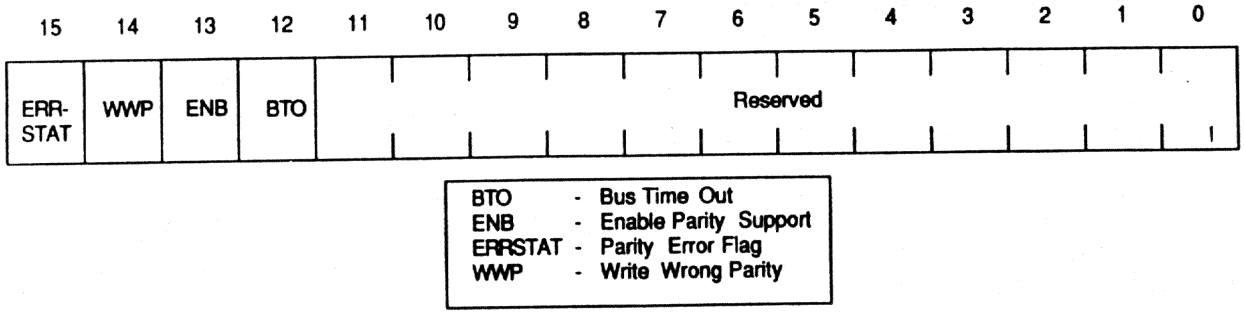


Figure 4-5: The DYRC Control and Status Register

Table 4-1: Meanings of the Bits in the DYRC CSR

Field Name	Meaning and Use
BTO	Bus Timeout. The DYRC sets this bit if a bus timeout occurs. The bit is cleared when the MicroVAX reads the CSR, and at power up.
ENB	Enable Parity Support. The MicroVAX uses this bit to control parity checking. By setting this bit, parity checking is enabled. At power up, this bit is cleared.
WWP	Write Wrong Parity. The DEC MicroServer's on-board test program uses this bit to generate inverted parity and so test the parity generators/checkers. In normal use, this feature is not used, and the bit is clear. The bit is also cleared at power up.
ERRSTAT	Parity Error Status Flag. This bit is set if a parity error is detected in the memory. The bit is cleared when the MicroVAX reads the CSR, and at power up.

The FAR is also 16 bits long, and contains information on the location of a parity error. Figure 4-6 shows the format of the FAR, and Table 4-2 shows the meaning of each bit in the register.

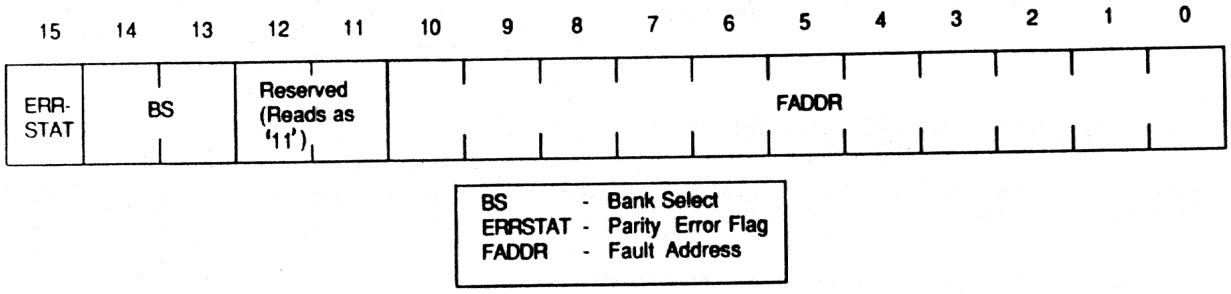


Figure 4-6: The DYRC Fault Address Register

Table 4-2: Meanings of the Bits in the DYRC FAR

Field Name	Meaning and Use
FADDR	Fault Address. This is the address of the page in which the error occurred. These bits are cleared when the MicroVAX reads the register, and at power up.
BS	Bank Select. These bits identify the bank in which the error occurred. They have the same value as the Bank Select pins had during the erroneous cycle. In the DEC MicroServer, these pins are derived from bits 20 and 21 of the MicroVAX address.
ERRSTAT	Parity Error Status Flag. This bit is set when a parity error occurs. The bit is cleared when the CPU reads this register, and at power up.
Note: Bits 11 and 12 are clear at power up, but become set at the end of the first memory access. They remain set until the device is reset or the next power up occurs.	

NOTE

The DYRC used in the first DEC MicroServer units did not produce reliable information in the FAR. This is expected to be resolved in future versions of the device.

4.5.3 Memory Partitioning

Simple memory protection is provided by the system image being written so that:

- Code and data are separate, and
- The code occupies the lower addresses

In this case, writing to the code part of the RAM is prevented and a partition error produced.

The DEC MicroServer provides such a protection mechanism through its partition register. This divides the System RAM into two 512K byte parts. When partitioning is enabled, any attempt to write to the lower 512K bytes generates a partition error. To enable this feature, the software simply sets the PRTNEN bit in the System Control/Status Register.

4.5.4 Parity Checking

The memory has parity, and so uses devices that are 256K by 9 bits (8 bits of data and 1 parity bit). This gives 4 parity bits for each longword.

The parity for each byte is checked using a type 74F280 parity generator/checker. To distinguish between parity generation and checking, a 74F257 multiplexer is used.

If the parity for a certain byte is incorrect, the appropriate UVPERR signal is asserted. The UVPERR signals are fed to an error-checking PAL which asserts the UVERROR signal to the DYRC's PARIN pin. This in turn asserts the UVERR signal that causes a machine check in the MicroVAX processor. Note that the UVERR signal is only asserted if enabled by the DYRCENB bit, and that DYRCENB is cleared on an error.

4.5.5 Clocks

The DYRC uses the same 40 MHz clock as the MicroVAX processor. The DYRC also divides this 40 MHz clock to generate a 100 Hz interval timer, which is fed back to the MicroVAX processor as the TIMERINT interrupt.

4.6 Firmware ROM

The DEC MicroServer design allows for two sizes of ROM for the firmware. The present design contains 32K longwords stored in 27256 type EPROMs, but allows for double-sized 27512 EPROMs to be added later, if demanded by an increase in the size of the firmware (this is described below).

The firmware memory space appears in the address map between 20040000 and 2007FFFF, and contains the system's firmware, as detailed in Part III of this manual. Figure 4-7 is a simplified diagram of the Firmware ROM.

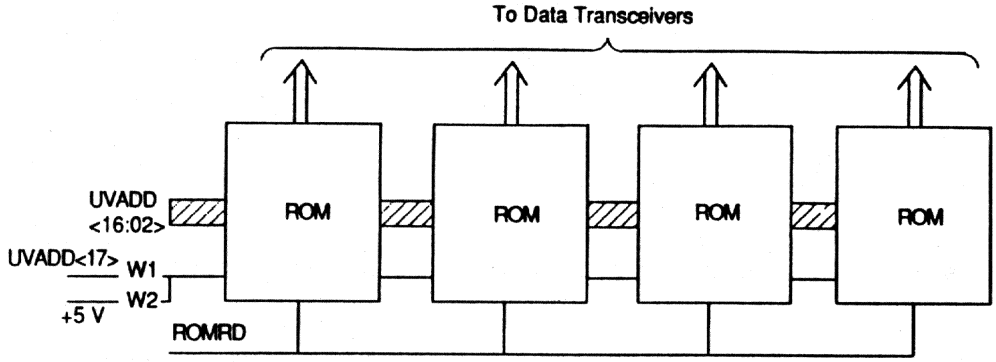


Figure 4-7: Arrangement of the Firmware ROM

The complete bank is selected when the ROMRD signal is asserted (see Section 4.10 for details of how ROMRD is produced). Then, the latched address lines (UVADD <16:02>) are used to select the appropriate longword.

More ROM may be necessary if the firmware increases in size. The design allows for double-sized devices to replace the 27256s. A pair of links (W1 and W2) allow the extra address line to be applied to the larger 27512 device.

Table 4-3 shows how the links should be set for each type of device.

Table 4-3: Settings of W1 and W2 for Different ROM devices

	Link Settings	
Device	**W1**	**W2**
27256	O	I

Table 4-3 (cont.)

Device	W1	W2
27512	I	O

I = Link In
O = Link Out

4.7 NVRAM

The NVRAM is an X2864 type (EEPROM) device providing 8K bytes of non-volatile storage. The memory is used to hold some system parameters (such as the console password). It is also used by the firmware and software to log error conditions that are detected. Figure 4-8 shows how the unit is wired.

The NVRAM appears at addresses 22000000 to 220007FFF. The memory is selected using the address lines, and the NVRAM signal. The NVOE and NVWE signals indicate whether information is to be read from or written to the memory.

The element address is determined by the latched address lines (UVADD<14:2>). The device is byte-oriented, but notice that in the MicroVAX address space each byte appears on a longword boundary.

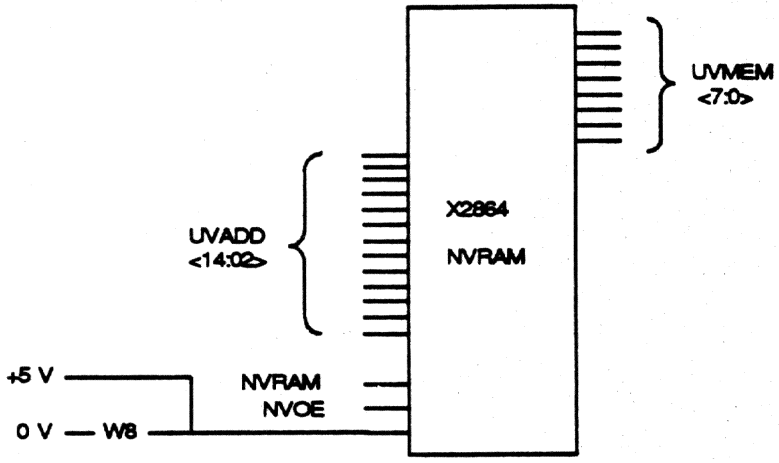


Figure 4-8: NVRAM Connection

See Chapters 16 and 17 for information on how the memory is used.

4.8 The Buffer RAM

The Buffer RAM has a similar design to the System RAM on the MicroVAX bus. This is also 1M byte, is controlled by a DYRC, and is parity checked by 74F280 devices. However, the memory is not partitioned. The RAM is enabled from the decode logic which asserts the IORAMSEL signal.

Errors are reported as an interrupt to the MicroVAX. This enables the MicroVAX processor to rapidly find the source of an error, and to determine that it originated on the I_O bus.

4.9 System Registers

The System Registers provide control and status functions for the entire DEC MicroServer system. Each of the functional blocks has its own internal registers. These registers, however, tend to be specific to the task of each block. So, the System Registers are needed to hold system wide information. For example, one of the register bits indicates when a system dump is to be performed.

The DEC MicroServer has a Control/Status register, and a Display register.

4.9.1 Control/Status Register

The Control/Status register is one word at address 27FFF808 (hex). Figure 4-9 shows the function of each bit in the register. Figure 4-10 shows the general arrangement of the register's circuitry.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDEN	TMS-MEM	PWRFL	DUMP	PRTN-ERR	WD-ERR	IO-ERR-SRC	CON-SOLE	TMS-CRL2	TMS-CRL1	RST-DSP	HW-RESET	PAGE 0	PRT-EN	DUM-PEN	DIS-PEN

CONSOLE - Console Port Status		PRTNERR - Partition Error	
DISPEN - Display Enable		PWRFL - Power Failure	
DUMP - Dump Switch Status		RSTDSP - TMS Processor Reset	
DUMPEN - Enable Dump Switch		TMSCRL1 - TMS Processor Control 1	
HWRESET - Hardware Reset		TMSCRL2 - TMS Processor Control 2	
IOERRSRC - I_O Bus Error Source		TMSMEM - TMS RAM Size	
PAGE0 - TMS Processor Page		WDEN - Watchdog Timer Enable	
PRTEN - Enable Partition Logic		WDERR - Watchdog Timer Expired	

Figure 4-9: The System Control/Status Register

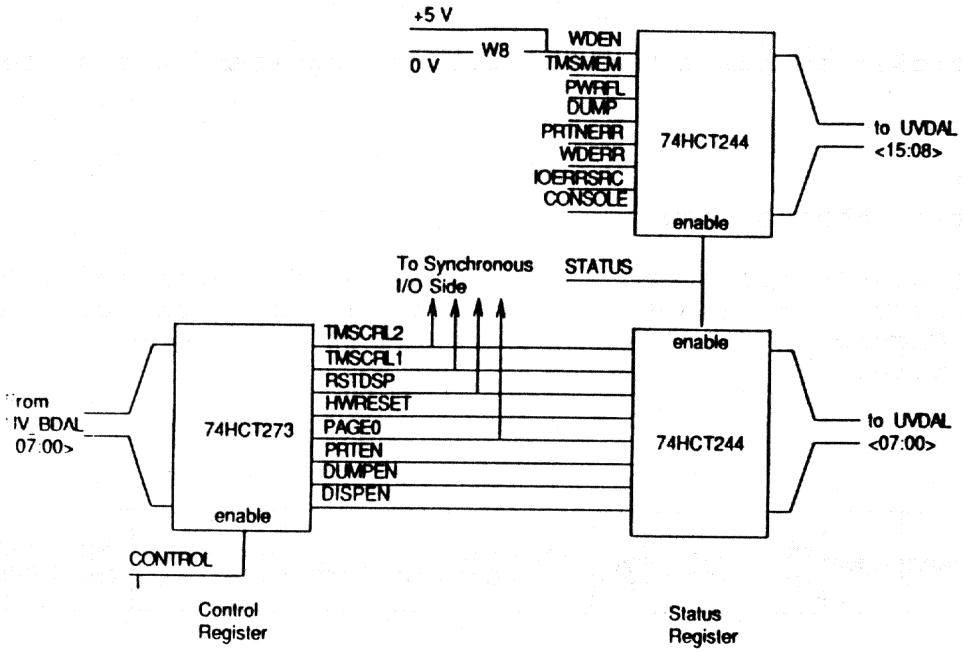


Figure 4-10: The General Arrangement of the Control/Status Register's Circuitry

The system's software can write information into the low byte only. To enable a particular function, the software sets the appropriate bit.

The software can read either byte, or the full word, as status information. In this case, the low byte will show what has been previously set up. The high byte contains error information.

Table 4-4 gives more information on the function of each bit in the register.

Table 4-4: Meanings of Bits in the System Control/Status Register

Field Name	Function
DISPEN	When set, this bit turns on the seven-segment display. The bit is cleared at power up.

Table 4-4 (cont.)

Field Name	Function
DUMPEN	When set, this bit enables operation of the DUMP switch on the control panel. This bit is cleared at power up.
PRTNEN	When set, this bit causes write operations to the Code/Data RAM to be checked against the partition register. See Section 4.5.3 for details of memory partitioning. This bit is cleared at power up.
PAGE0	Selects bank 0 or bank 1 of the TMS processor's program memory. Setting this bit selects bank 1. This bit is only useful when RSTDSP is set and when the TMSMEM bit indicates that there are two banks of TMS RAM. The MicroVAX uses this bit when loading the TMS firmware, because it can access only one bank at a time. See Chapter 14 for more information on how this bit is used.
HWRESET	When set, this bit resets all devices apart from the MicroVAX processor, the VIC, the DYRCs, the control part of the system control/status register, and the TMS processor. This bit allows the system's software to place the system in a known state and still retain the current operating context.

Table 4-4 (cont.)

Field Name	Function
RSTDSP	TMS Processor Reset. When cleared, this bit resets the TMS processor. This allows the MicroVAX software to reset the processor independently of the reset of the system. While this bit is cleared, the TMS processor remains halted and in the reset state. This enables the MicroVAX processor to access the TMS RAM. This bit is cleared on power up.
TMSCRL1	TMS processor control. Reserved for future use.
TMSCRL2	The MicroVAX processor uses this bit to generate the RINT interrupt to the TMS processor. The MicroVAX uses the interrupt to indicate that it wants to dump the system. The TMS processor responds by saving its contents in the TMS RAM. To generate the interrupt, the MicroVAX must set this bit, and then clear it. The time between setting and clearing the bit must be at least 1 microsecond.
CONSOLE	Normally, this bit is set. However, when equipment is attached to the Logic Interface, this bit is cleared. So, software can use this bit to determine whether a physical console or similar tool is attached.

Table 4-4 (cont.)

Field Name	Function
IOERRSRC	This bit can be set only when an error occurs on the I_O bus. Such an error causes the MicroVAX to restart and it can then use the value of this bit to determine if it or one of the devices on the I_O bus causes the error. When set, the error was caused by one of the DMA devices or by the Ethernet Interface.
WDERR	When set, this bit indicates that the Watchdog Timer expired.
PRTNERR	When set, this bit indicates that a partition error has occurred.
DUMP	When set, this bit indicates that the DUMP switch on the control panel has been pressed.
PWRFL	When set, this bit indicates that a power failure occurred. It is also set whenever the CSR is examined through the physical console. Used to help determine the cause of a PWRFL interrupt to the MicroVAX. That interrupt can be caused by an error on the I_O bus or by a power failure. On restarting, the MicroVAX processor uses this bit to determine which event occurred.
TMSMEM	Indicates whether there are one or two banks of TMS processor memory. When this bit is set there is only one bank. The MicroVAX processor uses this bit when loading the TMS firmware.

Table 4-4 (cont.)

Field Name	Function
WDEN	Watchdog enable. Indicates whether the link W8 is fitted to disable the watchdog timer. When set, the link is not fitted, and the watchdog timer is enabled. This should be the setting on all units in the field.
If a halt occurs but neither WDERR or DUMP are set, the source of the halt is the physical console, or the code executing a HALT while in Kernal mode.	

4.9.2 Display Register

The display register is a single byte at address 27FFF804 (hex) that controls the segments on the front panel display. The display does not use a decoder. Instead, to light a segment, the software simply clears the appropriate bit in the display register.

Each segment of the display is linked to the register through a resistor. When a bit is cleared, that end of the resistor is pulled down to 0 V. As the display is connected to 5 V, this causes the segment to light.

Figure 4-11 shows the display, and the allocation of bits in the register. Note that bit 7 is used as an error bit during RQT/PQT, and should be regarded as reserved for future use.

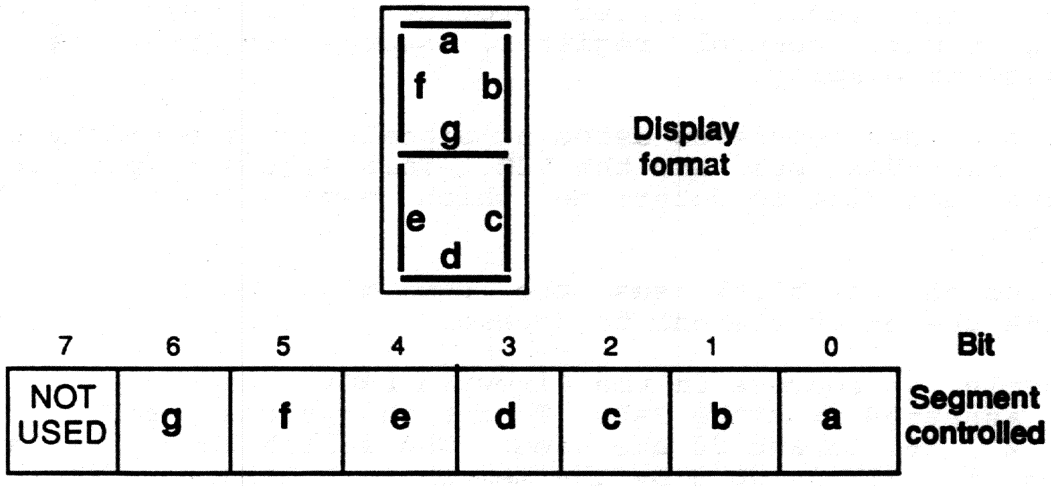


Figure 4-11: Display Register Bit Allocation

4.10 Address Decoding

When the MicroVAX is accessing either bus, the address decoding takes place on two levels:

1. The system block being accessed
2. The location in that block

The following sections show how this system works for both buses. The final section contains some notes on the addressing used by the Ethernet Interface and the DMA devices.

4.10.1 MicroVAX Bus

The top level decoding for the MicroVAX bus uses some of the address lines, together with the relevant control signals, and produces selection signals for specific blocks.

The address decoding uses the latched signals (UVADD). Each address passes through a pair of 74F138 decoders and a PAL uses this address, together with control signals from the MicroVAX bus. This produces select signals for each of the main blocks on the bus.

The first decoder sorts out the top five bits (25 to 29) of the address. The second decoder provides precise decoding of the watchdog timer, control register, status register, and the seven-segment display.

The PAL provides top-level decoding signals for the memory blocks on the MicroVAX bus and the VIC. This scheme simply produces high level decoding to determine which memory block is being addressed.

Once selected, the block uses the remaining address lines to determine the exact element to access.

For example, an address in the Firmware PROM causes signal ROMRD to be asserted from the PAL. To set this signal, address lines 29, 28, 27, 26, 25 and 24 are used. The selection of the actual location in PROM is done by the memory unit itself, using bits 2 to 16 of the latched address.

For a full address map of the MicroVAX processor, see Section B.2 in Appendix B. For a complete diagram of the decoding mechanism and the signals it generates, look at the Print Set.

4.10.2 I_O Bus

Like the MicroVAX bus, the I_O bus has logic that provides some high-level address decoding. This is used only when the MicroVAX is accessing the Buffer RAM, the Ethernet Interface, or one of the DMA devices. So, the logic also uses bits 25 to 29 of the latched MicroVAX address (UVADD).

The decoding logic is based on a 74F138 decoder and a PAL. These devices generate signals to select the Buffer RAM, the Ethernet Interface, and the DMA devices. From there the remaining address lines, in their I_O bus form, are used to access the exact element in the selected block.

Again, refer to the print set for a detailed diagram of the decoding mechanism and the signals it generates.

4.10.3 I/O Device Access to the I_O Bus

The high order byte of each address generated by the Ethernet Interface is set so that it points to the correct part of the address map. The interface supplies the remaining 24 bits based on information in its data structures. The MicroVAX sets up these structures, so it must ensure that the Ethernet Interface

does not try to access non-existent memory.

The DMA devices access memory using their internal page tables. Again, the MicroVAX is responsible for setting up these structures, and so must make sure that the devices only access the Buffer RAM.

4.11 The I_O Bus Isolator

The I_O bus isolator has two basic functions:

1. Isolate the MicroVAX and I_O buses allowing them to work separately or as a single bus

2. Share the I_O bus between the MicroVAX processor, the Ethernet Interface, and the DMA devices

The bus isolator itself is a set of bus transceivers and a set of bus drivers. The transceivers carry data between the MicroVAX processor and the I_O bus. The direction ofthe data transfer is determined by the write signal (UVBWR) from the MicroVAX.

The bus drivers carry addresses from the MicroVAX to the I_O bus. This uses the latched address lines on the MicroVAX bus (UVADD) to address items on the I_O bus.

A pair of PALs control access to the bus. One determines which device should have access and the other generates the appropriate control signals.

The first PAL (called RROBIN) schedules bus access on a round robin basis. That is, it examines the bus request signals from each potential bus user in turn to see if they want access. Access is given to the first requestor and when they've finished the cycle continues to the next in line. Figure 4-12 is a state diagram of the scheduler.

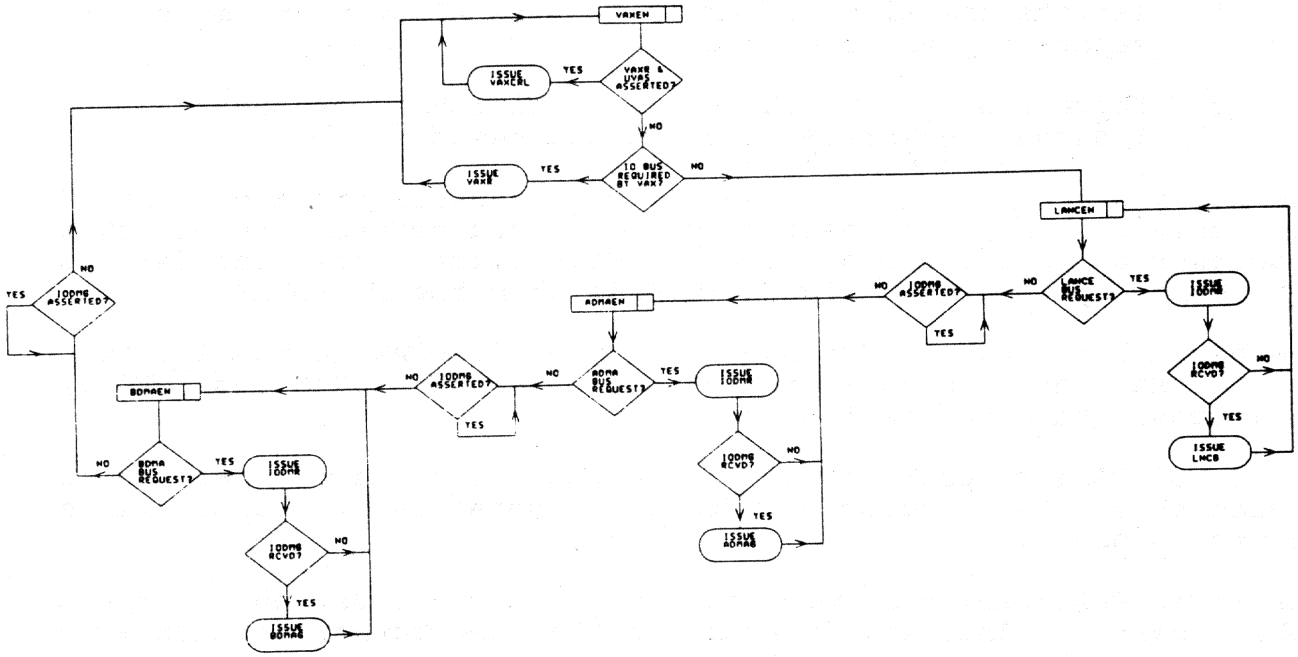


Figure 4-12: The Round Robin Scheduler for the I_O Bus

The second PAL (IOCRL) generates the relevant control signals in the correct order. For example, when the MicroVAX is accessing the I_O bus, the IOCRL PAL makes sure that the address is enabled at the correct time and is then followed by data.

IOCRL is also a state machine, and Figure 4-13 shows the state linkages. Table 4-5 summarizes the actions in each state. Table 4-6 summarizes the function of the input and output signals.

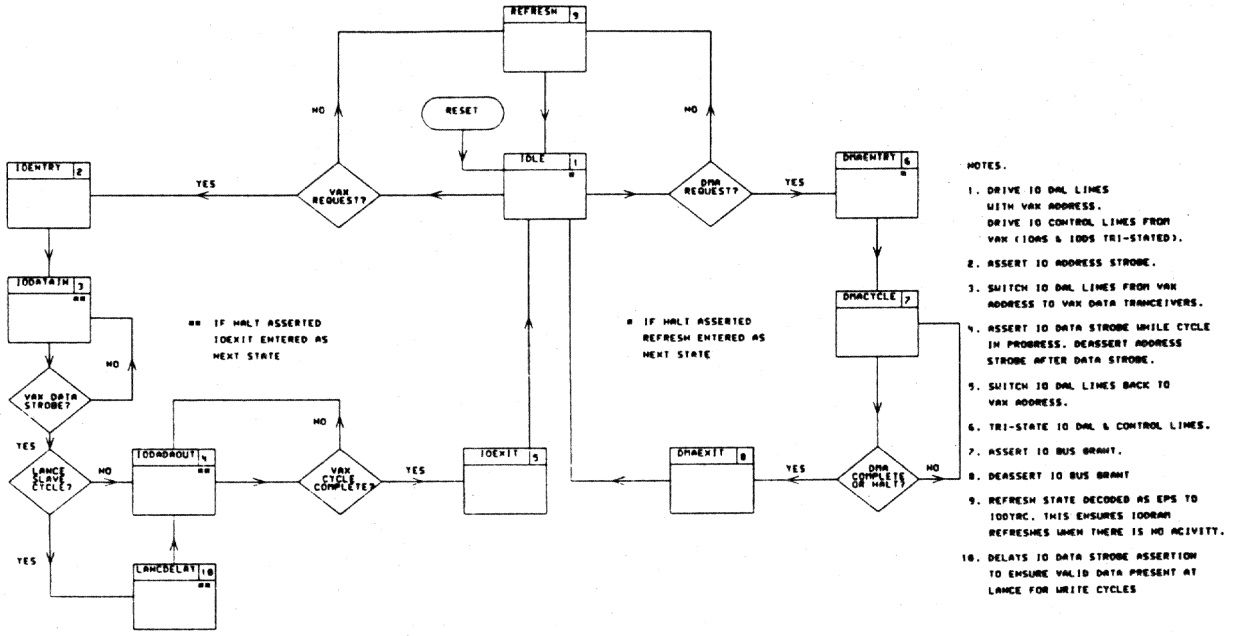


Figure 4-13: The I_O Bus Control State Machine

Table 4-5: States in the I_O Bus Controller

	State Name	List of Actions
1	IDLE	1. IODMG, IOAS, IODS deasserted 2. IODATEN deasserted 3. IOADDEN asserted 4. IOCRLEN asserted 5. IOAS and IODS to high impedence state
2	DMAENTRY	1. IOADDEN deasserted 2. IOCRLEN deasserted
3	DMACYCLE	1. IODMG asserted
4	DMAEXIT	1. IODMG deasserted
5	IOENTRY	1. IOAS asserted 2. IODS asserted 3. IOADDEN deasserted after IOAS assertion 4. IOADDEN and IOCRLEN asserted
6	IODATAIN	1. IOADDEN deasserted 2. IODATEN asserted 3. IOCRLEN asserted
7	IODATAOUT	1. IOCRLEN asserted 2. IODS deasserted at the same time as UVDS 3. IOAS and IODATEN deasserted after either UVAS or IODS have become deasserted
8	IOEXIT	1. IOADDEN asserted 2. IOCRLEN asserted

Table 4-5 (cont.)

	State Name	List of Actions
9	REFRESH	1. IOEPS asserted 2. IOADDEN and IOCRLEN asserted

Table 4-6: Signals Used and Generated by IOCRL

Signal Name	Description
	Input Signals
IODMR	DMA request from the Ethernet Interface or one of the DMA devices
IORINPRG	Indicates the the DYRC for the Buffer RAM is perfomring a memory refresh
RESET	Reset signal generated at power up or by setting the HWRESET bit in the System CSR
UVAS	Address strobe form the MicroVAX bus
UVDS	Data strobe from the MicroVAX bus
UVWR	Write signal from the MicroVAX that indicates whether the operation is write or read
VAXR	Request signal from the MicroVAX that it wishes to use the I_O bus
WDHALT	Interrupt from the watchdog timer.

Table 4-6 (cont.)

Signal Name	Description
	Output Signals
IOADDEN	Enables the address drivers in the I_O bus isolator
IOAS	Address strobe for the I_O bus
IOCRLEN	Enables I_O bus control signals from the controlling device
IODATEN	Enables the data transceivers in the I_O bus isolator
IODMG	DMA Grant signal set in reply to IODMR
IODS	Data strobe for the I_O bus

4.12 Error Reporting

On both buses there can be two types of error:

1. Non-existent address
2. Parity error

On the MicroVAX bus, both of these are gathered by the System RAM's DYRC and are fed back to the MicroVAX through the ERROR interrupt.

In the case of the I_O bus, the errors are fed through to the MicroVAX's PWRFL interrupt. Two bits in the System CSR (PWRFL and IOERRSRC) show whether the interrupt was caused by such an error and whether the MicroVAX had control of the I_O bus at the time.

For more information on error detection and reporting, look at Chapter 10.

Chapter 5
Ethernet Interface

5.1 Overview

The DEC MicroServer communicates over the LAN through the Ethernet interface attached to the I_O bus. Figure 5-1 shows the Ethernet I/O block in more detail.

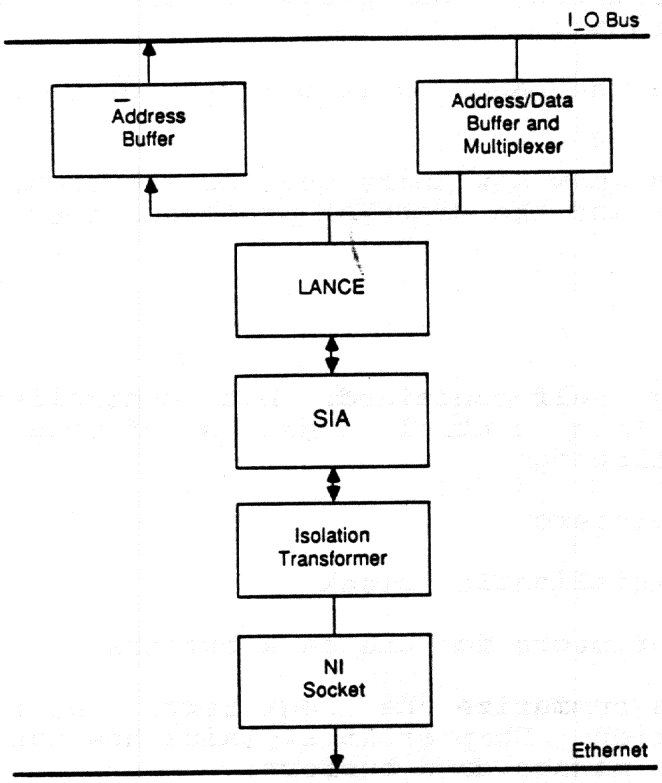


Figure 5-1: Ethernet I/O Block

The interface is based on a set of Ethernet devices: the LANCE (Local Area Network Controller for Ethernet), and the SIA (Serial Interface Adapter). These devices do much of the work necessary to send and receive information over the Ethernet, relieving the

MicroVAX processor of this work. For example, the devices handle conditions such as collision detect and retrying transmission.

Most of the work that the interface needs to do is carried out by the LANCE. This transfers information between the Ethernet and the Buffer RAM. These are DMA transfers during which the LANCE is master of the I_O bus.

The information in the Buffer RAM is held in buffers that the LANCE and the MicroVAX share. The descriptors for these buffers are used by the two devices to exchange information. For example, the LANCE puts the completion status of data transfers in these descriptors.

The SIA provides the interface between the TTL level signals that the LANCE uses and the differential signals used on the Ethernet. It also checks for collisions and provides the LANCE with a collision detection signal.

The Multiplexer adapts the LANCE (a 16-bit device) to the 32-bit I_O bus.

The following sections show how these devices are used, and how the Ethernet interface and the MicroVAX processor communicate.

5.2 LANCE

The LANCE is largely self-contained, but controlled by the MicroVAX software. This control consists of programming the device through the following:

- The LANCE registers
- The LANCE initialization block
- Rings of descriptors for the data buffers

The following sections summarize the registers, initialization, and sequence of operation. Chapter 15 explains how the LANCE and the MicroVAX processor manage data buffers.

5.2.1 Registers

The device has four 16-bit Control and Status Registers (CSRs), numbered from 0 to 3. Unlike registers in other devices, these are not directly accessable on the I_O bus. Instead, the MicroVAX accesses these registers through two ports, as Section

5.2.2 shows.

5.2.1.1 Device Control and Status Word (CSR0) - CSR0 is one word long, and provides control and status information for the whole device. Figure 5-2 shows the names used for each bit in the word, and Table 5-1 explains the meaning and use of each bit.

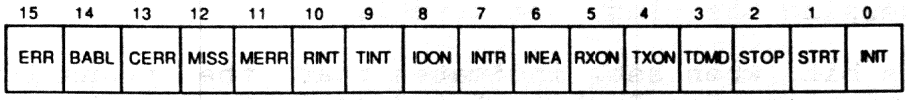


Figure 5-2: LANCE CSR0 Bit Definitions

Table 5-1: Content of the LANCE CSR0

Name	Meaning and Use
INIT	When set, this bit causes the chip to initialize itself. To do this, it reads an initialization block from the Buffer RAM.
STRT	When set, this bit makes the LANCE start processing information. It can send information, receive information, and do direct memory access on the I_O bus.
STOP	When set, this bit ends all external activity the chip may be involved in (such as data transmission), and clears the internal logic. Setting this bit is the same as the hardware performing a RESET operation. This bit is automatically cleared if either the INIT or STRT bit is set.

Table 5-1 (cont.)

Name	Meaning and Use
TDMD	When set, this bit causes the device to access the ring of transmit buffer descriptors without waiting for the poll time interval to expire. If clear, the device waits for the interval to expire before accessing the ring. See note 1.
TXON	This bit, when set, indicates that the transmitter is enabled. See Section 5.2.3 on initialization for details of how this bit is set. See note 2.
RXON	This bit, when set, indicates that the receiver is enabled. See Section 5.2.3 on initialization for details of how this bit is set. See note 2.
INEA	This bit controls whether the device generates interrupts through its INTR pin. When set, interrupts can be generated.
INTR	Indicates that one or more of the conditions that could cause an interrupt have occurred. The conditions are BABL, MISS, MERR, RINT, TINT, and IDON. The device generates an interrupt only if this bit and INEA are set. See note 2.
IDON	The device sets this bit when it has finished its initialization sequence. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.
TINT	The device sets this bit after updating an entry in the ring of transmit buffer descriptors. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.
RINT	The device sets this bit after updating an entry in the ring of receive buffer descriptors. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.
MERR	This device can set this bit only when it is master of the I/O bus. The bit indicates that the device did not receive a READY signal from memory within 25.6 microseconds of putting an address on the bus. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.

Table 5-1 (cont.)

Name	Meaning and Use
MISS	The device sets this bit when it loses a packet, or message, because no receive buffer was available. In waiting for the buffer, the internal silo has overflowed. This means that some data has been lost. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.
CERR	The device sets this bit if the collision indicator from the SIA comes on more than 2 microseconds after a transmission. That is, a collision has occurred despite the prevention measures built into the Ethernet. A collision after transmission of this sort is often used to test transceivers. See note 3.
BABL	This error bit indicates that the transmitter has been on longer than is necessary to send the longest possible packet, or message. This is equivalent to a transmit timeout error. When this bit and the INEA bit are set, the INTR bit is also set. See note 3.
ERR	This bit is generated from a logical OR of the BABL, CERR, MISS, and MERR bits. It indicates that one of the error conditions has occurred. The MicroVAX software can use this bit to determine the cause of an interrupt from the device.

Notes:
1. TDMD is a write-only bit, and can be set only.
2. TXON, RXON, and INTR are read-only bits.
3. IDON, TINT, RINT, MERR, MISS, CERR, and BABL are read- and clear-only bits. To clear one of these bits, write a 1. These bits are also cleared during reset, and when the STOP bit is set.

5.2.1.2 Initialization Block Address (CSR1 and CSR2) - These words hold the address of the initialization block (IADR). CSR1 holds the least significant 16 bits, and CSR2 the remaining 8 high order bits.

Bit 0 of CSR1 must be clear (0) because the LANCE accesses the initialization block in words, not bytes.

5.2.1.3 Bus Master Interface Definition (CSR3) - Only 3 bits of the 16 available are used in this register. These allow the controlling processor to modify the bus master interface, so allowing the LANCE to be used with many different types of processor. The bits in this word are cleared when the device is reset (through a RESET signal on the I_O bus) or when the STOP bit in CSR0 is set.

Figure 5-3 shows the allocation of bits in the word, and Table 5-2 gives more information on the use of each bit.

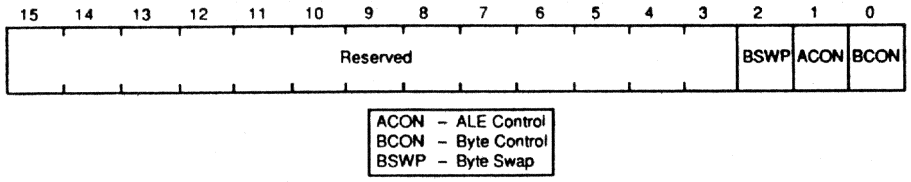


Figure 5-3: LANCE CSR3 Bit Allocation

Table 5-2: Content of the LANCE CSR3

Name	Meaning and Use
BCON	The value of this bit determines the function of pins 15 (BM0), 16 (BM1), and 17 (HOLD) on the LANCE. In the DEC MicroServer, BCON is cleared at power up or reset. In this case, pin 15 is a byte access indicator, pin 16 is a bus acknowledge output, and pin 17 is a bus request pin. Pin 15 is used to indicate whether a byte or word of data is being accessed. Pins 16 and 17 are used in DMA operations.

Table 5-2 (cont.)

Name	Meaning and Use
ACON	The value of this bit determines the assertive state of the ALE pin on the device when acting as master of the I_O bus. In the DEC MicroServer, this bit is clear, so the ALE pin has a high assertive state at the start of a cycle and a low assertive state during the cycle.
BSWP	In some processor architectures bits 15 to 8 of a word is the least significant byte, rather than the most significant (as in the DEC MicroServer). This bit tells the LANCE which byte ordering the controlling processor uses. This bit is always clear in the DEC MicroServer.

5.2.2 Access to the CSRs

The MicroVAX processor accesses the CSRs through two ports on the LANCE device:

- Register Address Port (RAP)
- Register Data Port (RDP)

These ports appear as words (aligned on a longword boundary) in the MicroVAX's address map. To read a register, the MicroVAX writes the CSR number (between 0 and 3) in the RAP and then reads the contents of the RDP.

Similarly, to set a value in one of the CSRs, the MicroVAX writes the CSR number in the RAP and then writes the desired value in the RDP.

The MicroVAX processor can read or write CSR0 at any time. However, it can access the other registers only when the STOP bit in CSR0 is set. Trying to access the remaining registers when the STOP bit is clear causes undefined data to be returned for a read, and the device ignores a write operation.

5.2.3 Initialization Block

During initialization, the device reads a block of information from I/O Buffer RAM. The MicroVAX indicates the start address of the block (IADR) by loading CSR1 and CSR2. When the INIT bit in CSR0 is set, the LANCE reads the initialization block and uses it to set the device's internal data structures. When the device has finished reading the block, it sets the IDON bit in CSR0. This causes a device interrupt if the INEA (also in CSR0) is set.

Figure 5-4 shows the content of the initialization block, and the succeeding sections explain the content of each field.

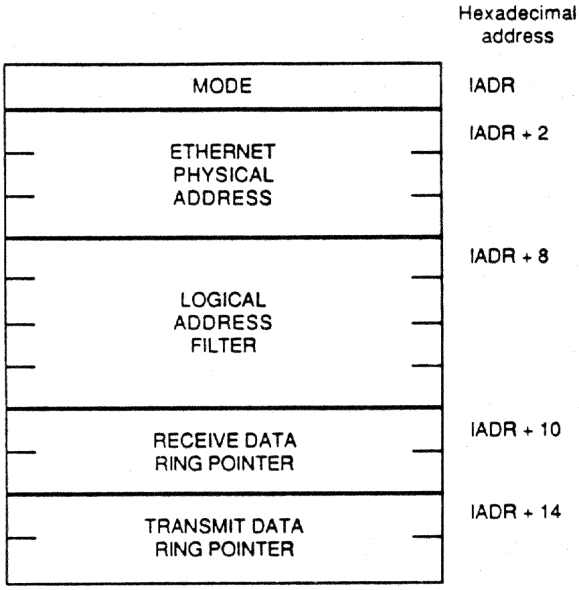


Figure 5-4: Format of the LANCE Initialization Block

5.2.3.1 Mode Field - The Mode field has 16 bits, and allows the MicroVAX to modify the LANCE's operating parameters. In normal operation, this field is clear. Its main use is for test software such as the DEC MicroServer's on-board test (see Chapter 12). Figure 5-5 shows the allocation of bits in the MODE field, and Table 5-3 gives more information on each bit.

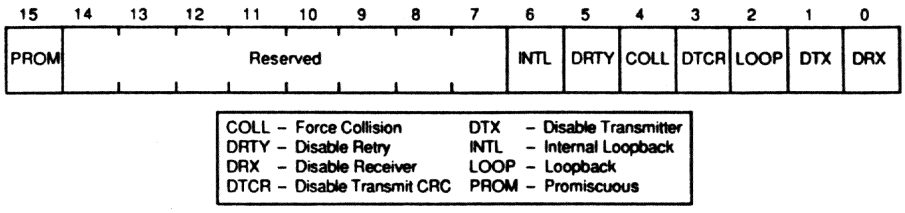


Figure 5-5: MODE Field Bit Allocation

Table 5-3: Content of the Mode Field

Bit Name	Meaning and Use
DRX	Setting this bit causes the Ethernet interface to reject all incoming packets. In addition, the receive buffer rings are not accessed. Setting this bit causes the RXON bit in CSR0 to be cleared when initialization finishes.
DTX	Setting this bit prevents the device from accessing the transmit buffer ring and so from sending any packets. Setting this bit causes the TXON bit in CSR0 to be cleared set when initialization finishes.
LOOP	Setting this bit allows the device to operate in full duplex mode. This feature is used by the DEC MicroServer's on-board test program to check the Ethernet connection. When set, this bit limits transmit packetsize to 32 bytes. The test software sends out these small packets and waits for them to return. Those returned may have four CRC bytes added if DTCR is clear.
DTCR	Normally, the transmitter generates and adds four CRC bytes for each packet it sends. By setting this bit, software can disable the CRC generation.

Table 5-3 (cont.)

Bit Name	Meaning and Use
COLL	This bit allows software to test the collision logic in the device. When set, a collision will be caused during the subsequent transmission. This results in 16 transmission retries and an error report in the transmit message descriptor (TMD). To use this bit, the LOOP bit must be set as well.
DRTY	Normally, the device will try up to 15 retransmissions of a packet if an error occurs on the original transmission. Setting this bit limits the number of retries to 1.
INTL	This bit is used with the LOOP bit to determine the sort of loopback operation to do. If this bit is clear, an external loopback is done. An internal loopback is done if this bit is set. The setting of this bit is ignored if the LOOP bit is clear.
PROM	Setting this bit causes the device to accept all incoming packets.

5.2.3.2 Ethernet Physical Address Field - The Ethernet Physical Address Field has 48 bits (three words) and contains the physical address that the device is to use. Figure 5-6 shows how the 48 bits are allocated to the three words. Notice that the least significant bit (PADR<0>) must be zero.

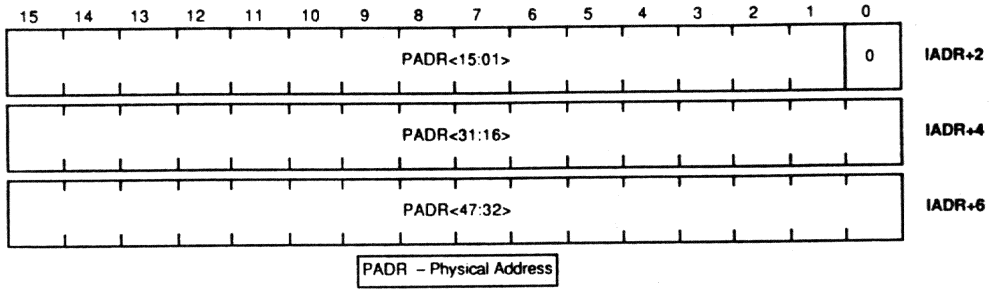


Figure 5-6: Ethernet Physical Address Field Bit Allocation

5.2.3.3 Logical Address Filter Field - This field determines which of up to 64 logical addresses the unit will recognize. Logical addresses have their least significant bit set to 1 to distinguish them from physical addresses.

To process a logical address, the device:

1. Puts the incoming 48 bit address through the CRC generator.
2. Copies the most significant six bits of the resultant CRC in a register.
3. Uses this number as a bit offset into the address filter.
4. If the appropriate bit is set, the address is accepted and the packet will be transferred to memory.

So by setting one or more bits in this field, the MicroVAX can establish the logical address(es) that the device will recognize.

Figure 5-7 shows the bit allocation of the filter.

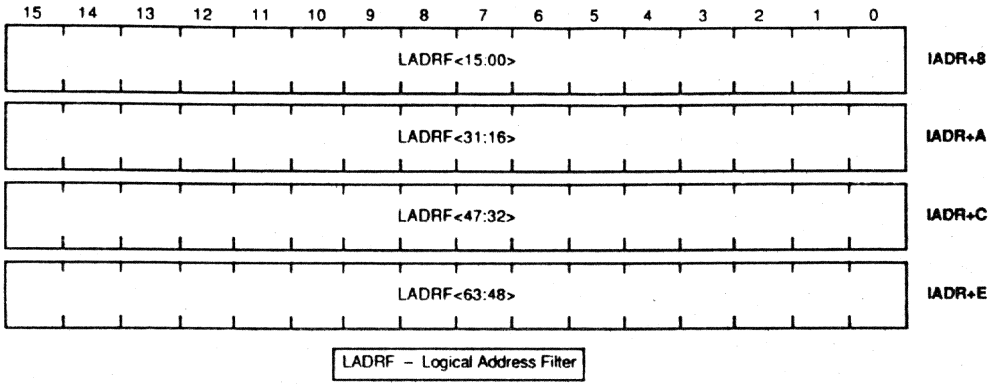


Figure 5-7: Logical Address Filter Bit Allocation

NOTE

The device recognizes the Ethernet Broadcast Address (address consisting only of 1s) as a special case and does not use this translation technique.

5.2.3.4 Receive Data Ring Pointer - This field contains the information on the ring of buffer descriptors for receive operations. Figure 5-8 shows the format of this field, and Table 5-4 explains the content of each field.

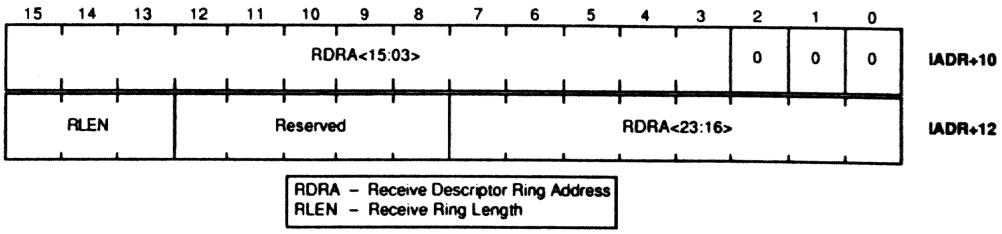


Figure 5-8: Format of the Receive Data Ring Pointer

Table 5-4: Content of the LANCE Receive Ring pointer

Field Name	Meaning and Use
RADR	The start address of the ring of receive buffer descriptors. Note that the lower three bits of the address are zero, and so the ring must be aligned on a quad word boundary.
RLEN	The number of entries in the ring. It is not an absolute number, but expresses the number of entries as a power of 2. Table 5-5 shows how many entries are indicated for each value of RLEN.

Table 5-5 shows how the value of RLEN corresponds to the number of entries in the ring.

Table 5-5: Meaning of the Ring Entry Field

Value of Length Field	Number of Ring Entries
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

5.2.3.5 Transmit Descriptor Ring Pointer - This field contains information on the ring of buffer descriptors to be used for transmit operations. Figure 5-9 shows the format of the field, and Table 5-6 gives more information about each entry.

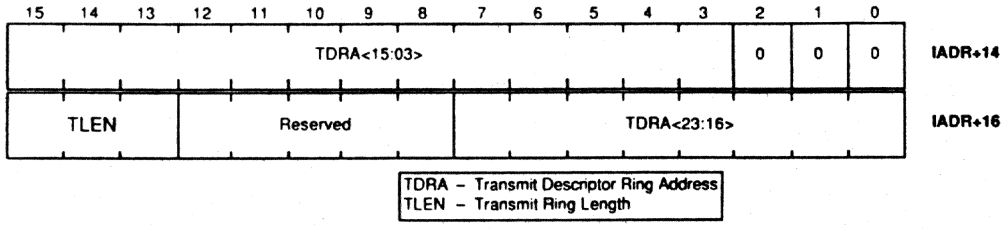


Figure 5-9: Format of the Transmit Data Ring Pointer

Table 5-6: Content of the LANCE Transmit Ring Pointer

Field Name	Meaning and Use
TADR	The start address of the ring of transmit buffer descriptors. Note that the lower three bits of the address are zero, and so the ring must be aligned on a quad word boundary.
TLEN	The number of entries in the ring. It is not an absolute number, but expresses the number of entries as a power of 2. Table 5-5 shows how many entries are indicated for each value of TLEN.

5.2.4 Operation Sequence

So, to summarize the programming sequence for the LANCE:

1. Set up the rings of receive and transmit buffer descriptors.
2. Set up the Initialization Block.
3. Set the address of the Initialization Block (IADR) in CSR1 and CSR2.
4. Set the value of CSR3 to 0.
5. Set the INIT bit (plus any other bits such as INEA) in CSR0.
6. When the resultant interrupt occurs, set the STRT bit in CSR0.

5.3 SIA

The SIA (device type AM7992) is a Manchester Encoder/Decoder compatible with both the IEEE 802.3 and the Ethernet specifications. The SIA converts the TTL level signals that the LANCE uses to the differential signals used on the Ethernet. The device also watches for collisions on the Ethernet, and provides the LANCE with a collision detection signal when necessary.

The SIA works under control of the LANCE and so needs no programming.

The device does not use the system clocks, and instead uses its own clock generated from a 20 MHz crystal, as Figure 5-10 shows.

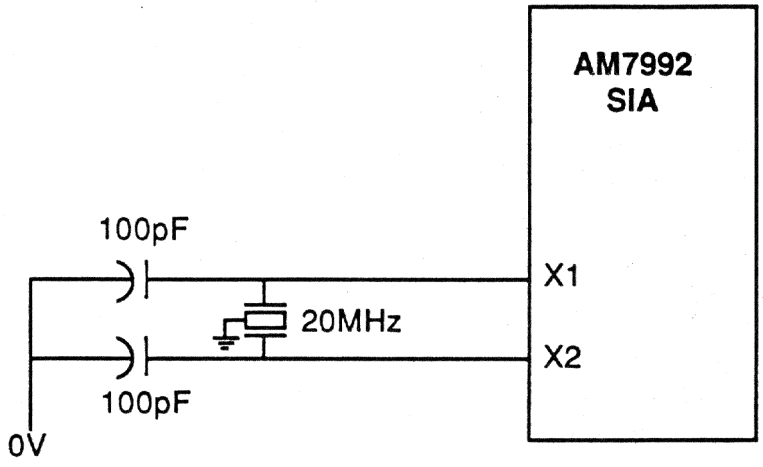


Figure 5-10: SIA Clock Source

5.4 Bus Multiplexer

The LANCE accesses data in 16 bit quantities. However, the Buffer RAM and the I_O bus handle 32 bit quantities. So the multiplexer is provided to interface the device to the I_O bus and Buffer RAM.

The multiplexer switches either the low order word of the bus or the high order word to the 16 address/data pins on the LANCE. Whether the high or low word is selected depends on the state of a control signal (called HISEL).

The low order word is selected in the following cases:

- When the LANCE is bus master and is sending out an address. In this case, the LANCE uses its data and address pins to send the lower 16 bits of the address.
- Data transfers that use low word addresses. This is determined by the state of Bit 1 on the I_O bus. If that bit is clear, the low word is being addressed. So in the subsequent data cycle, the low word is switched to the LANCE data and address pins.

- All transfers when the LANCE is a bus slave. For example, when the MicroVAX processor is setting the contents of the CSRs.

The high word is selected only when the LANCE is bus master and is transferring information to or from a high word address. Again, the state of Bit 1 of the address on the I_O bus determines this and, during the subsequent data cycle, the high word is switched to the LANCE data and address pins.

5.5 Interrupts Generated

The LANCE is the only device in the Ethernet Interface that can generate an interrupt. It can generate just one interrupt, but this can indicate a number of different events. To enable the interrupt, the MicroVAX must set the INEA bit in the LANCE CSR0 register.

Subsequently, the MicroVAX is interrupted (through the VIC) whenever any of the following events occur.

- Initialization complete -- the LANCE has read the initialization block.

- Transmission complete -- the LANCE has finished sending a packet. This event occurs when the final buffer in a chain is sent. On receiving the interrupt, the MicroVAX can check the status information in the buffer descriptor to make sure that the transmission was successful. See Chapter 15 for more information on buffers and their descriptors.

- Reception complete -- the LANCE has finished receiving a packet. This event occurs when the final buffer in a chain is received. When it receives the interrupt, the MicroVAX can check the status information in the buffer descriptor to make sure that the receive completed successfully. See Chapter 15 for more information on buffers and their descriptors.

- Memory error -- this event indicates that a memory access made by the LANCE has timed out.

- Missed packet -- this event indicates that the LANCE missed an incoming packet because no buffer was available to store it.

- Babbling -- this event indicates that the LANCE has been trying to send a message longer than the maximum allowed on an Ethernet.

To find out the cause of the interrupt, the MicroVAX examines the CSR0 register in the LANCE. This has bits allocated for each of the events, and so the MicroVAX can take appropriate action.

5.6 Ethernet Address PROM

Each DEC MicroServer unit has a unique Ethernet address. This is held in a 32 byte PROM on the MicroVAX bus. Since the PROM is a byte memory, each byte is aligned on a longword boundary, between addresses 26000080 and 260000FF (hexadecimal). This arrangement means that the bytes appear in the least significant byte of each longword.

The memory contains multiple copies of the address, together with checksums. All this ensures the integrity of the address.

Chapter 6
TMS Processor and Synchronous Control

6.1 Overview of the Synchronous I/O Logic

This chapter, together with Chapter 7, gives more detail on the synchronous side of the DEC MicroServer's architecture. Figure 6-1 is a detailed block diagram of the synchronous circuitry.

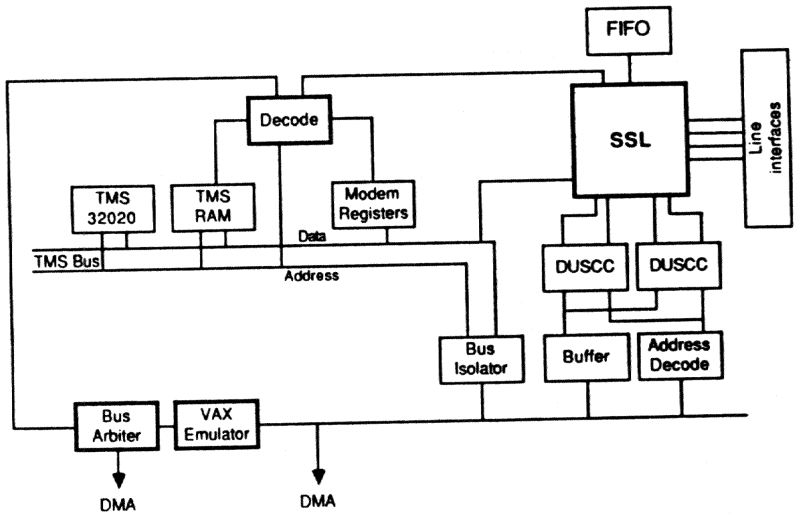


Figure 6-1: Detailed Block Diagram of the TMS and SCC Buses

Notice that the TMS bus has separate address and data lines. This is the only bus in the DEC MicroServer with this arrangement. All the other buses have multiplexed address and data lines.

Chapter 7 deals with the Serial Support Logic (SSL), the DUSCCs, and the line interfaces. This chapter deals with the rest of the blocks in Figure 6-1.

6.2 The TMS 32020 Processor

At relatively low data speeds, the MicroVAX processor could directly handle all the system's communications ports. However, the MicroVAX processor doesn't have the capacity by itself to handle the synchronous data speeds that the DEC MicroServer provides.

Therefore, the DEC MicroServer uses a second processor (a TMS 32020 device) to manage the synchronous side of the system. The TMS 32020 is a specialized signal processor and carries out these tasks:

- Setting up transfers between the I/O ports and Buffer RAM
- Checking incoming packets for errors
- Providing a convenient interface to the LES modem control and status registers

Using the TMS 32020 also enables a simpler interface between the MicroVAX software and the synchronous I/O ports. The MicroVAX uses control blocks and ring buffers under a similar scheme to the Ethernet Interface. The TMS 32020 provides detailed control of the I/O ports, using the information in the control blocks.

The TMS processor has many important and unusual features. Features of particular interest in the DEC MicroServer are:

- Separate program, data, and I/O address spaces
- Word (16 bit) architecture
- 544 words of RAM on the device
- Ability to move large amounts of data quickly due to fast cycle time (200 nanosecond) and high-speed program execution (between 2 and 3 mips for a typical instruction mix)

You can get full details of the device by reading the literature listed in Appendix I. The following subsections give some explanation of how the device is used in the DEC MicroServer.

6.2.1 Address Spaces

The architecture of the TMS 32020 has three separate address spaces:

- Program
- Data
- I/O

Addresses in each space start at zero, so for each address put on the address bus, the processor has to indicate which space it applies to. It does this by asserting one of the PROGMEM, DATMEM and IOADD signals. For example, when DATMEM goes low, the address bus contains the address of a location in data memory that the processor is accessing.

In the DEC MicroServer, the program and data address spaces use the same area of physical memory (see Section 6.3).

In the DEC MicroServer, I/O address space is used for:

- Synchronous control and status registers (see Section 6.4 for more details of these registers)
- Accessing the registers in the SSL
- Triggering and resetting an interrupt to the MicroVAX

Appendix B has details of the TMS 32020 address maps.

6.2.2 Interrupts

The DEC MicroServer uses four of the TMS 32020's maskable interrupts:

- **INT1** -- comes from either DMA controller (DMAINT). The synchronous control and status registers show which device generated the event. Note that the DMA request from the channel 3 receive logic is wired to INT1 and to the BIO line to enable the TMS to simulate a DMA channel. This acts to free a "real" DMA channel to act as a MicroVAX window into the I/O memory space.
- **INT2** -- comes from either DUSCC (SERINT). The synchronous control and status registers show which DUSCC generated the event.

- **RXINT** -- used for DUMP REQUEST from the MicroVAX.
- **TIMERINT** -- used for timer functions.

The device has one other maskable interrupt (called INT0), which is wired to the SSL board (see Section 7.4). Chapter 10 has details of the interrupts into the TMS processor, and those it generates.

6.3 TMS RAM

The TMS RAM is organized in banks of 16K words. There can be up to two banks in the system, and Figure 6-2 shows the general arrangement of the memory. The initial product uses one bank, leaving the other available for future expansion.

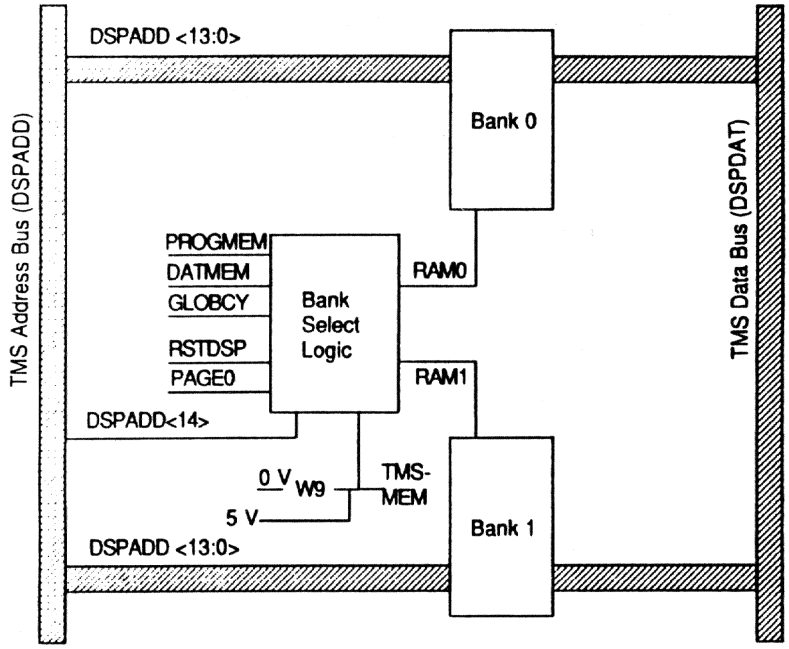


Figure 6-2: The TMS RAM Block

Most of the time the TMS processor is the only user of the memory. The bank selection logic uses bit 14 of the TMS address bus together with the PROGMEM, DATMEM and GLOBCY signals to select the appropriate memory bank.

The selection logic also enables the MicroVAX to access the memory (when loading the TMS firmware or when dumping the system). Here, the logic decodes the RSTDSP and PAGE0 signals to determine the correct bank to use.

The link (W9) is fitted only when the system includes two memory banks. Without the link, Bank 1 is not recognized, even if the bank itself is connected. The link also provides the appropriate level for the TMSMEM status indicator.

The way that the RAM is used depends on:

1. Whether one or two banks are fitted
2. Which processor is accessing the RAM

The following sections cover:

1. TMS access when there is only one memory bank
2. TMS access when there are two banks
3. MicroVAX access

6.3.1 TMS Access With One Memory Bank Fitted

With only one memory bank in place, any TMS access to the lower 32K of address spaces asserts the RAM0 selection signal. Both program and address spaces run from 0 to 32K, so overlap in physical memory. This gives maximum flexibility in the way the memory can be used.

6.3.2 TMS Access With Two Memory Banks Fitted

Two memory banks would be used when all the code and data for the TMS processor cannot fit into one bank. So using two banks means that one can be used as the program memory and the other as the data memory. However, this would lose some flexibility and restricts each address space to a maximum of 16K words.

The DEC MicroServer uses a compromise between separate data banks and overlapping address spaces. Table 6-1 shows which bank is selected for program and data addresses up to 32K.

Table 6-1: TMS Address Decoding for Two Memory Banks

Address Value (Hex)	Bank Selected	
	Program Address	Data Address
0 to 3FFF	RAM0	RAM1
4000 to 7FFF	RAM1	RAM0

To summarise: between addresses 0 and 3FFF, the RAM0 bank is used for program and the RAM1 bank for data. Between addresses 4000 and 7FFF, the RAM0 bank is used for data and the RAM1 bank is used for program.

6.3.3 MicroVAX Access

MicroVAX access to the TMS RAM is restricted. The selection logic only recognizes the VAX generated signals when the TMS processor is reset (through the RSTDSP bit in the System CSR).

The MicroVAX accesses the TMS memory through the DMA window. The MicroVAX sees the TMS memory as a 16K word address space in the DMA window. Thus, when two banks of TMS RAM are fitted (total memory space 32K words), the MicroVAX cannot directly address all 32K of available memory. Therefore, the MicroVAX also has to indicate which bank it is accessing.

The MicroVAX can use the TMSMEM bit in the System CSR to see how many memory banks there are. If there are two memory banks (TMSMEM is cleared) the MicroVAX uses the PAGE0 bit in the CSR to indicate the bank being addressed. When PAGE0 is clear, bank 0 is accessed when addresses between 0 and 3FFF (hex) are used on the TMS bus.

The TMS RAM is strictly a word memory. So the MicroVAX (through the DMA controllers) must use word access instructions only.

6.4 Synchronous Control and Status Registers

The I/O space in the TMS 32020 is used for registers that control the synchronous I/O ports and the SSL. Figure 6-3 shows how the I/O space is used.

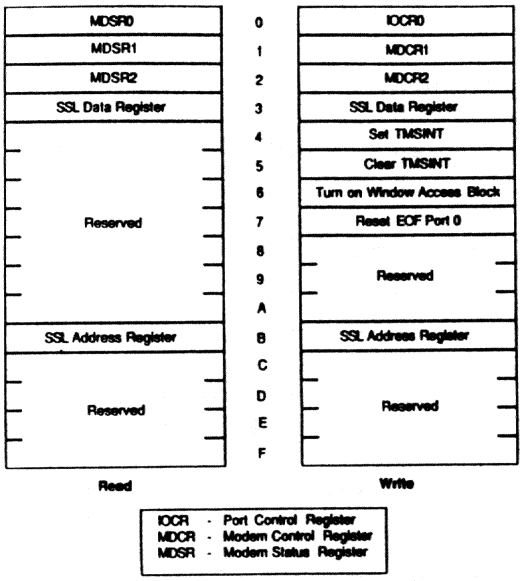


Figure 6-3: Allocation of the TMS32020 I/O Space

The first three addresses contain both the control and status registers. Writing to one of these locations changes the information in the appropriate control register. Reading that location fetches the content of the corresponding status register. For example, writing to location 1 sets the contents of MDCR1, while reading that location fetches the contents of MDSR1.

The control registers are:

- **IOCR0** -- Port control
- **MDCR1** -- Modem control for ports 0 and 1
- **MDCR2** -- Modem control for ports 2 and 3

The status registers are:

- **MDSR0** -- Cable code
- **MDSR1** -- Modem status for ports 0 and 1

- **MDSR2** -- Modem status for ports 2 and 3

Sections 6.4.1 to 6.4.7 give more details on how these locations are used, except for those containing the SSL registers. Section 7.4 deals with the SSL registers.

6.4.1 Port Control (IOCR0)

The OBT and the TMS firmware uses this register to set and reset certain port events:

1. Test mode -- used by the OBT so it can check out the ports
2. Reset End of Frame -- clear the end of frame flags
3. Receiver Interrupts -- generate the port receiver interrupts to the MicroVAX processor
4. Transmitter Interrupts -- generate the port transmitter interrupts to the MicroVAX processor
5. Enable the window access blocking mechanism

The register has five fields, as Figure 6-4 shows. Table 6-2 explains the content of each field.

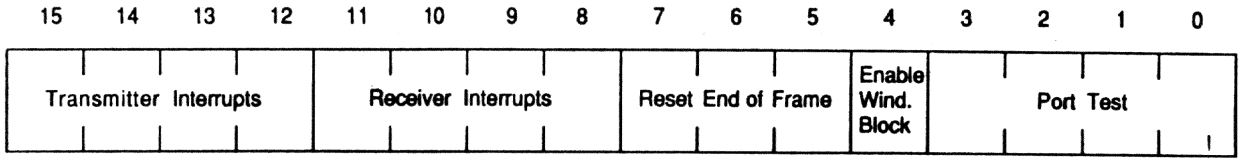


Figure 6-4: The Port Control Register (IOCR0)

Table 6-2: Fields in the Port Control Register

Field Name Active on LOW/HIGH	Meaning
Port Test LOW	The On-Board Test clears these bits to indicate that it is running and testing the ports. Once the normal firmware runs, it sets these bits. One of these bits is allocated to each of the synchronous ports. Figure 6-5 shows the bit allocation.
Enable Window Block HIGH	This field is the master control on the window access blocking mechanism used in 2M bits/s operation. Section 6.4.6 explains this mechanism in detail.
Reset End of Frame LOW	At the end of every received frame the DUSCC generates an End of Frame signal. This stops the clocking of data out of the receive FIFOs giving the TMS processor time to process the frame. To start reception of the next frame the TMS processor has to reset the flag, and it does this by clearing the appropriate bit in this field, and then setting it again. This field contains bits for ports 1, 2, and 3. The bit for port 1 is the least significant in the field. Section 6.4.7 explains the Reset End of Frame mechanism used for port 0.

Table 6-2 (cont.)

Field Name Active on LOW/HIGH	Meaning
Receiver Interrupt HIGH	The TMS processor uses four interrupts (named RXINT0 to RXINT3) to tell the MicroVAX that it has finished processing a ring entry for the appropriate receiver. The TMS processor generates these interrupts by setting and then clearing the appropriate bits in this field. One of these bits is allocated to each of the synchronous ports. Figure 6-5 shows the bit allocation.
Transmit Interrupt HIGH	The TMS processor uses a further four interrupts (named TXINT0 to TXINT3) to indicate that it has finished processing a ring entry for the appropriate transmitter. The TMS processor generates these interrupts by setting the appropriate bits in this field. One of these bits is allocated to each of the synchronous ports. Figure 6-5 shows the bit allocation.

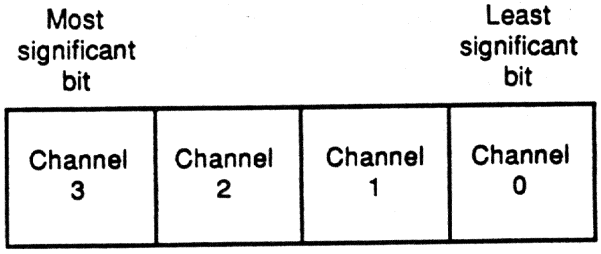


Figure 6-5: Allocation of Bits to I/O Ports

6.4.2 Modem Control Registers (MDCR1 and MDCR2)

The two modem control registers provide identical levels of control over each port, so the format of the two registers is very similar. MDCR1 contains information for ports 0 and 1, while MDCR2 contains information for ports 2 and 3. Only bit 15 is unique to each register.

Figure 6-6 shows the format of the two registers, and Table 6-3 explains what each field means.

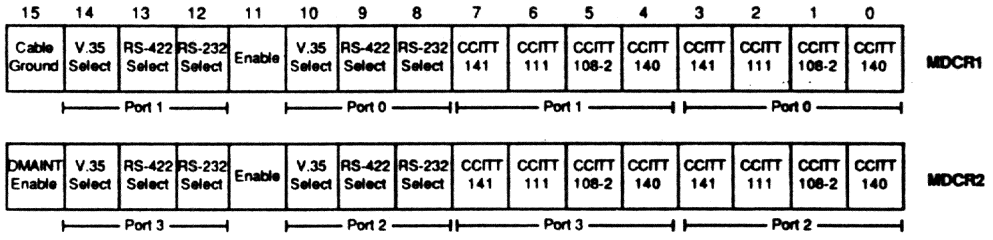


Figure 6-6: Modem Control Registers (MDCR1 and MDCR2)

Table 6-3: Fields in the Modem Control Registers (MDCR1 and MDCR2)

Field Name Active on LOW/HIGH	Meaning
CCITT 140 LOW	Set remote loopback (CCITT circuit 140). Tells the modem to set up a remote digital loopback.

Table 6-3 (cont.)

Field Name Active on LOW/HIGH	Meaning
CCITT 108-2 LOW	Data Terminal Ready (CCITT circuit 108-2). Indicates to the modem that the DEC MicroServer is ready to send or receive information on the channel.
CCITT 111 LOW	Data Rate Signaling Selector (CCITT circuit 111). For modems that can use one of two signaling rates, this circuit lets the DEC MicroServer select which to use on this channel. When this circuit is enabled, the higher of the two rates is used.
CCITT 141 LOW	Set Local Loopback (CCITT circuit 141). Tells the local modem to loopback data.
RS-232 Select LOW	Enables RS-232 as the line protocol.
RS-422 Select LOW	Enables RS-422 as the line protocol.
V.35 HIGH	Enables V.35 as the line protocol.

Table 6-3 (cont.)

Field Name Active on LOW/HIGH	Meaning
Enable HIGH	Enables the modem control values set up in bits <7:0> of the register. Notice that there is only one Enable bit in each register. So, setting this bit enables the modem signals for the appropriate pair of channels, and enabling individual channels is not possible. Enable switches on the line driver (for example, RS-422) selected by the previous bits. Whatever values are contained in bits <7:0> are now driven onto the line. When not enabled, the modem control lines are in the high impedance state (tri-stated).
Code Ground LOW	Controls the sense of the cable code values in MDSR0. This enables the On-Board test to invert the code values to check that the register has no bits stuck at particular values. Section 6.4.3 shows you how this field is used.

Table 6-3 (cont.)

Field Name Active on LOW/HIGH	Meaning
DMA INT Enable LOW	The BIO pin is always connected to the TMS. DMA INT Enable enables the DMA Interrupt from the channel 3 receiver. The BIO pin is used in the interupt service to distinguish between a normal DMA INT and a channel 3 interrupt. <paragraph> The DMA channel is permanently configured as a window into the I/O memory, so the TMS software emulates the DMA function by windowing data through to the I/O memory from the DUSCCs. <paragraph> If DMA INT Enable is clear, the processor will be interrupted each time the channel is needed for data transfer.

Notes:

1. The system software must make sure that only one of the protocol select bits is set at any one time.
2. On reset, all the CCITT circuits are disabled, V.35 is clear (0), and RS-232 and RS-422 are set (1).
3. Except for V.35 Select and Modem Control Enable, the processor disables a feature by setting the appropriate bit to 1. In the case of V.35 Select and Modem Control Enable, the processor clears the appropriate bit to 0.

6.4.3 Cable Code (MDSR0)

Each type of adapter cable has a unique code. By reading the Cable Code register, the TMS processor can determine which cable is attached to which port. Figure 6-7 shows the format of the register.

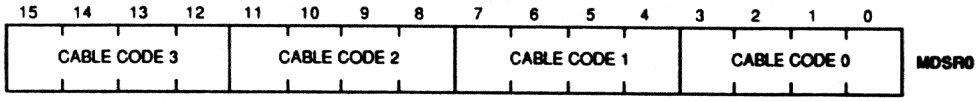


Figure 6-7: The Cable Code Register (MDSR0)

There is one field for each channel, and the values that can appear are controlled by the Code Ground field of MDCR1. If Code Ground is set, the values that can appear in each field are:

0 No cable connected

1 V.35 cable

2 RS-423/V.24 cable

3 X.21 cable

4 RS-422/V.36 cable

B Manufacturing loopback connector

F H3199 loopback connector

If Code Ground is clear, the opposite sense is used and the values that can appear are:

1 H3199 loopback connector

4 Manufacturing loopback connector

B RS-422/V.36 cable

C X.21 cable

D RS-423/V.24 cable

E V.35 cable

F No cable connected

6.4.4 Modem Status Registers (MDSR1 and MDSR2)

The two modem control registers provide identical status information for each port, so the formats of the two registers is very similar. MDSR1 contains information for ports 0 and 1, while MDSR2 contains information for ports 2 and 3. Only bits 0 and 8 are unique to each register.

The registers are directly linked to the modem lines and so provide instantaneous indications of their state.

Figure 6-8 shows the format of the two registers, and Table 6-4 explains what each field means.

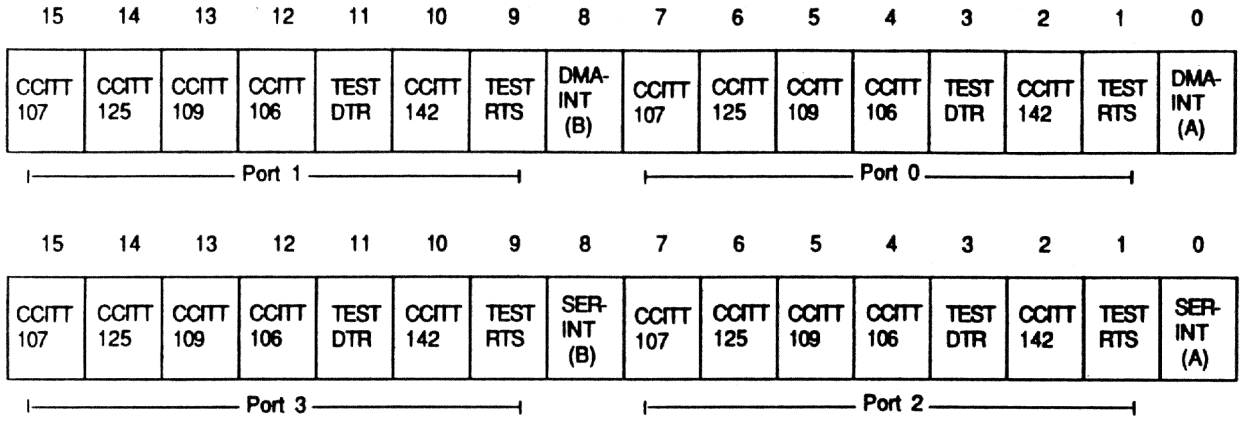


Figure 6-8: Modem Status Registers (MDSR1 and MDSR2)

Table 6-4: Modem Status Bits for MDSR1 and MDSR2

Field Name	Meaning
Test RTS	These fields, and the Test DTR fields are used to test the RS-232 interface. When used in full loopback there are insufficient line receivers to fully test RS-232. So, these fields are used to test the "spare" signals. To test one of these signals, the OBT enables the appropriate signal, and then examines this register to see if the field is enabled.
CCITT 142	Current state of Test Indicator (CCITT circuit 142). If this bit is clear, then circuit 140 or 141 has previously been used to set up data loopback.
Test DTR	See the description of Test RTS.
CCITT 106	Status of the Ready for Sending signal (CCITT circuit 106, or Clear To Send). This bit is asserted in response to a Request to Send signal to indicate that the modem is ready to send information.
CCITT 109	Carrier Detect (CCITT circuit 109). Used to indicate that the line carrier signal is within the prescribed limits.
CCITT 125	Calling indicator (CCITT circuit 125, or RING). Indicates whether a calling signal is being received on the line.
CCITT 107	Data Set Ready (CCITT circuit 107). Indicates the the modem is ready to receive data.

Table 6-4 (cont.)

Field Name	Meaning
DMA INT	These bits contain the interrupt status of the DMA devices. The INT2 interrupt on the TMS processor is asserted when either of the DMA devices generates an interrupt. The processor can then tell which device generated the interrupt by reading these bits. The bit that is asserted shows which device interrupted.
SER INT	These bits contain the interrupt status of the DUSCC devices. The INT1 interrupt on the TMS processor is asserted when either of the DUSCCs generates an interrupt. The processor can then tell which device generated the interrupt by reading these bits. The bit that is asserted shows which device interrupted.
In these registers, all bits use negative logic. That is, the presence of a signal is indicated by the appropriate field being clear. For example, when one of the DMA devices generates an interrupt, the appropriate field in MDSR1 is clear while the other DM INT field is set.	

6.4.5 TMS Interrupt Control (TMSINT)

An extra signal (called TMSINT) allows the TMS processor to interrupt the MicroVAX processor. The TMS generates the interrupt each time it changes the Global Status Register or any of the Port Status Registers. These registers are part of the Synchronous I/O Control Block, which Section 6.6 deals with in detail.

To generate the interrupt, the TMS processor performs a write operation on I/O address 4. The supplied data is not important - it is the write operation itself that triggers the interrupt. The processor clears the interrupt by performing a write operation on I/O address 5. Again, it is the operation that

resets the interrupt, not the data supplied.

6.4.6 Window Access Hold-Off

The device used in the I/O ports (the DUSCC) has a small transmit FIFO. During a transmission, this FIFO must not be allowed to empty or else the transmission may underrun and cause the packet to be aborted.

The DUSCC is supplied with data through the DMA devices, which the TMS processor also uses to fetch control information from the Buffer RAM. So, there are two potential users of the SCC. At data speeds below 256K bits/s, this presents no problem as the port can use the bus often enough to prevent the FIFO from emptying.

At higher speeds, however, there is a danger that the TMS processor can hold off the DUSCC long enough for the FIFO to empty. To avoid this problem, the DEC MicroServer includes a blocking mechanism to prevent the TMS processor accessing the SCC bus while a frame is being transmitted. This prevents transmitter underrun at high speeds.

The blocking mechanism is under full control of the TMS processor and is reset at the end of each frame.

The TMS processor enables the blocking mechanism by setting the Enable Window Block field in IOCR0. Then, on starting the transmission of each frame, the processor performs a write operation on I/O address 6. This enables the blocking mechanism, preventing a TMS window access overriding a DMA transfer to the DUSCC.

At the end of the frame, the TMS processor is interrupted and the blocking mechanism is disabled. The TMS can then use the SCC bus to access the Buffer RAM without restriction. The TMS processor enables the blocking mechanism again before sending the next frame.

6.4.7 Reset End of Frame 0

The Reset End of Frame field for Port 0 is in the TMS I/O space instead of in IOCR0. To reset the flag, the TMS processor simply performs a write operation on I/O address 7.

6.5 Control of the I/O Processor

Ultimately, the software in the MicroVAX has control of the TMS processor. The software gives "commands" to the I/O processor to do tasks such as:

- Turn lines on and off
- Initialize the receivers and transmitters
- Send and receive information over the synchronous lines

In return, the TMS processor returns status information that shows whether a command completed successfully, or whether there's been a change in the state of the I/O ports.

The two processors use an area of the Buffer RAM to exchange commands and status information. This area is called the **Synchronous I/O Control Block**.

Every 2 ms, the TMS processor looks in this block to see if there are any new commands. The MicroVAX is interrupted if there are any changes in status information.

Buffers for the information that the system sends and receives are also in the Buffer RAM. Each buffer has a header that the TMS processor and the MicroVAX use to exchange information such as the location of buffers, the size of messages in those buffers, and error information.

Section 6.6 explains the format and content of the control block. Chapter 15 covers the details of the buffers and their headers.

The MicroVAX processor can also control the TMS processor through the RSTDSP bit in the system CSR. When the TMS processor is running, the MicroVAX cannot access the TMS RAM or the TMS processor's address space. However, to reload code or to dump information, the MicroVAX has to access these devices. To do this, the following protocol is used:

1. The MicroVAX clears, then sets, the RSTDSP bit in the system CSR, forcing the TMS to dump its on-chip memory.
2. The MicroVAX waits approximately 10 ms for the TMS to complete the dump.
3. The MicroVAX then clears the RSTDSP bit to allow itself access to all locations in the TMS address space (whilst the TMS is reset).

4. The MicroVAX then sets the RSTDSP bit to restart the TMS.

6.6 Synchronous I/O Control Block

The Synchronous I/O Control Block is the interface between the MicroVAX and the TMS processor. Each of the fields in the block is written by one processor and read by the other. This, together with the inherent exclusive access to the I_O bus, means that none of the fields is shared. So, there is no need for the overhead of an interlocking mechanism.

The block is 128 bytes long, and Figure 6-9 shows its overall format. The offsets give the relative addresses of each section for both processors. The block is aligned on a longword boundary and must be in the same 16K byte block as the synchronous ring buffers.

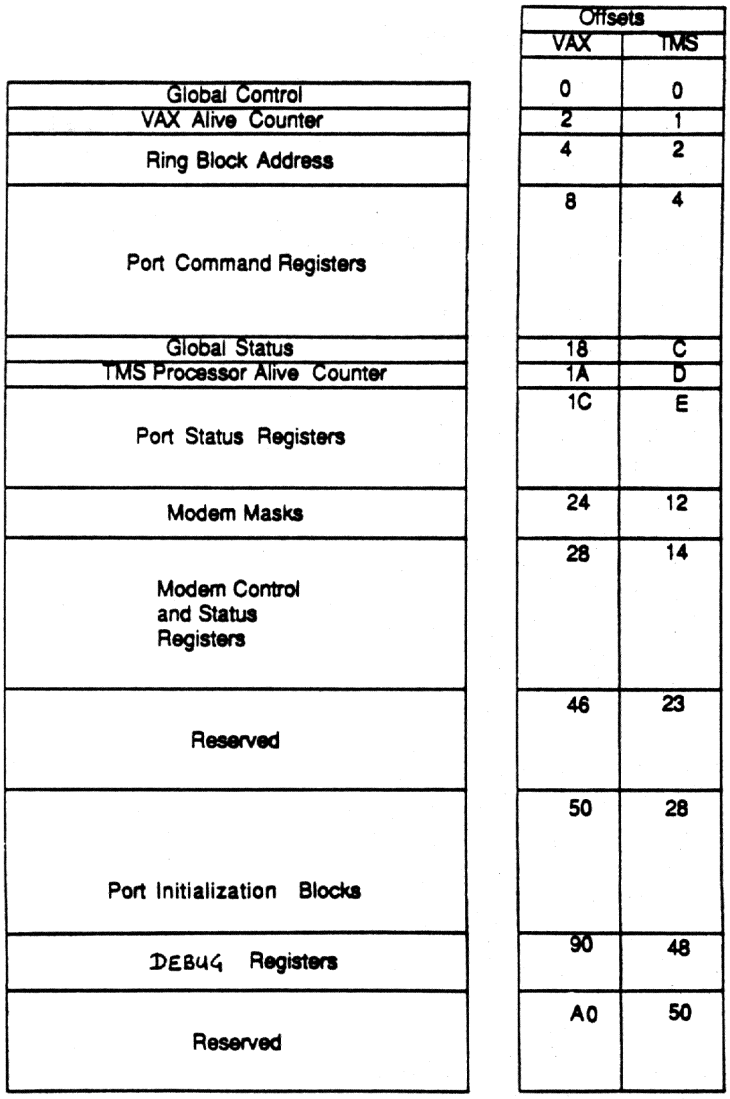


Figure 6-9: Format of the Synchronous I/O Control Block

The following sections explain the content of each section of the control block in more detail. These sections use figures to show the make up of each section more clearly. The offsets shown are those from the start of the Synchronous I/O Control Block.

6.6.1 Global Control

The Global Control register provides overall control of the TMS processor. At present, only bit 0 of the register is used. When set, this bit indicates that high data speeds (greater than 256K bits/s) are to be used. The TMS processor's firmware uses this bit to determine whether to use the Window Blocking Mechanism against underrun at high speeds.

6.6.2 Ring Block Address

This section contains the address of the synchronous ring block. The first word contains the low order bits. See Chapter 15 for details of the synchronous ring block and how it's used.

6.6.3 Port Command Registers

The port command registers contain four pairs of registers, one pair for each port as Figure 6-10 shows.

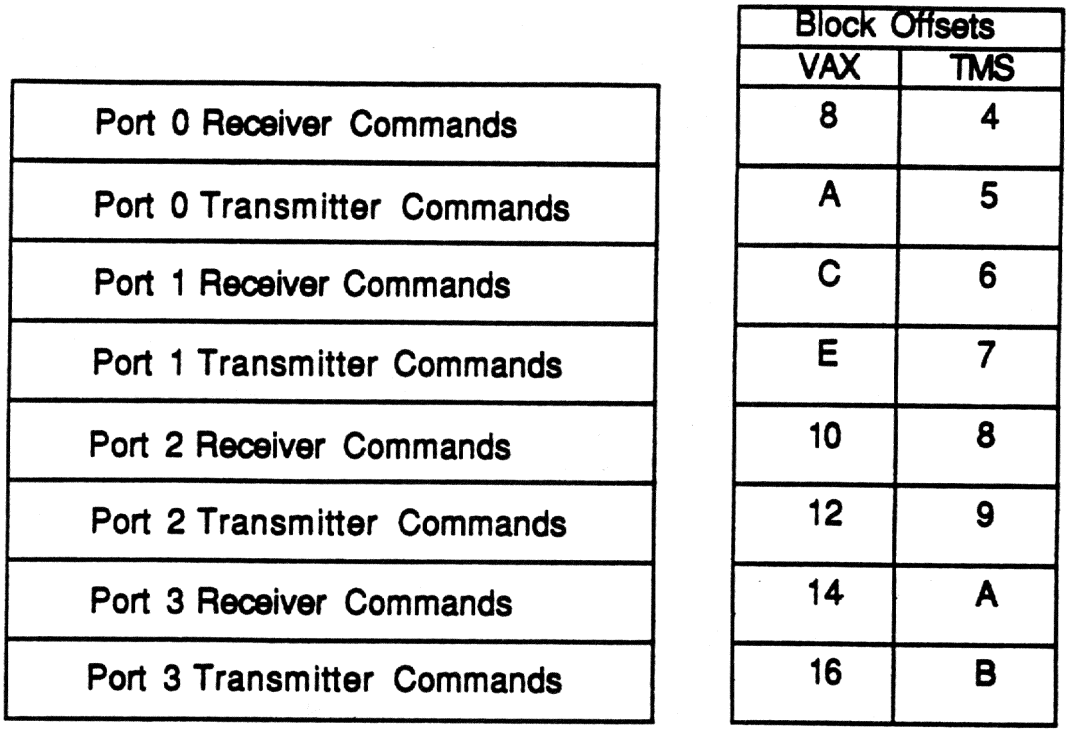


Figure 6-10: Port Command Registers

As the figure shows, there are separate transmit and receive registers for each port. Table 6-5 summarizes the values that can appear in each register, and what each value means. For details of each command, see Chapter 14.

Table 6-5: Synchronous I/O Commands

Value (Hex)	Function
	Receiver Commands
0	TMS processor has read the previous command
1	Initialize receiver

Table 6-5 (cont.)

Value (Hex)	Function
2	Start receiver
3	Stop receiver immediately
6	Turn the line off
	Transmitter Commands
0	TMS processor has read the previous command
1	Initialize transmitter
2	Start transmitter
3	Stop transmitter and abort current packet
4	Stop transmitter after the current packet
5	Abort the current packet and flush any pending packets
6	Switch the line off
7	Complete the current packet and flush any pending packets

6.6.4 Global Status

The Global Status Register gives the overall status of the synchronous side of the DEC MicroServer. Figure 6-11 shows the format of the register, and Table 6-6 lists the meaning of each bit.

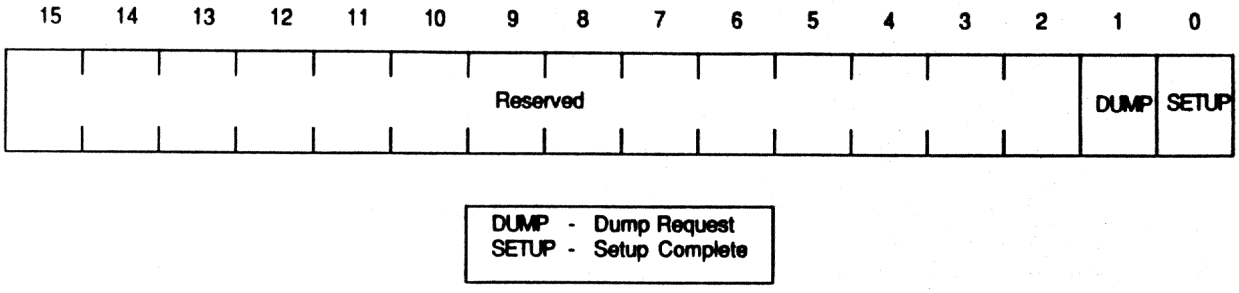


Figure 6-11: The Format of the Global Status Register

Table 6-6: Meanings of Bits in the Global Status Register

Field Name	Meaning and Use
SETUP	When set, this indicates that the firmware has been loaded and the TMS processor has finished initializing itself.

Table 6-6 (cont.)

Field Name	Meaning and Use
DUMP	When set, this indicates that the TMS processor is requesting a system dump. This occurs if the processor detects an internal error, or in response to a dump request from the MicroVAX processor. The TMS processor saves its context in the TMS RAM, and sets this status bit to indicate that it is ready to dump.

NOTE

The TMS processor generates the TMSINT interrupt each time it changes the value of this register.

6.6.5 TMS Processor Alive Counter

The TMS processor alive counter helps the MicroVAX determine whether the TMS processor is operating normally. The TMS processor tries to increment this counter on each of its own timer ticks (that is, every 2 ms). The MicroVAX reads the counter every 100 ms. If it reads the same value on two successive occasions, the MicroVAX assumes that the TMS processor has become deadlocked and so initiates a system dump.

6.6.6 Port Status Registers

There are four port status registers, one for each of the synchronous ports. These registers reflect the current state of the ports, and the TMS processor changes them each time the ports change. Figure 6-12 shows the offsets for these registers.

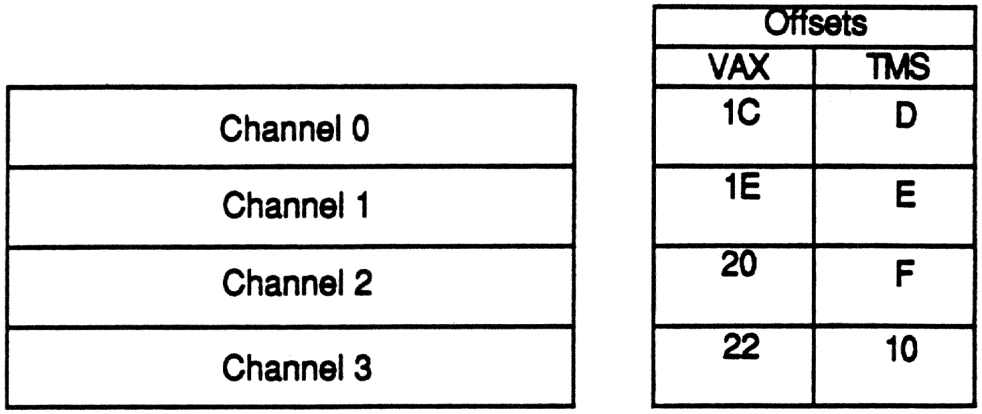


Figure 6-12: The Port Status Registers

Each of the registers has the same format, and Figure 6-13 shows this. The two status fields contain values that indicate the port's status. Table 6-7 shows the values that can appear and what each means.

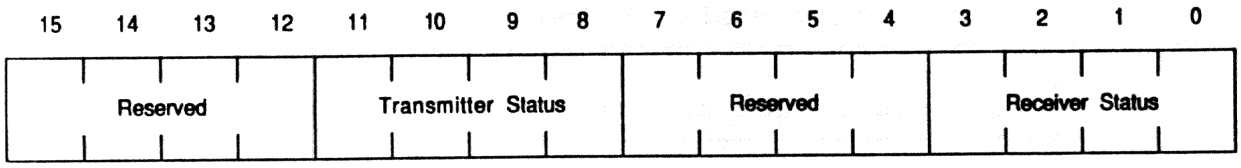


Figure 6-13: Format of a Port Status Register

Table 6-7: Values in a Synchronous Port Status Register

Value (Hex)	Meaning and Use
	Receiver Status Values
0	Channel not active
1	Channel booted but not initialized
2	Receiver stopped
3	Receiver running

Table 6-7 (cont.)

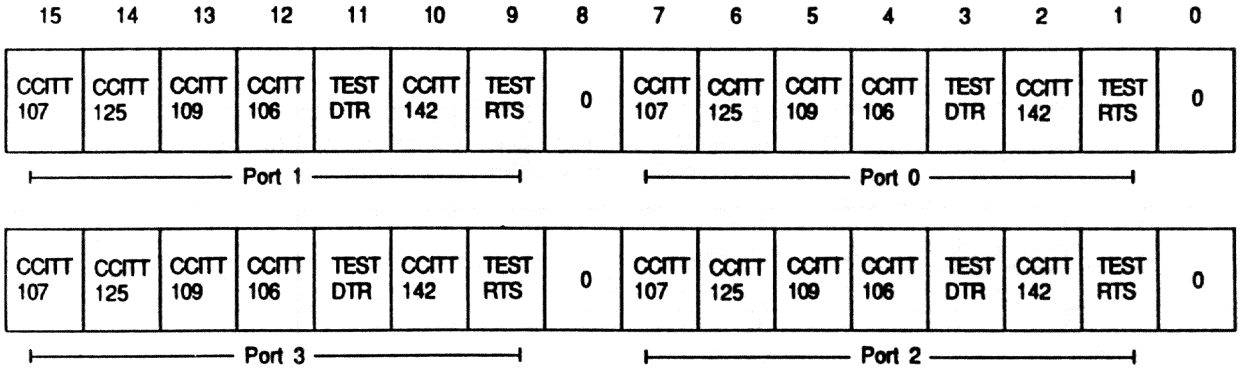
Value (Hex)	Meaning and Use
E	Invalid configuration
	Transmitter Status Values
0	Transmitter not active
1	Transmitter booted but not initialized
2	Transmitter stopped
3	Transmitter running
E	Invalid configuration

NOTE

The TMS processor generates the TMSINT interrupt to the MicroVAX each time it changes the value of this register.

6.6.7 Modem Masks

Figure 6-14 shows the format of the modem mask registers.



	Offsets	
	VAX	TMS
Port 1/0	24	12
Port 3/2	26	13

Figure 6-14: Modem Mask Registers

For each port, the bits correspond to those in the Modem Status Registers (see Section 6.4.4). By setting a bit in the mask, the MicroVAX will receive the MODCHGINT interrupt each time that bit in the appropriate Modem Status Register changes.

Only one interrupt is generated for any of the many possible changes in a port's modem status. This occurs whether or not that change caused more than one of the masked bits to change, but only when the changed bit corresponds to the modem masks.

6.6.8 Modem Control and Status Registers

The registers in this section let the MicroVAX set values in the modem registers and read the modem status registers without needing access to the actual registers on the TMS bus. The section is divided into four parts (one for each port) and Figure 6-15 shows the format of each part.

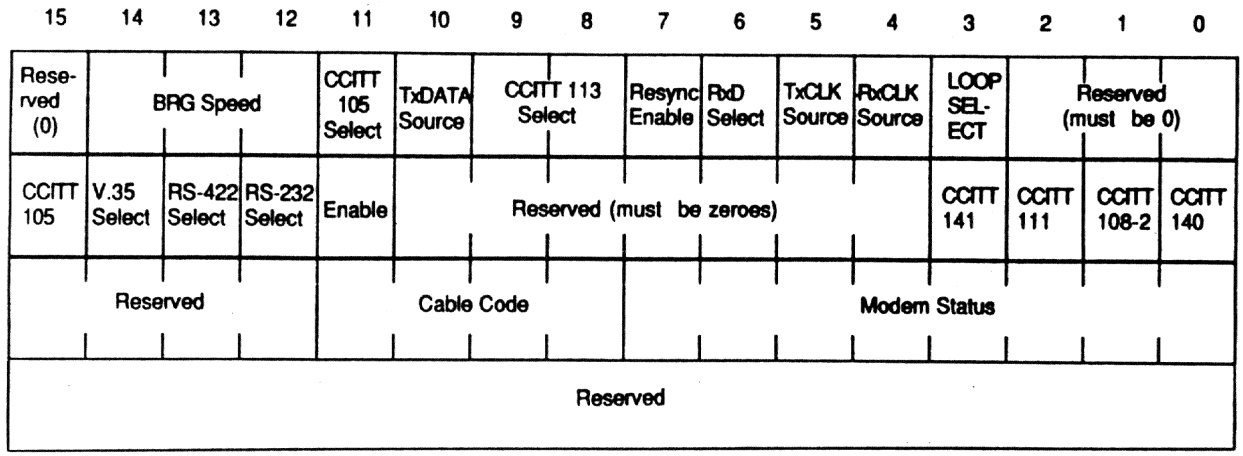


Figure 6-15: Modem Control and Status Registers in the Synchronous I/O Control Block

The Clock and Data Select register lets the MicroVAX set the initial clock and data sources for the SSL without accessing locations on the TMS bus. Section 7.4.3.1 deals with the SSL's register that holds this information. See that section for information on each field.

To enable any function in the control registers, the MicroVAX sets the appropriate bit. The TMS processor's firmware translates this into the correct setting for the actual control register.

The status part of the third register is a modified copy of that channel's modem status. However, the TMS processor modifies the settings so that a bit set means the function is enabled. The TMS processor tries to update this part on every clock tick. However, some changes can take up to 3 clock ticks to be reflected in the Synchronous I/O Control Block.

The cable code part of the second register contains a copy of the port's cable code.

Changing the sense of some signals simplifies the interface to the MicroVAX. Any bit that is set means that the appropriate feature is enabled. The MicroVAX does not have to determine whether individual features are active low or high.

6.6.9 Port Initialization Blocks

This port initialization section contains the information each port uses when it executes an initialization command.

The section has four equal parts, one for each synchronous port. Figure 6-16 shows the offset of each block in the section. Figure 6-17 shows the format of an initialization block.

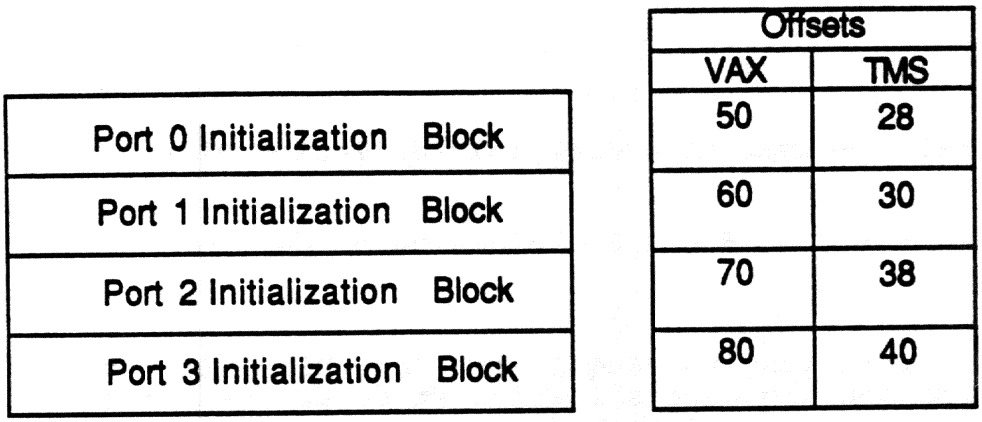


Figure 6-16: Offsets of the Initialization Blocks For Each Port

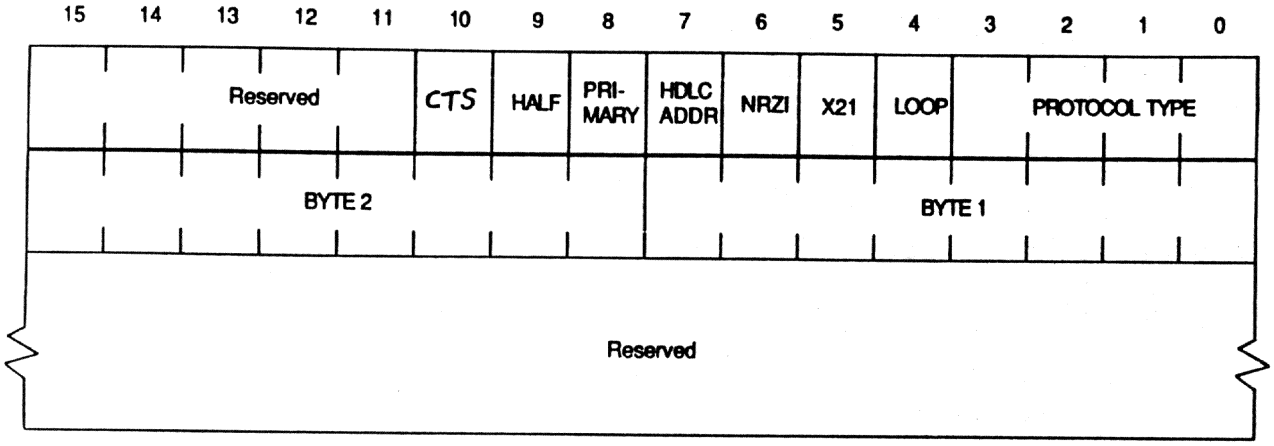


Figure 6-17: Format of a Port Initialization Block

Sections 6.6.9.1 to 6.6.9.2 explain each register in more detail.

6.6.9.1 Initialization Register - The MicroVAX can use this register to set the port's initial state at load time. Table 6-8 explains the meaning of each field.

Table 6-8: Meaning of Fields in the Initialization Register

Field Name	Meaning and Use
PROTOCOL TYPE	This field sets the port's protocol. The values that can appear in this field are: 0. No protocol 1. DDCMP-sync 2. HDLC
LOOP	Reserved for future use.
X21	Reserved for future use.
NRZI	When set, the port operates in NRZI mode.
HDLC ADDR	HDLC station addresses can use one or two bytes. The initialization block can handle both sorts of address, but the port needs to know which to use. The port uses the value of this field to determine the length of the station address. If the field is set, both bytes of the Station Address register are used. If the field is clear, only the low-order byte is used.
PRIMARY	When this field is set, the port acts as the primary station in an HDLC environment.
HALF	When set, all communication through the port uses half duplex protocols.
CTS	When set, this enables the transmitter. The CTS bit is tested prior to starting a transmit.

Table 6-8 (cont.)

Field Name	Meaning and Use
Note: The HDLC ADDR and PRIMARY fields are meaningful only if the PROTOCOL TYPE is set to HDLC.	

6.6.9.2 Station Address - When the HDLC protocol is used, the port needs to have a station address. This register contains the address it is to use, which can use one or both bytes. The value of the HDLC ADDR field in the Initialization Register determines how many bytes to use.

When HDLC ADDR is set, both bytes are used. When the field is clear, only the low order byte (Byte 1) is used.

Chapter 7
Synchronous I/O ports

7.1 Overview

This chapter deals with the synchronous I/O ports themselves, which use the following major components:

- Two DUSCCs
- The SSL
- The FIFOs
- The line drivers and connectors

Figure 7-1 shows these sections of the Synchronous I/O part in the DEC MicroServer.

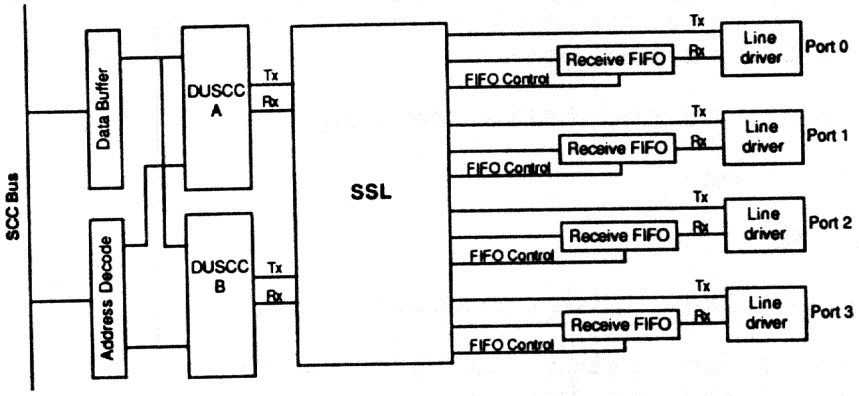


Figure 7-1: The Architecture of the Synchronous I/O Ports

Data flows between the Buffer RAM and the line connectors through the DMA devices, the DUSCCs, the SSL, and the line drivers. The TMS processor sets up and controls each transfer. The processor also sets up the DUSCCs and the SSL through their control

registers.

The DUSCCs do much of the necessary protocol processing and deal with CRCs. The SSL provides miscellaneous functions such as bit rate generation and FIFO control. The FIFOs themselves are each 16K bits long and there is one for each receive line. These help the DEC MicroServer to handle high data rates and help to isolate the DUSCCs from variations in line data rate.

The following sections give more details on each of the major components.

7.2 DUSCC

The DUSCC (**D**ual **U**niversal **S**ynchronous **C**ommunications **C**ontroller) provides two receiving and two transmitting communications channels, and so the DEC MicroServer uses two of the devices (device type SCN68562) to support its four synchronous lines.

The DUSCC can handle bit- and character-oriented synchronous protocols and can generate the necessary CRC values. It can also handle asynchronous communications, but the DEC MicroServer does not use this feature.

Data rates can be up to 4 MHz, and the channels can operate in full duplex or half duplex modes. There is also an internal loopback mode that the DEC MicroServer's on-board test uses to check that the DUSCCs are working.

Each channel contains the following components:

- Transmitter
- Receiver
- Counter/Timer (programmable)
- Bit-Rate Generator (programmable) -- one for each transmit and receive pair
- Two internal 4-character FIFOs -- one for the transmitter and one for the receiver

The channels in a device are referred to as Channel A and Channel B. The DUSCCs are known as DUSCC A and DUSCC B. Table 7-1 shows which DUSCC channel provides each DEC MicroServer communications port.

Table 7-1: MicroServer ports and DUSCC channels

Port No.	DUSCC Device	DUSCC Channel
0	A	A
1	A	B
2	B	A
3	B	B

Although the DUSCC can handle asynchronous protocols, the following description deals solely with its synchronous capabilities. If you need information on the asynchronous capabilities, and how to use them, refer to the DUSCC literature listed in Appendix I.

7.2.1 Registers

The DUSCC is controlled by setting appropriate values in its registers. Unlike the LANCE, these registers are directly addressable by both the MicroVAX and the TMS processor.

The registers appear as part of the TMS processor's code/data space and so can be accessed like any other memory locations. The MicroVAX accesses the registers through the DMA devices, as Chapter 8 shows.

7.2.2 Types of Register

In all, there are 51 registers in the device. Some of these appear twice (one for each channel), and others apply to the complete device. Table 7-2 lists the registers, their addresses, and their functions. To avoid unnecessary duplication, the channel-dependent registers appear only once in the table.

Section 7.2.4 summarizes the registers and how they are used. For more detailed information on a particular register, turn to the section shown in the right hand column of Table 7-2. Note that the relative addresses given in column two are word offsets from the device's base address in the memory map.

Table 7-2: DUSCC registers

Register Name	Relative Address	Function	Section Number
CMR1	c00000	Channel Mode Register 1	7.2.5
CMR2	c00001	Channel Mode Register 2	7.2.5
S1R	c00010	SYN 1/Secondary Address Register 1	7.2.6
S2R	c00011	SYN 2/Secondary Address Register 2	7.2.6
TPR	c00100	Transmitter Parameter Register	7.2.7
TTR	c00101	Transmitter Timing Register	7.2.7
RPR	c00110	Receiver Parameter Register	7.2.8
RTR	c00111	Receiver Timing Register	7.2.8
CTPRH	c01000	Counter/Timer Preset Register High	7.2.9
CTPRL	c01001	Counter/Timer Preset Register Low	7.2.9
CTCR	c01010	Counter/Timer Control Register	7.2.9

Table 7-2 (cont.)

Register Name	Relative Address	Function	Section Number
OMR	c01011	Output and Miscellaneous Register	7.2.12
CTH	c01100	Counter/Timer High	7.2.9
CTL	c01101	Counter/Timer Low	7.2.9
PCR	c01110	Pin Configuration Register	7.2.12
CCR	c01111	Channel Command Register	7.2.4
TXFIFO	c100xx	Transmitter FIFO	7.2.7
RXFIFO	c101xx	Receiver FIFO	7.2.8
RSR	c11000	Receiver Status Register	7.2.10
TRSR	c11001	Transmitter and Receiver Status Register	7.2.10
ICTSR	c11010	Input and Counter/Timer Status Register	7.2.10
GSR	x11011	General Status Register	7.2.10

Table 7-2 (cont.)

Register Name	Relative Address	Function	Section Number
IER	c11100	Interrupt Enable	7.2.11
IVR	011110	Interrupt Vector Register -- Modified	7.2.11
IVRM	111110	Interrupt Vector Register -- Unmodified	7.2.11
ICR	011111	Interrupt Control Register	7.2.11

Notes:

1. The value of "c" selects the register for a particular channel. If c is 0, the register for channel A is selected. If c is 1, the register for channel B is selected.

2. The letter "x" indicates a "Don't Care" value. In these positions, either a 1 or a 0 can be used to address the register.

7.2.3 Register Addressing

The DUSCC registers appear in both the TMS processor and the MicroVAX memory maps. The addresses in Table 7-2 are the word offsets from the device's base address in a particular memory map. The registers are each 8 bits long, and appear in the least significant byte of each word in the processor address maps.

7.2.4 Register Usage

The registers fall into nine functional groups:

1. Command
2. Channel Mode
3. Secondary Address
4. Transmitter
5. Receiver
6. Counter/Timer
7. Device and Channel Status
8. Interrupt
9. Miscellaneous

The TMS processor keeps general control of the device and what it does through the Channel Command Registers (CCR). Here it can start and stop the transmitter, receiver, and counter/timer for each channel.

The overall behavior of a channel is controlled through its Channel Mode registers.

The specific behavior of each component in the channel is controlled through the appropriate registers. For example, the TMS processor determines how the counter/timer behaves by programming the counter/timer registers.

The Interrupt registers allow the TMS processor to set the conditions when the device will generate interrupts.

7.2.5 Channel Command Register (CCR)

The TMS processor uses commands to control the action of each DUSCC channel. It enters these commands as values in the appropriate Channel Command Register (CCR).

Figure 7-2 shows the format of a Channel Command Register.

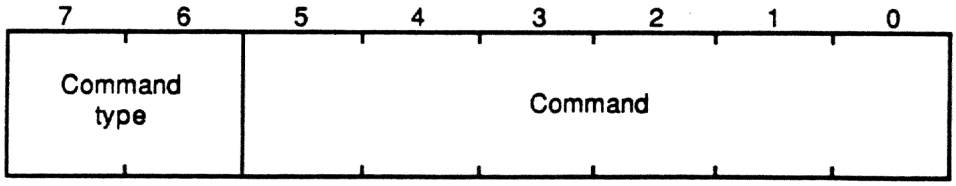


Figure 7-2: The DUSCC Channel Command Register

As the figure shows, the top two bits (Bits <07:06>) identify the type of command:

0. Transmitter command
1. Receiver command
2. Counter/Timer command
3. Digital Phase-Locked Loop command

Bits <04:00> contain the command itself. Table 7-3 shows the values this field can take for each of the command types.

Table 7-3: DUSCC Commands

Register Value (Hex)	Command Name and Meaning
	Transmitter Commands
0	RESET. Immediately stops the transmitter once it has sent the current character. Any remaining characters in the FIFO are cleared out. This command also clears the transmitter status bits in the TRSR (see Section 7.2.11.3) and the appropriate TXRDY bit in the GSR (see Section 7.2.11.1).
1	RESET CRC. This command is associated with the next character to go into the FIFO. Just before transmitting that character, the CRC will be reset to its initial value.
2	ENABLE. Enables the transmitter to send information controlled by the value of the "Clear to Send Control" bit in the TPR (see Section 7.2.8.1).
3	DISABLE. Stops the transmitter the next time that the FIFO is empty. Also sets the TXD pin in a mark state.

Table 7-3 (cont.)

Register Value (Hex)	Command Name and Meaning
4	SEND START OF MESSAGE. Sends the start of message sequence followed by the characters in the FIFO. This command is issued after the "Enable Transmitter" command to begin character transmission. Either this command or the "Send Start of Message with PAD" command must be used to begin data transfer. The command causes the SYN (for COP) or FLAG (for BOP) sequence to be sent. If multiple SYN or FLAG sequences are necessary, the I/O processor simply issues this command the necessary number of times. After the last command, the characters in the FIFO are sent. If the FIFO is empty, the device continues to send SYN or FLAG sequences until there is a character to send.
5	SEND START OF MESSAGE WITH PAD. Similar to "Send Start of Message" but it sends a bit pattern to synchronize the digital phase-locked loop before sending the SYN or FLAG character.
6	SEND END OF MESSAGE. This command is associated with the next character to go into the FIFO. After sending that character, the device will send the appropriate end of message sequence. Typically, this sequence is a Frame Check Sequence, such as a CRC. In BOP mode, a FLAG character is sent after the Frame Check Sequence. If the "Transmit EOM" (TEMC) bit in the TPR is set, this command will be executed automatically under certain conditions. See Section 7.2.8.1 for more details of these conditions.

Table 7-3 (cont.)

Register Value (Hex)	Command Name and Meaning
7	ABORT TRANSMISSION. **BOP Only.** Sends a sequence of 8 ones after sending the current character. After this, the transmitter will send MARK or FLAG characters depending on the value in the Underrun Control field of the TPR. This command also causes any remaining characters in the FIFO to be cleared; that is, not sent. To restart transmission, use one of the "Send Start of Message" commands.
8	SEND DLE. **COP Only.** This command is associated with the next character to go into the FIFO. Just before sending this character, the transmitter sends a DLE character. In the BISYNC protocol, an extra DLE has to be sent for each DLE in the message. When the channel is set up for this protocol (see Section 7.2.6), the transmitter automatically sends this second DLE. So, this command is not necessary.
9	GO ACTIVE ON POLL. This command is used in BOP loop mode only. It is not used in the DEC MicroServer.
A	RESET GO ON POLL. This command resets the "Go Active on Poll" command. It is not used in the DEC MicroServer.
B	GO ON LOOP. This command is used in BOP loop mode only. It is not used in the DEC MicroServer.
C	GO OFF LOOP. This command is used in BOP loop mode only. It is not used in the DEC MicroServer.

Table 7-3 (cont.)

Register Value (Hex)	Command Name and Meaning
D	EXCLUDE FROM CRC. Excludes the next character to go into the FIFO from the CRC calculation.
	Receiver Commands
0	RESET RECEIVER. Stops the receiver, and clears the FIFO. This command also clears the receiver status bits in the RSR and TRSR. Finally, the command clears the appropriate bit in the GSR.
1	Reserved
2	ENABLE RECEIVER. Starts the receiver dependent on the value of the "Enable Data Carrier Detect Control" bit in the appropriate RPR. Once started, the receiver looks for START, SYN, or FLAG depending on the channel's protocol. This command is ignored if the receiver has already been enabled.
3	DISABLE RECEIVER. Stops the receiver and discards the character currently being received, if any. This command, however, does not affect the status bits or the contents of the FIFO.
	Counter/Timer Commands
0	START. Starts the counter/timer, using the parameters set in the CTCR.

Table 7-3 (cont.)

Register Value (Hex)	Command Name and Meaning
1	STOP. Stops the counter/timer. The I/O processor can issue this command at any time compared to the clock source for the counter/timer. However, the counter/timer stops on completion of the next source clock cycle, so sometimes it will not stop immediately the command is issued.
2	PRESET TO FFFF. Loads the counter/timer with the hex value FFFF. This command also causes the DUSCC's C/T pin to go low.
3	LOAD FROM CTPR. Loads the counter/timer with the value in the appropriate preset registers. The command also causes the DUSCC's C/T pin to go low.
Digital Phase-Locked Loop Commands	

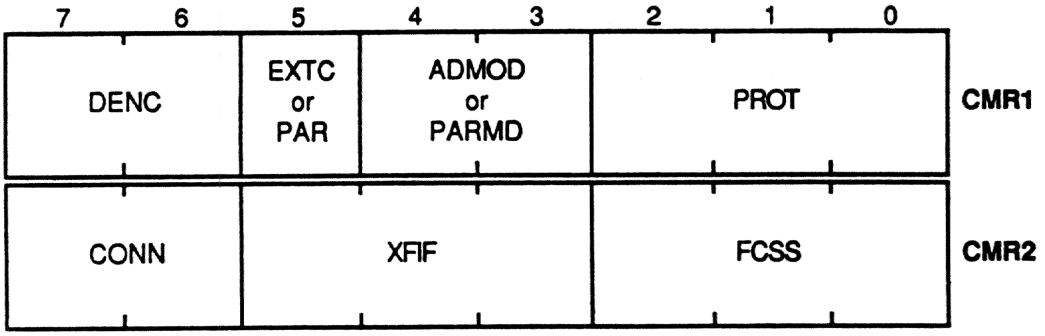
Table 7-3 (cont.)

Register Value (Hex)	Command Name and Meaning
0	START SEARCH. **NRZI Mode.** Synchronizes the DPLL and the data without the need for preframe transitions. The DPLL does this by: 1. Loading the DPPL counter with the decimal value of 15, and holding the clock output high. 2. Starting the counter when a transition occurs on the data line. At this point, the clock output is held low. 3. Resetting the counter to 0 when it reaches 1F (hexadecimal, 31 decimal). The clock output now changes from low to high. From this point the DPLL returns to normal operation. **FM Mode.** Clears the DPLL counter, and holds the clock output high. When a transition on the data line occurs, the DPLL starts normal operation. DO NOT use this command in either mode if you are using the DPLL to supply the clock for the transmitter, and the transmitter is active.
1	DISABLE. Stops the DPLL.
2	SET FM MODE. Starts the DPLL and puts it into FM mode. This mode is used if FM0, FM1, or Manchester (NRZ) data encoding is selected in CMR1 (see Section 7.2.6).
3	SET NRZI MODE. Starts the DPLL and puts it into NRZI mode. This mode is used if NRZ or NRZI data encoding is selected in CMR1 (see Section 7.2.6).

7.2.6 Channel Mode Registers (CMR1 and CMR2)

Each channel has two Channel Mode Registers (CMR1 and CMR2). CMR1 determines the type of protocol (bit-oriented or character-oriented) and the data encoding to be used on the channel. CMR2 determines the way the channel communicates with the MicroVAX or TMS processor, the error checking protocol to be used, and the transmit path.

Figure 7-3 shows the format of the registers. Table 7-4 gives more information on each field in the registers.



ADMOD	Address Mode	PAR	Parity type
CONN	Channel Connection	PARMD	Parity Mode
DENC	Data Encoding	PROT	Channel Protect Mode
EXTC	Extended Control	XFIF	Data Transfer Interface
FCSS	Frame Check Sequence Select		

Figure 7-3: DUSCC Channel Mode Registers

Table 7-4: DUSCC Channel Mode register fields (CMR1 and CMR2)

Field name	Function/Values
PROT	This field determines the type of protocol that the channel will use: bit-oriented, or character-oriented. These protocols are referred to by the abbreviations BOP and COP. The type of protocol this field selects affects the meaning of other fields in the DUSCC registers. The values that this field can take are as follows. 0. BOP primary 1. BOP secondary 2. Reserved 3. Reserved 4. COP dual SYN 5. BISYNC -- a form of COP dual SYN 6. COP single SYN 7. Not used in the DEC MicroServer
ADMOD	**BOP protocols.** This field selects the addressing mode. The values that this field can take are: 0. 8-bit addressing 1. Extended addressing 2. 16-bit addressing 3. 16-bit addressing with group
PARMD	**COP protocols.** This field determines the sort of parity to be used on the channel. The values this field can have are: 0. No parity -- required when BISYNC protocol is in use 1. Reserved 2. With parity -- odd or even as determined by the PAR field. 3. Force parity -- parity bit is forced to the value of PAR field

Table 7-4 (cont.)

Field name	Function/Values
EXTC	**BOP protocols.** This field determines whether extended address control is to be used. If this bit is set, extended control is used and a two-byte control field will follow the address field in a packet. When the bit is clear, extended control is not used and the control field consists of just one byte.
PAR	**COP Protocols (except BISYNC).** This bit determines the sort of parity to be used. When set, odd parity is used, and even parity is used when the bit is clear. **BISYNC.** Setting this bit indicates that 8-bit ASCII coding is to be used, and clearing it indicates that EBCDIC coding is to be used.
DENC	This field determines the sort of encoding used on the communications channel. The values this field can have are: 0. NRZ/MANCHESTER encoding 1. NRZI encoding 2. FM0 (or bi-phase 0) encoding 3. FM1 (or bi-phase 1) encoding

Table 7-4 (cont.)

Field name	Function/Values
DFCSS	The value in this field determines the sort of frame checking sequence to use on the channel. The values that this field can have are: 0. No frame check 1. Reserved 2. LRC8 with dividend preset to 0 -- valid for COP protocols only 3. LRC8 with dividend preset to 1 -- valid for COP protocols only 4. CRC 16 with dividend preset to 0 5. CRC 16 with dividend preset to 1 6. CCITT CRC with dividend preset to 0 7. CCITT CRC with dividend preset to 1
XFIF	This field sets the way data transfer takes place between the device's internal FIFO's and the I/O processor. The values this field can have are: 0. Half duplex DMA using single addresses 1. Half duplex DMA using dual addresses 2. Full duplex DMA using single addresses 3. Full duplex DMA using dual addresses 4. Not used in the DEC MicroServer 5. Not used in the DEC MicroServer 6. Not used in the DEC MicroServer 7. Non-DMA transfer using normal bus read and write cycles in response to polling status registers or interrupts

Table 7-4 (cont.)

Field name	Function/Values
CONN	This field sets the way that the channel operates. The values that can appear here are: 0. Normal Mode. The transmitter and receiver operate independently in either a half duplex or full duplex fashion (depending on how they are enabled) 1. Automatic Echo. All data received on a channel is automatically retransmitted after a delay of one bit. 2. Local Loopback. The transmitter output is internally connected to the receiver input; this is used by the OBT as part of the channel test 3. Reserved

7.2.7 Secondary Address Registers (S1R and S2R)

Each channel has a pair of Secondary Address Registers (S1R and S2R). Their function is different for COP and BOP protocols:

BOP - The registers contain an 8- or 16-bit address. The address in received packets is checked against these registers, and appropriate action taken on a match. These registers are not used in BOP primary protocols or in BOP secondary protocols where extended addressing is used.

COP - The registers contain a 1- or 2-character synchronization sequence. Incoming packets are checked against this field. Each character can be between 5 and 8 bits long, and is right justified.

Figure 7-4 shows the format of these registers.

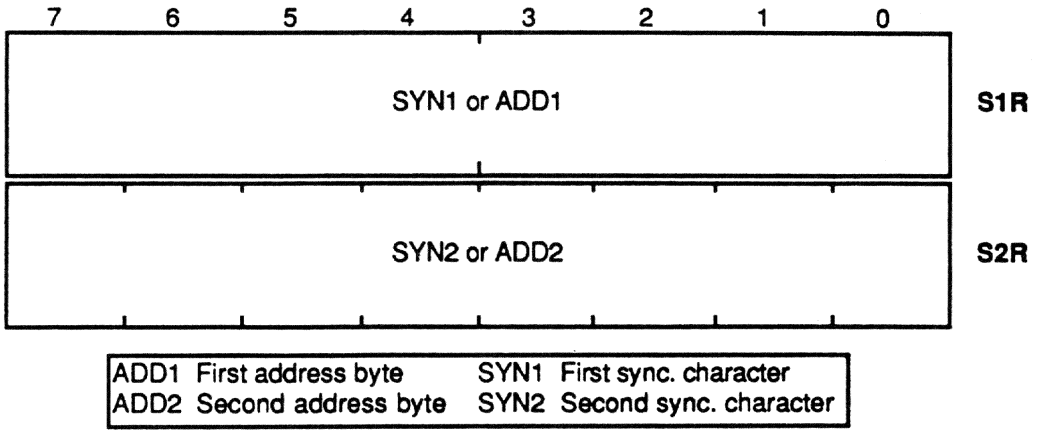


Figure 7-4: Synchronization/Secondary Address Registers for the DUSCC

7.2.8 Transmitter Registers

Each channel on the device has three transmission-related "registers":

- Transmitter Parameter Register (TPR)
- Transmitter Timing Register (TTR)
- Transmitter FIFO (TXFIFO)

The TPR and TTR are true registers. The FIFO is an internal data structure that is addressable in the same way as the other registers.

The following sections describe these registers in more detail.

7.2.8.1 Transmitter Parameter Register (TPR) - There is one Transmitter Parameter Register (TPR) for each channel. The controlling software uses this register to set up transmission conditions such as:

- Number of bits in each character

- Whether the transmitter is controlled by an external modem
- Idle state
- Response to underrun errors

Figure 7-5 shows the format of the register, and Table 7-5 explains the content of each field.

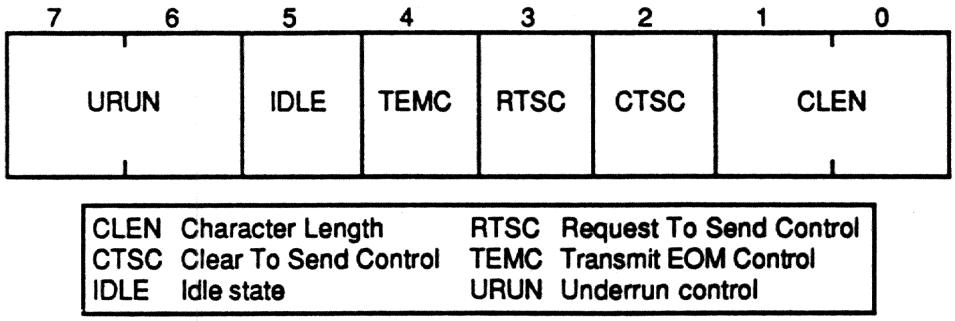


Figure 7-5: Transmitter Parameter Register for the DUSCC

Table 7-5: Content of the DUSCC Transmitter Parameter Register

Field Name	Function and Value
CLEN	The value of this field sets the number of bits to be used to send each character. Each value sets a different number of bits for each character: 0. 5 bits 1. 6 bits 2. 7 bits 3. 8 bits

Table 7-5 (cont.)

Field Name	Function and Value
CTSC	The value of this bit determines whether an external Clear To Send (CTS) signal controls the operation of the channel's transmitter. When this bit is set, the transmitter cannot run on (that is, send a character) until the external CTS signal is asserted at the channel's CTS pin. When this bit is clear, the state of the CTS pin has no effect on whether the transmitter can send a character.
RTSC	The value of this bit determines whether the transmitter must first assert the Request To Send (RTS) pin before sending information. This pin exists only if the device is not working in full duplex DMA mode. This is the mode used by the DEC MicroServer, and so this function is not applicable.
TEMC	For both COP and BOP, the value of this bit determines whether an End Of Message indicator is to be transmitted when either of the following events occur: 1. When the counter/timer reaches zero and it has been set to count characters transmitted. In this case, the DONE pin is also asserted. 2. If the DONE pin was asserted after a character was moved out of the DUSCC's transmit FIFO. In the case of BOP, the End Of Message indicator is known as the FCS-FLAG. In COP, it is known as FCS.

Table 7-5 (cont.)

Field Name	Function and Value
IDLE	The value of this field determines what idle state to use on the line when no data is being transmitted. A value of 1 indicates that idling is to use SYN characters in COP, or FLAG characters in BOP. If this bit is clear, the idle state asserts the MARK state.

Table 7-5 (cont.)

Field Name	Function and Value
URUN	This field tells the transmitter what to do if an underrun occurs. There are different options for BOP and COP modes. **For BOP:** 0. Perform a normal end of message sequence -- FLAG, followed by MARKs or FLAGs as determined by the IDLE field 1. Reserved 2. Send an Abort character to the remote station, and then place the TXD pin in a MARK condition until further instructions are received 3. Send an Abort character to the remote station, and then send FLAG characters until further instructions are received **For COP:** 0. Perform a normal end of message sequence -- FCS, if selected by Bits <2:0>, followed by MARKs or SYNs as determined by the IDLE field 1. Reserved 2. Place the TXD pin in a MARK condition until further instructions are received 3. Send SYN characters until further instructions are received

7.2.8.2 Transmitter Timing Register (TTR) - To send data, each channel needs some timing signals. Information on the source and

nature of this timing is kept in the channel's Transmitter Timing Register (TTR).

Figure 7-6 shows the format of this register, and Table 7-6 explains the value of each field.

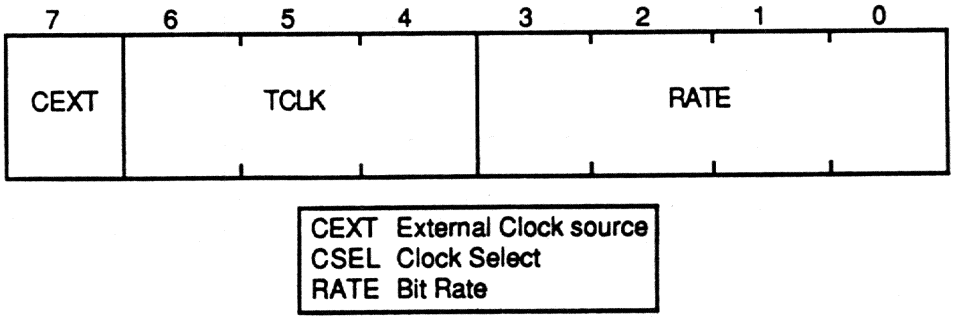


Figure 7-6: Transmitter Timing Register for the DUSCC

Table 7-6: Content of the DUSCC Transmitter Timing Register

Field Name	Function and Value
RATE	This field selects the bit rate the transmitter will use if the internal Bit Rate Generator is selected to provide the timing. The rates that this field can indicate are shown in Table 7-7.

Table 7-6 (cont.)

Field Name	Function and Value
TCLK	This field sets the source of the data clock: 0. External clock at the shift rate 1. External clock at 16 times the shift rate 2. Internal clock from the Phase-Locked Loop at the bit rate. Not used in full duplex operation 3. Internal clock from the Bit Rate Generator -- the value in the RATE field determines the rate used 4. Internal clock from the counter/timer of the device's other channel at the shift rate 5. Internal clock from the counter/timer of the device's other channel at 32 times the shift rate 6. Internal clock from this channel's counter/timer at the shift rate 7. Internal clock from this channel's counter/timer at 32 times the shift rate
CEXT	The channel takes an external clock through either its RXTC or TRXC pins. When the TCLK field specifies an external source, this field selects which pin the clock is present on. When this field is set, the TRXC pin is used, and when clear the RTXC pin is used.

Table 7-7: Bit Rate Values

Value in RATE Field	Bit Rate Used	Value in RATE Field	Bit Rate Used
0	50	8	1050
1	75	9	1200
2	110	10	2000
3	134.5	11	2400
4	150	12	4800
5	200	13	9600
6	300	14	19.2K
7	600	15	38.4K

7.2.8.3 Transmitter FIFO - The DUSCC has two internal FIFOs for each channel -- one FIFO for the receiver, and one for the transmitter.

The Transmitter FIFO is a four-character queue that helps to prevent underrun and also reduces the DMA overhead. The elements in the queue can be addressed directly in the same way as the DUSCC registers.

NOTE

Do not confuse these FIFOs with those on the receive line of each DEC MicroServer port. The FIFOs mentioned here are inside the DUSCC and help to improve the device's performance. The other FIFOs help to improve the performance of the complete DEC MicroServer (see Section 7.3).

7.2.9 Receiver Registers

There are three registers that control the operation of the receiver of each channel:

- Receiver Parameter Register (RPR)
- Receiver Timing Register (RTR)

- Receiver FIFO

The following sections describe the registers in more detail.

7.2.9.1 Receiver Parameter Register (RPR) - As with the transmitter, the Receiver Parameter Register establishes the size of characters that the receiver is to deal with, and how the receiver is to operate on those characters.

The meaning of the register's contents is different for BOP and COP. Figure 7-7 shows the format of register for both types of protocol, and Table 7-8 explains the content of each field. The table also has two parts, one for each type of protocol.

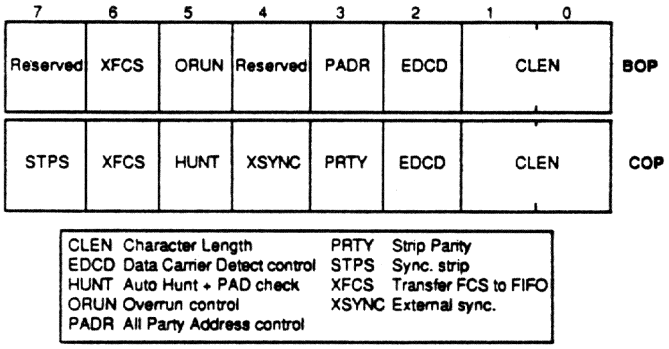


Figure 7-7: Receiver Parameter Register

Table 7-8: Content of the DUSCC Receiver Parameter Register

Field Name	Function and Value
	Bit Oriented Protocols
CLEN	This field sets the number of bits to expect in each character. Each value sets a different number of bits for each character: 0. 5 bits 1. 6 bits 2. 7 bits 3. 8 bits This is the size of each character in the information field; header and control information always consists of 8 bit characters.
EDCD	This field determines whether the receiver is controlled by the Data Carrier Detect (DCD) from the channel's modem. When this field is set, the receiver is controlled by DCD.
PADR	This field determines whether the receiver responds to All Party addresses -- Hex values: FF and FF FF. When set, the receiver will respond to these addresses.
ORUN	This field controls what the receiver does if there is a data overrun. Overrun occurs when a character is received while the Receiver FIFO and the receiver shift register are both full. When this field is set, the receiver continues to receive the frame, but the overrunning character is discarded. When clear, the receiver terminates an overrunning frame and waits to receive a Flag character.

Table 7-8 (cont.)

Field Name	Function and Value
XFCS	When set, this field causes the receiver to put the Frame Check Sequence attached to each message into the Receiver FIFO.
	Character Oriented Protocols
CLEN	This field sets the number of bits to expect in each character. Each value sets a different number of bits for each character: 0. 5 bits 1. 6 bits 2. 7 bits 3. 8 bits This number excludes the parity bits.
EDCD	This field determines whether the receiver is controlled by the Data Carrier Detect (DCD) from the channel's modem. When this field is set, the receiver is controlled by DCD.
PRTY	This field controls whether the receiver strips parity from the incoming characters. If set, the receiver strips parity.
XSYNC	When set, external synchronization is provided and so the receiver does not have to handle synchronization patterns. Instead, the receiver is enabled when the SYNI pin is asserted. XSYNC and the EDCD fields cannot both be set. This is because the two signals share the same pin on the device. **This field applies to Character Protocols that use only one synchronization character.**

Table 7-8 (cont.)

Field Name	Function and Value
HUNT	When set, causes the receiver to hunt for synchronization characters after receiving certain EOM characters. Setting the field also causes the receiver to check for a closing PAD of four 1s after a EOT/NAK sequence. **This field is used in BISYNC protocols only.**
XFCS	When set, this field causes the receiver to put the Frame Check Sequence attached to each message into the Receiver FIFO.
STPS	When set, the receiver strips the synchronization character patterns from each message, and does not put these in the receiver FIFO.

7.2.9.2 Receiver Timing Register (RTR) - To receive data correctly, each channel needs some timing signals. Information on the source and nature of this timing is kept in the channel's Receiver Timing Register (RTR).

Figure 7-8 shows the format of this register, and Table 7-9 explains the value of each field.

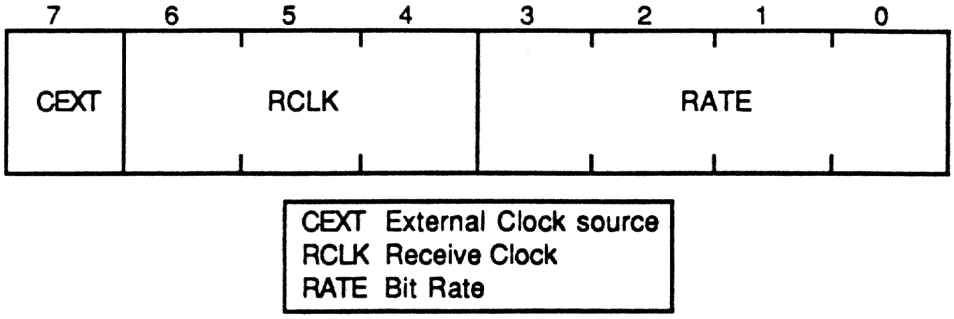


Figure 7-8: Receiver Timing Register (RTR) for the DUSCC

Table 7-9: Content of the DUSCC Receiver Timing Register

Field Name	Function and Value
RATE	The value of this field selects the bit rate the receiver will use if the internal Bit Rate Generator is selected to provide the timing. The rates that this field can indicate are shown in Table 7-7, which follows this table.

Table 7-9 (cont.)

Field Name	Function and Value
RCLK	The value in this field determines the source of the data clock: 0. External clock at the shift rate 1. Not used in the DEC MicroServer 2. Not used in the DEC MicroServer 3. Not used in the DEC MicroServer 4. Internal clock from the Phase-Locked Loop 5. Internal clock from the Phase-Locked Loop -- The loop itself is clocked from an external source 6. Internal clock from the Phase-Locked Loop -- The loop is clocked from the Bit Rate Generator 7. Internal clock from the Phase-Locked Loop -- The loop is clocked from this channel's counter/timer
CEXT	The channel takes an external clock through either its RXTC or TRXC pins. When the RCLK specifies an external source, this field selects which pin the clock is present on. When this field is set, the clock comes from the TRXC pin, and when clear the clock comes from the RTXC pin.

7.2.9.3 Receiver FIFO (RXFIFO) - The receiver has an internal FIFO that buffers incoming characters before transfer to buffer memory. Like the transmitter's FIFO, the locations in this queue are addressable as DUSCC registers. This FIFO is in addition to the FIFOs on the synchronous ports, and helps to improve the

DUSCC's performance.

7.2.10 Counter/Timer Registers

Each channel has a 16-bit counter/timer that can be used to count characters in a message or to provide timing for data communication. The counter/timer starts with a preset value and as each event occurs (in the form of a clock pulse) the value decreases by one. When the value is zero, the counter is reloaded (and may interrupt the TMS processor).

Each timer is controlled by two registers:

- Preset Register
- Control Register

In addition, there is a register that gives the current value of the counter/timer.

The following sections explain the use of these registers in more detail.

7.2.10.1 Counter/Timer Preset Register (CTPRL and CTPRH) - The preset register holds the value that the counter/timer will initialize itself with when it is started. Examples of this are when the device is used to count characters in a message, and the preset value is the number of characters.

DUSCC registers are only 8 bits wide, and so two are needed to hold the complete 16-bit value. Figure 7-9 shows the names of these registers, and what they contain.

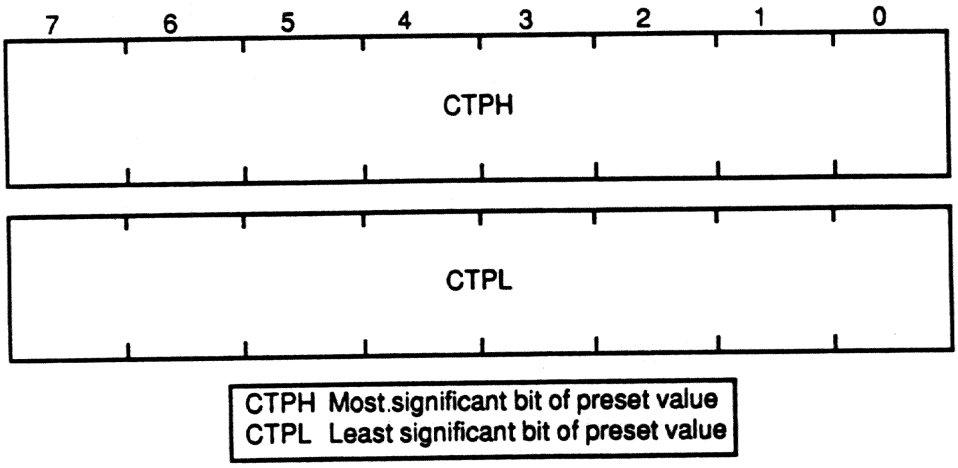


Figure 7-9: The DUSCC Counter/Timer Preset Registers

7.2.10.2 Counter Timer Control Register (CTCR) - The overall control of the counter/timer is through the Counter/Timer Preset Register (CTCR). Figure 7-10 shows the format of the register, and Table 7-10 explains the value of each field.

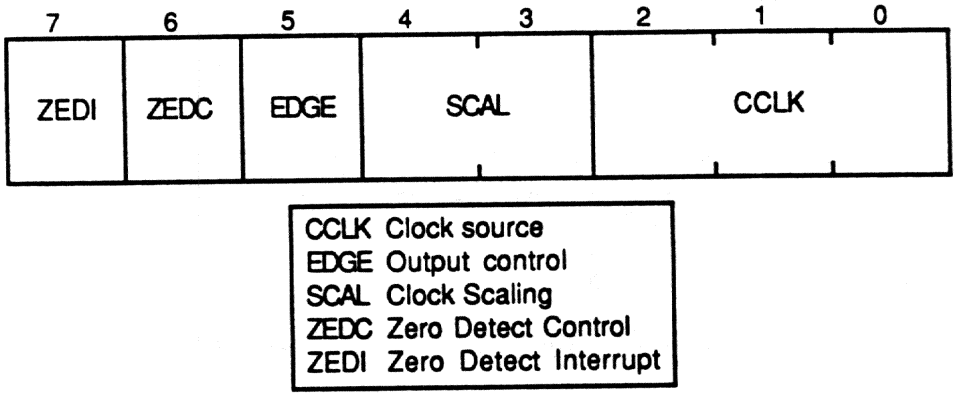


Figure 7-10: DUSCC Counter/Timer Control Register

Table 7-10: Content of the DUSCC Counter/Timer Control Register

Field Name	Function and Value
CCLK	This field sets the source of the clock pulse that controls the value in the counter: 0. RTXC pin on the device 1. TRXC pin on the device 2. The device's clock input pin 3. The device's clock input pin, but counting is delayed until the RXD pin goes low; counting continues until RXD goes high again; the TMS processor can use the resultant value to determine the bit rate of incoming data 4. Bit Rate Generator of the channel's receiver 5. Bit Rate Generator of the channel's transmitter 6. On reception of a character 7. On transmission of a character
SCAL	This field determines whether the clock source is scaled before being used to drive the counter/timer: 0. No scaling 1. Source frequency is divided by 16 2. Source frequency is divided by 32 3. Source frequency is divided by 64

Table 7-10 (cont.)

Field Name	Function and Value
EDGE	The DUSCC can be set up to produce an output signal on the TRXC or RTXC pins (see the Pin Configuration Register). The value of this bit determines the type of signal it provides: 0. The logic state of the signal changes each time the counter/timer reaches zero -- the state is set to zero each time you load a value into either of the preset registers. 1. A positive clock pulse each time the counter/timer reaches zero -- this pulse lasts for one clock period.
ZEDC	This field determines the reset value of the counter/timer. If set, the reset value is FFFF (hex). If clear, the reset value is that in the Counter/Timer Preset Registers.
ZEDI	This field controls whether the DUSCC generates an interrupt when the counter/timer reaches zero. If set, the device sets the C/T Zero Count bit in the Interrupt Control Register, which causes an interrupt to occur. If clear, the C/T Zero Count bit is not set and so no interrupt occurs.

7.2.10.3 Counter Timer (CTH and CTL) - Each channel has a pair of counter/timer registers. Each pair gives the current value of the counter/timer for the appropriate channel. As with the preset registers, two are needed for each channel to hold the 16-bit value. Figure 7-11 shows the names of the registers, and the values they hold.

NOTE

Always stop the counter/timer before reading these registers. This will prevent errors occurring if the read happens at the same time as the device is trying to update the register. To stop the counter/timer, set the appropriate value in the command register (see Section 7.2.5). The counter restarts when you have read the value.

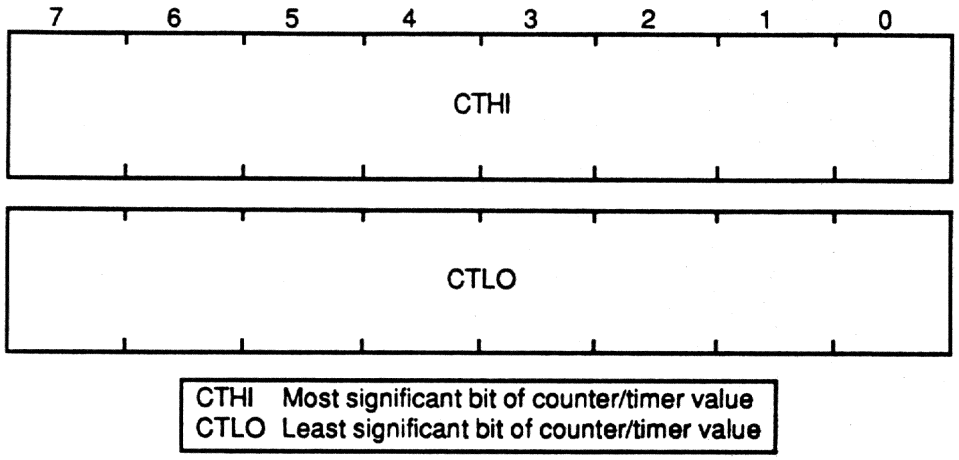


Figure 7-11: The DUSCC Counter/Timer Registers

7.2.11 Device and Channel Status Registers

The DUSCC has a General Status Register (GSR) that provides a summary of the current state of both channels. More detailed information can be found in the three detailed status registers for each channel:

- Receiver Status Register (RSR)
- Transmitter and Receiver Status Register (TRSR)
- Input and Counter/Timer Status Register (ICTSR)

The following sections explain the format and content of these registers in more detail.

7.2.11.1 General Status Register (GSR) - There is one General Status Register (GSR) for the entire device which summarizes the status of both channels. Figure 7-12 shows the format of the register.

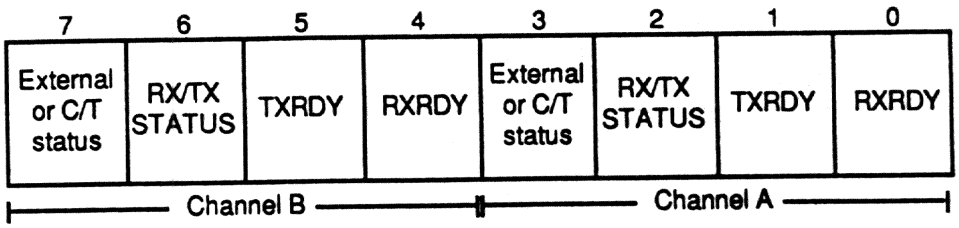


Figure 7-12: General Status Register for the DUSCC

As the figure shows, the GSR is divided into two parts: bits <07:04> contain information for Channel B, while bits <03:00> contain information for Channel A. The register holds the same information for both channels.

Table 7-11 explains the meaning of each bit in the register.

Table 7-11: Content of the DUSCC General Status Register

Field Name	Meaning
RXRDY	Indicates that there are characters in the receiver FIFO for the TMS processor to read. This bit is cleared when the DUSCC is reset or when the "Reset Receiver" command is issued on that channel.
TXRDY	Indicates that the transmitter FIFO can accept characters for transmission along the serial line. This bit can appear as set only when the transmitter is enabled. At all other times, the bit is clear.

Table 7-11 (cont.)

Field Name	Meaning
RX/TX Status	Indicates that one or more of the status bits in the appropriate RSR or TRSR are set.
External or C/T Status	Indicates that one of the status bits in the appropriate ICTSR is set.

7.2.11.2 Receiver Status Register (RSR) - There is one register for each channel that tells the controlling processor the status of the appropriate receiver. The general name for these registers is RSR.

Some of the bits in the register refer to the "current" character. This is the character at the front of the FIFO: that is, the character that would be read from the queue first. This feature allows the TMS processor to check the status of every character, or of every frame.

To check every character, the status bits have to be read and reset before reading each character from the FIFO. To check each block, the register is reset at the start of each block, and read at the end of that block. To reset a bit you write a 1 to that bit in the RSR. So, for example, to clear the entire register, write a value of FF (hex).

Figure 7-13 shows the format of the register, and illustrates how the bits have different meanings depending on the type of protocol being used. Table 7-12 explains what each bit means for each type of protocol.

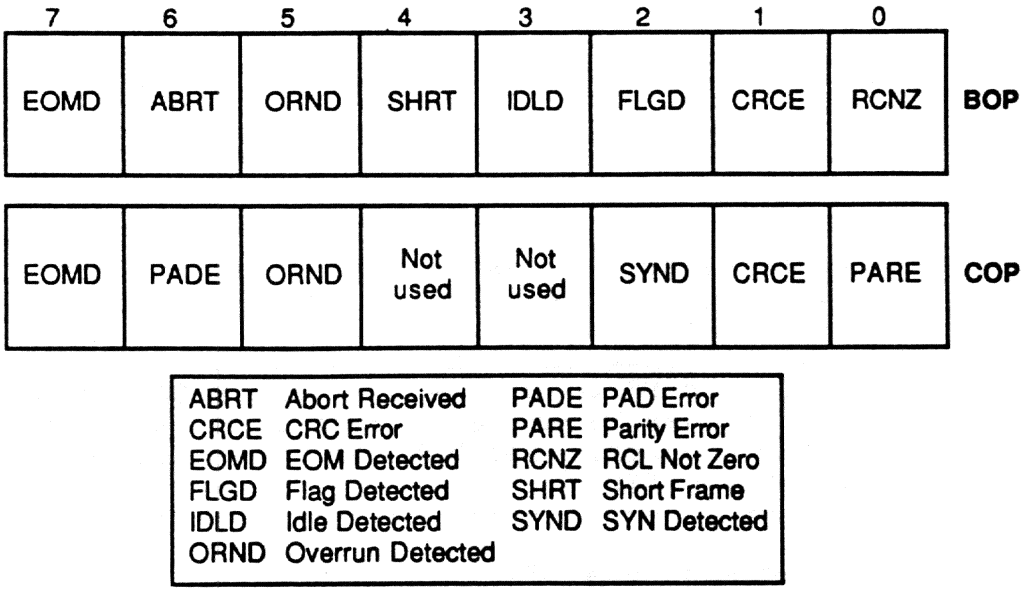


Figure 7-13: Receiver Status Register for the DUSCC

Table 7-12: Content of the DUSCC Receiver Status Register

Field Name	Function and Meaning
	Bit Oriented Protocols (BOP)
RCNZ	The final character received in the Information field of a message did not have the correct number of bits. This number is set up in the CLEN field of the RPR and is used by the receiver to repackage the information into its constituent characters. The actual number of bits in this final character is held in the RSLEN field of the TRSR. In the case of an incorrect length being set in the RPR, this status information can help in correcting errors.

Table 7-12 (cont.)

Field Name	Function and Meaning
CRCE	The received CRC was not the same as that calculated by the receiver. This bit is set at the same time as the last character in the frame's Information field is put into the receiver's FIFO. However, if the XFCS field in the Receiver Parameter Register is set, this bit is set when the last byte of the received CRC is put in the FIFO.
FLGD	A flag sequence of 01111110 has been received.
IDLD	An idle sequence of a 0 followed by fifteen 1s has been received. If received in the middle of a valid frame, this sequence must be preceded by an Abort command. Outside a valid frame, this sequence is recognized at any time.
SHRT	An End Of Frame indicator was received, but some fields in that frame were missing. This bit is set if the receiver does not detect the appropriate number of Address fields, a Control field (as specified in CMR1), and the necessary FCS bytes.
ORND	A character was received when the Receiver FIFO was already full. Each character received in this state is ignored. Once the FIFO becomes empty, the device begins receiving information on the next character boundary.

Table 7-12 (cont.)

Field Name	Function and Meaning
ABRT	An ABORT sequence of 01111111 was received after the first address byte of a frame but before the closing flag character. When this bit is set, the I/O processor should read all the characters in the FIFO and determine if any of them are valid characters from the preceding frame. If none of the characters has EOM Detect associated with it, they are all from the current frame and are to be ignored. The processor should also ignore any other characters previously received in this frame. When this condition occurs, the receiver automatically starts to search for a flag character. When the next frame begins, the CRC will be reset.
EOMD	The character currently at the front of the FIFO is the last in the frame. If the XFCS field in the RPR is set, this bit will be set when the last byte of the FCS reaches the front of the FIFO. An overrun error may mean that the EOM character is lost. However, this bit will still be set to show the character was received.
Character Oriented Protocols (COP)	
PARE	A character was received with the incorrect parity.

Table 7-12 (cont.)

Field Name	Function and Meaning
CRCE	Set when the received BCC characters do not agree with those the receiver has calculated. If set, this bit is normally associated with the character preceding the first BCC character. However, if the XFCS field in the RPR is set, this bit is associated with the last of the BCC bytes.
SYND	A synchronization pattern was received. This pattern is set up in the Secondary Address Registers. Note that the receiver sets this bit only when a completely valid synchronization sequence is received. Synchronization characters that have parity errors are ignored.
ORND	A character was received when the Receiver FIFO was full. Each character received in this state is ignored. Once the FIFO becomes empty, the device begins receiving information on the next character boundary.
PADE	**BISYNC Only.** The closing PAD of four 1s was missing after the EOT and NAK sequence. Whether this bit is set is controlled by the HUNT field in the RPR.

Table 7-12 (cont.)

Field Name	Function and Meaning
EOMD	**BISYNC Only.** The character currently at the front of the FIFO is either a text message terminator, or the start of a control sequence received outside a text or header field. If the XFCS field in the RPR is also set, this bit becomes set when the last byte of the FCS reaches the front of the FIFO. The control sequences that can set this bit are: Header Field, Normal Text Field, Transparent Text Field, and Control Message Field. An overrun error may mean that the EOM character is lost. However, this bit will still be set to show the character was received.

7.2.11.3 Transmitter and Receiver Status Register (TRSR) - There is one register for each channel that gives the controlling processor information on the status of the appropriate transmitter and receiver. The general name for such a register is TRSR.

The receiver information in this register is in addition to that in the RSR.

Any bits of the register not used in a particular mode will always read as zero.

Figure 7-14 shows the format of the register, and illustrates that the bits have different meanings depending on the type of protocol in use. Table 7-13 explains the meaning of each bit for both types of protocol.

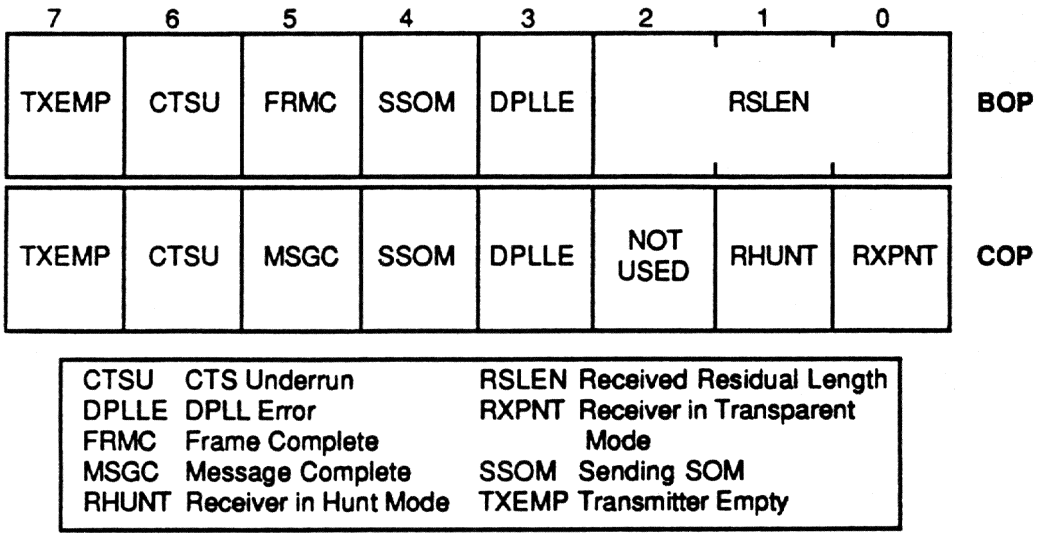


Figure 7-14: The DUSCC Transmitter and Receiver Status Register

Table 7-13: Content of the DUSCC Transmitter and Receiver Status Register

Field Name	Function and Meaning
	Bit Oriented Protocols (BOP)
RSLEN	If the RCNZ field in the RSR is set, this field contains the number of bits found in that final character.
DPLLE	The Digital Phase-Locked Loop (DPLL) could not find a data transition for two consecutive bits in the detection window. As a result, the DPLL has gone into search mode. This bit will not be set if the DPLL is selected as the clock source of the transmitter. The source is selected in the TTR.

Table 7-13 (cont.)

Field Name	Function and Meaning
SSOM	This bit is set when the transmitter starts to send the Flag or ABORT character in response to a Send Start of Message, Send Start of Message with PAD, or Abort Transmission command. The bit is set each time any of these commands is issued, and so the I/O processor can use this flag to control the number of Flag or ABORT characters that are sent.
FRMC	The transmitter has started to send the End Of Message sequence in response to a Send End Of Message command. This bit can also be set if the TEMC field in the TPR is set and one of the controlling conditions occurs (see Section 7.2.8.1). It can also be set if underrun occurs and the URUN field in the TPR is zero.
CTSU	The CTS pin on the device was negated when the transmitter was ready to send a character. This bit can be set only if the CTSC field in the TPR is set.
TXEMP	The transmitter has sent a character and the FIFO contains no further characters to send. The transmitter treats this as an underrun condition, and takes the action specified in the URUN field of the TTR.

Table 7-13 (cont.)

Field Name	Function and Meaning
	Character Oriented Protocols (COP)
RXPNT	**BISYNC Only.** The receiver has been sent a DLE-STX sequence and so is operating in transparent mode. The receiver reverts to normal mode (and this bit is cleared) when one of the transparent text terminators is received.
RHUNT	The receiver is searching for a synchronization sequence. This bit is set when the receiver is reset or disabled and is cleared as soon as synchronization occurs.
DPLLE	The Digital Phase-Locked Loop (DPLL) could not find a data transition for two consecutive bits in the detection window. As a result, the DPLL has gone into search mode. This bit will not be set if the DPLL is selected as the clock source of the transmitter. The source is selected in the TTR.
SSOM	The transmitter has started to send a synchronization pattern in response to the "Send Start Of Message" or "Send Start Of Message With PAD" commands. If either of these commands are repeated, the bit will be set again. The controlling processor can use this flag to control the number of synchronization sequences sent.

Table 7-13 (cont.)

Field Name	Function and Meaning
MSGC	The transmitter has started to send the End Of Message sequence in response to a "Send End Of Message" command. This bit can also be set if the TEMC field in the TPR is set and one of the controlling conditions occurs (see Section 7.2.8.1). It can also be set if underrun occurs and the URUN field in the TPR is zero.
CTSU	The CTS pin on the device was negated when the transmitter was ready to send a character. This bit can be set only if the CTSC field in the TPR is set.
TXEMP	The transmitter has sent a character and the FIFO contains no further characters to send. The transmitter treats this as an underrun condition, and takes the action specified in the URUN field of the TTR.

7.2.11.4 Input and Counter/Timer Status Register (ICTSR) - A pair of registers (one for each channel) give information on the status of the counter/timers. The general name for these registers is ICTSR.

Figure 7-15 shows the format of the register, and Table 7-14 explains the meaning of each bit.

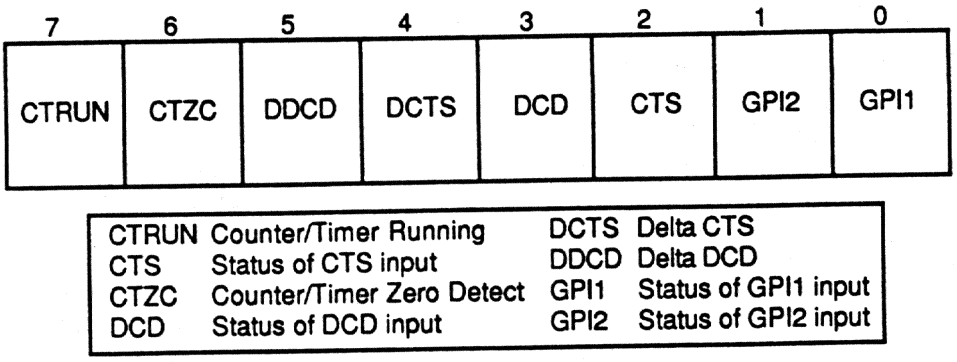


Figure 7-15: The DUSCC Input and Counter/Timer Status Register

Table 7-14: Content of the DUSCC Input and Counter/Timer Status Register

Field Name	Function and Meaning
DCD, CTS, GPI1, and GPI2	The current state of these inputs for this channel.
Delta CTS	Indicates that the level at the CTS pin has changed. This condition can cause an interrupt if the ECTS field in the IER and the appropriate INEA field in the ICR are set.
DDCD	Indicates that the level at the DCD pin has changed. This condition can cause an interrupt if the ECTS field in the IER and the appropriate MINT in the ICR are set.

Table 7-14 (cont.)

Field Name	Function and Meaning
CTZC	The counter/timer has reached zero. This condition can cause an interrupt if the ECTS field in the IER and the appropriate MINT in the ICR are set.
CTRUN	The counter/timer is running. This bit is set on issuing the Start Counter/Timer command, and is cleared on issuing the Stop Counter/Timer command.

7.2.12 Interrupt Registers

The Interrupt Control Register (ICR) provides overall control of the DUSCC's interrupt features. Detailed control of the interrupts is handled through another set of registers:

- Interrupt Enable Register (IER)
- Interrupt Vector Register (IVR)
- Interrupt Vector Register, Modified (IVRM)

The following sections explain the format and content of these registers in more detail.

7.2.12.1 Interrupt Control Register (ICR) - There is one Interrupt Control Register (ICR) on the DUSCC. This gives the TMS processor top-level control of the device's interrupt capabilities. For example, by using this register, the processor can enable or disable interrupts from either channel.

Figure 7-16 shows the format of the register, and Table 7-15 explains the meaning of each bit.

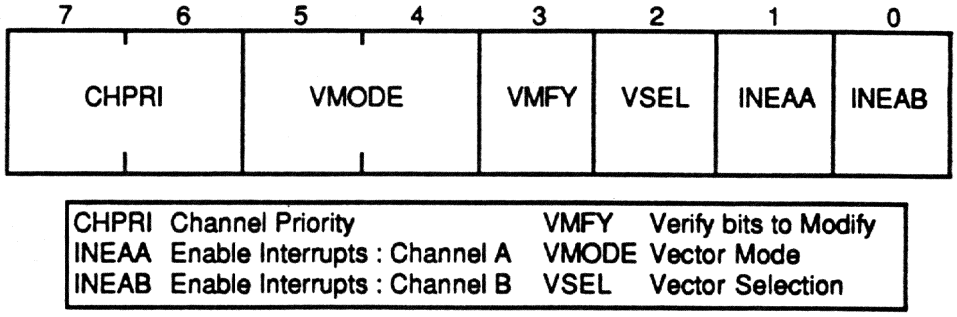


Figure 7-16: DUSCC Interrupt Control Register

Table 7-15: Content of the DUSCC Interrupt Control Register

Field Name	Function and Meaning
INEAB	Enables the generation of interrupts from Channel B. The exact conditions that can generate an interrupt are set up in Channel B's IER.
INEAA	Enables the generation of interrupts from Channel A. The exact conditions that can generate an interrupt are set up in Channel A's IER.
VSEL	Determines which vector is sent as part of the interrupt acknowledgment cycle. If this bit is clear, the vector in the IVR is sent. Otherwise, the vector in the IVRM is sent.

Table 7-15 (cont.)

Field Name	Function and Meaning
VMFY	Selects the bits of the vector to modify when indicating the highest priority interrupt that's pending on the DUSCC. If this bit is clear, bits 0, 1, and 2 of the vector are modified. Otherwise, bits 2, 3, and 4 are affected. Information on interrupt priority follows this table.
VMODE	Sets the response of the DUSCC to an interrupt acknowledge cycle from the I/O processor. A values of 0, 1, or 2 in this field indicates that a vector is to be sent in reply. If the device has an interrupt pending it returns the appropriate vector and asserts the DTACKN pin. If the device has no interrupt pending, it ignores the interrupt acknowledge cycle. A value of 3 indicates that the response does not include sending the vector. The device still calculates the appropriate value and sets up the IVR accordingly. The controlling processor can read the IVR to find the source of the interrupt. Notice that the device does not respond directly to the interrupt acknowledge cycle.
CHPRI	Selects the relative priority between the two channels on the device, and the priority of events that can occur on each channel.

There are four events in a DUSCC channel that can generate an interrupt. The DUSCC assigns the following priority to these events:

1. Receiver Ready
2. Transmitter Ready
3. Receiver/Transmitter Status
4. External Timing or Counter/Timer Status

As there are two channels, there is a total of eight events to be put in priority order. Rather than provide every combination of these events, the device provides the four most useful ones:

1. All channel A events have priority over channel B, giving a complete priority order of:

   A(1), A(2), A(3), A(4), B(1), B(2), B(3), B(4)

2. The events are grouped according to their type: channel A events always have priority over channel B events. This gives a priority order of:

   A(1), B(1), A(2), B(2), A(3), B(3), A(4), B(4)

3. All channel B events have priority over channel A, giving a priority order of:

   B(1), B(2), B(3), B(4), A(1), A(2), A(3), A(4)

4. The events are ordered by their type, but channel B events have priority over channel A events. This gives a complete order of:

   B(1), A(1), B(2), A(2), B(3), A(3), B(4), A(4)

You use the CHPRI field in the ICR to set up the priority order you require. The values this field can have are:

0. Channel A has priority over channel B.
1. Events are ordered by type, with channel A events having priority over channel B.
2. Channel B has priority over channel A.
3. Events are ordered by type, with channel B events having priority over channel A.

7.2.12.2 Interrupt Enable Register (IER) - There are two interrupt enable registers (one for each channel) that you can

use to control which events can cause an interrupt. The generation of an interrupt for an event is controlled by the values in these registers, and the setting of the **INEAA** and **INEAB** fields in the ICR.

Figure 7-17 shows the format of the Interrupt Enable Register (IER).

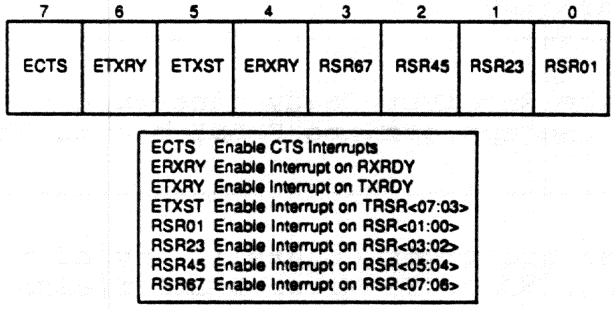


Figure 7-17: DUSCC Interrupt Enable Register

The RSR10, RSR02, RSR45, and ETXST fields set the conditions that can generate the Transmitter/Receiver Status interrupt. The remaining bits enable the other interrupts. Table 7-16 explains the use of each bit.

Table 7-16: Content of the DUSCC Interrupt Enable Register

Field Name	Function and Meaning
RSR01	If set, an interrupt occurs if bits 1 or 0 in the RSR are set.
RSR23	If set, an interrupt occurs if bits 3 or 2 in the RSR are set.
RSR45	If set, an interrupt occurs if bits 5 or 4 of the RSR are set.

Table 7-16 (cont.)

Field Name	Function and Meaning
RSR67	If set, an interrupt occurs if bits 7 or 6 of the RSR are set.
ERXRY	If set, the Receiver Ready interrupt occurs whenever the appropriate RXRDY bit in the GSR is set.
ETXST	If set, an interrupt occurs if any of the TXEMP, CTSU, FRMC, MSGC, SSOM, or DPLLE fields in the TRSR are set.
ETXRY	If set, the Transmitter Ready interrupt occurs whenever the appropriate TXRDY bit in the GSR is set.
ECTS	If set, the External or C/T Status interrupt occurs whenever either of the DDCD or DCTS fields is set in the ICTSR.

7.2.12.3 Interrupt Vector Register (IVR) - There is one Interrupt Vector Register (IVR) on the DUSCC. If the VSEL field in the ICR is zero, the IVR is sent as part of the interrupt acknowledge cycle. It contains the interrupt vector that the controlling processor can use to dispatch to the correct interrupt handler.

When the DUSCC is reset, the value in this register is set to 0F (in hex). To change the value, the controlling processor simply writes the appropriate value to the register.

7.2.12.4 Interrupt Vector Register, Modified (IVRM) - There is one Interrupt Vector Modified Register (IVRM) in the DUSCC. If the VSEL field in the ICR is set, this register is sent as part of the interrupt acknowledge cycle. Its value is a modified form

of the IVR. The modification shows the highest priority interrupt currently active.

The value is modified by replacing three of the bits with the indication of the event being reported. The bits affected are either bits <2:0> or <4:2>, depending on the value of the VMFY field in the ICR. Table 7-17 shows the values used to indicate status.

Table 7-17: Values Used to Modify the Interrupt Vector

Value	Event
0	Receiver ready on channel A
1	Transmitter ready on channel A
2	Receiver or transmitter status change on channel A
3	External or counter/timer status change on channel A
4	Receiver ready on channel B
5	Transmitter ready on channel B
6	Receiver or transmitter status change on channel B
7	External or counter/timer status change on channel B

As long as the TMS processor has set up a value in the IVR, the IVRM will be created and stored in the DUSCC. This occurs irrespective of the value of the Vector Selection bit in the ICR. If the processor has not set a value in the IVR, the IVRM is never calculated.

7.2.13 Miscellaneous Registers

The two remaining registers in the device provide more general features:

- The Pin Configuration Register (PCR) sets the function of the programmable pins on the device for each channel.
- The Output and Miscellaneous Register (OMR) sets a residual character length, and reports the status of the FIFOs in the device.

The following sections give more information on these registers.

7.2.13.1 Pin Configuration Register (PCR) - Each channel has a number of multifunction pins. At any one time, these pins can each handle only one function, and the Pin Configuration Register (PCR) defines the function for each.

Figure 7-18 shows the layout of the register, and Table 7-18 explains the use of each bit.

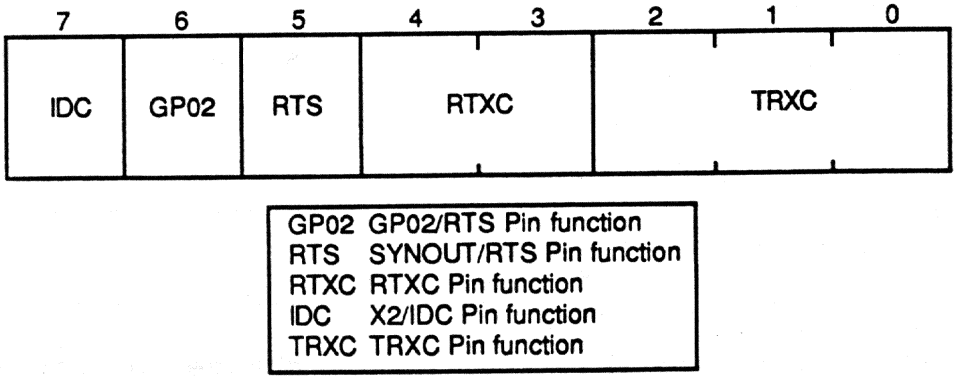


Figure 7-18: DUSCC Pin Configuration Register

Table 7-18: Content of the DUSCC Pin Configuration Register

Field Name	Function and Meaning
TRXC	This pin can be an input or an output. However, in the DEC MicroServer it is used only as an input, and so this field should contain 0.
RTXC	This pin can be an input or an output. However, in the DEC MicroServer it is used only as an input and so this field should contain 0.
RTS	This pin can act as a synchronization/flag detection pin or as a request-to-send output. The DEC MicroServer uses the pin as an output and so the value of this bit should be 1.
GPO2	Not used in the DEC MicroServer.
IDC	This pin can be a crystal clock input or an interrupt daisy-chain output. The DEC MicroServer uses neither of these features, and the pin is connected to ground. So, to ensure correct operation of the device, set this field to 1.

7.2.13.2 Output and Miscellaneous Register (OMR) - The Output and Miscellaneous Register (OMR) for each channel controls the state of the RTS output pin, shows the state of the FIFOs on the device, and sets the length of the last information character sent for each packet. Figure 7-19 shows the format of the register, and Table 7-19 explains the use of each bit.

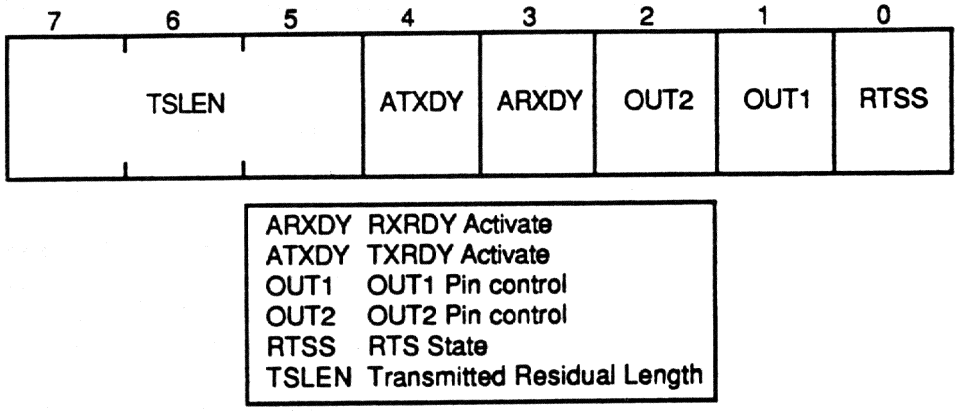


Figure 7-19: The DUSCC Output and Miscellaneous Register

Table 7-19: Content of the DUSCC Output and Miscellaneous Register

Field Name	Function and Meaning
RTSS	Sets the state of the RTS pin for the channel. If this bit is set, the pin is asserted.
OUT1	Not used in the DEC MicroServer.
OUT2	Not used in the DEC MicroServer.
ARXRY	When set, this bit indicates that the receive FIFO is full and needs to be emptied.
ATXRY	When set, this bit indicates that the transmit FIFO is empty.

Table 7-19 (cont.)

Field Name	Function and Meaning
TSLEN	**BOP Only.** Sets the number of bits in the last character of the information field for each packet. The length of all other characters in the information field is set by the Character Length bits in the TPR. To make this character field as long as all the others, use a value of 7 for this field. Otherwise, use a value that is one less than the number of bits in the character. For example, if you want the last character to have 5 bits, put a value of 4 in this field.

7.2.14 Interrupts Generated

Each DUSCC has one interrupt output. These are wired through an OR gate to both the TMS processor and to the VIC. This enables either the TMS processor or the MicroVAX to handle these interrupts, although the TMS processor would normally handle them.

This wiring arrangement means that the processor will be interrupted when either DUSCC creates the appropriate signal. Therefore, bits in the synchronous I/O registers show which of the devices caused the interrupt.

Section 6.4 has more information on these registers.

7.2.15 Effects of Resetting the Device

The device's RESET pin is controlled by the board's RESET signal. This becomes active when one of these events occurs:

1. Power up.
2. The software sets the HWRESET bit in the System Register.

3. A device attached to the external logic port asserts the EXTRESET signal.

When RESET becomes active, the DUSCC:

- Resets the transmitters and receivers
- Sets the IVR and IVRM to 0F (hex)
- Clears these registers:
  - CMR1
  - CMR2
  - TPRA and TPRB
  - RPRA and RPRB
  - OMRA and OMRB
  - PCRA and PCRB
  - RSRA and RSRB
  - TRSRA and TRSRB
  - GSR
  - IERA and IERB
  - ICR

7.3 FIFO

Each DEC MicroServer synchronous port has a FIFO attached to its receive line (CCITT circuit 104).

The FIFOs enable the DEC MicroServer to handle high data speeds at high line utilization. They also isolate the DUSCCs from fluctuations in the line data rate, giving them information at a constant speed.

The information enters the FIFOs at line speed. It leaves them at a steady rate of either 1 MHz or 4 MHz, going to the appropriate Received Data pin on a DUSCC. The TMS processor selects the output speed using registers in the SSL (see Section 7.4).

The logic that looks after the FIFOs -- putting data in, retrieving data, and controlling its clock -- is part of the SSL. Section 7.4 explains the workings of this logic in more detail.

The FIFO for each port is provided by a 4K x 4 static RAM device (NEC PD4362C). The SSL intercepts the information on CCITT Circuit 104, and passes it through the RAM. The SSL generates all the necessary address, data and control signals to do this.

In some protocols that use NRZI coding the data clock is part of the line signal. In these cases, buffering the information in a FIFO would lose this timing information. So, each of the FIFOs can be bypassed and the incoming information sent directly to the appropriate DUSCC. The SSL activates the "switch" that bypasses each FIFO. However, it sets this switch under control of the TMS processor.

Each queue can also be cleared independently of the others. This feature is used during error recovery. Again, the SSL handles this operation under instruction from the TMS processor.

The TMS processor controls the use of the FIFOs using registers in the SSL. Section 7.4 has details of these registers.

NOTE

Don't confuse these FIFOs with those in the DUSCC. The FIFOs discussed here are to improve the overall system performance. Those in the DUSCC are to reduce the DMA overhead of that device.

7.4 Synchronous Support Logic (SSL)

The DUSCCs cannot provide all the functions that the DEC MicroServer needs, so some support logic for the serial interfaces is necessary. This support logic is on its own circuit board, as explained in Chapter 2.

The following sections outline the functions this logic provides, and how they can be controlled by the TMS processor or the MicroVAX.

7.4.1 Functions

The SSL has these parts:

- Transmission synchronizer (one for each transmitter)
- Bit rate generator (one for each channel)
- FIFO controller (one for each receiver)
- Loopback controller

The following paragraphs give more details of what each of these

blocks does.

7.4.1.1 Transmission Synchronizers - As with all logic there is some delay between the DUSCC's transmit clock input being asserted and the device presenting valid data. In the DUSCCs this delay is a maximum of 360 ns. This delay is sufficient for the data and the transmission clock to get out of step at high data speeds (that is, above 1.38M bits/s). Once the data and clock are out of step, incorrect information will be sent by the transmitter.

The SSL avoids such errors by including a transmission synchronizer on each channel. This resynchronises the transmitted data with the transmission clock. There is one synchronizer for each transmit line, and each can be bypassed when using data speeds lower than 1.38M bits/s.

7.4.1.2 Bit Rate Generators (BRGs) - Each bit rate generator can provide eight different clock frequencies for use when sending data. The rate for each channel can be selected separately so that the system can work on mixed line speeds.

The bit rate frequency is in the range 8 kHz to 8 MHz, and can be directed to:

1. Transmit clock input on a DUSCC channel
2. DCE transmit clock circuit (CCITT circuit 114)
3. Data loopback clock for CCITT circuit 115

The frequency and destination of the clock are chosen by setting values in the device's internal registers (see Section 7.4.3).

The BRGs are primarily for the diagnostic firmware to use, and so the speeds offered are not industry-standard ones. However, if two DEC MicroServers were operating "back to back" the BRG clocks could be used for normal data communications.

7.4.1.3 FIFO Controllers - Each FIFO controller accepts information from the serial line, and puts it in the FIFO RAM. The controller also takes the information from the RAM and passes it to the DUSCC using a fixed-rate clock.

The controllers also have a clock control circuit, to halt the clock when any of these conditions occurs:

1. The FIFO becomes empty.
2. The DUSCC is waiting for or processing a DMA transaction through the DMA controllers.
3. The DUSCC detects an end of frame.

The TMS processor controls the FIFOs through registers in the SSL. This enables the processor to bypass the FIFOs, clear them, and select the rate at which data is passed to the DUSCCs.

7.4.1.4 Loopback Controllers - These controllers allow information sent from the DUSCC to be looped back to the DUSCC's receive line. This allows an internal loopback to take place.

Information used in such a loopback should stay within the DEC MicroServer. Therefore, this part of the SSL also includes a switch to force the transmit line itself into an Idle state.

Like other features in the SSL, the TMS processor controls these facilities through an internal register.

7.4.2 Architecture

Basically, the SSL has five functional blocks. One block is the overall controller. For example, this section contains the registers and their access logic.

The remaining blocks contain the control logic for individual channels. Each block contains a FIFO controller, data resynchronizer, bit rate generator, and loopback controller.

Figure 7-20 shows the architecture of one of the channel sections.

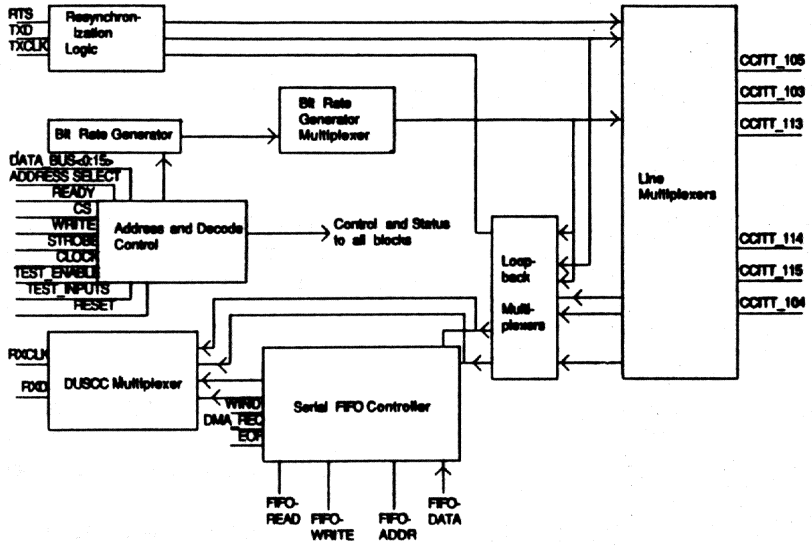


Figure 7-20: Architecture of a Channel in the SSL

7.4.3 Registers

The SSL has 5 registers, each of them 16 bits long. There is one **Channel Control Register (DBCR)** for each channel and they are identified by the names DBCR0, DBCR1, DBCR2, and DBCR3. In addition, there is an **SSL Identification Register (DBIDR)** that gives the revision and ECO levels of the SSL.

The TMS processor cannot access these registers directly. Instead, they are accessed through ports on the SSL, as Section 7.4.4 shows.

The following sections deal with the content of each type of register in detail.

7.4.3.1 Channel Control Registers (DBCR) - The TMS processor uses these registers to set up the behavior of a particular channel. Figure 7-21 shows the format of the register, and Table 7-20 explains the meaning and use of each field.

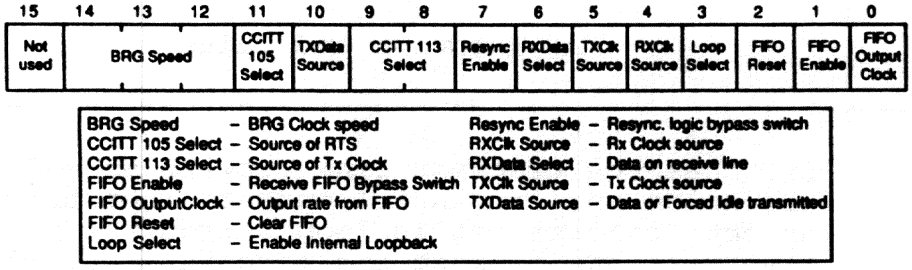


Figure 7-21: The SSL's Channel Control Register

NOTE

In the current implementation of the SSL, each DBCR is a write-only register.

Table 7-20: Content of the SSL's Channel Control Registers

Field Name	Function and Value
FIFO O/P CLK	Selects the rate that information leaves the FIFO on route to the DUSCC. If this field is set, the transfer rate is 4 MHz. If clear, the rate is 1 MHz.
FIFO ENABLE	Switches the FIFO in and out of the receive line. When this field is set, the FIFO is included in the receive line. When clear, the FIFO is bypassed and the DUSCC receives information directly from the communications link.
FIFO RESET	Allows the TMS processor to clear the FIFO. To do this, the processor clears and then sets this bit. In normal use, this bit is set.

Table 7-20 (cont.)

Field Name	Function and Value
LOOP SELECT	Switches the transmit line between normal operation and internal loopback. When this field is set, information is looped from the DUSCC transmit line back to its receive line. When clear, information is transferred in the normal way along the transmit line.
RXCLK SOURCE	Selects the source of the clock used in the receive logic. When set, the information on CCITT circuit 114 is used. When clear, the information on CCITT circuit 115 is used. If LOOP SELECT is set, the clock is always taken from the channel's BRG.
TXCLK SOURCE	Selects the source of the clock used in the DUSCC transmitter. When this field is clear, the line clock on CCITT circuit 114 is routed to the DUSCC. When set, the appropriate BRG on the SSL is used.
RXDATA SELECT	Selects the CCITT circuit used as the received data input to the DUSCC. When this field is set, circuit 115 is used. When clear, circuit 104 is used.
RESYNC ENABLE	Switches the transmit resynchronizer in and out of the transmit line. When this field is clear, the resynchronizer is included in the line. When set, the resynchronizer is bypassed, and the DUSCC output is routed directly to the communications line. This feature is used in X.21.

Table 7-20 (cont.)

Field Name	Function and Value
CCITT 113 SELECT	Selects the source of information routed on CCITT circuit 113. Each value of this field selects a different clock source: 0. Transmit clock from CCITT circuit 114 1. SSL BRG 2. Steady state value of 1 3. Steady state value of 0
TXDATA SOURCE	Selects the data sent on the transmit line. When this field is clear, the line is held in an idle state. When set, information is taken from the appropriate data output pin on the DUSCC. In normal operation, this field would be set. However, when LOOP SELECT is set, this field needs to be cleared to make sure that the looped data is not sent out on the communcations line.
CCITT 105 SELECT	Selects the source of the Request to Send signal for CCITT circuit 105. When this field is set, the signal is taken from the SSL's BRG. When clear, the RTS output from the DUSCC is used. Usually, this field is clear. However, for some X.21 operations, the field is set. This field is automatically set when TXDATA SOURCE is cleared.

Table 7-20 (cont.)

Field Name	Function and Value
BRG SPEED	Selects the speed of the Bit Rate Generator (BRG). Each value in this field selects a different rate: 0. 3.906 kHz 1. 7.812 kHz 2. 15.625 kHz 3. 31.25 kHz 4. 62.5 kHz 5. 250 kHz 6. 1 MHz 7. 2 MHz These BRGs are primarily used by diagnostic software when doing loopback tests

7.4.3.2 SSL Identification Register (DBIDR) - This register contains information on the revision and ECO levels of the SSL. This enables firmware and software to recognize later enhancements to this logic and to act accordingly.

The register is read only, and is shown in Figure 7-22.

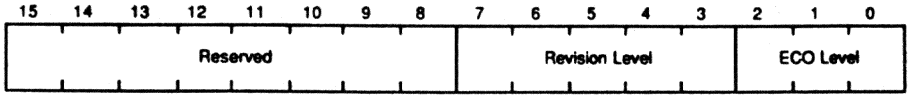


Figure 7-22: The SSL's Identification Register

Both the REV and ECO fields contain numbers. The greater the value, the later the revision or ECO level.

7.4.4 Accessing the Registers

The TMS processor accesses the registers through two ports on the SSL:

- Address Register
- Data Register

Both of these are locations in the TMS processor's I/O address (see Chapter 6 for information on address spaces).

Each register is identified by a number, as Table 7-21 shows.

Table 7-21: Register numbers in the SSL

Register Number (hex)	Register Name
0	DBIDR
1, 2, and 3	Not used
4	DBCR0
5	DBCR1
6	DBCR2
7	DBCR3
8 to A	Reserved for future use

Reading and writing the SSL registers is similar to the way the MicroVAX processor reads and writes the LANCE CSRs. To read a register, the TMS processor puts the appropriate number in the SSL Address Register and then reads its content from the SSL Data Register. Conversely, to set a value in a register, the TMS processor stores the register number in the SSL Address Register, and then writes the register's contents in the SSL Data Register.

7.5 Line Drivers and Connectors

7.5.1 Distribution Panel

The line drivers and the port connectors are on a separate circuit board. This includes the drivers for all the physical line protocols that the DEC MicroServer supports, and with the 50-way ports that the adapter cables are attached to.

Four 40-way ribbon cables connect the main logic module to the distribution panel. Each of these cables carries the signals for one of the synchronous I/O ports.

The drivers and associated circuitry are the same for each port. There is more than one line transmitter and receiver for each port -- the ones used depend on the protocol in use. This protocol information is taken from the modem control registers MDCR1 and MDCR2. The protocol bits of these registers are fed directly to the distribution panel, through the ribbon cables. Chapter 6 has details of the modem control registers and their content.

7.5.2 The 50-Way Socket

All the relevant signals for a port (for all protocols) are routed through to a single 50-way connector that is directly attached to the distribution panel. Figure 7-23 shows the shape and pin numbering on the socket. Table 7-22 lists the function associated with each pin. The table also shows how the signals are used in each electrical interface's socket.

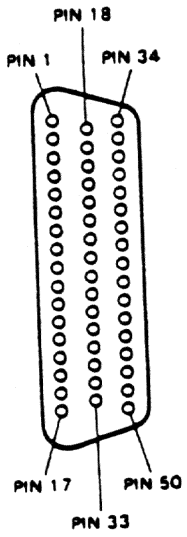


Figure 7-23: The 50-Way Synchronous Socket

Table 7-22: Signals on the 50-way Synchronous socket

Pin	Signal Name	V.35 Pin	V.24 Pin	RS-422 Pin	RS-423 Pin
1	Code Ground	earth	earth	earth	earth
2	Code 0	earth			
3	Code 1		earth		earth
4	Code 2			earth	
5	Code 3				
6	Tx Data A			4	
7	Tx Data B			22	
8	Tx Data		2		4
9	RTS/C A			7	
10	RTS/C B			25	
11	Rx Data A		3	6	6
12	Rx Data B		earth	24	24
13	Local Loop		18	10	10
14	Test 4				
15	Test I		25	18	18
16	Remote Loopback		21	14	14
17	Ring Indicator	J	22	15	15
18	Rx Clock A		17	8	8
19	Rx Clock B		earth	26	26
20	Tx Clock A		15	5	5

Table 7-22 (cont.)

Pin	Signal Name	V.35 Pin	V.24 Pin	RS-422 Pin	RS-423 Pin
21	Tx Clock B		earth	23	23
22	Clock		24		17
23	V35 Tx Clock A	Y			
24	V35 Tx Clock B	a			
25	V35 Clock A	U			
26	V35 Clock B	W			
27	V35 Rx Data A	R			
28	V35 Rx Data B	T			
29	V35 Tx Data A	P			
30	V35 Tx Data B	S			
31	V35 Rx Clock A	V			
32	V35 Rx Clock B	X			
33	DTR	H	20		12
34	DSR A	E	6	11	11
35	DSR B	earth	earth	29	29
36	RTS	C	4		7
37	DCD/I A	F	8	13	13
38	DCD/I B	earth	earth	31	31
39	CTS A	D	5	9	9
40	CTS B	earth	earth	27	27
41	DCE Ground	earth	earth	20	20
42	Test 1				
43	Test 2				
44	DTE Ground	B	7	19, 37	19, 22, 25, 30, 35, 37

Table 7-22 (cont.)

Pin	Signal Name	V.35 Pin	V.24 Pin	RS-422 Pin	RS-423 Pin
45	DTR A			12	
46	DTR B			30	
47	Clock A			17	
48	Clock B			35	
49	Test 3				
50	Speed Select		23	16	16

Notes:

1. The cable code pins (numbers 2 to 5) are connected to the cable ground signal where they are indicated as "earth".
2. Signal pins are connected to the DCE ground signal where they are indicated as "earth".
3. The Test 1 to Test 4 signals are used when the 50-way loopback connector is attached and allow all the signals to be tested.

7.5.3 Adapter Cables

The 50-way socket is general purpose and provides all the signals available on one port. This, of course, cannot be directly connected to a modem which needs the signals for only one interface. The required signals are provided by the use of an interface specific cable, called an adapter cable. Each adapter cable is 2 feet long, and is connected to a modem by a modem extension cable.

In all there are five adapter cables, one for each line protocol. Table 7-23 lists the available adapter cables and their part numbers.

Table 7-23: Synchronous adapter cables

Cable Type	Digital Part Number	Description
BC19B-02	17-01108-01	RS-422 interface adapter
BC19C-02	17-01109-02	X.21 interface adapter
BS19D-02	17-01110-01	V.24 interface adapter. This cable includes the V.24 to RS232C adapter (Part Number: 12-27591-00)
BC19E-02	17-01111-01	RS-423 interface adapter
BC19F-02	17-011112-01	V.35 interface adapter

Chapter 8
DMA devices

8.1 Introduction

The DEC MicroServer uses two of the MicroDMA devices to connect the I_O and SCC buses. Through these DMA devices the MicroVAX processor can access the I/O subsystem. More importantly, they:

1. Handle the transfer of information between the synchronous I/O ports and the Buffer RAM.

2. Enable the TMS processor to get commands from and put status information into the Buffer RAM.

The following sections summarize the device's capabilities and show how two of them are used in the DEC MicroServer. As before, this chapter concentrates on the features of the device that the DEC MicroServer uses. If you want more detailed information of what the device can do, refer to the literature listed in Appendix I.

8.2 Summary of the MicroDMA

The MicroDMA has four independent DMA channels that can transfer information from the I_O bus to the SCC bus. The transfer speed can be up to 10M bytes/s.

Software in the MicroVAX or in the TMS processor controls the behavior of the device and of each channel, by setting appropriate values in the device registers. Section 8.3 has more information on the device registers.

Each channel has three ways of operating. In the DEC MicroServer, only two of these are used:

1. **DMA** -- transferring information between the I_O bus and the SCC bus.

2. **WINDOW** -- allowing the TMS processor to access the Buffer RAM on the I_O bus.

The device also provides another window, separate from the main

channels. This I_O ACCESS window gives the MicroVAX access to the devices on the SCC and TMS buses.

Sections 8.5 and 8.6 show how the processors use these devices.

8.3 Device Registers

In all, there are 56 registers in each DMA device. Both the MicroVAX and the TMS processor can access these registers. When the system is running, the TMS processor is the main user of the registers to set up data transfers, and to fetch commands.

The MicroVAX uses the registers when loading or dumping the system. The TMS processor uses them when the system is running.

8.3.1 Types of Register

There are two types of register in the device:

- **Global Registers** -- that control the operation of the entire device
- **Channel Registers** -- that control the operation of one channel

8.3.2 Global Registers

Each DMA device has one set of Global Registers. All of these registers can be read by either processor in the DEC MicroServer. However, only the MicroVAX processor can set values in any of these registers.

Table 8-1 lists these registers and summarizes their function. The sections after the table give more information on each register.

Table 8-1: The MicroDMA Global Registers

Mnemonic Name	Full Name	Function
DGCTL	Global Control Register	Sets up overall operation of the device and provides information on the version of the device.
DSBR	System Base Register	Address of the system page table in the MicroVAX.
DGBR	Global Base Register	Address of the global page table in the MicroVAX.
DGTRN	Translation Temporary	Internal temporary storage used during address translation.
DGPA	Peripheral Address	Address of a device on the SCC bus. Used in maintenance and test only.
DGTEMP	Global Temporary	Internal temporary register used for maintenance and testing only.
DGBC	IO Access Byte Count	Used for maintenance and testing only.

8.3.2.1 Global Control Register (DGCTL) - The MicroVAX uses the DGCTL to set up the overall operation of the DMA device. The register also contains some information on the device (such as its version number).

Figure 8-1 shows the format of the register. Table 8-2 explains the use of each field.

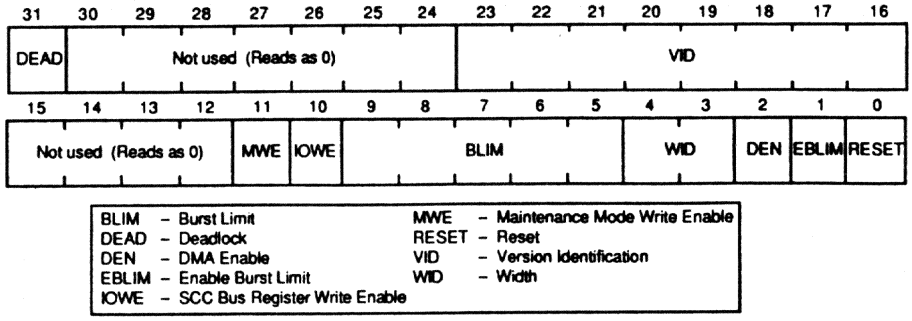


Figure 8-1: Format of the DGCTL Register in the MicroDMA

Table 8-2: Fields in the MicroDMA's Global Control Register

Field Name	Values and Function
RESET	Resets the device and stops any operations currently in progress. Clears the ENABLE bit in each of the channel control registers to stop these operations. All other register values are preserved.
EBLIM	Limits the size of any burst transfer to the number of bus cycles specified in the BLIM field.
DEN	Allows the device to ask for, and carry out, DMA operations on the I_O bus. Normally, this bit is set to allow DMA operations. However, by clearing this bit the MicroVAX can stop the DMA device from using the I_O bus. **Reviewers:** Do we use this ?

Table 8-2 (cont.)

Field Name	Values and Function
WID	This field tells the DMA device how wide the SCC bus is. The device can handle bus widths between 8 and 32 bits, and to work properly it needs to know the width of the bus. In the DEC MicroServer, this field always has a value of 3.
BLIM	Limits the number of bus cycles that the DMA device can use on the I_O bus at any one time. The limit in this field is used only if the EBLIM field is set.
IOWE	The MicroVAX processor can set this bit to let the TMS processor alter the value of the channel registers.
MWE	Normally the DGTRN, DGPA, DGTEMP, and DGBC registers are read-only. However, maintenance software can set this bit to set values in these registers.
VID	This is a read-only field that contains the version number of the DMA device.
DEAD	Indicates that a deadlock occurred in the operation that the device last carried out. This bit helps the MicroVAX or TMS processor determine the cause of a machine check. To clear this condition, the MicroVAX needs to write a 1 in this bit.

8.3.2.2 System Base Register (DSBR) - This register contains a copy of the MicroVAX processor's System Base register. The MicroDMA uses the page table that this register points to when accessing locations on the I_O bus. Bits 31, 30, 1 and 0 must be zero.

8.3.2.3 Global Base Register (DGBR) - This register contains the starting address of the Global page table. This address is a System Virtual address.

The MicroVAX needs to load this register before the MicroDMA does any address translation. Doing this will avoid Buffer RAM being corrupted.

Once the MicroDMA is running, writing a new value to this register stops all the device's DMA channels.

8.3.2.4 Translation Temporary (DGTRN) - A temporary register that the device uses when translating addresses. Normally this register is read only, but maintenance software can load the register by setting the MWE field in the Global Control Register.

8.3.2.5 Peripheral Address (DGPA) - A register that can be used by maintenance or test software. Normally this register is read-only, but maintenance software can load the register by setting the MWE field in the Global Control Register.

8.3.2.6 Global Temporary (DGTEMP) - A register that can be used by maintenance or test software. Normally this register is read-only, but maintenance software can load the register by setting the MWE field in the Global Control Register.

8.3.2.7 IO Access Byte Count (DGBC) - A register that can be used by maintenance or test software. Normally this register is read-only, but maintenance software can load the register by setting the MWE field in the Global Control Register.

8.3.3 Channel Registers

Each MicroDMA device has four sets of channel registers -- one set for each DMA channel. These registers can be read by either the MicroVAX or TMS processors.

The MicroVAX can write to them at any time, but the TMS processor can write to them only if the IOWE field in the Global Control Register is set. If the TMS processor tries to write to the registers when IOWE is clear, the write completes normally, but

the register values do not change.

Table 8-3 lists these registers and summarizes their functions. The sections after the table give more information on each register.

Table 8-3: The MicroDMA Channel Registers

Mnemonic Name	Full Name	Function
DCCTL	Channel Control Register	Sets up overall operation of the channel and provides channel status information.
DCINT	Channel Interrupt Register	Holds the MicroVAX Interrupt Vector associated with the channel.
DCIOBA	Peripheral Base Address	The starting address of the device or memory on the SCC bus that will send or receive information.
DCIDS	DMA Source Address	Address of device or memory on the SCC bus. Used in a DMA mode that the DEC MicroServer does not use.
DCIBC	Initial Byte Count	The number of bytes to transfer in a particular DMA operation.
DCWM	Window Mask	Contains an address mask used in address translation when the channel is accessed in WINDOW mode.

Table 8-3 (cont.)

Mnemonic Name	Full Name	Function
DCBO	Byte Offset	Indicates the byte of the page in Buffer RAM where a DMA operation will begin. The page number is held in DCSPTE.
DCUPA	MicroVAX Physical Address	Contains the starting physical address of a block of memory in Buffer RAM. Used in DMA and WINDOW modes when virtual address translation is not used on the I_O bus.
DCSPTE	SAVPTE Register	Contains the address of the page table entry for the page in Buffer RAM where a DMA transfer is to start. Used with the DCBO register in DMA and WINDOW modes.
DCIDD	Destination Address	Destination on the SCC bus of a DMA transfer. Used only in a mode that the DEC MicroServer does not use.
DCCSV	Current SAVPTE Register	Current value of SAVPTE when an error occurred. Used for error recovery or maintenance only.
DCIOA	Current SCC bus address	Used in maintenance only.

Table 8-3 (cont.)

Mnemonic Name	Full Name	Function
DCBC	Current Byte Count	Contains the number of bytes still to be transferred in this operation.
DCPTE	Current Page Table Entry	Contains the current page table entry being used in this operation.
DCPA	Current Physical Address	Contains the current physical address on the I_O bus being used in this operation.

Note: Each register occurs four times in the device. To distinguish between them, the number of the DMA channel they control or report on is added to the mnemonic name (channel numbers start at 0). So, for example, DCCTL0 is the Channel Control Register for DMA channel 0.

8.3.3.1 Channel Control Register (DCCTL) - This register controls the operation of the channel. It also provides status information once an operation has completed. Figure 8-2 shows the format of the register. Table 8-4 explains the use of each field.

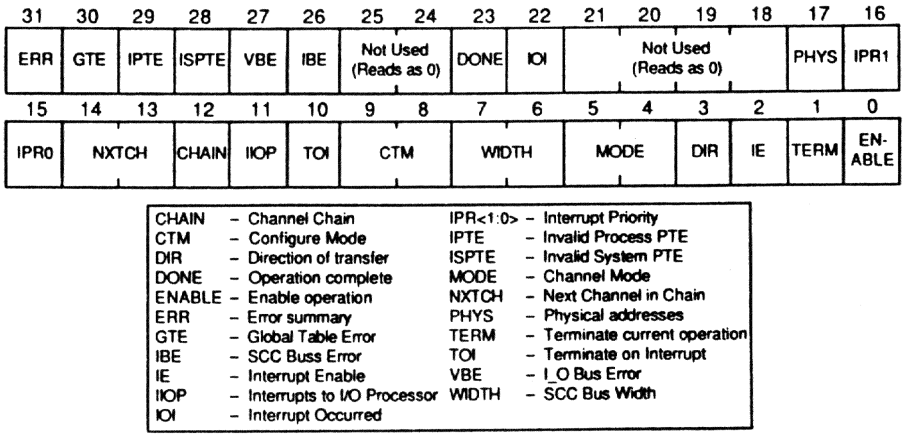


Figure 8-2: The Channel Control Register for the MicroDMA

Table 8-4: Fields in the MicroDMA's Channel Control Register

Field Name	Function
ENABLE	Starts or stops a channel operation. To carry out a DMA or WINDOW operation, the configuration fields of this register and other complete registers need to be set accordingly. To start the operation, the controlling processor sets this ENABLE bit. To abort an operation subsequently, the processor clears this bit.
TERM	Forces the premature termination of a channel operation. However, unlike clearing ENABLE, setting this bit also causes the channel's internal buffers to be flushed. The normal termination action, including generating an interrupt, also occurs.
IE	Tells the device to generate an interrupt when an operation is complete, or ends in error. This bit is set before, or at the same time as, ENABLE is set.

Table 8-4 (cont.)

Field Name	Function
DIR	For DMA operations, this bit determines the direction of data transfer. When set, the transfer occurs from the SCC bus to the I_O bus. When clear, the transfer occurs in the opposite direction. This bit is not used for WINDOW mode where the direction is determined by whether the TMS processor is doing a read or write operation.
MODE	Sets the mode that the channel is to use for the next operation. In the DEC MicroServer only two values are used: 2 -- for DMA mode 3 -- for WINDOW mode
WIDTH	Tells the device how wide the device on SCC bus is. In the DEC MicroServer, this field contains 0 or 1 when communicating with the DUSCCs, and contains 2 when communicating with the TMS memory or the modem status registers.

Table 8-4 (cont.)

Field Name	Function
CTM	Controls the way addresses on the SCC and the I_O bus are updated during a DMA operation. After each byte is transferred, either or both addresses can be HELD or INCREMENTED. The value of this field determines which is held and which is incremented. In the DEC MicroServer, only two values are used. A value of 2 is used during DMA operations between SCC devices and the Buffer RAM. This value causes the SCC address to be held after each transfer, and the Buffer RAM address to be incremented. A value of 3 is used during dump operations to recover information from the TMS RAM. This value causes both addresses to be incremented after each transfer.
TOI	Tells the channel to end the transfer on receiving an interrupt from the appropriate device on the SCC bus. This interrupt comes through the appropriate IIR pin on the device, and in the DEC MicroServer is used for receive operations only.
IIOP	When set, tells the DMA to send any interrupt for this channel to the TMS processor rather than the MicroVAX. Unless the MicroVAX is in full control of the synchronous I/O ports, this is the normal setting for this bit.
CHAIN	Makes the channel chain an operation to another channel. The DEC MicroServer does not use this feature.

Table 8-4 (cont.)

Field Name	Function
NXTCH	Specifies the number of the channel that this operation is to be chained to, when it completes. This field is used only with IIOP. These fields enable the processors to set up a chain of DMA operations, each starting after the previous one ends. The DEC MicroServer does not use this feature.
IPR	Sets the priority level of any interrupt sent to the MicroVAX. The IPL is between 14 and 17. When this field contains 0 the IPL is 14, and when it contains 3, the IPL is 17.
PHYS	Determines whether the addresses on the I_O bus are physical (and so can be used directly) or virtual (and so need translating using a page table). If this bit is set, the addresses are taken as physical, and so the channel uses the DCUPA register as the start of the buffer this transfer is to use. In this case, the DCSPTE register is not used. If the bit is clear, addresses on the I_O bus are assumed to be virtual. The DCSPTE register points to a page table entry that the device uses to determine the physical address of each byte to transfer. The DEC MicroServer can work with either setting of this bit.
IOI	Indicates that an interrupt occurred on the appropriate IIR pin of the device. If the TOI bit is also set, this interrupt terminates the DMA operation. This bit is cleared on reset and when the ENABLE bit is set.

Table 8-4 (cont.)

Field Name	Function
DONE	Indicates that a channel operation has finished. The bit is cleared on reset and when the ENABLE bit is set.
IBE	Indicates that an error on the SCC bus occurred during the previous operation. This bit is cleared at reset, on setting the ENABLE bit, or by writing 1 to this bit.
VBE	Indicates that an error on the I_O bus occurred during the previous operation. The bit is cleared at reset, when the ENABLE bit is set, or by writing 1 to this bit.
ISPTE	Indicates that an invalid System Page Table Entry was fetched during an address translation. This bit is cleared at reset, when the ENABLE bit is set, or by writing 1 to this bit.
IPTE	Indicates that an invalid Process Page Table Entry was fetched during an address translation. This bit is cleared at reset, when the ENABLE bit is set, or by writing 1 to this bit.
GTE	Indicates that an error occurred while translating a virtual address to a physical one. IPTE and ISPTE may also be set. This bit is cleared on reset, when the ENABLE bit is set, or by writing 1 to this bit.
ERR	Indicates that an error occurred during the previous operation. This bit is set if any of the IBE, VBE, ISPTE, IPTE, or GTE fields are set. The bit is cleared at reset, when the ENABLE bit is set, or by writing 1 to this bit.

8.3.3.2 Channel Interrupt Register (DCINT) - This register contains the interrupt vector for the MicroVAX processor. Figure 8-3 shows the format of the register.

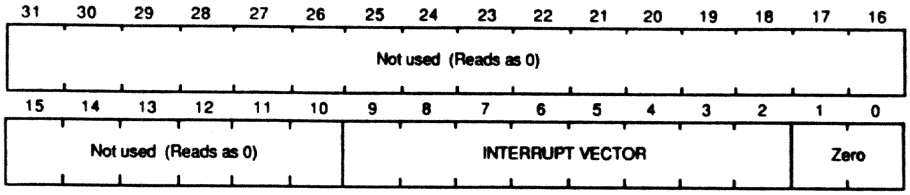


Figure 8-3: Channel Interrupt Register for the MicroDMA

8.3.3.3 Peripheral Starting Address (DCIOBA) - The starting address on the SCC bus to be used in a DMA transfer. The controlling processor loads this register before setting the ENABLE bit in the DCCTL.

To start the transfer, the device copies this register to the DCIOA register. This copy may be modified during the DMA operation depending on the value of the CTM field in the DCCTL register.

Only the least significant 24 bits of this register are used. Bits 31 to 24 should contain zero.

8.3.3.4 DMA Source Address (DCIDS) - The device uses this register for one of the operating modes other than DMA and WINDOW. Therefore, the DEC MicroServer does not use this register.

8.3.3.5 Initial Byte Count (DCIBC) - The controlling processor uses this register to set up the size of a DMA transfer. When the transfer starts, the device makes a copy of this number in the DCBC register. This copy is decremented for each byte transferred and the operation completes when DCBC reaches zero.

The least significant 30 bits of this register are used, which gives a maximum DMA transfer of 1G byte. Bits 31 and 30 should be zero.

8.3.3.6 Window Mask (DCWM) - Contains the channel window mask when the DMA is used in window mode. There is a DCWM register for each of the four channels.

8.3.3.7 Byte Offset (DCBO) - The number of bytes left to be transferred in this operation. At the start of a DMA operation, this register is loaded with a copy of the DCIBC register.

8.3.3.8 MicroVAX Physical Address (DCUPA) - This register holds the starting physical address of the buffer or window in the Buffer RAM to be used in a DMA or WINDOW operation.

This register is used when the MicroVAX addresses memory only through physical addresses. This is the recommended mode for the DEC MicroServer as it does not have the overhead of address translation.

8.3.3.9 SAVPTE Register (DCSPTE) - This register contains the virtual address of a page table entry. That entry points to the page in the Buffer RAM that the channel is using. The DCBO register holds the offset of the byte in that page.

8.3.3.10 Destination Address (DCIDD) - Used only in one of the operating modes other than DMA and WINDOW. The DEC MicroServer does not use this register.

8.3.3.11 Current SAVPTE Register (DCCSV) - This register contains a copy of the current SVAPTE. If an address translation error occurs, the error recovery software can get the address of the offending Page Table Entry from this register.

Usually this register is read-only, but can be written in maintenance mode (when the MWE field in the Global Control Register is set).

8.3.3.12 Current SCC Bus Address (DCIOA) - If an error occurs on the SCC bus, this register contains the address that caused the error.

8.3.3.13 Current Byte Count (DCBC) - The number of bytes that have yet to be transferred in the current DMA operation. If the operation is prematurely stopped, the controlling processor can read this register to see how many bytes did not get transferred.

8.3.3.14 Current Page Table Entry (DCPTE) - This register is usually only of use to the MicroDMA. It contains the current page table entry that the channel is using. However, if an invalid page table entry error occurs, the error recovery software can use this register to determine the possible cause of the error.

Usually this register is read-only, but can be written in maintenance mode (when the **MWE** field in the Global Control Register is set).

8.3.3.15 Current Physical Address (DCPA) - Not used in the DEC MicroServer.

8.3.4 Register Addressing

Each of the MicroDMA's registers is 32 bits wide, and so different addressing mechanisms have to be used by the MicroVAX and TMS processors. The following sections show how each processor addresses the registers.

8.3.4.1 MicroVAX Addressing - To the MicroVAX, the registers appear as a 512 byte block that starts on a longword boundary.

The registers are in the following order:

1. Global registers
2. DMA Channel 0 registers

3. DMA Channel 1 registers

4. DMA Channel 2 registers

5. DMA Channel 3 registers

Table 8-5 shows the offsets of the global registers, and Table 8-6 shows the offsets of the channels registers. See the address map in Appendix B for the position of the register blocks in the MicroVAX address space.

Table 8-5: MicroVAX Offsets for the DMA Global Registers

Register Mnemonic	Offset	Register Mnemonic	Offset
DGCTL	000	DGTRN	020
DSBR	004	DGPA	024
DGBR	008	DGTEMP	028
00C to 01F are RESERVED		02C to 037 are RESERVED	
		DGBC	038

Note: All offsets are in hexadecimal.

Table 8-6: MicroVAX Offsets for the DMA Channel Registers

Register Mnemonic	Channel 0 Offset	Channel 1 Offset	Channel 2 Offset	Channel 3 Offset
DCCTL	040	080	0C0	100
044 to 047, 084 to 087, 0C4 to 0C7, and 104 to 107 are RESERVED				
DCINT	048	088	0C8	108
DCIOBA	04C	08C	0CC	10C
DCIDS	04C	08C	0CC	10C
DCIBC	050	090	0D0	110
DCWM	050	090	0D0	110

Table 8-6 (cont.)

Register Mnemonic	Channel 0 Offset	Channel 1 Offset	Channel 2 Offset	Channel 3 Offset
DCBO	054	094	0D4	114
DCUPA	054	094	0D4	114
DCSPTE	058	098	0D8	118
DCIDD	058	098	0D8	118
05C to 05F, 09C to 09F, 0DC to 0DF, and 11C to 11F are RESERVED				
DCCSV	060	0A0	0E0	120
DCIOA	064	0A4	0E4	124
DCBC	068	0A8	0E8	128
DCPTE	06C	0AC	0EC	12C
DCPA	070	0B0	0F0	130
Note: All offsets are in hexadecimal.				

Section 8.3.4.3 shows how some offsets are shared by more than one register.

8.3.4.2 TMS Processor Addressing - The TMS processor is a 16-bit word machine, and so the registers appear as a block of 128 words. Each register is divided into a high and a low portion. The least significant portion appears in the register block first.

The registers appear in the block in the following order:

1. Global registers
2. DMA Channel 0 registers
3. DMA Channel 1 registers
4. DMA Channel 2 registers
5. DMA Channel 3 registers

Table 8-7 shows the offsets of the global registers, and Table 8-8 shows the offsets of the channels registers. See the address map in Appendix B for the position of the register blocks in the TMS processor's address space.

Table 8-7: TMS 32020 Offsets for the DMA Global Registers

Register Mnemonic	Offset	Register Mnemonic	Offset
DGCTL_L	000	DGTRN_L	010
DGCTL_H	001	DGTRN_H	011
DSBR_L	002	DGPA_L	012
DSBR_H	003	DGPA_H	013
DGBR_L	004	DGTEMP_L	014
DGBR_H	005	DGTEMP_H	015
006 to 00F are RESERVED		016 to 01B are RESERVED	
		DGBC_L	01C
		DGBC_H	01D

Notes:

1. All offsets are in hexadecimal.
2. "_L" refers to the least significant portion of each register, and "_H" refers to the most significant portion.

Table 8-8: TMS 32020 Offsets for the DMA Channel Registers

Register Mnemonic	Channel 0 Offset	Channel 1 Offset	Channel 2 Offset	Channel 3 Offset
DCCTL_L	020	040	060	080
DCCTL_H	021	041	061	081
022, 023, 042, 043, 062, 063, 082, and 083 are RESERVED				
DCINT_L	024	044	064	084
DCINT_H	025	045	065	085
DCIOBA_L	026	046	066	086
DCIOBA_H	027	047	067	087
DCIDS_L	026	046	066	086
DCIDS_H	027	047	067	087
DCIBC_L	028	048	068	088
DCIBC_H	029	049	069	089

Table 8-8 (cont.)

Register Mnemonic	Channel 0 Offset	Channel 1 Offset	Channel 2 Offset	Channel 3 Offset
DCWM_L	028	048	068	088
DCWM_H	029	049	069	089
DCBO_L	02A	04A	06A	08A
DCBO_H	02B	04B	06B	08B
DCUPA_L	02A	04A	06A	08A
DCUPA_H	02B	04B	06B	08B
DCSPTE_L	02C	04C	06C	08C
DCSPTE_H	02D	04D	06D	08D
DCIDD_L	02C	04C	06C	08C
DCIDD_H	02D	04D	06D	08D
02E, 02F, 04E, 04F, 06E, 06F, 08E, and 08F are RESERVED				
DCCSV_L	030	050	070	090
DCCSV_H	031	051	071	091
DCIOA_L	032	052	072	092
DCIOA_H	033	053	073	093
DCBC_L	034	054	073	093
DCBC_H	035	055	075	095
DCPTE_L	036	056	076	096
DCPTE_H	037	057	077	097
DCPA_L	038	058	078	098
DCPA_H	039	059	079	099

Notes:

1. All offsets are in hexadecimal.
2. "_L" refers to the least significant portion of each register, and "_H" refers to the most significant portion.

Section 8.3.4.3 shows how some offsets are shared by more than one register.

8.3.4.3 Sharing Offsets Between Registers - As Tables 8-6 and 8-8 show, some offsets are used by more than one register. The registers control mutually exclusive functions, and so can share the same position in the register map.

Figure 8-4 shows which registers share locations, and Table 8-9 shows how these offsets are shared.

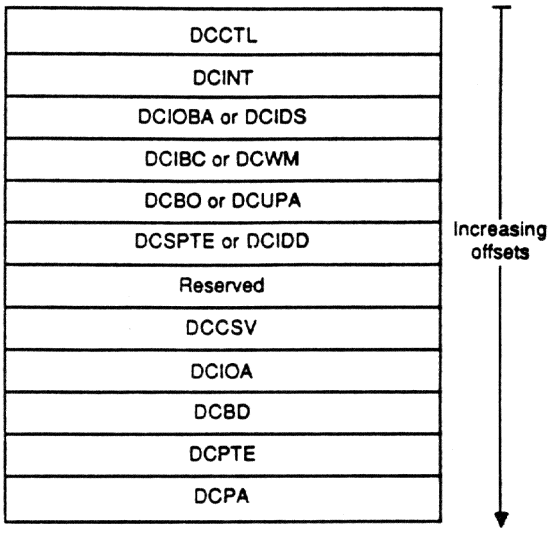


Figure 8-4: Some Positions in the Register Map are Shared

Table 8-9: Shared Offsets in the MicroDMA Register Maps

Register Mnemonic	Condition
DCIOBA	DMA mode only
DCIDS	Not used in the DEC MicroServer
DCIBC	Used in DMA mode only
DCWM	Used in WINDOW mode only
DCBO	Used if the system uses virtual addresses (PHYS field in the DCCTL is 0)
DCUPA	Used if the system uses physical addresses (PHYS field in the DCCTL is 1)

Table 8-9 (cont.)

Register Mnemonic	Condition
DCSPTE	Used in DMA or WINDOW modes if the system uses virtual addresses
DCIDD	Not used in the DEC MicroServer

8.4 DMA Channel Priorities and Device Chaining

Within each DMA device, the channels have a fixed priority of:

- Channel 0
- Channel 1
- Channel 2
- Channel 3

That is, for any event, DMA Channel 0 will always be serviced ahead of the others.

The two devices in the DEC MicroServer are daisy-chained together. This means that one device always has priority over the other, and so effectively gives a fixed priority for all eight DMA channels:

- **MicroDMA A:**
  - Channel 0
  - Channel 1
  - Channel 2
  - Channel 3
- MicroDMA B:

- Channel 0
- Channel 1
- Channel 2
- Channel 3

This inherent priority of the channels is used when assigning the tasks to each channel. Tasks that are high priority (such as high speed data transfers) are always assigned to the channels on the higher priority device.

Conversely, tasks with a relatively low priority (such as the TMS processor accessing new commands in the Buffer RAM) use low priority channels on the low priority device.

8.5 How the MicroVAX Uses the MicroDMA

The MicroVAX processor uses the MicroDMA as a window to the TMS and the SCC bus. Normally, it does not need to access this part of the DEC MicroServer as it is looked after by the TMS processor. However, the MicroVAX needs this access either when:

- o Loading the TMS processor with code
- o Controlling the synchronous I/O ports without using the TMS processor

The MicroVAX uses the window capability of the DMA devices, and can use either device. This window is completely separate from the DMA channels themselves. Appendix B shows the position of the windows in the MicroVAX address map, and also the contents of each window.

To prevent potential system lock up, the MicroVAX can only access the I/O subsystem when the TMS processor is in a reset state. To reset the TMS processor, the MicroVAX sets the RSTDSP bit in the system control register. Chapter 4 has more information on the system control register.

8.6 How the TMS Processor Uses the DMA

The main use of the DMA devices is to transfer information between the Buffer RAM and the synchronous I/O ports. Each line is full duplex, and so a total of eight DMA channels are needed. Table 8-10 shows which DMA channel is used for each synchronous line.

Table 8-10: Allocation of the DMA Channels Between the Synchronous Lines

DMA		
Device	**Channel**	**Sync. Line**
A	0	Port 0 Tx
A	1	Port 0 Rx
A	2	Port 1 Tx
A	3	Port 1 Rx
B	0	Port 2 Tx
B	1	Port 2 Rx
B	2	Port 3 Tx
B	3	Port 3 Rx

Key:

Tx = Transmit line
Rx = Receive line

The TMS processor also needs a DMA window to get commands from the Buffer RAM. This would need another DMA channel. This means that a total of nine DMA channels are needed, but only eight are available. So the lowest priority channel (Channel 3 on DMA B) is used only in window mode for passing commands. Data received on DMA B Channel 3 is read by the TMS or transferred into I/O memory on a window cycle.

When operating for a synchronous line, each DMA channel works in one of two ways:

- **DMA** -- to transfer information between the Buffer RAM and the appropriate transmit or receive line.
- **WINDOW** -- (DMA B Rx 3 only) so the TMS processor can access the ring buffer entry for that channel. (See chapter 15 for more details of the ring buffer entries.)

8.7 Interrupts

The DEC MicroServer uses just one interrupt from each DMA device. These are combined in an OR gate to provide a single interrupt (DMAINT) that is routed to the TMS processor.

Normally, the TMS processor handles the interrupt. In this case, the MicroVAX masks off the interrupt. However, the processor can pass the interrupt on to the MicroVAX through the TMISINT register. DMAINT already goes directly to the MicroVIC.

When the interrupt occurs, the appropriate processor needs to know which device wants attention. The Synchronous status register for Ports 0 and 1 contains two bits -- one for each DMA device. When the DMAINT occurs, one of these bits will be set accordingly. Section 6.4 has more information on this control register.

Chapter 10 has more information on DMAINT and its use.

Chapter 9
Miscellaneous Functions

9.1 Overview

The preceding chapters have covered the major parts of the DEC MicroServer architecture. This chapter contains information on the remaining miscellaneous parts:

1. Watchdog timer
2. LED display
3. External logic interface

All of these devices are attached to the MicroVAX bus.

9.2 Watchdog Timer

The watchdog timer is a simple mechanism that helps prevent the system from hanging in a software loop. The timer uses a counter incremented by a 100 Hz clock. When the counter reaches a preset value, it generates an interrupt to the MicroVAX. This interrupt occurs through the MicroVAX's HALT pin.

The DYRC that manages the System RAM provides the 100 Hz timing. This is permanently enabled, and is derived from the 40 MHz system clock that drives the DYRC.

In normal operation, the software regularly resets the timer before it reaches the expiry period, and so the interrupt never occurs. However, if the software gets into a never-ending loop, it cannot reset the timer and so the interrupt occurs. The interrupt causes a system dump and reboot.

The watchdog timer is built around a HCT4020 device, as shown in Figure 9-1.

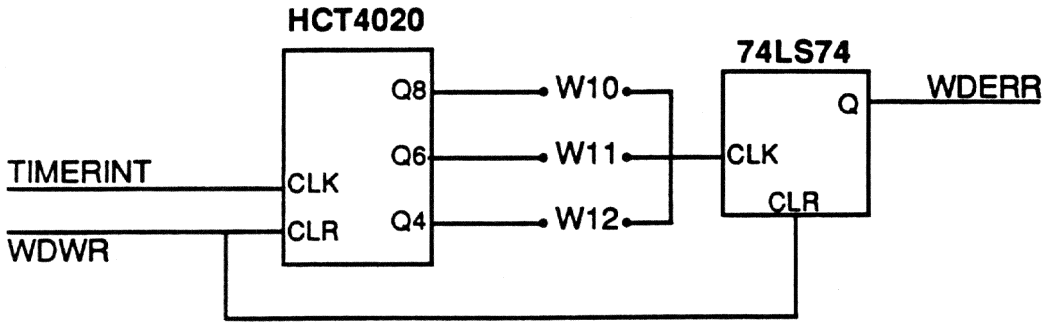


Figure 9-1: Simplified Diagram of the Watchdog Timer

The HCT4020 has 14 outputs, but here only Q4, Q6, and Q8 are used. One of these is wired to the clock input of the 74LS74 through the appropriate jumper (W10, W11, or W12). The jumper used determines the length of the timer period (that is, the time before the counter produces its interrupt):

W10 2.56 seconds

W11 640 milliseconds

W12 160 milliseconds

All of these are nominal values.

The jumper is put in during manufacture and cannot be altered in the field. The default jumper used is W12, giving a watchdog period of 160 ms.

If the timer expires, the appropriate Q output from the HCT4020 drives the clock input of the 74LS74 which sets the WDERR interrupt. To prevent the timer expiring, the system software writes to address 27FFF800 in the MicroVAX address space. In hardware, this write operation generates the WDWR signal that clears the counter and the interrupt.

In the field, the watchdog timer is permanently enabled. However, for manufacturing purposes adding a jumper (W8) to the board disables the timer.

9.3 LED Display

The DEC MicroServer includes a 7-segment LED display to give the user status information. The display shows the progress of the power up and boot sequence. It is also used to indicate errors, and to indicate that the system is running.

9.3.1 Hardware

Figure 9-2 is a simplified connection diagram of the display.

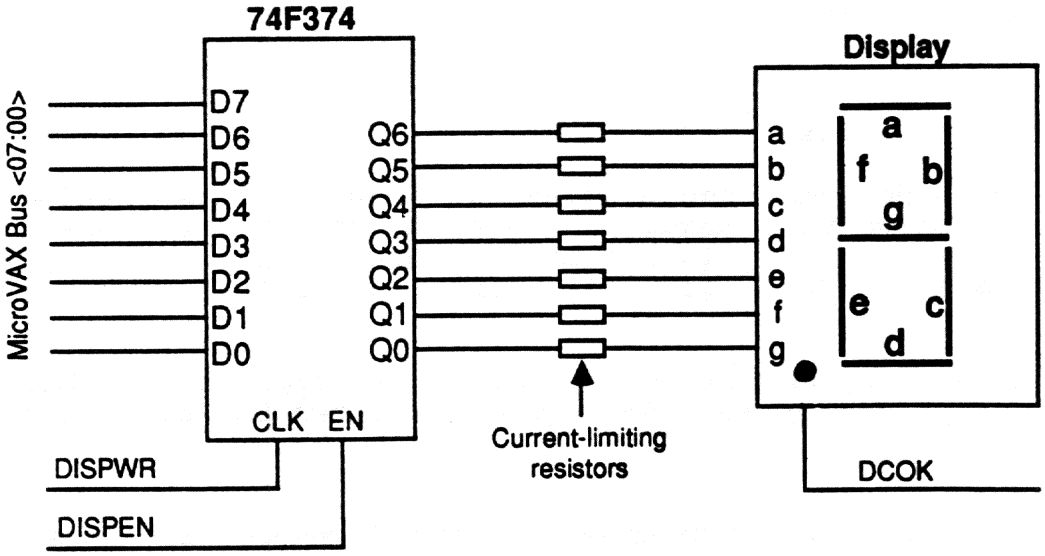


Figure 9-2: Simplified Diagram of the LED Display

Essentially the display has two parts:

- The display register (74F374)
- The display itself

Notice that there is no decoder, so to display a particular value the software has to set the appropriate bit pattern in the register. (Section 4.9.2 deals with the register and its bit allocation.) The register drives the display by acting as a current sink. The amount of current that it sinks is limited by the resistors connected to each Q output.

The register is loaded from the least significant eight bits of the Buffered MicroVAX bus (UVBDAL<7:0>) when the DISPWR signal is set. This signal comes from the MicroVAX address decoder and is

set when the processor writes to address 27FFF804.

The Q outputs of the register are routed directly to the seven-segment display. The DISPEN signal acts as a switch for the Q outputs. When DISPEN is high, the outputs are turned on and the value in the register appears on the display. When DISPEN is low, the Q outputs of the register are tri-stated, and so the display is blank. DISPEN is one of the bits in the system control register that the MicroVAX software can set.

The decimal point of the display is used as the DEC MicroServer's DC power indicator. This connection on the display is wired to the DCOK signal that is asserted only when the DC supplies are within tolerance limits.

9.3.2 Display Values

As there is no decoder, the display can show any combination of the seven segments. However, certain values are used for specific conditions. Table 9-1 shows these values and explains what each means. Sections 9.3.2.1 and 9.3.2.2 list some display sequences.

To illuminate a segment of the display, the appropriate bit in the register is cleared. Section 4.9.2 shows the allocation of bits in the register.

Table 9-1: Values That Can Appear on the Seven-Segment Display

Display Value	Meaning
	Power up and Loading Codes
	Power applied, OBT starting.
One segment lights at a time	OBT in progress. The display cycles through the segments one after another, in a figure-of-eight pattern.

Table 9-1 (cont.)

Display Value	Meaning
1	Waiting to load software. The tests have completed successfully. The DEC MicroServer has broadcast the load request and is waiting for a load host to reply.
2	Loading software. A load host has responded to the load request and the DEC MicroServer's software is being loaded across the Ethernet.
3	Backing off to retry loading. The load request failed because no load host responded. The DEC MicroServer will wait a while and then try to send the load request again. Note that this delay is not a fixed period of time: this prevents all DEC MicroServers on an Ethernet requesting a load simultaneously after a power out. When the DEC MicroServer sends the message, the display reverts to 1. After the OBT has completed during a power up, the value 3 will appear for a while. After a short time the display will change to 1.
4	Software loaded. Once the load is complete, and just before the software starts, this value appears. It should only appear for a short while, before the software's running pattern appears.
Circling pattern (two segments light at the same time)	Software running normally.

Table 9-1 (cont.)

Display Value	Meaning
	Dump Codes
5	Waiting to dump. The DEC MicroServer has broadcast a dump request on the Ethernet and is waiting for a host to respond, accepting the dump.
6	Dump in progress. A host has responded to the dump request, and the DEC MicroServer is dumping the system's contents. When the dump is complete, the display reverts to the OBT cyclic pattern, and the system is reloaded.
7	This value is allocated to a back off state. However, the DEC MicroServer's firmware does not currectly use this value. If a dump fails, the DEC MicroServer abandons the attempt, runs the OBT, and reloads.
	Halt Codes
8	System halted. Appears if the system is halted through the remote console. Note that this value remains while console tests are run.
	Error Codes
C	The NVRAM test in the OBT failed. See Chapter 12 for details of the conditions that can cause this display to occur.

Table 9-1 (cont.)

Display Value	Meaning
d	One of the tests on the synchronous ports failed. This display can appear only if the 50-way loopback connector is in place and the drivers or 50-way socket are faulty. A problem with the DUSCC or other synchronous port components is reported using code F.
E	No connection to the Ethernet. The OBT checks to see if the DEC MicroServer is connected to the Ethernet. This display indicates that the test failed. This may indicate faults in the DEC MicroServer's Ethernet hardware, the connection to the Ethernet, or the Ethernet itself.
F	Basic internal fault. One or more major components in the DEC MicroServer has failed the OBT.
Chapter 12 has details of the OBT and the conditions that cause any of the error codes to appear.	

9.3.2.1 Power Up and Boot Sequence - During a load sequence, the display shows the following sequence of values:

Ripple 1 2

If the DEC MicroServer has to back off, the sequence is slightly different:

1 3 or 3 1

The display will alternate between 1 and 3 until the DEC MicroServer's load request is answered. The display then continues as follows:

1 2

9.3.2.2 Dump and Restart Sequence - During a dump sequence, the display shows the following sequence of values:

5 6

The display then continues with the power up and boot sequence (with or without back off as necessary).

If no dump host responds to the dump request, the DEC MicroServer abandons the dump attempt and reloads the software. So, the display shows '5' and then continues with the power up sequence (with or without retry as necessary).

9.4 External Logic Interface

A 40-way IDC connector on the main circuit board allows test equipment (such as a physical console) to be attached to the board. This socket provides:

- 8 bit unbuffered DAL bus (the lower byte of the MicroVAX bus -- UVDAL<7:0>)
- An address decode line (address line 27 on the MicroVAX bus -- UVDAL<27>) -- see Appendix B for details of how this address line is used.
- Address strobe
- Data strobe
- Read/write signal
- External reset (allowing test equipment to reset the board)
- Two interrupt sources to the VIC
- A sense pin that enables the MicroVAX software and firmware to detect that equipment is connected to the port. See Chapter 13 for details of how this signal is used.
- External break to halt the MicroVAX processor
- 5 V power supply

Table 9-2 shows the signals assigned to each pin.

Table 9-2: Signals on the External Logic Interface

Pin Number	Signal Name	Meaning and Use
	Output Signals	
1, 2	-12 V	
3, 4, 7, 8, 9, 10, 13, 14, 31, 32, 33, 34, 35, 36	Ground	
5, 6	+12 V	
11, 37, 38, 39, 40	+5 V	
16	UVBWR	MicroVAX bus signal indicating that a write cycle is in progress.
18	UVBAS	MicroVAX bus address strobe.
20	UVBDS	MicroVAX bus data strobe.
21	UVDAL<27>	MicroVAX bus line 27, used as address decode for the interface.

Table 9-2 (cont.)

Pin Number	Signal Name	Meaning and Use
	Input pins	
12	CONSOLE	Sense pin used on the logic module to determine if equipment is attached to the interface. This is fed to the system control and status register.
15	EXTIRQ1	Interrupt request pin connected to the MicroVAX interrupt controller.
17	EXTIRQ2	Interrupt request pin connected to the MicroVAX interrupt controller.
19	EXTRESET	Resets the logic module. Also connected to the system control and status register.
22	BREAK	Halts the MicroVAX processor.

Table 9-2 (cont.)

Pin Number	Signal Name	Meaning and Use
Bidirectional signals		
23, 24, 25, 26, 27, 28, 29, 30	UVDAL<0:7>	Bits 0 to 7 of the MicroVAX bus. Lower numbered pins carry the least significant bits. These signals are UNBUFFERED and so any device attached to the interface should present MOS loads.

Chapter 10
System control

10.1 Introduction

The DEC MicroServer uses three basic forms of system control:

- Interrupts
- Machine checks
- Interprocessor communication

In addition, there are hardware and software mechanisms to control how devices on one bus access devices on other buses.

This chapter deals with interrupts, machine checks and cross-bus accessing. The interprocessor communication is part of the buffer management scheme, and this is covered in Chapter 15.

10.2 Use of Interrupts and Machine Checks

Broadly speaking, interrupts are used to report events, and machine checks are used to report errors. However, in one or two instances, an interrupt is used to report an error on the I_O bus to the MicroVAX.

Interrupts report events such as the completion of an I/O transfer. Sections 10.3 to 10.4 give more information on the interrupts and how they are handled.

The type of errors reported through machine checks are nonexistent memory references and memory parity errors. Section 10.5 deals with these errors and their reporting mechanism in more detail.

10.3 Event Reporting

The parts of the system that can generate interrupts to report events are:

- Ethernet Interface
- DUSCCs
- Modem Status Registers
- TMS processor
- DMA Controllers
- External Logic Interface
- Power Supply
- Clock

In addition, the MicroVAX can interrupt the TMS processor by setting the TMSCRL1 and TMSCRL2 bits in the System CSR.

The following sections show the interrupts produced in the system, and how they are processed.

10.3.1 Types of Interrupt

In all, there are 18 interrupts in the system. Table 10-1 shows the name of each interrupt, the part of the system that generates it, and the event it is used to report.

Table 10-1: MicroServer Interrupt Names

Interrupt Name	Event Reported
	Ethernet Interface Interrupts
LNCIRQ	Reports any of the following events: o Initialization complete o Transmission complete o Reception complete o Memory error o Missed packet o Babbling Section 5.5 has more information on what these events mean.
	DUSCC Interrupts
SERINT	Each DUSCC can generate one interrupt, but these are combined at a system level to create this one interrupt. SERINT occurs when either DUSCC asserts its interrupt pin. Fields in the Modem Status Registers indicate which device generated the interrupt. The conditions that cause either DUSCC to cause an interrupt are set in the device's Interrupt Enable Register. See Section 7.2.12 for information on DUSCC interrupt conditions.
	Modem Status Interrupts
MODCHGINT	This interrupt can be generated from the Modem Status Registers each time there is a change in the modem status information.

Table 10-1 (cont.)

Interrupt Name	Event Reported
	TMS Processor Interrupts
TMSINT	The TMS processor uses this interrupt to report major events to the MicroVAX. For example, the TMS processor uses this interrupt to ask the MicroVAX to carry out a dump sequence when a severe error has occurred.
RX0INT, RX1INT, RX2INT, RX3INT	The TMS processor uses these interrupts to tell the MicroVAX when it has completed receiving a packet on any of the synchronous ports. These interrupts are used as part of the buffer management scheme explained in Chapter 15.
TX0INT, TX1INT, TX2INT, TX3INT	The TMS processor uses these interrupts to tell the MicroVAX when it has completed transmitting a packet on any of the synchronous ports. These interrupts are used as part of the buffer management scheme explained in Chapter 15.

Table 10-1 (cont.)

Interrupt Name	Event Reported
	MicroDMA Interrupts
DMAINT	Each MicroDMA device generates one interrupt but, like the DUSCC interrupts, they are combined into this one interrupt at a system level. Fields in the Modem Status Registers indicate which device caused the interrupt. Each device generates an interrupt on successful completion of a DMA operation. They can also generate an interrupt if an error occurs, depending on how the Channel Control Registers are set up. See Section 8.3.3.1 for information on the MicroDMA Channel Control Registers.
	MicroVAX/TMS processor Communication Interrupts
TMSCRL2	The MicroVAX uses this signal to tell the TMS processor to save its context so that the system can be dumped. When the TMS has written the information to TMS RAM, it sets a control bit in the Synchronous I/O Control Block. Once set, the MicroVAX can complete the system dump. To trigger this interrrupt, the MicroVAX must set the TMSCRL2 bit in the System CSR and then clear it. There must be at least 1 ms between setting and clearing the bit.

Table 10-1 (cont.)

Interrupt Name	Event Reported
External Logic Interface Interrupts	
EXTIRQ1 and EXTIRQ2	The exact meaning of interrupts from the external logic interface depends on the equipment attached to the interface, and the action of the MicroVAX software when the interrupt occurs. The manufacturing console uses EXTIRQ1 line as a receive interrupt, and EXTIRQ2 as a transmit interrupt.
Power Supply Interrupts	
PWRFL	This interrupt is driven off the POK signal from the power supply. If the supply drops out of operating range, this interrupt is asserted.
Clock Interrupts	
TIMERINT	This interrupt is generated by the DYRC associated with the System RAM. It provides an interrupt every 10 ms and is used by the MicroVAX software for time keeping operations.

NOTE

There is one further interrupt signal allocated on the board called X21INT. This is not used on the current implementation, and is reserved for future use.

10.3.2 Event Processing

The interrupt processing is carried out by three devices:

- MicroVAX processor
- MicroVAX Vectored Interrupt Controller (VIC)
- TMS processor

The PWRFL and TIMERINT signals are routed to the MicroVAX processor and are always handled by that device.

The SERINT and DMAINT are routed to both the VIC and the TMS processor. This enables either device to field the interrupt, and for either the TMS processor or the MicroVAX to process them. In practice, however, the TMS processor handles both these signals, and the MicroVAX masks them from the VIC.

The TMSCRL1 and TMSCRL2 signals are routed to the TMS processor. In the current design, only TMSCRL2 is used.

The remaining interrupts are all routed to the VIC and are handled by the MicroVAX. Section 10.4 summarizes the function of the VIC and shows how it is used in the DEC MicroServer.

10.4 The MicroVAX Vectored Interrupt Controller (VIC)

The MicroVAX processor only has four device interrupt inputs (IRQ0 to IRQ3), but the DEC MicroServer has many more interrupt sources that need attention. The VIC accepts interrupts through its 16 PIRQ pins, and activates the appropriate IRQ line to the MicroVAX.

When the MicroVAX processor executes interrupt acknowledge cycles, the VIC provides the necessary interrupt vectors at the appropriate points. Software in the MicroVAX sets up the conditions that the VIC operates under, including the value of the interrupt vectors.

The following sections summarize the operation of the VIC, concentrating on the features used in the DEC MicroServer. If you want more detailed information on the device, refer to the literature listed in Appendix I.

10.4.1 Interrupt Properties

Each interrupt that drives the VIC has three properties:

1. The input pin it is connected to
2. The VAX Interrupt Priority Level (IPL) -- a number between 14 and 17 (decimal)
3. The signal state that defines the interrupt condition -- rising edge, falling edge, low level, or high level

The input pin is fixed, of course, but the other properties can be programmed using the VIC's registers. Generally, the properties are permanently assigned for a complete system and the DEC MicroServer is no exception. Table 10-2 lists the system interrupts that the VIC can process, and shows the properties they have.

Table 10-2: DEC MicroServer Interrupt Properties

Interrupt Name	Input Pin	IPL	Signal Condition
DMAINT	PIRQ2	16	Low level
EXTIRQ1	PIRQ15	15	Falling edge
EXTIRQ2	PIRQ0	15	Falling edge
LNCIRQ	PIRQ14	17	Low level
MODCHGINT	PIRQ1	16	Rising edge
RX0INT	PIRQ13	16	Rising edge
RX1INT	PIRQ12	16	Rising edge
RX2INT	PIRQ11	16	Rising edge

Table 10-2 (cont.)

Interrupt Name	Input Pin	IPL	Signal Condition
RX3INT	PIRQ10	16	Rising edge
SERINT	PIRQ3	16	Low level
TX0INT	PIRQ9	16	Rising edge
TX1INT	PIRQ8	16	Rising edge
TX2INT	PIRQ7	16	Rising edge
TX3INT	PIRQ6	16	Rising edge
TMSINT	PIRQ5	16	Falling edge
X21INT	PIRQ4	16	Low level

10.4.2 Interrupt Priorities

Each PIRQ pin has a fixed priority in relation to all the others; PIRQ15 has the highest priority, and PIRQ0 the lowest. Simply using this as a way of processing interrupts can mean that devices attached to high priority pins can monopolize interrupt cycles. The VIC provides two priority mechanisms that give more sophisticated control:

1. Fixed Priority
2. Round Robin

In the **fixed priority** scheme, the VIC takes into account the IPL level that the MicroVAX is acknowledging. The PIRQ serviced is the highest priority one with the appropriate IPL associated with it. For example, if the RX0INT and RX1INT were both outstanding,

RX0INT would be serviced on the next acknowledge cycle for IPL 16. This is because it is attached to the higher priority PIRQ pin.

In the **round robin** scheme, the IPL level takes precedence over the PIRQ priority. When the MicroVAX acknowledges interrupts on a particular IPL, the VIC selects the highest priority PIRQ on that level. Once that interrupt has been processed, the VIC prevents it from causing another until all other outstanding interrupts at that IPL have been serviced. This blocking happens for each successive interrupt at that level. Only when there are no more outstanding interrupts at the IPL does the VIC free up the blocked PIRQ lines (allowing them to generate further interrupts).

10.4.3 Registers and Register Types

The MicroVAX software determines the IPLs, signal conditions, interrupt vectors, and priority mechanism by setting values in the VIC's registers. The registers also enable the MicroVAX to mask out those interrupts it would not normally handle, and to find out what interrupts still need attention.

The VIC has a total of 9 control registers, and 16 vector registers that can be grouped like this:

- Signal condition
- IPL
- Interrupt priority scheme
- Interrupt vectors
- Masking
- Interrupt pending status

The following sections look at each type of register in more detail.

10.4.4 Signal Condition

There are two registers that the MicroVAX software can use to determine the signal conditions for each of the 16 interrupts:

- Level/Edge (LE) register
- POLARITY register

Both are 16-bit registers with one bit allocated to each interrupt. The bit number in each register corresponds to the PIRQ pin number of the appropriate interrupt. For example, bit 0 in both registers refers to the signal connected to PIRQ0. The combination of values in the two registers determine the signal condition for each interrupt, as Table 10-3 shows.

Table 10-3: Interrupt Signal Conditions

Value in LE	Value in POL	Signal Condition
0	0	Falling edge
0	1	Rising edge
1	0	Low level
1	1	High level

For the interrupt conditions shown in Table 10-2 the values of these registers (in hexadecimal) are:

- LE -- 401C
- POLARITY -- 3FC2

10.4.5 IPL

Each of the MicroVAX processor's interrupt pins is associated with one IPL. The VIC has corresponding output pins, one of which is asserted when an interrupt occurs on one of the PIRQ pins.

The IPL for each interrupt is set up through the IMAP registers. There are four of these registers -- one for each output pin -- and each is 16 bits long. The bits in each register are allocated in the same way as the LE and POLARITY registers (one for each PIRQ input).

Each IMAP register relates to a different IPL:

IMAP0 -- IPL14

IMAP1 -- IPL15

IMAP2 -- IPL16

IMAP3 -- IPL17

To put an interrupt at any IPL, the appropriate bit in the corresponding IMAP register is set. For example, to put RX0INT at IPL 16, bit 13 in IMAP 2 is set.

For the IPLs shown in Table 10-2, the hexadecimal values of these registers are:

IMAP0 -- 0000

IMAP1 -- 8001

IMAP2 -- 3FFE

IMAP3 -- 4000

10.4.6 Priority Scheme

The MicroVAX software can set up different priority schemes for each IPL. The schemes used for each level is held in the ROBIN register, as Figure 10-1 shows.

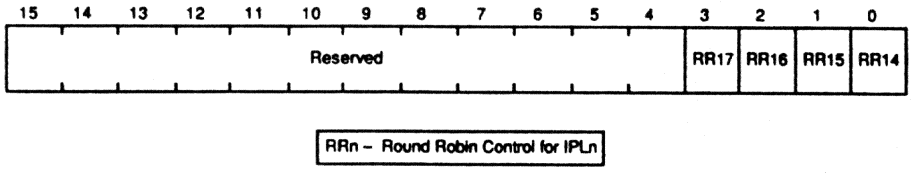


Figure 10-1: Fields in the VIC's ROBIN Register

To use the round robin scheme for any IPL, set the appropriate RR bit to 1. To use the fixed priority scheme, clear the bit.

10.4.7 Interrupt Vectors

The VIC contains 16 interrupt vector registers (IVEC0 to IVEC15). Each of these registers contains the interrupt vector associated with the appropriate PIRQ. The VIC supplies the appropriate one of these vectors to the MicroVAX when the interrupt acknowledge cycle for a particular IPL takes place.

For example, if the MicroVAX acknowledges interrupts at IPL16, the VIC supplies it with the vector for the highest priority interrupt yet to be serviced at that IPL. The scheme used to determine the highest priority interrupt is defined in the ROBIN register.

Figure 10-2 shows the format of a register as used in the DEC MicroServer. Notice that the two least significant bits are zero.

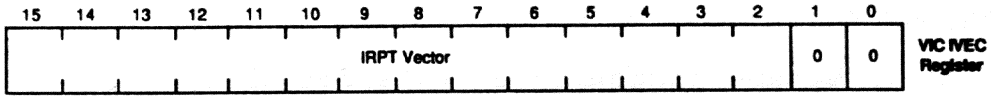


Figure 10-2: Format of the VIC's Interrupt Vector Registers

10.4.8 Masking

The MicroVAX does not want to handle all of the interrupts all of the time. In particular, the DMAINT and SERINT signals are normally handled by the TMS processor. In these cases, the MicroVAX needs to prevent the VIC from responding to those interrupts when the TMS processor is running.

The MicroVAX can enable and disable the recognition of the 16 interrupts through the VIC's Interrupt Enable Register (IEN). This has one bit allocated to each PIRQ input. The bit number in the register corresponds to the PIRQ input number.

To enable interrupts on a particular PIRQ, the MicroVAX sets the corresponding bit in the IEN. To mask an interrupt, the appropriate bit is cleared.

So, in normal operation of the DEC MicroServer, the hexadecimal value of this register would be FFF3.

10.4.9 Pending Interrupts

The VIC includes one register (the Pending Summary Register, or PSR) that provides a summary of the interrupts that are waiting to be serviced. The MicroVAX software can also use this register to clear individual interrupts.

The register has 16 bits, one for each of the PIRQ pins. When one of these bits is set, the device attached to the corresponding PIRQ pin has requested an interrupt. A bit becomes set whenever the signal condition set in the LE and POLARITY registers occurs.

The conditions that clear a bit in the PSR depend on the signal condition set up in the LE and POLARITY registers. For each level-sensitive interrupt, the bit will be cleared when the appropriate PIRQ pin is de-asserted. For each edge-sensitive interrupt, the bit in the PSR is cleared when one of the following occurs:

- An interrupt acknowledge cycle for that PIRQ
- A clear operation on that bit in the PSR
- Writing the LE register

10.4.10 Effects of Reset

When the VIC is reset, the POLARITY, LE, ROBIN, IEN and PSR registers are all set to zero. None of the IMAP and IVEC registers is reset and so these need to be set to appropriate values before normal system operation is resumed. This is also the case at power up.

10.4.11 Register Addressing

Although the VIC registers are all 16 bits long, they each appear on a longword boundary. The registers appear in a block that starts at location 26000000 (hex) in the MicroVAX processor's address space. Figure 10-3 shows the offset of each register in this block.

Register	Hexadecimal offset
POLARITY	0
LE	4
PSR	8
IEN	C
IMAP0	10
IMAP1	14
IMAP2	18
IMAP3	1C
ROBIN	20
Addresses <40:24> are not used and are not decoded by the VIC	
IVEC0	44
IVEC1	48
IVEC2	4C
IVEC3	50
IVEC4	54
IVEC5	58
IVEC6	5C
IVEC7	60
IVEC8	64
IVEC9	68
IVEC10	6C
IVEC11	70
IVEC12	74
IVEC13	78
IVEC14	7C
IVEC15	80

Figure 10-3: Offsets in the VIC Register Block

The MicroVAX can read from or write to any register providing it uses word access instructions only. Byte accesses are not supported and longword accesses could result in incorrect data being transferred between the two devices.

10.5 Error Handling

The types of errors the system can trap are:

- Nonexistent memory or device location
- Parity error
- Memory partition error
- Watchdog expiry
- Dump request through the switch on the rear panel

The following sections show how the system handles each type of error.

10.5.1 Nonexistent Memory

The MicroVAX address map is incomplete, and many locations do not exist. If the processor tries to access any of these locations on the MicroVAX or I_O buses, the read or write operation will eventually time out.

Bus timeout is monitored by the DYRC devices on both buses, but the action taken when a timeout is different for each bus.

10.5.1.1 MicroVAX Bus Accesses - On the MicroVAX bus, the DYRC asserts the UVERR signal that is connected to the MicroVAX processor's ERR pin. This, in turn, causes the machine check and the MicroVAX firmware takes appropriate action.

The DYRC contains the reason for the machine check (nonexistent memory) and the address that caused the error in its status registers.

10.5.1.2 I_O Bus Accesses - The I_O bus can operate asynchronously of the MicroVAX and so a similar scheme cannot be used. Instead, a nonexistent memory access asserts the PWRFL interrupt to the MicroVAX. If one of the I_O bus devices generated the condition, the IOERRSRC bit in the system CSR is set. If the MicroVAX generated the erroneous memory access, the IOERRSRC bit in the system CSR is cleared. In both cases, the PWRFL bit is cleared.

The IOERRSRC and PWRFL bits enable the error handling routine to determine whether the assertion of the PWRFL signal was a real power failure or a memory error. They also enable the routine to determine who was in control of the I_O bus when the error occurred: the MicroVAX or an I_O bus device.

As with the MicroVAX bus, the DYRC on the I_O bus contains the detailed error information including the address that caused the error.

10.5.1.3 SCC Bus Accesses - The MicroVAX accesses memory and devices on the SCC bus through the MicroDMA controllers. If such an access refers to a nonexistent location, the DMA operation ends in an error and this is reported back to the MicroVAX.

10.5.2 Parity and Partition Errors

Parity and partition errors on the MicroVAX bus are handled in a similar way. When the error occurs, the UVERROR signal is asserted which causes the MicroVAX processor to execute a machine check.

At the same time, the PRTNERR bit in the system CSR is set if the error is a partition fault. By using this bit, and the information in the appropriate DYRC registers, the MicroVAX can determine the type of error and the address that caused it.

10.5.3 Watchdog Expiry

If the watchdog timer expires, the following occurs:

1. The WDERR bit in the system CSR is set to indicate the error.

2. The MicroVAX's HALT pin is asserted, causing the processor to carry out a restart operation. During the restart operation, the system will be dumped and reloaded.

10.5.4 Dump Switch

When the system's dump switch is pressed, the following occurs:

1. The DUMP switch in the system CSR is set to indicate the cause of the dump.

2. The MicroVAX's HALT pin is asserted causing the processor to restart in the same way as it does when the watchdog timer expires.

10.5.5 Use of NVRAM in Error Processing

The DEC MicroServer keeps a log of parity and memory accessing errors in the NVRAM. This can help a manufacturing plant diagnose a system that has been returned for repair. See Chapter 16 for more details of the information kept in the NVRAM.

10.6 Cross-Bus Accessing

There are some restrictions on what devices are able to access on other buses, and on their own bus. These restrictions specifically concern:

1. MicroVAX access to the TMS processor memory
2. TMS processor access to the I_O bus
3. LANCE access to the I_O bus

Under the correct conditions, the MicroVAX can access any part of the system, but the TMS processor and the LANCE have more restricted capabilities. The TMS processor can, of course, access anything in the Synchronous I/O part of the system, but can access the I_O bus only on the MicroVAX side of the DMA devices.

The LANCE is restricted to using the I_O bus, and does not have access to any other bus.

The following sections explain the types of bus access available to the MicroVAX, the TMS processor, and the LANCE.

10.6.1 MicroVAX Access to the SCC Bus

The MicroVAX has free access to its own and the I_O buses. However, there are restrictions on the way it can access the SCC bus.

These restrictions apply to the TMS RAM and the TMS processor's I/O space. When the TMS processor is running, the MicroVAX software should not try to access these parts of the system. Doing so can cause a deadlock that can be resolved only by powering the system off and then back on again.

To access TMS RAM or the TMS processor's I/O space, the MicroVAX uses the RSTDSP bit in the system CSR. Setting this bit puts the TMS processor into a reset state, allowing the MicroVAX to access the memory through a DMA window. When the MicroVAX has finished, it clears the RSTDSP bit allowing the TMS processor to continue.

10.6.2 TMS Processor Access to the I_O Bus

Through the DMA controllers, the TMS processor has access to the I_O bus only. Here, it has access only to the Buffer RAM. Any attempt to access other parts of the system result in "nonexistent memory" errors.

This access allows the TMS processor to transfer data between the RAM and the synchronous ports, as well as receive commands from the MicroVAX. However, it prevents the TMS processor from disturbing the operation of the rest of the system.

10.6.3 LANCE Access to the I_O Bus

The LANCE has very restricted access to the system. First, it can access only its own bus. Secondly, that access is restricted to the Buffer RAM.

These restrictions allow the LANCE to transfer data between the RAM and the Ethernet interface, but prevent it from disturbing other parts of the system.

PART III

FIRMWARE

Chapter 11
Overview of the Firmware

11.1 Types of Firmware

The DEC MicroServer has three pieces of firmware:

- On-board tests
- VAX firmware
- Synchronous control code

This part of the book shows how each of these pieces operates, explaining the major functions of each. The rest of this chapter gives an introduction to each piece of code. Turn to the referenced chapters for more information.

11.2 On-Board Tests

The on-board tests have three parts. The most commonly used is the power-up test which runs each time the system boots. It checks that major parts of the system's hardware are working. This test helps to ensure that a system can be successfully loaded and run.

The second part is a set of manufacturing tests that enable engineers to carry out concentrated testing on one particular area of the hardware.

The third part is a set of tests that are used from the DEC MicroServer's remote console. These are loopback tests that enable the control and data paths in a communications link to be tested.

Chapter 12 contains details of what the tests do and their use of the seven-segment display.

11.3 VAX Firmware

After the power-up test has run, the VAX firmware takes control and loads the operational software from a load host. The VAX firmware also provides support functions for the system such as dumping a "bad" system, or writing information into NVRAM. The VAX firmware also handles the remote console.

Chapter 13 contains details of what the VAX firmware does.

11.4 Synchronous Control Code

The synchronous control code runs in the TMS processor and controls the synchronous I/O ports. In doing this it looks after the DUSCCs, the SSL, the DMA controller, and the modem control/status registers. This means that the MicroVAX processor does not have to manage these devices. This control code, and the operational software, communicate through the Synchronous I/O Control Block.

Chapter 14 contains details of what the synchronous control code does.

Chapter 12
On-Board Test

12.1 Overview

The on-board tests are part of the code in the Firmware ROM. The code contains three sorts of test:

- Power-up tests
- Manufacturing tests
- Console driven tests

The **basic tests** run each time the system starts up, before the software is loaded. These tests check that the hardware works and can run operational software. The **manufacturing tests** need to be run from a console, and can be used only when the manufacturing console is attached to the main logic module. These tests help manufacturing engineers to test exhaustively a faulty area of hardware. The **console driven tests** are a set of loopback tests that allow a user or field service engineer to test one or more of the synchronous communications links. These tests are run from the DEC MicroServer's remote console. This chapter explains these tests in more detail, and shows how they are run.

12.2 Power-Up Tests

The power-up tests are automatically run each time the system boots to give a confidence check on the hardware. That is, the tests check that the hardware can successfully load and run a communications system.

Any problems that the tests uncover are logged in the NVRAM to help future repair diagnosis. In most cases, a test failure means that the unit is unusable, and in these cases a fault code appears on the seven-segment display.

The basic tests take around 2 minutes to complete and run in 4 phases. Each phase checks out a different part of the hardware:

- Low level
- Ethernet Interface
- Synchronous I/O
- Exerciser

The phases test increasing amounts of the DEC MicroServer, each phase depending on successful completion of its predecessor. The sequence stops at the first error found.

Sections 12.2.1 to 12.2.4 explain each phase in more detail. These explanations mention some of the test patterns used to test memory locations and device registers. Section 12.2.5 contains a summary of these test patterns.

The final three subsections deal with test startup, error logging, and use of the seven-segment display.

12.2.1 Low Level Phase

The low level phase consists of 12 separate tests. These check out the computational parts of the system that are not directly involved in any I/O operation. For example, one test checks that the System RAM works properly, while another checks the MicroVAX processor.

The following tests run in this phase:

1. Firmware ROM CRC
2. System RAM
3. MicroVAX processor
4. VIC
5. NVRAM
6. Ethernet PROM
7. Control and status registers
8. Interval timer
9. Watchdog timer

10. Buffer RAM

11. MicroVAX memory management registers

12. MicroVAX memory management functions

12.2.1.1 Firmware ROM CRC - The last word in the Firmware ROM is a 16-bit Cyclic Redundancy Check (CRC-16) value. This value is calculated using the other bytes in the ROM.

The first test recalculates this CRC value and checks it against the stored value. This checks that the ROM contains the right information, and also demonstrates that the following operate correctly:

1. The bus between the MicroVAX processor and the ROM

2. The ROM's decode circuitry

12.2.1.2 System RAM - This test uses the Nair, Thatte, and Abrahams algorithm for checking semiconductor RAMs. This checks the System RAM for:

- Stuck-at and address coupling faults in the RAM devices
- Correct connection of the devices to the MicroVAX data bus (using a march pattern)
- Correct parity error detection (by forcing incorrect parity)
- Correct trapping of accesses to nonexistent memory

Successful completion of this test demonstrates that the following also work:

1. The bus between the MicroVAX processor and the RAM block

2. The RAM's refresh circuitry

This test is one of the longest in the phase (taking around 15 seconds to run).

12.2.1.3 MicroVAX Processor - This test checks the internal registers and the critical timing paths of the MicroVAX

processor. In doing this, the test also checks much of the processor's instruction set.

12.2.1.4 VIC - This test checks the following areas of the VIC:

1. Checks that the interrupt mechanism works as expected
2. Tests the control and status registers by writing and reading walking 1 and walking 0 patterns
3. Checks for unsolicited interrupts from other parts of the board (which would indicate faults in those areas -- for example, VIC inputs stuck at an active level due to a short circuit elsewhere in the system)
4. Checks the VIC and MicroVAX interaction by making sure that interrupts occur in the MicroVAX at the correct IPL (this test does not use external devices)

12.2.1.5 NVRAM - This test operates on just one location in the NVRAM. It checks that two complementary bit patterns can be written to and read from the NVRAM.

Only one location is used because:

1. Other locations contain permananent data (such as the console password) that would otherwise be destroyed
2. The NVRAM has a limited life (10 000 write operations)

Successful completion of this test also demonstrates that the following work:

1. The bus between the MicroVAX processor and the NVRAM
2. The NVRAM's address decode circuitry

Notice that this test does not check the address uniqueness of each NVRAM location.

12.2.1.6 Ethernet Address PROM - This test checks that the Ethernet Address PROM is correctly formatted by:

1. Comparing the first copy of the address with one of the duplicate copies

2. Calculating the CRC checksum from the address and comparing this with the checksum stored in the PROM

Successful completion of this test demostrates that the following are working properly:

1. The bus between the MicroVAX processor and the Ethernet PROM

2. The PROM's address decode circuitry

12.2.1.7 Control and Status Register - This test checks the system control/status register and the memory partition mechanism.

The read-only bits of the register are checked for coupling faults. All the other bits are checked to make sure they're not stuck at a particular value by writing values to each bit and reading them again.

To check the partition mechanism, the test sets the PRTEN bit in the register and then tries to write to the protected half of the System RAM. This causes the PRTNERR bit in the register to be set and machine check to occur through the MicroVAX's ERR pin. To make sure the mechanism is working correctly, the test also tries to write to a location in the unprotected part of the System RAM. This, of course, should not generate a machine check.

Successful completion of this test also shows that the following work correctly:

1. The bus between the MicroVAX processor and the System CSR

2. The register's address decoding for read and write operations

12.2.1.8 Interval Timer - This checks that the 10 ms clock generated by the System RAM's DYRC is working properly. This clock generates the TIMERINT interrupt, and the test uses this to determine that:

1. The interrupt occurs
2. The time interval is correct
3. The interrupt occurs at the correct IPL

12.2.1.9 Watchdog Timer - This test is run only if the disabling link (W8) is not fitted. It tests that the watchdog timer functions correctly by:

1. Resetting the timer
2. Waiting for the restart to occur, and checking that the time interval is one of those supported.

12.2.1.10 Buffer RAM - The Buffer RAM is checked in the same way as the System RAM. However, this test takes longer to complete because of the cross bus access (from the MicroVAX bus to the I_O bus) for each location read/write.

12.2.1.11 MicroVAX Memory Management Registers - This is the first of two tests that checks memory management features of the MicroVAX processor. This test checks the registers used in memory management. The test writes values to the registers and reads those values back. This makes sure that the registers are working correctly.

12.2.1.12 MicroVAX Memory Management Functions - The second memory management test checks out the functions. This reads and writes to both physical and virtual addresses. The test also checks the access control mechanism by trying to write to read-only pages.

12.2.2 Ethernet Interface Phase

The Ethernet interface phase consists of three tests that check out the LANCE and SIA parts of the DEC MicroServer. This phase has the following tests:

1. LANCE Control and Status registers
2. LANCE internal loopback
3. Ethernet loopback

12.2.2.1 LANCE Control and Status Registers - The first test checks that the control and status registers in the LANCE are working properly. This involves writing values in the registers, reading the register values, and checking that these are the same as those that were written. The values written to the registers are walking 0 and walking 1 patterns, that can detect bits stuck at a particular value, and coupling faults between registers.

Successful completion of this test also shows that the following are working correctly:

1. The bus between the MicroVAX processor and the LANCE (including the data paths inside the LANCE)
2. The address decode logic for the LANCE

12.2.2.2 LANCE Internal Loopback - This test checks as many of the LANCE's functions as possible using an internal loopback. This helps to ensure that the Ethernet is not adversely affected by messages from a faulty LANCE.

The following functions are tested:

1. Basic transmission and reception
2. Detection of missed packets
3. Buffer overflow and other receive buffer errors
4. Detection of memory errors
5. Generation and detection of CRC values
6. Detection of collisions
7. Rejection of physical addresses at the appropriate time
8. Filtering of logical addresses

9. Detection of a broadcast address

10. Promiscuous mode

Successful completion of this test shows the following:

1. The LANCE can perform DMA activities to and from the Buffer RAM.

2. There is correct arbitration of the I_O bus between the MicroVAX processor and the LANCE.

12.2.2.3 Ethernet Loopback - This test loops information through the SIA and out to the Ethernet connector. To do this the DEC MicroServer sends out a packet addressed to itself (a so-called "runt" packet).

This packet could, of course, collide with traffic on the Ethernet, so the test will try up to 32 times to send the packet and successfully receive it.

The test ignores any heartbeat signal that the Ethernet transceiver may produce. This means that:

1. The test can be run with a passive Ethernet loopback connector in place of a real Ethernet. This can be useful in manufacturing, installation testing, and fault finding in the field.

2. The DEC MicroServer can be connected to an Ethernet whose transceiver equipment does not provide any heartbeat signal.

NOTE

This test will fail if the Ethernet port is not connected to a network or an Ethernet loopback connector.

12.2.3 Synchronous I/O Phase

The synchronous I/O phase checks out the synchronous I/O parts of the DEC MicroServer. There are 8 tests in this phase:

1. MicroDMA controller

2. TMS processor RAM
3. DUSCC
4. Synchronous internal data loopback
5. Receive FIFOs
6. Modem status loopback signals
7. Synchronous external data loopback
8. TMS processor

12.2.3.1 MicroDMA Controller - The first test checks that the control and status registers in each MicroDMA controller are working properly. This involves writing values in the registers, reading the register values, and checking that these are the same as those that were written. The values written to the registers are walking 0 and walking 1 patterns, that can detect bits stuck at a particular value and coupling faults between registers.

At this stage, the DMA capabilities of the devices cannot be checked, until other parts of the synchronous I/O part of the system are known to work. The DMA capability of both devices is checked as part of the internal data loopback test. Until then, all tests use the DMA controllers in I_O Access mode which gives access to the synchronous ports and the TMS processor without using the DMA circuitry.

12.2.3.2 TMS Processor RAM - The TMS processor RAM is checked through DMA controller A from the MicroVAX processor. The test uses walking 1, walking 0, and incremental patterns to check for bits stuck at 1 or 0 and for coupling faults.

12.2.3.3 DUSCC - Here, one of the DMA controllers (DMA A) is used in I_O Access mode to test both DUSCCs. Both DUSCC devices are checked in the same way:

1. The control and status registers are checked using walking 1 and walking 0 patterns.
2. The device is set into internal loopback mode.

3. Single packets are transferred at a rate of 64K bits/s on each channel in turn using bit-oriented protocols.

Successful completion of this test also shows that the following are working properly:

1. The data path between the DUSCC devices and the MicroDMA controllers

2. The DUSCC timing clock

12.2.3.4 Synchronous Internal Data Loopback - Here, the DUSCCs are still run in internal loopback mode. However, this time the DMA capabilities of the MicroDMA controllers are used. The test:

1. Uses DMA assistance for all channels except channel 3 receive

2. Uses bit- and character-oriented protocols

3. Runs at 2M bits/s on ports 0, 1, and 2, and at 250K bits/s on port 3

Parts of this test are similar to the DUSCC test. Therefore, failure of this test more probably indicates a MicroDMA problem.

12.2.3.5 Receive FIFO - This test moves the loopback point from the DUSCCs to the SSL so that the FIFOs on each receive line can be checked. Information for each port is sent at that port's maximum speed.

During this test, the transmit data path to the distribution panel for each port is disabled. This prevents the test interfering with any equipment attached to the ports.

Successful completion of this test also shows that the FIFO enable switches in the SSL work properly.

12.2.3.6 Modem Status Loopback - This test is run only if any of the ports have 50-way or manufacturing loopback connectors attached. The OBT looks to see which ports have loopback connectors attached to them.

If a manufacturing connector is attached to one port, all the other ports must also have connectors added. These can be 50-way or manufacturing connectors.

For 50-way connectors, the test operates only on those ports with the connectors attached.

The test sends data on the modem control lines and checks that it returns on the appropriate status lines. The test also checks the active cable code circuitry. Successful completion of this test also shows the following:

1. The +12 V and -12 V supplies to the distribution panel are present (but the test cannot check their tolerance)

2. The loopback connector continuity

12.2.3.7 Synchronous External Data Loopback - Like the modem status test, this one operates only on ports with a 50-way or manufacturing loopback connector attached.

This test checks the data lines on each port using every line interface standard. The test uses both NRZ and NRZI data, and also runs each line interface at its maximum speed.

12.2.3.8 TMS Processor - This is the only test (apart from the Exerciser Phase) that uses code in the TMS processor (all the others have been run from the MicroVAX). This code is loaded from the Firmware ROM into the TMS processor's RAM.

This code tests:

1. The TMS processor's internal RAM using walking 1, walking 0, and incremental patterns

2. The basic ability of the TMS to execute instructions

12.2.4 Exerciser Phase

All the tests done so far have used bits of the system in isolation. The aim of this phase is to use all the parts at the same time, creating interaction between the components and the data paths. This helps to check out the various bits of arbitration logic throughout the system and to trap errors caused by system loading.

The test phase runs like this:

1. Code is loaded into the TMS processor from the Firmware ROM. This code is similar to that used in an operational system.

2. The TMS processor sets up the synchronous ports to loopback data at 64K bits/s. The loopback occurs at the I/O connector (if a loopback connector is attached) or at the SSL (if no loopback connector is attached to the appropriate channel).

   Different protocols and buffer sizes are used on each port.

3. The LANCE is set up to loop packets internally which involves DMA transfers to and from the Buffer RAM.

4. The MicroVAX transfers information from the Buffer RAM to the System RAM to increase system loading.

5. The TMS processor sets up ports 0 and 1 to run at 256K bits/s. The LANCE continues to loop packets internally and the MicroVAX continues transferring packets to and from System RAM.

Steps 2, 3 and 4 operate simultaneously.

During this testing, the system watches for unexpected exceptions, hardware errors, and data corruption.

12.2.5 Test Patterns

The tests on DRAM memories are based on the Nair, Thatte, and Abrahams algorithm for checking semiconductor memories. For more detailed information on this algorithm, refer to the article listed in Appendix I.

The tests on the static RAM and on the system and device registers use three main types of test pattern:

1. Walking 1

2. Walking 0

3. Incremental

These tests detect bits that are stuck at a particular value and device coupling faults. The following sections summarize how

these test patterns are used.

12.2.5.1 Walking 1 and Walking 0 - A walking 1 pattern moves a single 1 through each bit of a location or register in turn. Initially, all bits are clear except for one (say the least significant, or left-most bit). This value is read, the 1 shifted to the next bit, and written back. This continues until the 1 has "walked through" all the bits.

A walking 0 pattern is similar but uses a single 0 instead of a single 1.

When testing registers in a device using this method, extra read operations are done. Each time the register is read back, the OBT also reads all the other registers in the device to make sure that they have not changed.

12.2.5.2 Incremental - This test simply starts with the location at 0 and then increments the value until all bits have the value 1. Again, at each change of value, the location is read back to make sure it has the correct value.

12.2.6 Start Up Processing

The basic tests run each time the DEC MicroServer is booted. This can occur for a number of reasons and, before starting, the on-board tests must determine the reason.

If the reboot is due to the mains power being applied, the tests can carry on. For all other reasons, a running system has been interrupted, and may need to be dumped before the tests can run.

Doing this check before starting the on-board tests makes sure that no useful information in the system is lost. This information can be useful in determining the reason for a system crash (for example).

If a dump is necessary, the OBT calls the appropriate routine in the MicroVAX firmware. When the dump is complete, the tests can run.

One other activity that the OBT does at start up is to check the status of the DUMP switch. If you hold the DUMP switch in on power up, you erase the remote console password. So, the OBT must check this switch on power-up. If the DUMP switch is held

in, the OBT indicates this to the firmware when it hands over control. The firmware then executes a routine to clear the password.

12.2.7 Error Reporting

If one of the tests fails, the OBT:

- Displays an error code on the seven-segment display
- Logs the error in NVRAM

Section 12.2.8 explains how the OBT uses the display, and the error codes.

The information in NVRAM is in two parts:

1. Error summary
2. Last error details

Figure E-3 contains an example NVRAM error log display generated by the remote console SHOW ERROR command.

12.2.7.1 Previous Error Summary - The error summary is a table with up to ten entries. Each entry contains information on a particular type of error:

1. **Error code** -- each error is identified by a hex number
2. **Number of times it has occurred**
3. **The boot number of its last occurrence** -- the DEC MicroServer keeps track of how many times it is booted

Including the boot number helps to determine if the error was a recent one, or a one-off occurrence that happened in the past.

Note that the OBT logs only a total of 10 kinds of error that occur during the unit's life. So, for example, if there were more than 10 kinds of error occurred on a unit, the extra errors would not be logged. The only exception would be if the later error was the last error on that unit: the last error is always recorded.

12.2.7.2 Last Error Details - The second part of the error information is more detailed information on the last error that the OBT detected. This information is a copy of the MicroVAX processor registers R0 to R11, as they were when the error occurred.

The contents of most of these registers are particular to each test, but two contain more general information:

R1 contains the error code (as recorded in the error log)

R11 contains the OBT's status longword that gives more information on what was going on when the error occurred

NOTE

The last error details always contains information on the last error to occur. This is irrespective of whether that error code appears in the previous error summary.

12.2.7.3 OBT Status Longword (R11) - The OBT maintains status information in the MicroVAX register R11. Figure 12-1 shows the format of this word, and Table 12-1 explains the meaning of each field.

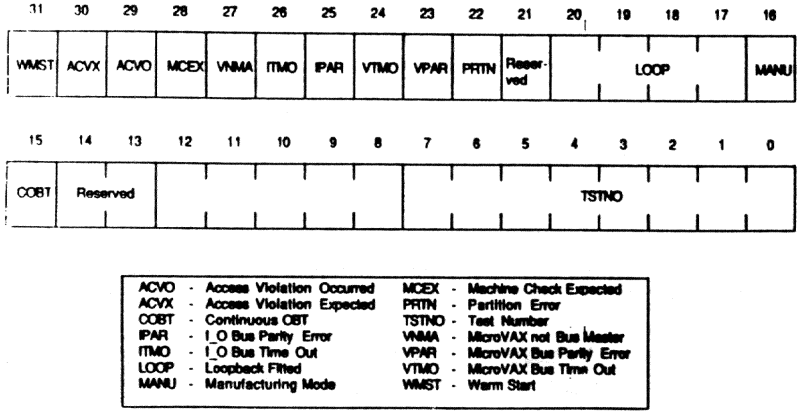


Figure 12-1: The OBT Status Longword (R11)

Table 12-1: Content of the OBT Status Longword

Field Name	Function and Meaning
TSTNO	The number of the current test.
COBT	The unit is running the test continuously. This field is set by using SPECIAL TEST 19 or the TEST CONTINUOUS console command.
MANU	The unit is working in manufacturing mode. This is determined by manufacturing loopback connectors being attached to one or more ports.
LOOP	Indicates which ports have loopback connectors attached to them. One bit in this field is used for each port, and if that bit is set, the port has a connector attached. The least significant bit shows the status for Port 0, and the most significant the status for Port 3.
PRTN	Indicates that a machine check was caused by a partition error.
VPAR	Indicates that a machine check was caused by a parity error on the MicroVAX bus.
VTMO	Indicates that a machine check was caused by a bus timeout on the MicroVAX bus.
IPAR	Indicates that a machine check was caused by a parity error on the I_O bus.
ITMO	Indicates that a machine check was caused by a timeout on the I_O bus.

Table 12-1 (cont.)

Field Name	Function and Meaning
VNMA	Indicates whether the MicroVAX processor was bus master when either a parity error or timeout occurred on the I_O bus. When this field is clear, the MicroVAX was the bus master. The LANCE, TMS processor, or one of the MicroDMA devices was master if this bit is set.
MCEX	Some of the tests force machine checks to occur. The OBT uses this field to differentiate between these and unexpected checks. If this bit is set, the OBT was expecting a machine check to take place.
ACVO	This field is used only in the memory management test. If set it indicates that an access violation occurred during these tests.
ACVX	Some of the memory management tests force access violations to occur. In these cases, the OBT sets this bit to show that a violation is to be expected. This field is used with ACVO.
WMST	This bit indicates whether the OBT was run because of power-up or because of a system restart (such as through the remote console). When this bit is clear, the OBT is running because of power-up.

12.2.8 Use of the Seven-Segment Display

The OBT uses the unit's seven-segment display to show that it's running, and to report any failures.

Table 12-2 shows the values that the OBT uses, and indicates which tests can use each value.

Table 12-2: The OBT's Use of the Seven-Segment Display

Display Value	Meaning
. (decimal point)	The unit is receiving electrical power, and the OBT is about to start. If this display remains for more than 10 seconds there is a serious problem with the main logic module.
One segment lights at a time	While the OBT is running, it lights the segments of the display one after the other. They light in a figure-of-eight pattern.
F	One of the tests in the low level phase failed. The reason for failure should be logged in the NVRAM, and it may be possible to read this using the manufacturing console. (Remote console will not work.) If this proves impossible, move the NVRAM to another board that is known to work, and read it from there.
E	The Ethernet Loopback test failed. Again, the NVRAM will contain more detailed information that can be read only through the manufacturing console. This test will fail if nothing is connected to the Ethernet port. Before doing further investigation, make sure that there is an Ethernet or an Ethernet loopback connector fitted to the port.

Table 12-2 (cont.)

Display Value	Meaning
d	One of the tests in either the Synchronous I/O or Exerciser phases has failed. In this case, the error information in NVRAM can be accessed through the manufacturing console or through the remote console.
C	The OBT could not write error information into the NV RAM. This occurs when a test fails, but the OBT can't record the failure in the NVRAM. All information in the NVRAM has a checksum that helps to check that it was written correctly. To do this check, the OBT reads the written information, recalculating the checksum. If this and the stored checksum are not the same, the OBT assumes that the write has failed. The OBT stops, and displays this value. The NVRAM may still contain some useful information. However, treat any information in the memory with caution.

12.3 Manufacturing Tests

The manufacturing tests can be run with the manufacturing console attached to the External Logic Interface. These tests allow concentrated testing of a faulty area of the hardware. They also allow monitoring of intermittent faults on specific areas of hardware by removing the interference of other tests.

The tests provide the manufacturing plant with these facilities:

1. Ability to run individual OBT tests.
2. Continue testing after an error has occurred so that a fault can be localized.

3. Test failures (with one exception) are reported on the seven-segment display and not logged in NVRAM. This reduces wear on the NVRAM and maximizes its field life.

4. Ability to run the OBT continuously with or without loopback connectors fitted.

The following sections show how to use these tests, and lists those available.

12.3.1 Using the Tests

All the tests are started by the same console command (SPECIAL TEST). The format of the command is:

SPECIAL TEST test c

where:

test is a hexadecimal number identifying the test to run. Section 12.3.2 lists the values you can use for this parameter.

c is an optional parameter that determines whether the test stops on error. If c is 1, the test runs continuously, and if c is 0, the test stops on error. The default value of c is 0.

12.3.2 Available Tests

Table 12-3 shows the number of each manufacturing test, and gives a short description of each. For details of each test, refer to Sections 12.2.1 to 12.2.4.

Table 12-3: Special Tests Available From the Manufacturing Console

Test Number	Description
1	Firmware ROM CRC

Table 12-3 (cont.)

Test Number	Description
2	System RAM -- Each run takes around 15 seconds
3	MicroVAX processor
4	VIC
5	NVRAM -- Use this test with care because of the limited life of the component
6	Ethernet PROM
7	Control and Status Registers
8	Interval Timer
9	Watchdog Timer
A	Buffer RAM -- Each run takes around 20 seconds
B	MicroVAX Memory Management Registers
C	MicroVAX Memory Management Functions
D	LANCE Control and Status Registers
E	LANCE Internal Loopback
F	Ethernet Loopback -- This test fails if the the unit is not connected to an Ethernet or there is no Ethernet loopback connector attached
10	MicroDMA Controller

Table 12-3 (cont.)

Test Number	Description
11	I/O Processor RAM
12	DUSCC
13	Synchronous Internal Data Loopback
14	Receive FIFO
15	Modem Status loopback -- Always completes successfully if there are no synchronous loopback connectors attached
16	Synchronous External Data Loopback -- Always completes successfully if there are no synchronous loopback connectors attached to the unit
17	I/O Processor
18	Exerciser Phase -- Each runs takes around 30 seconds to complete
19	Continuous Execution of OBT -- Always logs errors in NVRAM, and always stops on error

12.3.3 Use of the Seven-Segment Display

The Manufacturing Tests use the unit's seven-segment display to report the pass or failure of each test. Figure 12-2 shows the displays for pass and failure.

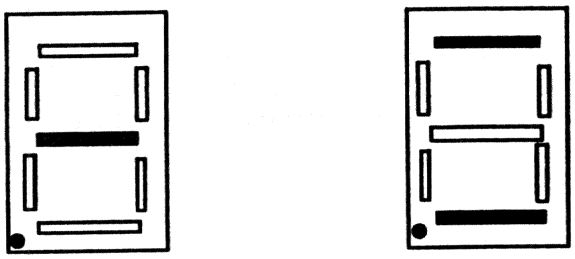


Figure 12-2: Displays Used by the Manufacturing Tests

When you start a test, the pass pattern is displayed. This changes to the failure pattern when the test fails.

You may also see all three bars lit if a short test is finding an intermittent fault. This is due to the fast switching between the pass and failure patterns.

12.3.4 Getting Error Information on Test Failure

Once you have started a test, the only way to gain the attention of the console is to press the BREAK key.

If a test stops on error, you can get the error information (that would normally be logged in NVRAM) like this:

1. Press the BREAK key.
2. Use the EXAMINE/G console command to see the register values. For example, to display the contents of R1 (the error code) use:

   ```
   EXAMINE/G 1
   ```

3. DO NOT use XDT to try and examine the registers. XDT modifies the register contents and so values left in them by the OBT will be lost.

12.4 Limitations to Test Coverage

Although the OBT is extensive, it cannot provide full coverage of all the DEC MicroServer's circuitry. The following sections summarize the limitations of the test coverage for these areas:

1. Basic hardware
2. Ethernet interface
3. Synchronous I/O
4. Memory management
5. Miscellaneous functions

12.4.1 Basic Hardware

There are the following limitations in the tests of the MicroVAX processor, the NVRAM, and both blocks of RAM:

1. It is not possible to test all of the MicroVAX instructions in all possible addressing modes.
2. It is not possible to test fully the NVRAM because of the component's limited life.
3. The tests cannot detect pattern sensitive faults in the RAM blocks. However, these errors are trapped through the parity mechanism built into each block.
4. The CPU clock and the DYRC clock (which provides the interval timer) use the same source. So, the tests cannot detect any inaccuracy in the main 40 MHz clock.

12.4.2 Ethernet Interface

There are the following limitations in the tests run on the Ethernet Interface:

1. The LANCE has no provision for testing transmit data chaining when doing internal loopback.
2. Receiving and sending of full length Ethernet packets cannot be tested because of the limited size of the LANCE's internal silos.

3. The LANCE's TRANSMIT DEMAND (TDMD) feature is not tested because the MicroVAX processor cannot tell exactly when the LANCE accesses the buffers.

4. The Collision error mechanism is not tested because there is no guarantee that the Ethernet transceiver provides a heartbeat signal. Without this signal, the error condition cannot be tested.

5. The babble timer is not tested.

6. The ACON and BSWP fields in the LANCE's CSR 3 are not tested because the DEC MicroServer does not use these functions.

7. The LCOL, LCAR, and TDR fields in the LANCE Transmit Buffer Descriptor cannot be tested.

8. The tolerance of the 12 V supply is not tested, and nor is the presence of the supply if the Ethernet transceiver is a passive one. However, using the Ethernet loopback connector will show whether a supply is present. The light at the back of the connector is lit when there is a supply present.

9. The test cannot detect faulty or wrong components in the interface that could cause it to fail with certain combinations of transceiver and transceiver cable length.

12.4.3 Synchronous I/O

There are the following limitations in testing of the Synchronous I/O parts of the DEC MicroServer:

1. Only data rates of 20K, 64K, 256K, and 2M bits/s are used.

2. Only DDCMP and HDLC protocols are tested, both with NRZ and NRZI.

3. It is not possible to test all the TMS processor's instructions in all the possible addressing modes.

12.4.4 Memory Management

The following memory management features are not tested:

1. Translation Buffer mechanism
2. Memory protection mechanism (except for Kernel Read, access code 0011)
3. P1 space virtual address translation
4. Address translation over the full virtual address range
5. Global address translation
6. Translation-Not-Valid and Length-Violation exceptions (other than to report an error if either of them occur)

12.4.5 Miscellaneous Functions

The other limitations in the test coverage are:

1. It is not possible to test that the MicroDMA controllers and the LANCE can perform DMA operations to and from all possible locations in the Buffer RAM. However, the buffers used in the OBT are spread throughout the RAM to give a reasonable degree of confidence.
2. Missing or open circuit capacitors that decouple the 5 V supply could cause incorect operation under certain circumstances.
3. Missing pull-up resistors (or ones of the wrong value) could cause incorrect operation under certain circumstances.
4. The OBT may work through loopback connectors even when there are incorrect resistor values in the serial line drivers/receivers. However, the system may not work with the customer's equipment.

Chapter 13
VAX Firmware

13.1 Overview

The main function of the VAX firmware is to load and start the software during system startup. The firmware also handles:

- System dumps
- Software support functions (such as supporting a console carrier session, or logging a machine check)
- Access to the Ethernet
- Access to NVRAM
- Console commands
- XDT commands

This chapter gives some details of all these functions, and also shows how the firmware uses system memory.

13.2 System Startup

A DEC MicroServer can start up for a number of reasons but in all cases the firmware uses the same procedure:

1. Setup and test
2. Request a load host
3. Load the software
4. Transfer control to the software

Sections 13.2.1 to 13.2.4 explain each of these steps in more detail. Following them are four more sections that cover related topics:

1. Use of the seven-segment display
2. The format of the file received from the load host
3. Special considerations on the use of the software identification
4. The effect of the manufacturing console on the load procedure

Finally, there's a summary of the loading process using a set of flowcharts.

13.2.1 Setup and Test

When the system is reset, the VAX processor starts executing from location 20040000. At this point, the CPU is:

- Running at IPL 1F
- In kernel mode
- Using the interrupt stack

The processor starts by running the OBT, after which the VAX firmware takes over:

1. See if the manufacturing console is attached -- this determines whether to enter console XDT or to use the normal procedure (see Section 13.2.8).
2. Dump the system if the OBT indicates this is necessary. Otherwise, return to the OBT.
3. Initialize the data used by the console firmware, and issue a request to load the system.

13.2.2 Request a Load Host

The DEC MicroServer is now ready to receive software from a load host. This process takes place over the Ethernet and uses DECnet's MOP protocol.

First, the DEC MicroServer broadcasts a "REQUEST PROGRAM" message that notifies all possible load hosts that the unit needs software. One or more of these hosts will have been set up as a load host for the DEC MicroServer. Those that can supply

software (which is also known as a **load image**) send an "ASSISTANT VOLUNTEER" message back to the DEC MicroServer. The load then begins (see Section 13.2.3).

If the DEC MicroServer receives no response to its request, it repeats the message every 5 seconds, for a maximum of 12 times. After this, the DEC MicroServer performs a backoff to prevent wasting resources on the Ethernet and on the Ethernet nodes.

To back off, the DEC MicroServer waits for a period of time (called the backoff period) before repeating the sequence of REQUEST PROGRAM messages. The first backoff period is one minute, but for each subsequent failure, the backoff period increases to a maximum of 20 minutes.

NOTE

At power up the DEC MicroServer uses a backoff period before sending the first REQUEST PROGRAM message. Doing this prevents a number of DEC MicroServers powering up at the same time (for example, when power is restored after a failure) congesting the Ethernet and their load hosts with REQUEST PROGRAM messages. The backoff period is calculated from the unit's Ethernet address, which makes sure that all units use a different backoff period.

13.2.3 Load the Software

When a load host responds to the load request, software can be loaded. Again, this uses a MOP protocol but this time the process is driven by the load host.

The general process is:

1. The load host sends a segment of the image using a MEMORY LOAD message.

2. The DEC MicroServer transfers the segment to System RAM, and responds with a "REQUEST MEMORY LOAD" message. This asks for the next segment and contains error information if the previous segment was in error. The DEC MicroServer then waits for the next segment.

3. The host responds by either sending the next segment or repeating the previous one (depending on whether there was an error). Again, this step uses the MEMORY LOAD message.

4. Steps 2 and 3 are repeated until all of the image has been sent to the DEC MicroServer.

5. Finally, the load host sends a message called PARAMETER LOAD WITH TRANSFER ADDRESS. The DEC MicroServer uses the contents of this message to transfer control to the software (see Section 13.2.4).

Simply waiting for the host during this process could result in the system hanging if the host or the Ethernet failed. To guard against this, the DEC MicroServer maintains a 5-second timer that's reset when each segment of the image file is received from the host.

If the timer expires, the firmware repeats the last REQUEST MEMORY LOAD message. If this also fails to generate a response, the firmware assumes that the load has failed and reverts to sending REQUEST PROGRAM requests.

13.2.4 Transfer Control to the Software

Once the image has been received from the host, the firmware:

- Sets up the System Parameter Block
- Initializes the system
- Passes control to the software

13.2.4.1 The System Parameter Block - The firmware passes configuration information to the software in a System Parameter Block (SPB). This is an argument list of nine longwords, as Figure 13-1 shows.

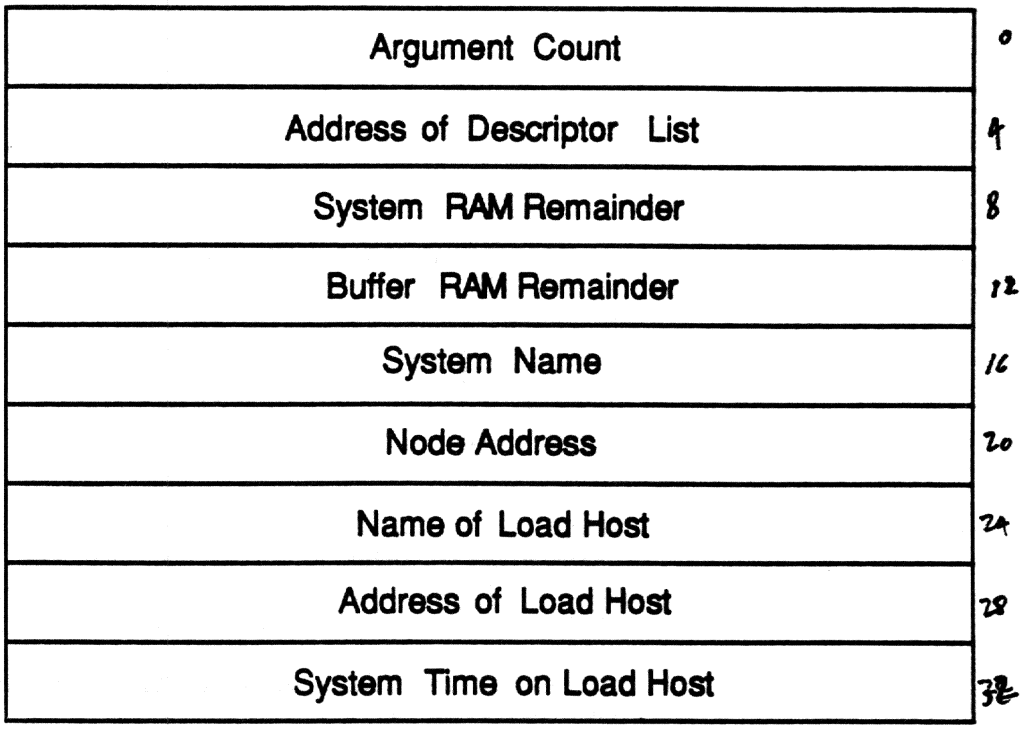


Figure 13-1: Format of the System Parameter Block (SPB)

Table 13-1 summarizes what each field contains.

Table 13-1: Content of the System Parameter Block

Name of Item	Content and Use
Argument count	Number of items in the rest of the list. This item always contains the number 8.
Address of descriptor list	Address of a list of descriptors that would be used in a dump file to retain the physical addresses in the system. The software can use these descriptors to determine physical addresses in the system. (Appendix H shows the content of the dump file and of these descriptors.)

Table 13-1 (cont.)

Name of Item	Content and Use
System RAM remainder	When the load is complete, some of the System RAM is free for the software to use for data storage. This item points to a pair of longwords that contain the first free address, and the number of bytes available.
Buffer RAM remainder	When the load is complete, some of the Buffer RAM will also be available for the software to use for data storage. This argument points to a pair of longwords that contain the first free address and the number of bytes available.
System name (*)	The address of a counted string that contains the node name that the DEC MicroServer system is to use.
Node address (*)	The address of a integer that contains the node address that the DEC MicroServer system is to use. The integer is a 16-bit, unsigned value giving the DECnet address that the system is to use.
Name of load host (*)	The address of a counted string that contains the node name of the load host.
Address of load host (*)	The address of an integer that contains the node address of the load host. The integer is a 16-bit, unsigned value that gives the DECnet address of the load host.

Table 13-1 (cont.)

Name of Item	Content and Use
System time on load host (*)	The address of a counted string that contains the system time on the load host when the load completed. The DEC MicroServer system can use this to set any time-of-day clock it keeps.

(*) These items are sent to the DEC MicroServer as part of the PARAMETER LOAD WITH TRANSFER ADDRESS message. Some items are optional and may not be present. For those that are missing, the firmware puts a null string in the appropriate items of the argument list.

13.2.4.2 Initializing the System - Before finally handing over control, the firmware initializes the system. This does the following:

1. Sets the the initial IPL
2. Sets the current stack to the interrupt stack
3. Disables all interrupts on the board (including the interval timer)
4. Enables parity checking on the System and Buffer RAMs
5. Clears all the MicroVAX general registers except for:
   - SP -- which contains the highest address in the System RAM
   - AP -- which will contain the address of the SPB
6. Sets the initial SCB (this is the SCB that the firmware used)

When all this has been set, control can be handed on to the software.

13.2.4.3 Passing Control to the Software - To pass control to the software, the firmware sets the AP and PC registers in the MicroVAX. It loads the AP with the address of the SPB, and it loads the PC with the transfer address supplied in the PARAMETER LOAD WITH TRANSFER ADDRESS message.

The software now starts to execute. It is responsible for setting its own SCB and stack. However, the firmware does set some default values for these during the initialization. This allows the software to do some work without setting these items. However, they should be set up as soon as possible.

13.2.5 Use of the Seven-Segment Display

Like the OBT, the VAX firmware use the seven-segment display to show the progress of the loading process. Table 13-2 shows the values used and what each indicates.

Table 13-2: The Use of the Seven-Segment Display When Software is Loaded

Display Value	Meaning
1	Sending REQUEST PROGRAM messages, and waiting for a load host to reply.
2	Software being loaded through the MEMORY LOAD and REQUEST MEMORY LOAD messages.
3	Backing off before repeating a sequence of REQUEST PROGRAM messages.
4	Load complete and the PARAMETER LOAD WITH TRANSFER ADDRESS message received. The firmware is now setting up the SPB and initial conditions before handing over control to the software.

13.2.6 Format of the Image File

To the load host, the image file is an executable VAX image that includes a start address. An image can be linked on a VAX/VMS system using the /HEADER and /SYSTEM qualifiers. For DEC MicroServer images, any base address specified with /SYSTEM must be less than F4000 (hex).

Once in the DEC MicroServer, the image has two parts:

1. The MicroVAX code plus initial data structures
2. TMS processor firmware

One of the software's first tasks is to load the TMS processor firmware, as Chapter 14 explains.

13.2.7 Use of Software Identification

In the basic form, a DEC MicroServer can be loaded by any host node that has the following:

1. A suitable image file
2. A record of the image file specification and the DEC MicroServer's Ethernet address in its volatile database
3. The ability to handle service operations (this is enabled through an NCP command)

If a number of hosts have information on a DEC MicroServer (but with different image files) there is no guarantee which image will be loaded. This is because:

1. The standard REQUEST PROGRAM message does not specify the image name
2. The DEC MicroServer always loads from the first host to respond to the REQUEST PROGRAM message

The DEC MicroServer contains a DECnet feature called Software Identification (or Software ID, for short) that enables it to ask for a specific image. The following sections show how this mechanism works, and gives some guidance on its use.

NOTE

Although the DEC MicroServer can use Software ID, it is not supported in the first release of any of the software products. However, the feature can be useful for loading test programs or specific versions of a software product when tracking problems.

13.2.7.1 How Software ID Works - A software ID is a string of up to 16 ASCII characters held in the DEC MicroServer. This string is the name of an image file on a load host.

When the unit is booted, it uses a modified form of the REQUEST PROGRAM message that includes the Software ID. In this case, the unit can be loaded by any host that has that particular file, and not from any other.

Of course, 16 characters is a little restrictive for specifying a file (especially when loading from VAX/VMS systems). So, the Software ID is treated as being just the file name. The load host looks for a file of that name, with a file type of .SYS, in the directory addressed by MOM$LOAD (a system logical name).

13.2.7.2 Limitations of Software ID - On the face of it, using Software ID is an attractive way of loading a particular image. The DEC MicroServer can ask for a particular image, and load hosts simply have to look for that file in MOM$LOAD. Any that has the file loads it to the requesting DEC MicroServer.

This would imply that the load host would not have to keep extensive information in its DECnet databases on the DEC MicroServer (particularly its Ethernet address). However, this is not the case because:

1. The DEC MicroServer does not hold information on its DECnet node name and node address

2. The image file does not contain the DECnet node name and node address

Instead, the node information is provided by the load host using information in its databases. To be able to provide the correct node information, the load host needs to keep information on the DEC MicroServer (in particular, its Ethernet address) in the DECnet databases. This applies to every load host on the Ethernet.

13.2.8 Effect of the Manufacturing Console

The load sequence is different when the manufacturing console is attached to the board. The action taken depends on whether the watchdog timer is disabled. Table 13-3 shows the possible actions that can occur.

Table 13-3: Actions Taken When the Manufacturing Console is Present

State of the Watchdog Timer	Action
Enabled	Load sequence continues, but can be interrupted by pressing the BREAK key on the console.
Disabled	Load sequence halts when OBT completes, and the console prompt appears on the terminal. If the OBT ran as a result of the DUMP switch being pressed, the console enters XDT instead of the normal console.

13.2.9 Summary

The flowcharts on the following pages show the complete loading sequence.

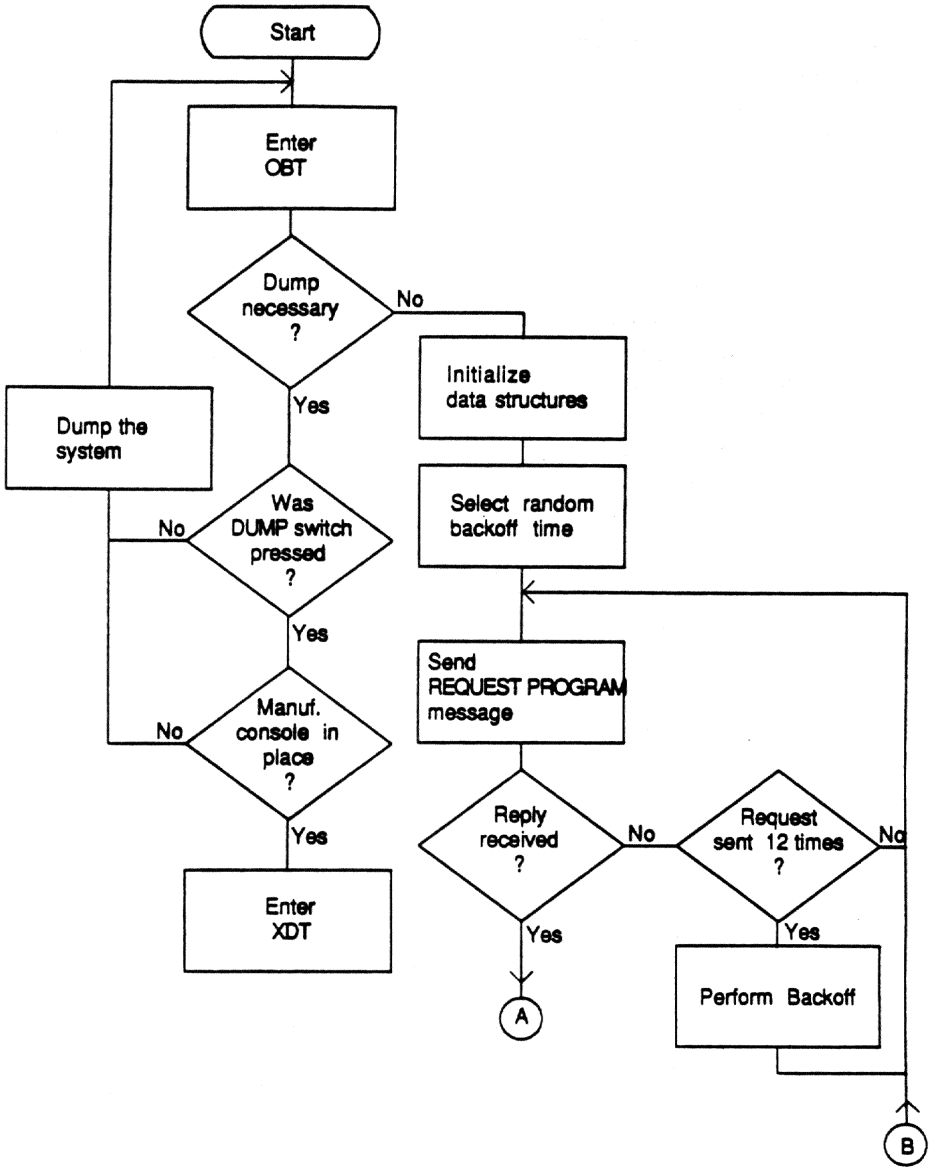
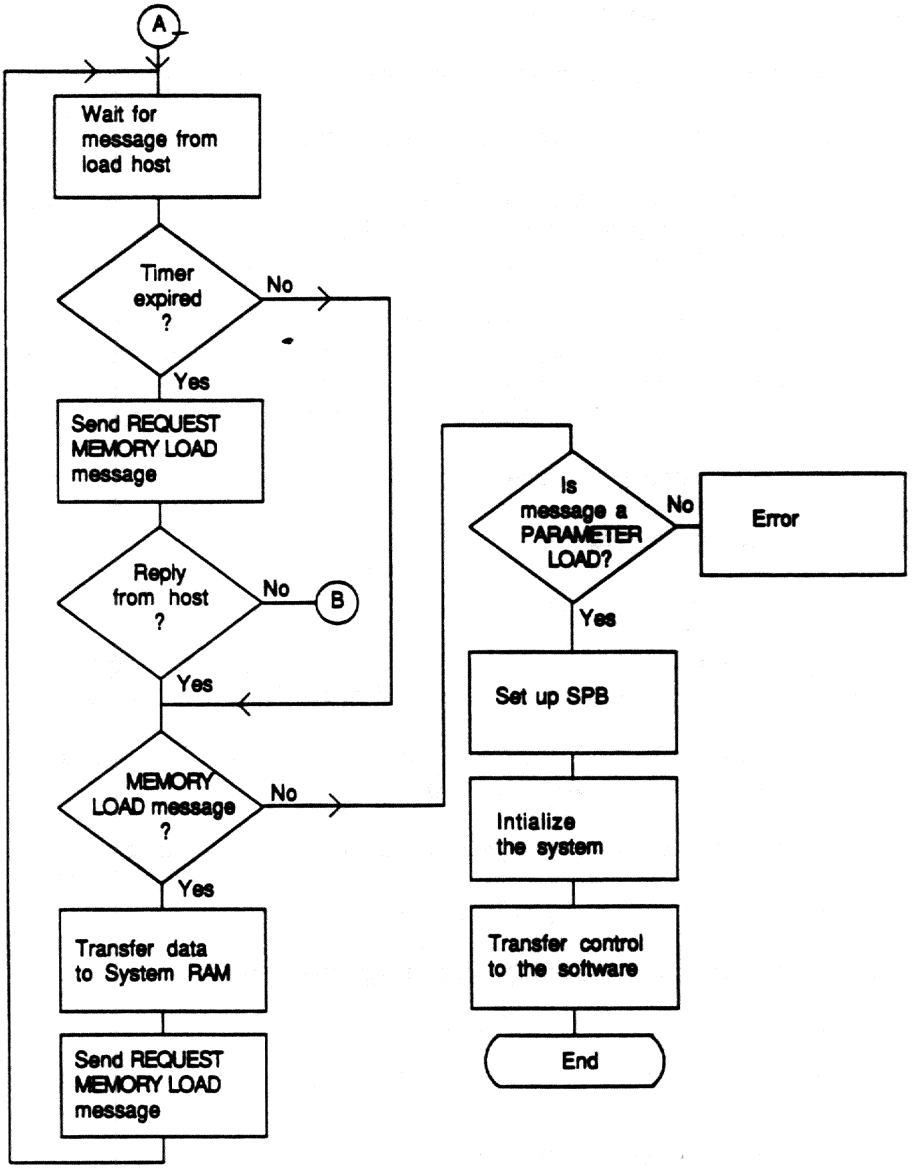


Figure 13-2: DEC MicroServer Loading Sequence

Figure 13-2 (Cont.): DEC MicroServer Loading Sequence



13.3 System Dump

A dump of a DEC MicroServer system preserves the entire context of the system (including memory, board registers, and device registers) and sends this to a host node. The host node puts all the dump information in a file for future analysis.

The system may need to dump for a number of reasons:

1. Someone pressed the DUMP switch on the unit
2. The software signaled a bugcheck
3. A severe error has occurred on the board that causes the DEC MicroServer to restart (for example, the watchdog timer expired)

In all cases, the same procedure is used. The need for a dump causes the MicroVAX processor to restart at location 20040000 and start the system startup sequence. During this, the OBT detects that a dump is necessary. From here the following occurs:

1. Request a dump server
2. Dump the system
3. Reboot the system

Like the loading process, all the messages exchanged between the DEC MicroServer and the dump server use DECnet's MOP protocol.

Sections 13.3.1 to 13.3.3 cover this process in detail. After that are four sections that cover:

1. Use of the seven-segment display during a dump
2. Format of the dumped file
3. Effect of the manufacturing console
4. A summary of the dump process

13.3.1 Requesting a Dump Server

The first stage in the dump process is to try and find a dump server that will accept the information. To do this, the DEC MicroServer sends a REQUEST DUMP SERVICE message on the Ethernet. A dump server that can handle the request replies with a ASSISTANT VOLUNTEER message and the system dump begins.

If the DEC MicroServer receives no response, it repeats the message every 5 seconds for a maximum of 12 times. If there is still no response, the DEC MicroServer abandons the dump procedure and reboots the system (see Section 13.3.3). It does not back off because:

1. Dump servers are optional parts of a network. So, continually asking for a server on a network that doesn't have one means that the DEC MicroServer would never reboot.

2. The DEC MicroServer is rebooted as soon as possible, so restoring communications services.

13.3.2 Dumping the System

When a dump server has responded, the system's contents can be transferred to it. The dump server drives the transfer by requesting the segments one after the other.

The general process is:

1. The dump server sends a REQUEST MEMORY DUMP message in response to the DEC MicroServer's REQUEST DUMP SERVICE message.

2. The DEC MicroServer sends the first segment of the dump file to the server using a MEMORY DUMP DATA message, and waits for a response.

3. The dump server sends another REQUEST MEMORY DUMP that asks for the next segment (or the previous segment again, if it was received in error).

4. The DEC MicroServer sends the next segment in another MEMORY DUMP DATA message, and waits for a response.

5. Steps 3 and 4 are repeated until all the dump information has been sent.

6. The dump server signals the end of the process by sending a DUMP COMPLETE message. The DEC MicroServer can now run the OBT and reboot.

NOTE

Some systems send a BOOT message instead of DUMP COMPLETE. The firmware treats this in the same way as DUMP COMPLETE.

Simply waiting for the dump server during this process could mean that the DEC MicroServer would hang if the Ethernet failed, or the dump server abandoned the dump attempt. To guard against this, the DEC MicroServer maintains a 150 second timer that's reset when each server response is received.

If the timer expires, the DEC MicroServer continues as if the dump had completed normally, runs the OBT, and reboots the system.

13.3.3 Rebooting the System

In normal use, the DEC MicroServer now continues with the startup process as explained in the Sections 13.2.1 to 13.2.4. However, a different process is used if the unit has a manufacturing console attached (see Section 13.3.6).

13.3.4 Use of the Seven-Segment Display

The VAX Firmware uses the seven-segment display to show the progress of the dump process. Table 13-4 shows the values used and what each indicates.

Table 13-4: The Use of the Seven-Segment Display During a System Dump

Display Value	Meaning
5	Sending the original REQUEST DUMP SERVICE message and waiting for a dump server to reply.
6	System being dumped through the REQUEST MEMORY DUMP, MEMORY DUMP DATA, and DUMP COMPLETE messages.

13.3.5 Format of the Dump File

The dump file contains information from many parts of the DEC MicroServer's components, all from noncontiguous areas of address space. So that a dump analyzer can properly reconstruct the system's contents, each part is held in its own area of the dump file, and is referenced by a descriptor. The descriptor contains, among other things, the physical addresses used by that area.

The file does not use the same format as a VAX/VMS dump file, so read Appendix F for a detailed explanation of its content.

13.3.6 Effect of the Manufacturing Console or the Remote Console

If the dump was started through a command from the manufacturing console (or the remote console) the DEC MicroServer does not simply reboot. Instead, control returns to the console waiting for a command (such as BOOT).

However, dump servers that end the dump with a BOOT message automatically cause the DEC MicroServer to load, irrespective of whether a console is attached. To avoid this problem, change the console password in the DEC MicroServer. This makes sure that the BOOT message is ignored because the password is incorrect.

13.3.7 Summary

The flowchart on the following page show the complete dumping sequence.

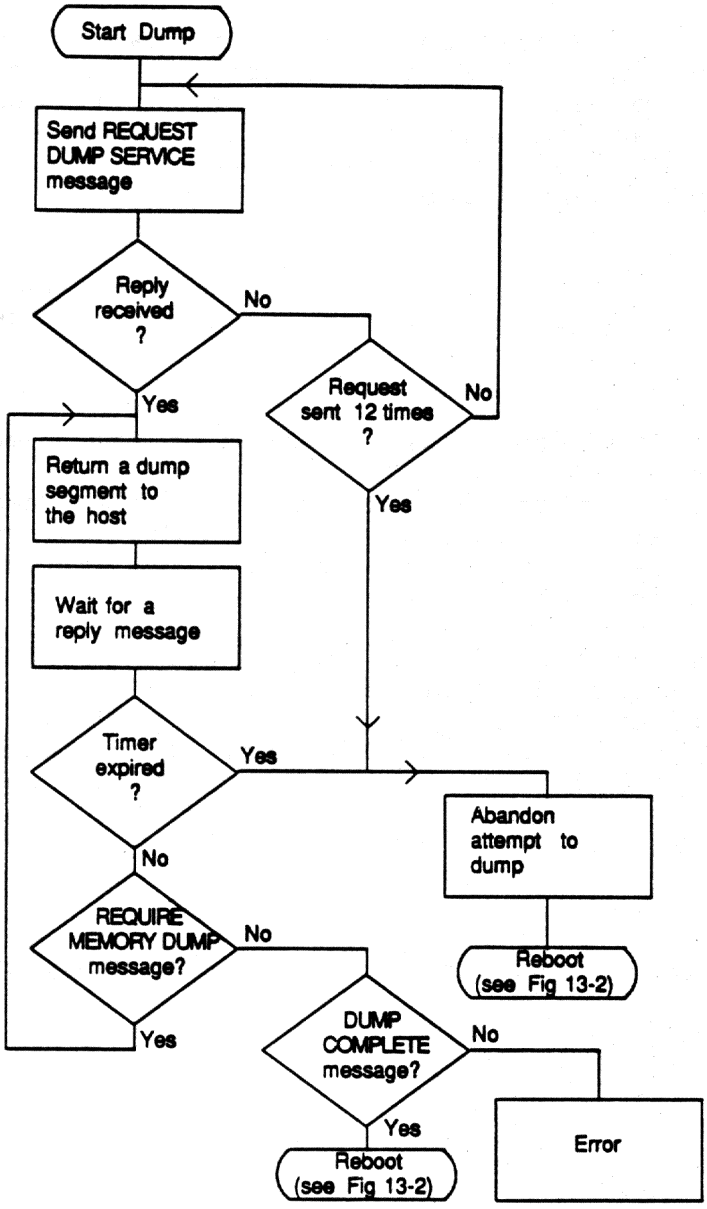


Figure 13-3: DEC MicroServer Dump Sequence

13.4 System Support Routines

The firmware provides a number of routines that the operational software can use. The following are the functions that the routines provide (and their names):

1. Boot the system (ROM$CALL_BOOT)
2. Dump the system (ROM$CALL_DUMP)
3. Maintain console carrier session (ROM$MOP_FN)
4. Log a machine check (ROM$MACHINE_CHECK)
5. Get parameters from NVRAM (ROM$GET_PARAMS)
6. Execute a console command (ROM$DO_COMMAND)
7. Load a specified image (ROM$LOAD_SYS)
8. Set the software state text (ROM$SET_TEXT)
9. Get the hardware version information (ROM$GET_VERSION)

The following sections explain what each of these routines does, their parameters, and how they should be called.

13.4.1 Boot the System (ROM$CALL_BOOT)

Boots the system, after running the OBT, using the startup sequence detailed in Section 13.2. A new image is loaded and so control does not return to the caller when this routine completes.

The routine does not take any parameters.

13.4.2 Dump the System (ROM$CALL_DUMP)

Calls the dump procedure detailed in Section 13.3. A new system image is loaded when the dump is complete, and so control does not return to the routine's caller.

The routine takes one parameter: the bug check code that describes the reason for the dump. This parameter is assumed to be on the top of the current stack.

NOTE

A more convenient way for a memory-mapped system to start a dump is to execute a HALT instruction.

13.4.3 Maintain Console Carrier Session (ROM$MOP_FN)

This routine processes MOP messages (such as console carrier commands, LOOP messages, and TRIGGER messages) that the system's software receives. The routine accepts a MOP message, processes it, and prepares a reply.

The routine accepts three parameters in registers R1, R2, and R3:

R1 contains the address of the message to process.

R2 contains the address of the buffer to accept the reply.

R3 contains the number of bytes in the message to process.

The routine returns values in R0 and R3:

R0 indicates whether the routine executed successfully. If this register contains 1, the routine executed successfully; if it contains 0, there was an error.

R3 contains the number of bytes in the reply message (pointed to by R2). If there is no reply, this register contains 0.

The value of R1 is modified during the routine's execution, but all other register values are preserved.

13.4.4 Log a Machine Check (ROM$MACHINE_CHECK)

The software can use this routine to log a machine check in the NVRAM. The routine requires a valid stack which has the machine check information. The routine can be used for all machine checks (including parity errors). However, it only logs the event in NVRAM; handling and recovery of the machine check is still the responsibility of the software.

When the routine returns to the caller, it will have examined all DYRC registers to find the cause of the error. The routine also resets these registers to re-enable parity checking.

The routine accepts a single parameter, passed in R2, which is the start address of the machine check information.

By using this parameter (and not simply using the top of stack), the routine can also be used to log check occurring on the I_O bus. If a check occurs on that bus when something other than the MicroVAX is bus master, a power fail interrupt occurs. In this case, the check information cannot be put on the stack. Instead, the following process is used:

1. The software sets up a dummy area of 5 longwords. These should contain the following values:

   - o 8
   - o 0
   - o 0
   - o The PC left on the stack by the power fail interrupt
   - o The PSL left on the stack by the power fail interrupt

2. The software then calls **ROM$MACHINE_CHECK** with the address of this dummy area in R2.

3. The routine then fills the second and third longwords like this:

   a. The high-order word of the second longword contains the value found in the **LANCE's CSR0**.

   b. The low-order word of the second longword contains the value in the System CSR.

   c. The high-order word of the third longword contains the value found in the CSR of the Buffer RAM's DYRC.

   d. The low-order word of the third longword contains the value found in the **FAR** of the Buffer RAM's DYRC.

   In some cases, the routine will not be able to read all these register values. If it can't, the appropriate word will contain 0.

4. The routine writes this information to the NVRAM.

The routine preserves all register contents, and the contents of the stack.

13.4.5 Get Parameters from NVRAM (ROM$GET_PARAMS)

The NVRAM contains two longwords of parameters for the software (see Chapter 16). This routine provides an easy access to those longwords.

The routine returns the first longword in R0, and the second in R1. The routine preserves the values in all other registers.

13.4.6 Execute a Console Command (ROM$DO_COMMAND)

This routine enables the software to parse and execute a console command. The software passes the address of a counted string (that contains the command) and receives a success or failure indication in return.

The routine takes just one parameter, passed in R2. This is the address of a counted string to be processed as a console command. This string can be a complete command or can be data that a previously passed command is to use.

While executing, the routine uses R1 in addition to the stack, so on return R1 is modified, and the success/failure indicator is in R0. All other registers are preserved while the routine executes.

NOTE

This routine cannot return any information that the command may create. Therefore, it is mainly used for setting up the contents of NVRAM.

13.4.7 Load a Specified Image (ROM$LOAD_SYS)

This routine loads the system with a specified software image (passed as an input parameter). Unlike ROM$CALL_BOOT, this routine does not run the OBT. Of course, control never returns to the caller of this routine, and instead passes to the loaded image.

The routine takes just one parameter, passed in R2. This is a pointer to a counted string of up to 16 characters that the firmware is to use as a Software ID.

13.4.8 Set the Software State Text (ROM$SET_TEXT)

One of the remote console commands enables the operator to find out the state of the system software. This command displays some text that describes the state (for example, Running or Waiting to Dump). The system's software uses this routine to set that text.

The routine accepts one parameter:

R2 the address of a counted string that contains the text

13.4.9 Get Hardware Version Number (ROM$GET_VERSION)

This routine returns a coded form of the hardware version and revision information stored in NVRAM. The routine returns this information in R0. All other register values are preserved.

The high word of the register contains the version (variant) number, converted to a binary value. The low word contains the revision level, again converted to a binary value. Only the alphabetic part of the revision code is used.

13.4.10 Calling the Routines

Table 13-5 lists the entry points of the routines.

Table 13-5: Entry Points for the System Support Routines

Routine Name	Entry Point
ROM$CALL_BOOT	20040008
ROM$CALL_DUMP	2004000C
ROM$MOP_FN	20040010
ROM$MACHINE_CHECK	20040014
ROM$GET_PARAMS	20040018
ROM$DO_COMMAND	2004001C

Table 13-5 (cont.)

Routine Name	Entry Point
ROM$LOAD_SYS	20040020
ROM$SET_TEXT	20040024
ROM$GET_VERSION	2004002C

Use the JSB mechanism to call the routines, with the following in effect:

1. Kernel mode
2. Memory mapping disabled

All the routines need some stack space, and this can be any valid stack, except for these routines:

- ROM$CALL_BOOT
- ROM$CALL_DUMP
- ROM$LOAD_SYS

These routines use the Interrupt stack.

Note that routines which access the **NVRAM** can take several milliseconds to complete. This is due to the amount of time it takes to change the memory's contents.

13.5 Ethernet Driver

The firmware includes an Ethernet driver that is mainly used to boot the system. Once the software is loaded, that takes over control of the Ethernet port.

The firmware Ethernet driver is also used to:

- Dump the system -- this ensures that the system can be dumped without losing or modifying any of the failed system's data structures.

- Handle MOP requests.
- Provide system identification information.

Sections 13.2 and 13.3 explain the load and dump processes in detail. The next three subsections deal with the following:

- MOP support
- System identification information
- Reaction to NCP TRIGGER commands

13.5.1 MOP Support

Apart from load and dump, the firmware handles the following MOP functions:

1. Remote console sessions -- These use the console carrier functions of MOP. This code is used only when the software is not running (for example, during a system load). At all other times, the console session is managed by the software which uses a system support routine to execute commands.
2. Loop data -- On receiving MOP loop data messages on the Ethernet, the driver returns them. This enables the Ethernet and the DEC MicroServer's Ethernet port to be tested even when no software is loaded.
3. System identification -- See Section 13.5.2.

13.5.2 System Identification Information

Any node on the Ethernet can use the REQUEST SYSTEM ID message to get information about other nodes. If the Ethernet driver receives such a message, it returns a SYSTEM ID message that contains:

1. Maintenance version of the MOP protocol used
2. Function information that shows which MOP operations the unit can handle. That is: boot, dump, multi-block loader, loop, console carrier, and reservations of console carrier.

3. An indication of whether a console session is already in progress.
4. The period of the reservation timer (60 seconds).
5. The size of the remote console's input buffer.
6. The size of the remote console's output buffer.
7. The DEC MicroServer's Ethernet hardware address.
8. The type of the Ethernet communications device.
9. The size of the transmit buffer that the Ethernet port uses.

13.5.3 Effect of the NCP TRIGGER Command

An NCP TRIGGER command sends a boot message to the DEC MicroServer. On receipt of this, the firmware will check the message (against the console password) and then cause the system to reboot.

13.6 Console

Console commands can come from one of two sources:

1. A remote console attached from a load host through the Ethernet
2. A manufacturing console attached directly to the external logic interface

The firmware deals with remote console sessions only if there is no software loaded. Otherwise, the software receives remote console commands, and forwards the output from those commands. However, the software uses one of the support routines to execute the commands.

The firmware always deals with commands from the manufacturing console. This enables engineers in a manufacturing or repair plant to find faults in a DEC MicroServer with or without software running on the unit.

The firmware recognizes just one set of console commands. However, certain of these commands can be used only from the manufacturing console. This prevents customers from

inadvertently damaging their systems.

Appendix E shows how to attach both sorts of console, and lists the commands you can use with each.

13.7 Console XDT

Console XDT is a primitive debugging tool that can be used with a minimum amount of the DEC MicroServer working. This can help to identify faults that prevent the OBT completing.

All that needs to be operational to use XDT is:

1. MicroVAX processor
2. Firmware ROM
3. External logic interface
4. Manufacturing console

Appendix E shows how to use XDT.

13.8 Identification Register

The Firmware ROM includes an identification register that contains the revision level of the DEC MicroServer firmware. This information is accessible from the console through the SHOW VERSION command. The register occupies one longword at address 20040004, and Figure 13-4 shows its format and content.

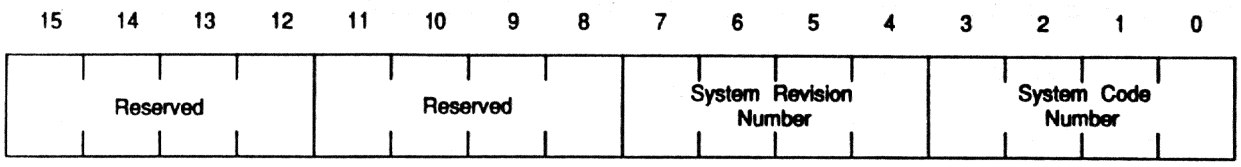


Figure 13-4: Identification Register in the Firmware ROM

13.9 Memory Usage

Some of the space in the System and Buffer RAM blocks is used by the firmware and the OBT. The amount used depends on whether the software is loaded or not, but some memory is always reserved by the firmware.

NOTE

All of the reserved RAM lies outside the region protected by the memory partitioning mechanism. So, it is up to the software to make sure that the reserved regions aren't used. Both regions lie at the high end of each RAM, and start at known addresses. These regions remain protected as long as the software makes sure it uses only those memory sections detailed in the System Parameter Block.

Sections 13.9.1 and 13.9.2 show the use of the two RAM blocks during and after system loading.

13.9.1 System RAM

Figure 13-5 shows the use of the System RAM when the OBT is running. Separate areas are kept for the OBT and the console firmware to make sure that they do not interfere with each other.

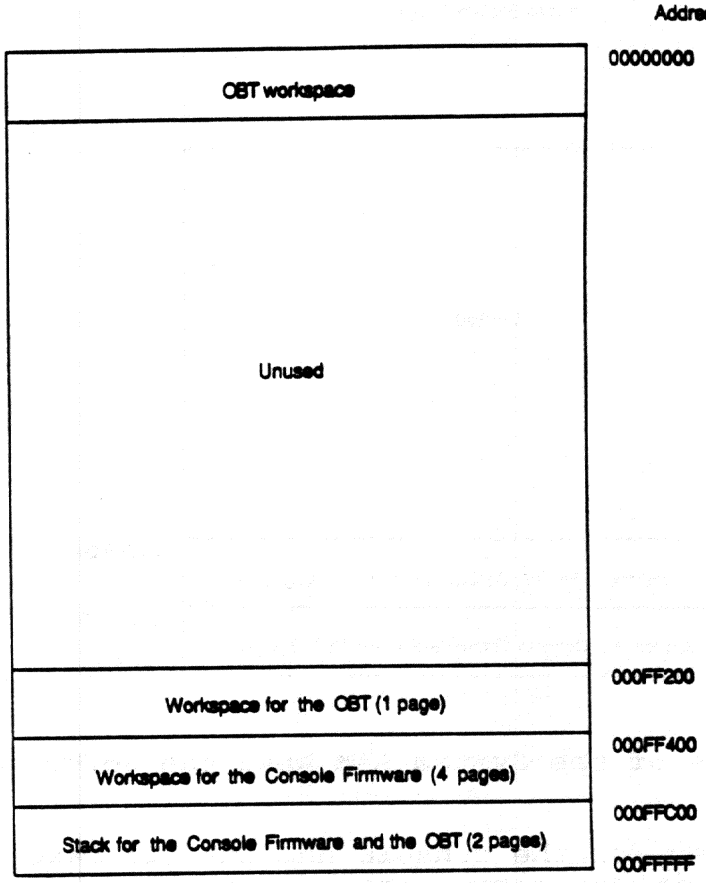


Figure 13-5: Use of the System RAM When the OBT is Running

Once the software has been loaded, the use of the memory changes to that shown in Figure 13-6. The first free address after the system image, and the size of the free area, are passed to the software in the System Parameter Block.

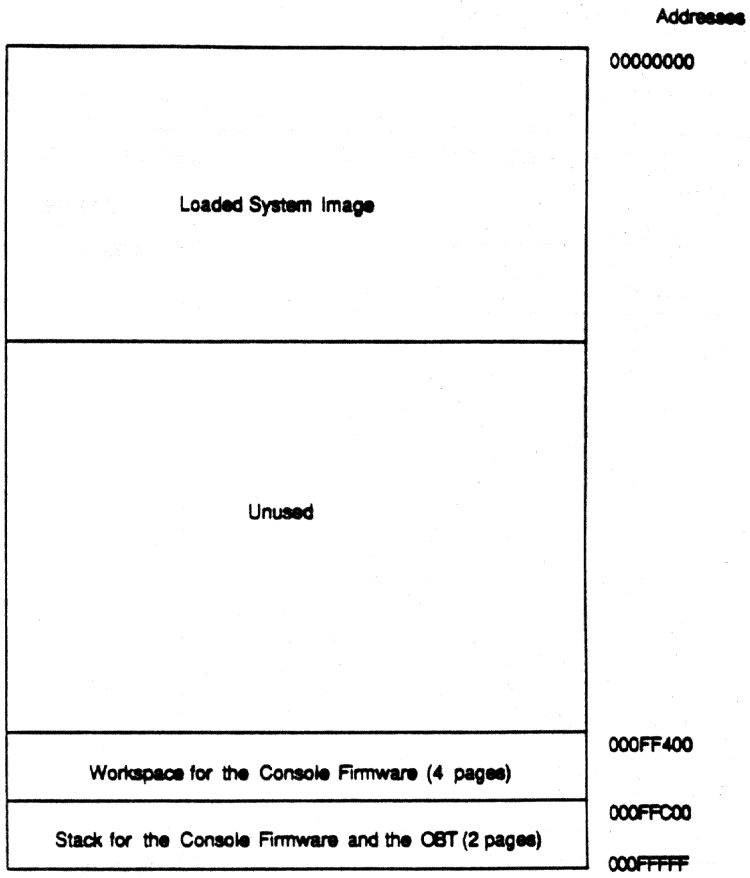


Figure 13-6: Use of the System RAM When the Software Has Loaded

The areas reserved for the console and the OBT make sure that the console can run without corrupting any system data.

13.9.2 Buffer RAM

The Buffer RAM is always used in the same way. It is allocated in the way Figure 13-7 shows.

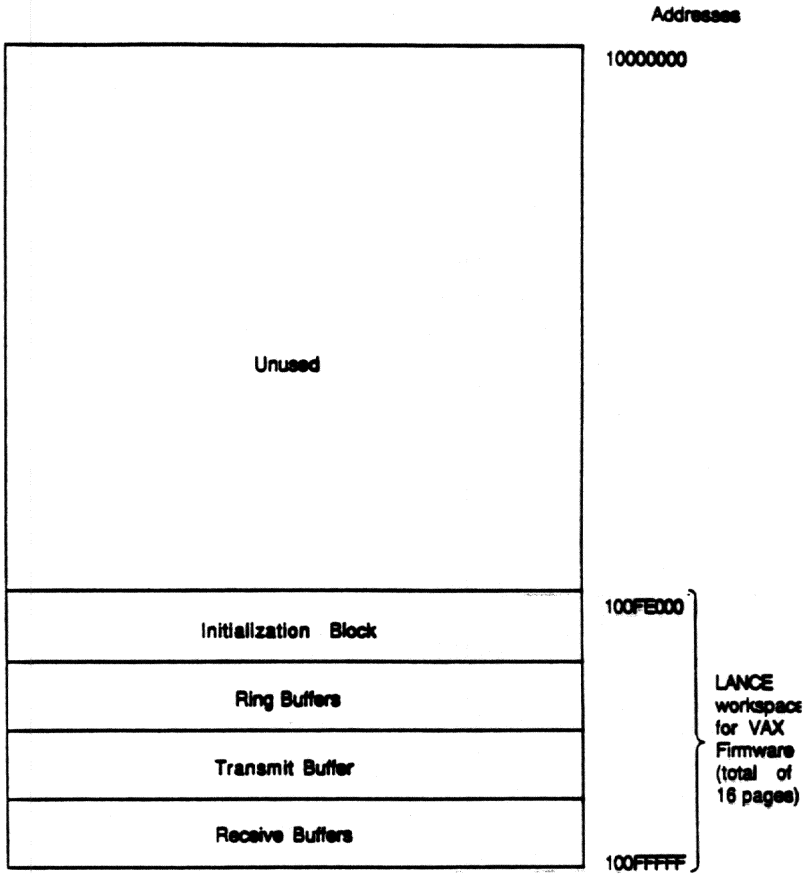


Figure 13-7: Use of the Buffer RAM

The initialization block, ring buffers, and buffers make sure that the firmware can dump and reload a system without corrupting any data structures.

The first free address and the number of bytes available for use as buffers are passed to the software as part of the System Parameter Block.

Chapter 14
TMS Processor Firmware

14.1 Introduction

Important Notice

> This chapter describes the characteristics of the REV V5.83 ROMs. It is possible that future development of the DEC MicroServer's firmware and software will have different characteristics. Each change to the REV denotes possible bug fixes or new features.

Strictly speaking, the TMS processor's code is not firmware because it is loaded as part of the system loading process. However, for all practical purposes, it can be treated as firmware.

The firmware works under control of the MicroVAX software and provides these functions:

1. A framing level interface between the MicroVAX and the synchronous ports
2. Message transmission and reception including CRC checking
3. Detailed control of the synchronous ports

Together, these functions allow the MicroVAX to be used more for processing the information in messages instead of looking after the ports.

The rest of this chapter deals with:

1. Control interface with the MicroVAX
2. Buffer passing
3. Channel control

4. High speed code
5. Loading the firmware
6. Dumping the TMS processor's environment
7. Accessing the Buffer RAM

Finally, there are some examples of how the system sends and receives messages.

14.2 Control Interface with the MicroVAX

The MicroVAX and the TMS processor exchange commands and status information through:

- The Synchronous I/O Control Block
- The Synchronous buffer descriptors

The control block carries information for the complete I/O side and for individual ports. The buffer descriptors contain detailed information on individual packets.

Sections 14.2.1 to 14.2.7 explain the major uses of the control block. See Chapter 15 for information on the buffer descriptors.

14.2.1 Commands to the TMS Processor

The MicroVAX controls each synchronous port through the command registers in the Synchronous I/O Control Block. In all, there are 11 commands (four for receive operations, and seven for transmit). The following sections give details of what each command does. This is followed by two more sections, one on command queuing and the other on port states.

14.2.1.1 Initialize Receiver - This command causes the TMS processor to read the appropriate port initialization block and initialize the registers for that port.

14.2.1.2 Start Receiver - This causes the port's receiver to begin. When a message is received on the line, the TMS processor receives the message and looks for an unused entry in the

appropriate ring of receive buffer descriptors.

14.2.1.3 Stop Receiver and Abort Current Packet - On receiving this command, the TMS processor abandons any packet being received. It also hands back to the MicroVAX any ring entries for the receiver. In each case, the ring status indicates that an abort occurred.

14.2.1.4 Turn Off the Receiver - This command is similar to the Stop Receiver command, but in addition the receiver is shut down. It can be restarted again only by issuing an Initialize Receiver command.

14.2.1.5 Initialize Transmitter - This command causes the TMS processor to read the appropriate port initialization block and initialize the registers for that port.

14.2.1.6 Start Transmitter - This command starts the transmitter. The TMS processor now begins scanning the entries in the appropriate ring of buffer descriptors, looking for a packet to send. It continues looking for ring entries and transmitting packets until one of the stop commands is issued.

As each packet is sent, the ring entry is handed back to the MicroVAX with status information added to the buffer descriptor.

14.2.1.7 Stop Transmitter and Abort Current Packet - This command causes the TMS processor to abandon sending the current packet (if any), to stop the transmitter, and to hand back to the MicroVAX all ring entries for the port's receiver.

Note that to abort the packet, the TMS processor stops sending information and also sends an incorrect CRC.

14.2.1.8 Stop Transmitter After Current Packet - This command causes the TMS processor to stop the transmitter once the current packet has been sent. The ring entries for that packet are returned to the MicroVAX in the normal way, but all others waiting to be processed are retained by the TMS processor.

14.2.1.9 Abort the Current Packet and Flush any Pending Packets - This command causes the TMS processor to abandon the current packet, flush the queue of packets waiting to be sent, and keep the transmitter running.

To flush the queue of packets, the processor sets an abort status in each of the buffer descriptors and returns them to the MicroVAX.

14.2.1.10 Switch Off the Transmitter - This command is similar to Stop Transmitter and Abort Current Packet. However, the TMS processor also switches off the transmitter.

14.2.1.11 Complete the Current Packet and Flush Any Pending Packets - This command causes the TMS processor to complete the current packet, and flush the queue of packets waiting to be sent on the port. The transmitter keeps running.

The ring entries for the current packet are handed back to the MicroVAX in the normal way. Those for queued packets are handed back with an abort status.

14.2.1.12 Command Queuing - Each time that the TMS processor reads a command, it clears the appropriate Port Command Register. This allows the MicroVAX to queue commands to the TMS processor.

The TMS processor clears the register only when it has successfully read the command and can accept it for processing. The MicroVAX can continue supplying commands (without waiting for interim ones to complete) as long as it only gives a new command when the Port Command Register is clear.

Notice that clearing the register does not mean that previous commands have been completed. The action only indicates that the TMS processor has accepted those commands for processing.

14.2.1.13 Port States - The receiver and transmitter for a port are always in a defined state. The MicroVAX uses the commands to change the state of the receiver and transmitter. However, certain commands can be issued only when the transmitter or receiver is in a particular state. This helps to prevent unexpected or unpredictable events occurring.

The possible states are:

- Off
- Stopped
- Running

Tables 14-1 and 14-2 are state tables that show:

- The commands that the TMS processor can accept in each state
- The new state that the receiver or transmitter enters on completing a command

To use one of the tables, begin with the Initial State noted at the foot of the table. In the appropriate column, only those commands that do not have an entry of "Error" can be issued. For those commands that are legal, the new state is shown. The column for that new state shows the commands available.

In both tables the "Error" state indicates that a command cannot be used in the current state. In these cases, the current state does not change, but the Port Status Register in the control block will indicate that the command could not be executed. If the MicroVAX software is queuing commands, it needs to make sure that the status is matched to the appropriate command.

Table 14-1: Receiver State Table

	State		
Command	**OFF**	**STOPPED**	**RUNNING**
Initialize receiver	STOPPED	Error	Error
Start receiver	Error	RUNNING	RUNNING
Stop receiver	Error	STOPPED	STOPPED
Turn off the receiver	Error	OFF	OFF

Table 14-1 (cont.)

Command	OFF	STOPPED	RUNNING
Initial State = OFF			

Table 14-2: Transmitter State Table

	State		
Command	**OFF**	**STOPPED**	**RUNNING**
Initialize transmitter	STOPPED	Error	Error
Start transmitter	Error	RUNNING	Error
Stop transmitter and abort current packet	Error	STOPPED	STOPPED
Stop transmitter after current packet	Error	STOPPED	STOPPED
Abort current packet and flush	Error	STOPPED	RUNNING
Switch off the transmitter	Error	OFF	OFF
Complete current packet and flush	Error	STOPPED	RUNNING
Initial State = OFF			

14.2.2 Boot and Dump Control

The times when the system is loading or dumping are important. To operate correctly the system must load correctly and complete initialization before starting work. Similarly, to dump properly, there must be as much information as possible available.

To help ensure that these two operations can be successful, the MicroVAX and the TMS processor use fields in the Global Status Register of the Synchronous I/O Control Block.

The following sections show how these fields are used. For a more detailed description of how the TMS processor is loaded and dumped see Section 14.5 and 14.6.

14.2.3 Loading

Part of the system load procedure provides the TMS processor with its code. Once loaded, the TMS processor initializes itself. Once these operations are complete the MicroVAX can begin to give commands to the TMS processor.

The MicroVAX must wait until the TMS processor is ready for commands. To ensure it does this, the TMS processor uses the SETUP field in the Global Status Register. When loading the TMS processor, the MicroVAX makes sure that this field is clear. Once the load is complete, the MicroVAX starts the TMS processor and waits for the SETUP field to be set.

When it is started, the TMS processor initializes its internal data structures and ensures that the rest of the Synchronous I/O elements are appropriately set. Then, it sets the SETUP field and waits for commands from the MicroVAX.

14.2.4 Dumping

A system dump needs to contain as much information as possible. This needs to include the register values from all the major devices. As the MicroVAX creates a system dump, it can ensure that most of the registers are saved. However, the MicroVAX cannot access the information in the TMS processor.

So, TMS processor information (registers and internal RAM) is copied into the TMS RAM. The two processors use a special protocol (involving the DUMP field of the Global Status Register) to indicate when this copy is complete.

Either the MicroVAX or the TMS processor can decide that a system dump is necessary. In either case, the TMS processor saves its information and then sets the DUMP field. Once this is done the MicroVAX can halt the TMS processor and include the TMS RAM in the dump.

14.2.5 Modem and Interrupt Control

The Synchronous I/O Control Block contains copies of the modem status registers (IOSR0, MDSR1, and MDSR2). It also contains registers that provide the same functions as the modem control registers (MDCR1 and MDCR2).

The MicroVAX could access the registers on the TMS bus directly, but only if the TMS were first reset. In a running system, this would lose any benefit from a parallel bus architecture.

So, the copies enable the MicroVAX to access the registers without having to access the TMS bus. Using the copies also gives these benefits:

1. **The active state of all fields can be made the same.** In the real registers, some bits are active high and others are active low. In the copies, all bits are active high with the TMS processor doing the necessary translation. This simplifies the work the MicroVAX has to do.

2. **The TMS processor can smooth the status values.** Some of the status values can change momentarily under certain circumstances. If the MicroVAX were to read the registers directly, it would have to take this into account. However, using the copies allows the TMS processor to do this extra processing and changing bit values only when a real change has occurred. This, again, simplifies the work the MicroVAX has to do.

3. **The MicroVAX can change the control values at any time.** There is no need for the MicroVAX to know what the TMS processor is doing to be able to change the values. Instead it can simply alter the values and the TMS processor will pick them up when it wants to.

14.2.6 Status Values

The control block includes a number of fields the two processors use to exchange status information:

- Global Status Register
- Port Status Registers
- Alive counters

The following sections discuss these features in more details.

14.2.6.1 Global Status Register - This register gives the overall status of the synchronous side of the DEC MicroServer. It enables the MicroVAX and the TMS processor to coordinate loading and dumping.

The TMS processor generates the TMSINT interrupt each time it changes the value of this register. This saves the MicroVAX from constantly having to scan for a change, while making sure that it notices any change.

14.2.6.2 Port Status Registers - There are four Port Status Registers, one for each of the synchronous ports. They show the current state of each port's transmitter and receiver as well as reporting errors (such as an invalid configuration). The TMS processor maintains these registers.

The TMS processor also uses the TMSINT interrupt to tell the MicroVAX when the value of any of these registers changes. This helps the MicroVAX know when commands have been completed or when there are changes in the status of individual channels.

14.2.6.3 Alive Counters - The control block contains an alive counter. This is used by the MicroVAX processor to detect any deadlock in the TMS processor.

The TMS processor tries to update the alive counter on each system clock tick. The MicroVAX processor reads the counter periodically (for example every 150 ms). If the same value is read twice, the TMS processor is probably deadlocked, so the MicroVAX processor starts a system dump.

14.2.7 MicroVAX/TMS Processor Protocol

Except for the command registers, only one processor at a time can write to a location in the Synchronous I/O Control block. Therefore, no strict synchronization of the processors is necessary.

In the DEC MicroServer, each processor periodically scans the control block looking for changes in the locations it reads. However, there is provision for interrupt driven change notification that can be used in later versions.

14.2.7.1 Location Polling - Each processor has its own polling routine that looks for changes in the control block. Also, each processor uses its own timer to make sure the polling routine runs regularly. These timers are derived from clock inputs to the processors.

The MicroVAX runs its polling routine each time the 100 Hz clock occurs (driven by the TIMERINT interrupt). The TMS processor runs its routine each time its internal timer expires (which drives the TINT interrupt every 2 ms).

For example, to supply a command to the TMS processor, the MicroVAX stores the command in the appropriate register as required. The next time that the TMS processor runs its polling routine, it notices the change in the command registers, reads the value and clears the register. In turn, the MicroVAX polling routine notices that the register is clear and so another command could now be issued.

Similarly, the TMS processor updates the modem status values in the control block as necessary. The MicroVAX picks up these changes the next time its polling routine runs, or through the EVENT interrupt.

14.2.7.2 Interrupt Notification - The polling method relies on each processor to scan the control block regularly. However, there may not be changes in the modem status for a number of polling intervals during which the MicroVAX could be doing other work. So, the product contains provision for interrupt-driven notification.

In this scheme, the TMS processor changes fields in the status block in the normal way. It can then generate the MODCHGINT interrupt to the MicroVAX on specific requested signals. Similarly, the MicroVAX would look for changes in the modem

status only when this interrupt occurred.

When the MicroVAX does not want to know every change in channel status, the interrupt scheme allows the Modem Mask registers in the control block to select the MODCHGINT interrupt only on individual signals.

14.3 Buffer Passing

The TMS processor and the MicroVAX processor exchange buffers through an Ownership bit in each buffer descriptor and interrupts to the MicroVAX.

The MicroVAX determines the address and size of each of the buffers. In selecting the size, it makes sure that any incoming or outgoing message will fit the buffer. This prevents buffer overflow errors.

The method of buffer passing is similar for transmit and receive. So, we'll use an example of sending a message on Port 1 to illustrate the process.

The MicroVAX finds a free descriptor and fills its buffer with the message. It sets the size of the message in the descriptor, and changes the ownership bit. The descriptor and buffer is now "owned" by the TMS processor and so has, in effect, been passed over to that processor.

When its TINT interrupt next occurs, the TMS processor scans the buffer descriptors and the control block. In the descriptors it will notice this new buffer is waiting to be transmitted. It sends the information and then:

1. Sets any necessary status
2. Resets the ownership bit
3. Generates the TX1INT interrupt

The interrupt tells the MicroVAX that the transmission is complete and the buffer has been passed back.

14.4 Channel Control

The MicroVAX controls all the synchronous ports in a similar way. To illustrate this process, the following sections use an example of channel 1.

The overall process has three parts:

1. Setup and initialization
2. Start and use
3. Shutdown

14.4.1 Setup and Initialization

First the MicroVAX needs to set the correct initial values in the Synchronous I/O Control Block and the rings of buffer descriptors.

In particular, the MicroVAX needs to set up the following areas of the control block:

1. Modem mask register
2. Port modem control register for Port 1
3. Port initialization block for Port 1
4. The ring block address

The MicroVAX must also make sure that all the rings of buffer descriptors have been set to the initial or default values.

The MicroVAX next initializes the port's receiver and transmitter by loading the appropriate values into the port command registers. When the initialization is complete, the TMS processor sets the port status register appropriately and generates the TMSINT interrupt.

14.4.2 Start and Use

On receiving the TMSINT interrupt, the MicroVAX examines the status register to make sure the initialization was successful. Then, it can start the port's receiver and transmitter by setting the appropriate values in the command registers.

Again, the TMSINT interrupt will occur when the commands have completed. From here, the TMS processor will scan the rings of buffer descriptors looking for messages to transmit. Also, the TMS processor looks for a buffer descriptor in the appropriate receiver ring to receive the next packet.

As each message is sent or received, the TMS processor triggers the TX1INT and RX1INT interrupts as appropriate.

While the port is in use, the MicroVAX can change the values in the modem control register at any time. The TMS processor will pick these changes up (and modify the real register) during its next poll of the block.

14.4.3 Shutdown

To shut down a port, the MicroVAX puts the appropriate stop command in the port command registers. The actual command used depends on the port state. Shutdown returns the port to the STOPPED state. Here the MicroVAX can turn off the line completely or can start again.

14.4.4 Occurrence of the TMSINT Interrupt

TMSINT can occur on completion of a number of TMS commands. Therefore, the MicroVAX should always check to see if more than one outstanding request has completed.

14.5 Loading the Firmware

As described earlier, the "firmware" is loaded as part of the software image. Before it loads this image, the VAX firmware halts the TMS processor by setting the RSTDSP bit in the System CSR. Once this bit is set, the MicroVAX can freely access the TMS RAM.

The software image uses one of the DMA devices as a window to pass the code into the TMS RAM.

The DMA window allows the MicroVAX to see only 16K words of the TMS RAM at a time. However, the board has provision for double that capacity. To load larger amounts of TMS RAM, the MicroVAX uses further bits in the System CSR.

If 32K of memory is fitted, the TMSMEM bit in the System CSR is clear. So, the MicroVAX uses the PAGE0 bit in the same register to indicate which 16K portion of memory is being loaded. By clearing and then setting that bit, the MicroVAX can load the entire TMS RAM.

Once the TMS RAM is loaded, the software can set up the Synchronous I/O Control Block and the rings of buffer descriptors that the system needs. The software puts the address of the control block into a predetermined pair of words in the TMS RAM (TMS processor addresses 20 and 21, hexadecimal).

At this point, the TMS processor is ready to start, and the MicroVAX does this by clearing the RSTDSP bit in the System CSR. Once running, the TMS initializes itself, and then initializes the synchronous ports.

14.6 Dumping the TMS Processor's Environment

A complete dump of the DEC MicroServer system can be necessary for a number of reasons. For example:

1. The MicroVAX or the TMS processor has detected an irrecoverable error.

2. The MicroVAX has detected no change in the I/O Processor Alive Counter.

In any case, the TMS processor needs to save its internal status in the TMS RAM so that as much information as possible can be included in the dump file.

A dump can be started by either processor and a similar process is used in both cases. The following sections set out the major differences. See Chapter 13 for details of how the collected information is dumped to a host system.

14.6.1 A Dump Started by the MicroVAX

If the MicroVAX detects an irrecoverable error, it calls the dump entry point in the ROM.

Then, to trigger the TMS RINT (DUMP) interrupt, the MicroVAX firmware sets and then clears the TMSCRL2 bit in the System CSR. The time between setting and clearing the bit must be at least 1 ms.

On receiving the interrupt, the TMS processor:

- Saves its internal information in the TMS RAM
- Sets the DUMP bit in the Global Status Register of the Synchronous I/O Control Block
- Generates the TMSINT interrupt

On receiving the TMSINT interrupt, or after timing out, the MicroVAX firmware:

- Halts the TMS processor (using the RSTDSP bit in the System CSR)
- Copies the contents of the TMS RAM (using a DMA window) into the dump image

In some circumstances, the TMS processor may not be able to save its context. In these cases the DUMP bit in the status register, and hence the TMSINT interrupt, will never occur. So, the VAX firmware sets a time limit of 3 ms for the interrupt to occur. If the limit expires, the firmware halts the TMS processor and uses whatever's in its RAM regardless of whether the internal information is there or not.

14.6.2 A Dump Started by the TMS Processor

If the TMS processor detects the irrecoverable error, it:

- Saves its internal information in the TMS RAM
- Sets the DUMP bit in the Synchronous I/O Control Block
- Triggers the TMSINT interrupt

The software will see the change in the status. With the DUMP bit set, the software saves its own internal information and then calls the firmware to dump the system.

14.7 Accessing the Buffer RAM

The TMS processor accesses the Buffer RAM through the DMA devices. There are three types of access that the TMS processor makes:

1. Accessing the descriptor rings

2. Accessing the Synchronous I/O Control Block

3. Accessing the buffers

As chapter 8 shows, the lowest priority DMA channel is dedicated to TMS window access to the Buffer RAM.

For as much time as possible this DMA channel is used as a window to the Synchronous I/O Control Block. When data arrives on Port 3, the window is moved so that it maps onto the appropriate data buffer. When the data transfer is complete, the window reverts to the control block.

The switching of the channel is driven by the RXDMR3 signal. DUSCC B asserts this signal each time characters arrive on Port 3's receiver. The signal is connected to the BIO pin on the TMS processor and - if enabled - raises DMAINT, which causes the TMS to process the character and eventually transfer it to the I/O buffer.

This channel sharing takes up time in the changing from one function to another. This is not significant, however, as the DMA channel is only used when all four ports are are in use. In this case, the maximum data speed is 64K bits/s. This comparatively low speed, gives the TMS processor sufficient time to change the function of the channel without losing any data. In all other configurations, the DMA channel is always free and so can be permanently used to access the control block.

Chapter 15
Buffer Management

15.1 Introduction

Each of the five I/O ports has its own set of buffers. Each set is divided into two parts. One is used for transmit and one for receive operations. To achieve the DEC MicroServer's high performance, there is more than one buffer for each direction (receive and transmit) on each port.

So, in all there are ten sets of buffers that need to be controlled to make sure that:

1. Unprocessed information is not overwritten

2. Space is available for new messages

In the DEC MicroServer each buffer has a descriptor containing information such as the buffer's location and size. To simplify the buffer management, each set of descriptors is ordered as a circular list called a **ring structure**.

This chapter gives an introduction to the way ring structures are used in the DEC MicroServer. The chapter also sets out the descriptors used for Ethernet and synchronous I/O operations.

15.2 Descriptor Rings and Buffer Management

The buffer descriptors for each port are grouped together in rings (one for transmit and one for receive), with each descriptor in a ring having the same size and layout (see Figure 15-1).

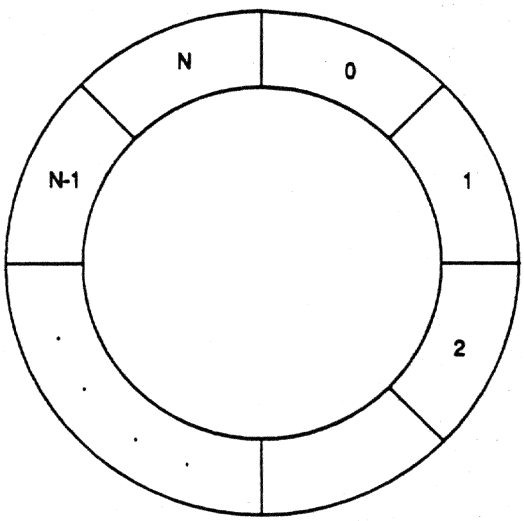


Figure 15-1: A Ring of Buffer Descriptors

Having more than one buffer for each direction (receive and transmit) allows both the MicroVAX processor and the port (through the LANCE or the TMS processor) to access a ring at the same time. For example, in a transmit ring the MicroVAX can be filling one buffer while the port is emptying another.

The rings are always dealt with sequentially, starting from the same place. Both the port and the MicroVAX access the rings in the same direction.

15.2.1 Descriptor Ownership

With the port and MicroVAX accessing a ring at the same time, some form of descriptor ownership is needed. Otherwise, the MicroVAX could overwrite a descriptor that (say) the port has yet to process and the original message would be lost.

To help prevent such errors, each descriptor always has an owner. During an I/O operation, the owner changes between the MicroVAX and the I/O port, but at all times only the current owner can change the contents of a descriptor (and its associated buffer).

So to continue the transmit example, the MicroVAX owns a descriptor when it is filling a buffer. When emptying the buffer the port owns the descriptor.

The current owner of a descriptor is recorded by a bit in the descriptor itself. When the bit is set, the I/O port owns the descriptor. When the bit is clear the MicroVAX owns the descriptor.

Ownership changes when the current owner decides it has finished with the descriptor. The owner then hands over the descriptor to the other partner by changing the value of the ownership bit.

For example, when the MicroVAX has finished filling a transmit buffer, it sets the ownership bit in the appropriate descriptor. The I/O port is now the owner of the descriptor and can use it to empty the buffer. When the buffer is empty the port hands the descriptor back to the MicroVAX by clearing the ownership bit.

So the ownership bit provides a simple buffer control mechanism. This, together with the sequential processing of buffers, makes sure that:

1. No messages are lost
2. Messages are always treated in the correct order

When a system starts, or when any ring is empty, the MicroVAX owns all the descriptors.

15.2.2 Descriptor Contents

The information held in each descriptor, and its structure, depends on:

- The type of I/O port (Ethernet or synchronous)
- The type of ring (receive or transmit)

Even so, there are some common features.

Descriptors for receive operations contain:

- Buffer address
- Buffer size
- Status and error information
- Size of the received message

- Ownership bit

Descriptors for transmit operations contain:

- Buffer address
- Message size
- Status and error information
- Ownership bit

The descriptors differ in the type of status information (Ethernet and synchronous ports have different events to report), and the structure of each descriptor.

Sections 15.3 and 15.4 show the various descriptor formats in detail.

15.2.3 Ring Sizes

The size of each ring is important to the system's performance. Too few entries in a ring becomes a problem at high line utilization. Here, the demand for entries to hold new messages outstrips the rate they can be emptied. So, the new messages have to be delayed until entries are available. This can mean that messages have to be retransmitted, reducing the maximum performance of the system. On the other hand, if there are too many entries, space in the Buffer RAM is wasted.

So, determining the size of each ring is a balance between:

1. Having enough buffers so that the MicroVAX or the port are not held up waiting for buffers
2. Not wasting space in the Buffer RAM through too many buffers and their descriptors (generally, more buffers require more processing time for each partner because the rings are wider)

The DEC MicroServer uses just two ring sizes:

- **Ethernet rings** -- for remote console operations (using the ROM code), the transmit ring has 4 entries, and the receive ring has 2. The number of entries used by the operational (non-ROM) software is system dependent.

- **Synchronous rings** -- all rings for synchronous I/O have 32 entries.

15.2.4 Using the Rings

All rings are used in a similar way. The following descriptions show how they are used to send a message and receive a message.

For simplicity, assume that each message fits into one buffer. Also, we'll assume no message has yet been sent and so all the ring entries are owned by the MicroVAX.

15.2.4.1 Transmit - The MicroVAX queues a message for transmission like this:

1. Find the first ring entry with an ownership bit that is clear.
2. Put the address of the message buffer in the correct field of the descriptor.
3. Set the status bits as necessary.
4. Put the length of the message in the correct field of the descriptor.
5. Set the ownership bit.

To send subsequent messages, the MicroVAX follows the same procedure. If, however, all the entries are owned by the I/O port, the MicroVAX must wait until one becomes available.

The LANCE and the TMS processor continually poll the entries in their rings, looking for ones with the ownership bit set. On finding such an entry, they take the information from the buffer and send it across the communications link. Status information from the transfer (for example, errors) is put in the descriptor before the port clears the ownership bit and interrupts the MicroVAX. The MicroVAX can then examine the status, taking appropriate action if needed, before reusing the entry.

15.2.4.2 Receive - The MicroVAX initializes all receive descriptors before handing them over to the I/O port. This initialization includes putting in the buffer address and the buffer length.

The LANCE or the TMS processor handle the reception of a message for their appropriate ports like this:

1. Find the first entry in the appropriate ring whose ownership bit is set.
2. Put the message in the buffer.
3. Set the status bits accordingly.
4. Enter the length of the message in the descriptor.
5. Hand the descriptor and buffer to the MicroVAX by clearing the ownership bit.
6. Interrupt the MicroVAX.

On receiving the interrupt, the MicroVAX can check the status information, taking any action that is necessary. Then the descriptor is re-initialized ready for future use.

15.3 Ethernet Ring Descriptors

Both types of LANCE descriptor are four words long, and are quadword aligned in the Buffer RAM. The following sections show the content and format of the two types of ring.

15.3.1 Receive Descriptor

Figure 15-2 shows the format of a receive descriptor. Table 15-1 contains more information on each of its fields.

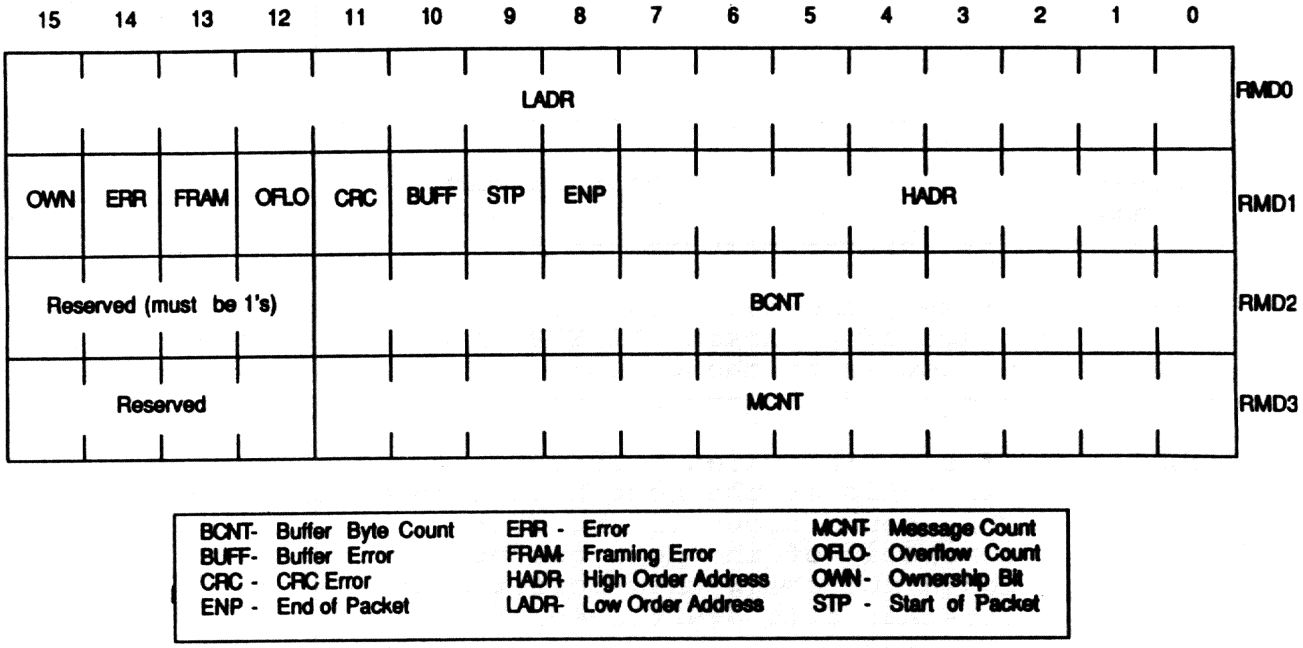


Figure 15-2: Format of a Descriptor for a LANCE Receive Buffer

Table 15-1: Fields in the LANCE Receive Buffer Descriptor

Field Name	Meaning and Use
	Buffer Address Information
LADR HADR	These two fields contain the start address of the buffer to receive a message. LADR contains the least significant 16 bits of the address, and HADR contains the remaining 8 high order bits.
	Status Information
ENP	If set, indicates that this is the last buffer in a chain used to hold a single packet. If STP is also set, the complete packet is in the buffer.

Table 15-1 (cont.)

Field Name	Meaning and Use
STP	If set, indicates that this buffer contains the start of a packet. If ENP is also set, the entire packet is in the buffer.
BUFF	If set, indicates that the LANCE could not get ownership of the next buffer to chain a packet. This could occur if the MicroVAX still had ownership of the next buffer, or if the LANCE could not check that buffer's status before its internal silo overflowed.
CRC	If set, indicates that the LANCE detected a CRC error on the packet.
OFLO	If set, indicates that the LANCE has lost all or part of a packet because it could not be stored in a buffer before the LANCE's internal silo overflowed.
FRAM	If set, indicates that the packet does not contain a multiple of eight bits (that is, it contains an incomplete number of characters) AND that a CRC error was detected.
ERR	If set, indicates that one of the error bits -- FRAM, OFLO, CRC and BUFF -- is set. So, it provides a summary of any error condition.
	Descriptor Ownership Information
OWN	The setting of this bit shows who has ownership of the descriptor (and hence of the buffer). When the bit is clear the MicroVAX has ownership, and when it's set the LANCE has ownership.

Table 15-1 (cont.)

Field Name	Meaning and Use
	Buffer Size and Message Size Information
BCNT	The size of the available buffer. The MicroVAX completes this field whose value must be at least 64.
MCNT	For messages that are successfully received, the LANCE uses this field to indicate how many bytes are in the message. For chained messages, this field is complete only in the descriptor that has the ENP bit set.

15.3.2 Transmit Descriptor Ring

Figure 15-3 shows the format of a transmit descriptor, and Table 15-2 contains more information on each of its fields.

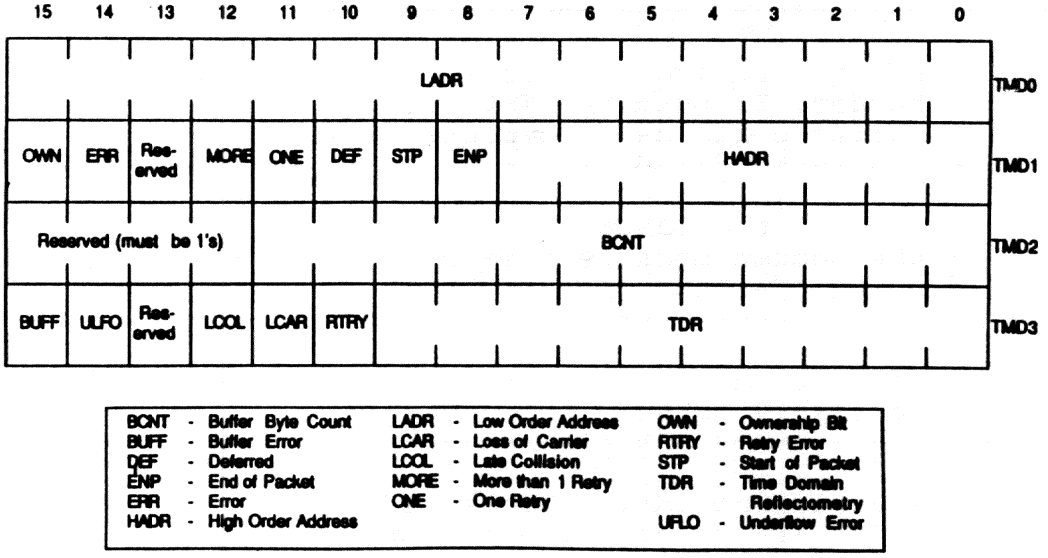


Figure 15-3: Format of a Descriptor for a LANCE Transmit Buffer

NOTE

TMD3 contains useful information only if the ERR bit in TMD1 is set.

Table 15-2: Fields in the LANCE Transmit Buffer Descriptor

Field Name	Meaning and Use
	Buffer Address information
LADR HADR	These two fields contain the start address of the buffer that contains the message. LADR contains the least significant 16 bits, and HADR contains the remaining 8 high order bits.
	Control and Status Information
ENP	If set, Indicates that this is the last buffer used to hold a packet. If STP is also set, the complete packet is in this buffer.
STP	If set, indicates that a packet starts in this buffer. If both the ENP and STP bits are set, the buffer contains a complete packet.
DEF	If set, the LANCE had to defer the transmission of the packet because some other node was already using the LAN.
ONE	If set, indicates that exactly one retry was necessary to send the packet.
MORE	If set, indicates that more than one retry was necessary to send the packet.

Table 15-2 (cont.)

Field Name	Meaning and Use
ERR	If set, indicates that one of the error bits -- RTRY, LCAR, LCOL, UFLO, and BUFF -- is set. So, it provides a summary of any error condition, with the detailed information being in TMD3.
TDR	The LANCE has an internal counter that measures the time between starting a transmission and the occurrence of a collision. This field contains the value of this counter if a collision occurs. The value can be used to determine the approximate distance to a cable fault. In the DEC MicroServer, this counter is inaccessible to the customer or Field Service Engineer.
RTRY	When set, indicates that the LANCE could not send a packet because of repeated collisions. The number of retries that the LANCE will have attempted is determined by the value of DRTY in the Initialization Block's MODE field.
LCAR	If set, indicates that the LANCE lost the presence of the carrier signal while transmitting a packet. In this case, the LANCE does not retry to send the packet.
LCOL	If set, indicates that a collision occurred after the slot time for a channel had passed. When this error occurs, the LANCE does not retry the transmission.

Table 15-2 (cont.)

Field Name	Meaning and Use
UFLO	If set, indicates that the LANCE truncated a message. When sending a message, the LANCE takes information from the buffers and queues for transmission through its internal silo. Until detecting the end of the packet, the LANCE has to keep information in the silo, so that transmission can be continuous. If the silo becomes empty, the message is terminated, and the LANCE sets this bit. One cause of this error can be that the next buffer is not available when the LANCE needs it.
BUFF	If set, indicates that the LANCE completed processing of the current buffer, and the next in the chain was not available -- that is, the OWN bit is not set. In this case, the LANCE will continue to send information until its internal silo is emptied, and then the UFLO bit is also set. This error can also occur if the LANCE owns the next buffer, but cannot read the status information before the silo is empty.
Descriptor Ownership Information	
OWN	The setting of this bit shows who has ownership of the descriptor (and hence of the buffer). When the bit is clear the MicroVAX has ownership, and when it's set the LANCE has ownership.
Buffer Size and Message Size Information	
BNCT	This field indicates how many bytes of information there are in the buffer. The LANCE sends this number of bytes, starting from the beginning of the buffer.

15.3.3 Buffer Chaining

Some messages will be too large to fit into a single buffer. So the message is split up and spread over a number of buffers. The first buffer will have the STP bit in its descriptor set, and the last will have its ENP bit set. Descriptors for intermediate buffers have neither bit set.

To keep messages in the right order, a buffer chain has to use sequential entries in the appropriate descriptor ring. Also, buffers have to be claimed on a one-by-one basis.

Normally, this presents no problems, but occasionally buffer availability errors may occur. Here, the LANCE wants to access or claim the next buffer in a chain but the buffer's OWN bit is not set. In these cases, the BUFF bit in the previous buffer's descriptor is set and the I/O transfer terminated.

Buffer problems occur because the LANCE is transferring information faster than the MicroVAX can process it. Thus, these errors would normally occur only on a very heavily loaded system.

15.4 Synchronous Ring Descriptors

All the synchronous rings are in a 16K byte block of the Buffer RAM. This block is aligned on a 1K byte boundary. Figure 15-4 shows the order of the rings in the block.

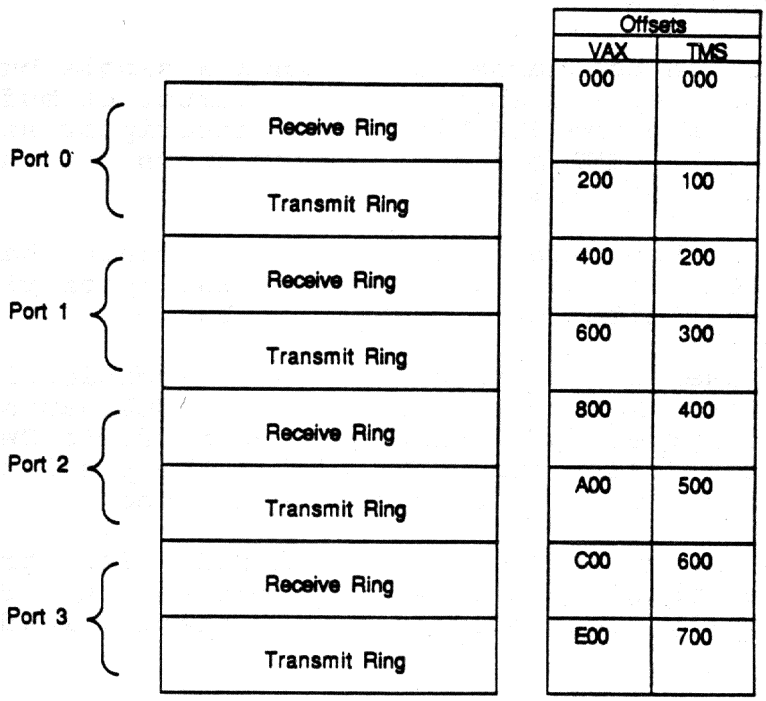


Figure 15-4: The Synchronous Ring Block

The figure shows the relative start address of each descriptor ring. There are addresses for both the MicroVAX processor and the TMS processor (which uses a different addressing scheme, see Chapter 6).

The descriptors are all eight words long, and the following sections show their format.

15.4.1 Receive

Figure 15-5 shows the format of the receive descriptor. Table 15-3 contains more information on each of its fields.

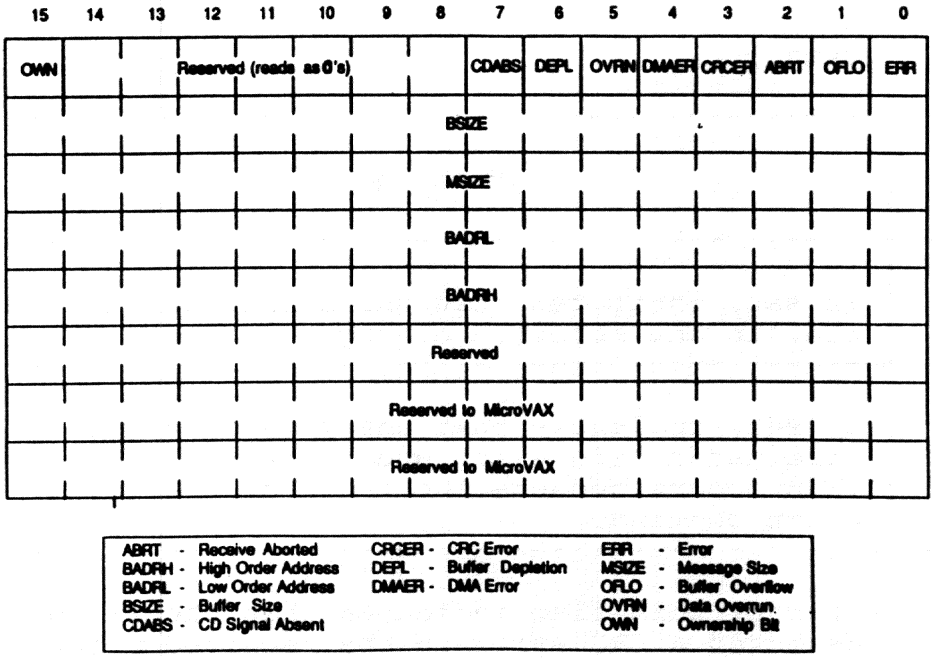


Figure 15-5: Format of a Descriptor for a Synchronous Receive Buffer

Table 15-3: Fields in the Synchronous Receive Buffer Descriptor

Field Name	Meaning and Use
	Buffer Address Information
BADRL BADRH	These two fields contain the start address of the buffer to receive a message. BADRL contains the least significant 16 bits of the address, and BADRH contains the 16 high order bits.
	Status Information
ERR	If set, indicates that one of the error bits -- OFLO, ABRT, CRCER, DMAER, and OVRN -- is set. So, it provides a summary of any error condition.

Table 15-3 (cont.)

Field Name	Meaning and Use
OFLO	If set, indicates that the buffer was too small to contain the received message.
ABRT	If set, indicates that the reception was terminated before the end of message was received. This occurs because the receiver was given an abort command.
CRCER	If set, this flag indicates that the I/O port detected a CRC error in the received message.
DMAER	If set, this flag indicates that the message reception could not complete because of a DMA error when transferring information to the buffer. Use of this flag is reserved.
OVRN	If set, this flag indicates that data overrun occurred on the receive line.
DEPL	If set, indicates that the TMS processor could not obtain a receive buffer (either for the current packet or for a previous packet).

Table 15-3 (cont.)

Field Name	Meaning and Use
CDABS	If set, indicates that the modem signal CD was absent when the TMS processor received the first byte of a DDCMP message. This field is used only when the DDCMP protocol is selected. It allows the system software to conform with DEC Standard 52 and to start up lines connected to DMR devices. Note, however, that the FIFO on a receive line holds data only. The modem signals present as each byte is received are not put into the FIFO. So if the FIFO contains a large amount of data, this field can be set incorrectly because the modem signal has been turned off before the first byte has travelled through the FIFO.
Descriptor Ownership Information	
OWN	The setting of this field shows who owns the descriptor (and hence the buffer). When clear, the MicroVAX has ownership, and when the field is set the TMS processor has ownership.
Buffer Size and Message Size Information	
BSIZE	The size of the available buffer, in bytes.

Table 15-3 (cont.)

Field Name	Meaning and Use
MSIZE	The number of bytes in the received message. The value of this field is the number of bytes processed by the TMS processor. When using the HDLC protocol, this value does not include the data check characters because the DUSCC automatically removes them. In the DDCMP protocol, however, the DUSCC does not remove the check characters and so the value of this field includes these characters.

15.4.2 Transmit

Figure 15-6 shows the format of the transmit descriptor. Table 15-4 has more information on each of the fields.

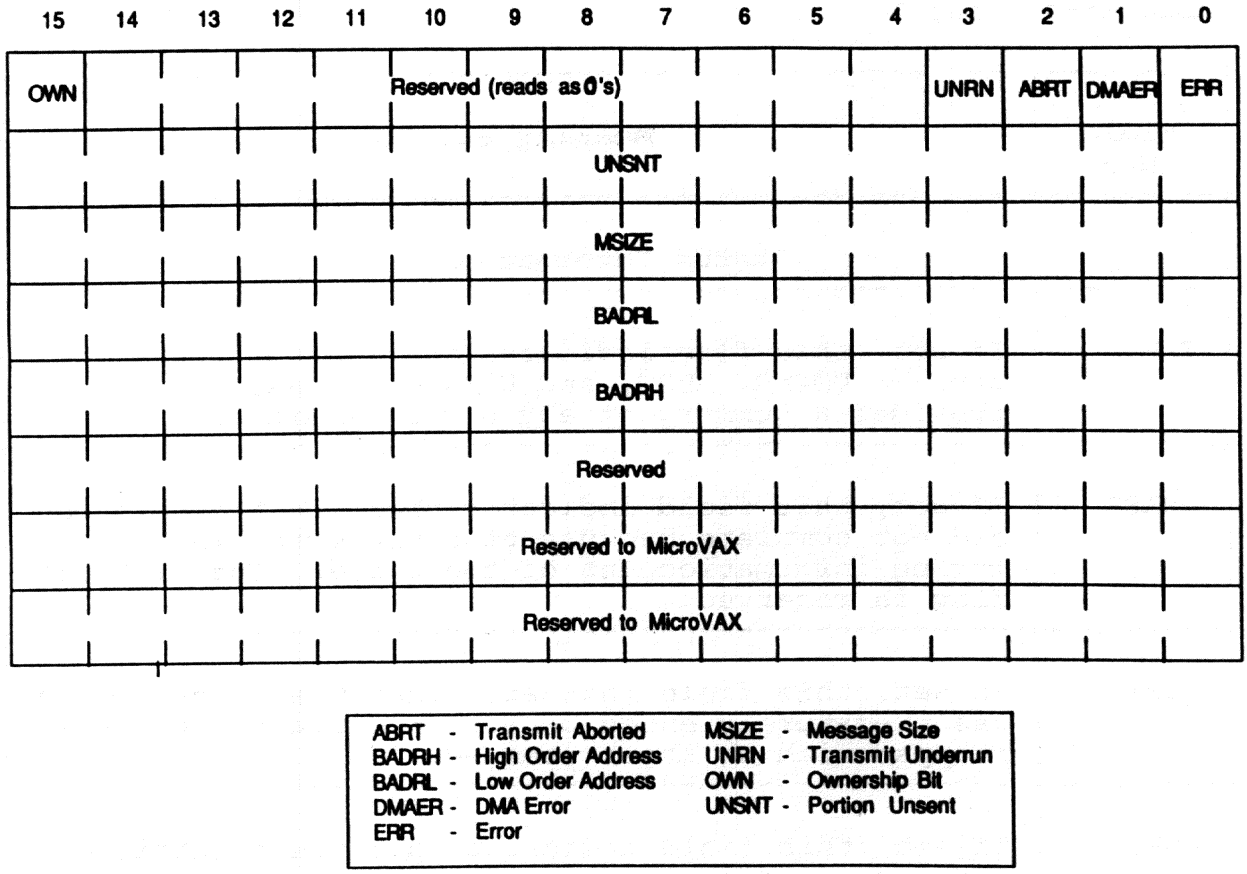


Figure 15-6: Format of a Descriptor for a Synchronous Transmit Buffer

Table 15-4: Fields in the Synchronous Transmit Buffer Descriptor

Field Name	Meaning and Use
	Buffer Address Information
BADRL BADRH	These two fields contain the start address of the buffer holding the message to be transmitted. BADRL contains the least significant 16 bits of the address, and BADRH contains the 16 high order bits.

Table 15-4 (cont.)

Field Name	Meaning and Use
	Status Information
ERR	If set, this flag indicates that one of the error bits -- DMAER, ABRT, and UNRN -- is set. So it provides a summary of any error condition.
DMAER	If set, this field indicates that the transmission did not complete because of a DMA error when moving information out of the buffer. Use of this flag is reserved.
ABRT	If set, this field indicates that the transmission was prematurely ended because the transmitter was stopped with an abort command.
UNRN	If set, this field indicates that the transmitter underran, indicating a hardware problem.
	Descriptor Ownership Information
OWN	The setting of this field shows who owns the descriptor (and hence the buffer). When clear, the MicroVAX has ownership, and when the field is set, the TMS processor has ownership.

Table 15-4 (cont.)

Field Name	Meaning and Use
	Buffer Size and Message Size Information
UNSNT	This field contains useful information only when the DMAER bit is set. In this case, this field contains the number of characters in the message that were still to be sent when the DMA error occurred. Use of this field is reserved.
MSIZE	The number of bytes in the message to transmit. This value does not include the two CRC characters that the TMS processor adds to each message.

15.4.3 Buffer and Descriptor Use

The synchronous descriptors are less complex than the LANCE ones because there is no buffer chaining. Instead, a message always uses just one buffer. However, the MicroVAX must make sure that receive buffers are large enough to hold any incoming message.

Some of the words in each descriptor are allocated to the MicroVAX. The TMS processor never uses these locations, and so the MicroVAX can use them for anything it wishes (for example, to hold temporary values).

There are some differences in buffer contents between the DDCMP and HDLC protocols.

For DDCMP:

- **Transmit** -- The message in the buffer contains a packet header and the message data. The header needs to include space for its CRC, but not a CRC value itself (the TMS processor calculates this). The data part does not need space allocated for the CRC value as the TMS processor automatically adds this when sending the message.

- **Receive** -- The buffer supplied by the TMS processor contains a complete packet, excluding the CRC values (though including a space or place holder for them). These place holders are returned whether or not a CRC error was detected during the reception.

For HDLC:

- **Transmit** -- The buffer contains the complete packet, excluding the packet CRC. The TMS processor will add this value.

- **Receive** -- the buffer should be large enough to contain the packet CRC as well as the rest of the packet. However, due to shortcomings in the DUSCC these locations will not be used. Later versions of the DUSCC may be able to provide the CRC and the time by which the locations will be needed.

15.5 Ring Headers

Both LANCE and synchronous rings are pointed to by a control block (called a ring header). This is usually part of another data structure and allows the appropriate device or firmware to locate the start of the rings.

For the LANCE, the rings are pointed to by the last two fields in the LANCE initialization block. For the synchronous ports, the rings are pointed to by the second longword of the Synchronous I/O Control Block. See Chapters 5 and 6 for more information on these data structures.

16.1 Overview of Structure and Content

The Non-Volatile RAM (NVRAM) is an 8K byte memory used to hold system parameters, error information, and similar information. The memory has a number of fixed-length fields. Table 16-1 shows the field names and the locations in the memory that they use. Section 16.2 gives details of the content of each field.

Table 16-1: Fields in the NVRAM

Field Name	Locations (Hex)	Description
Initialization Flag	0	Indicates whether the NVRAM has been initialized
Identification	2 to F	Identifies the NVRAM as being formatted for the DEC MicroServer
Error Log	10 to 5F	Contains the OBT's error log
Test Locations	60 to 63	Locations used by the OBT
Error Checksum	70	An error check used to verify the Last Error field
Last Error	74 to A8	Detailed information on the most recent error detected by the OBT

Table 16-1 (cont.)

Field Name	Locations (Hex)	Description
Boot Number	A9 to AA	The counter of the number of times the system has been rebooted
Boot Checksum	AB	An error check used to verify the value of the boot number
Serial Number	100 to 11F	The serial number of the unit
DYRC Information	124 to 12B	Copies of DYRC registers from the last machine check
Parity Errors	12C to 12F	Number of parity errors detected in the system
Last Parity Address	130 to 133	Address of the last parity error
Last Machine Check	134 to 19B	Information on the last unexpected machine check that occurred
Ethernet Address	19C to 1A1	Ethernet address of the unit when initialized
Software ID	1A2 to 221	Software ID that can be used to load the DEC MicroServer software

Table 16-1 (cont.)

Field Name	Locations (Hex)	Description
Service Password	222 to 229	Service password used to deter unauthorized access through the remote console
Software Parameters	22A to 231	The parameters returned by the ROM$GET_PARAMS system support routine
Firmware Options	232 to 239	Set operating options of the firmware
ECO Information	23A to 27A	Contains any ECO information entered
Firmware ID	27B to 2F1	Enables an alternative version of the firmware to be loaded
Part Number	300 to 30F	Reserved to hold a part number for the unit. Currently, only the variant and revision fields are used
Part Variant	309 to 30A	The unit's variant number
Part Revision	30C to 30F	The unit's revision number
Software Usage	400 to 1FFF	Reserved for use by the application software

NOTE

When accessing the NVRAM from the MicroVAX processor, use these values with care. See Section 16.6.

16.2 Details of Field Contents

Table 16-2 gives details of what each field contains.

Table 16-2: Contents of the Fields in the NVRAM

Field	Content and Format
Initialization Flag	Shows the version of the NVRAM layout in use. If this field contains 0, the NVRAM has not been initialized. In the initial product, this field contains 1, and each time the format is changed the value increases by 1.
Identification	Contains the word DEMSA in ASCII text. This shows that the NVRAM belongs to the DEC MicroServer.
Error Log	Contains the summary of errors that the OBT maintains. The log contains 10 entries, each containing: o Error Code (2 bytes) o Number of Occurrences (1 byte) o Checksum containing the two's complement sum of all other bytes in the entry (1 byte) o Boot number of the last occurrence (2 bytes) o Reserved space (2 bytes)
Test Locations	This field is set aside for use by the OBT. These are the only locations that it uses.

Table 16-2 (cont.)

Field	Content and Format
Error Checksum and Last Error	These fields contain the details of the most recent error that the OBT has detected. The Last Error field contains a copy of the MicroVAX registers R1 to R11 as they were when the error occurred. See Section 12.2.7 for more information on how to use this information. The Error Checksum field contains the two's complement sum of all the bytes in the Last Error field.
Boot Number and Boot Checksum	When the unit is sent to a customer, the Boot Number field contains zero. Then its value increases by one each time the system is booted. The value increases irrespective of the way that the system is started (power up, remote console, NCP, or pressing the DUMP switch). The Boot Checksum field contains the two's complement sum of the bytes in the Boot Number field.
Serial Number	This field contains the unit's serial number, stored as a counted ASCII string. This field does not have to be filled but can help when diagnosing problems.

Table 16-2 (cont.)

Field	Content and Format
DYRC Information	To help trace a problem (whether hardware or software) it is often useful to know the values in the unit's DYRC registers when a machine check occurs. This field is divided into four words that contain: o System RAM's CSR o System RAM's FAR o Buffer RAM's CSR o Buffer RAM's FAR
Parity Errors and Last Parity Address	The Parity Errors field contains a count of the number of parity errors detected in the RAMs since the NVRAM was initialized. The Last Parity Address field contains the page number of the last parity error that was detected.

Table 16-2 (cont.)

Field	Content and Format
Last Machine Check	This field contains some of the system's register values as they were when the last machine check occurred. The software can enter this information using the ROM$MACHINE_CHECK service routine. This field contains: o The value 8 as a longword. This is the number of words in the rest of the information. o The value of the LANCE's CSR0 (2 bytes) o The value of the System CSR (2 bytes) o The value in the CSR of the Buffer RAM's DYRC (2 bytes) o The value in the FAR of the Buffer RAM's DYRC (2 bytes) o The value of the PC (4 bytes) o The value of the PSL (4 bytes)
Ethernet Address	Contains the Ethernet address that the unit had when the NVRAM was initialized. See Section 17.4.

Table 16-2 (cont.)

Field	Content and Format
Software ID	This field is reserved for a software ID that can be used during the process. Normally, the system will broadcast a message to load hosts asking for any image to be loaded. However, the unit can ask for a specific image using the contents of this field. NOTE The initial software products do not use this feature. However, it is provided for future use, and is larger than current DECnet implementations allow, also for future expansion.
Service Password	This field contains the service password that a user must quote when: o Connecting a terminal as the remote console o Using the NCP commands BOOT and LOAD The password is held as a 64-bit number that the user expresses in hexadecimal. A value of 0 indicates that there is no password.
Software Parameters	This word contains two longwords of data that a running system can access through the service routine ROM$GET_PARAMS. The content and meaning of this field is software dependent.

Table 16-2 (cont.)

Field	Content and Format
Firmware Options	This value of this field sets the way that the VAX firmware operates. The field is a bit mask, with one bit allocated to each feature. Setting a bit enables the appropriate feature. The bits defined in this field are: 0. SNAP SAP processing for MOP messages. 1. Switch controlling whether RAM versions of the firmware can be used. 2. Determines whether the SPECIAL console command can be used from a remote console. 3. Determines whether MOP V3.1 is to be used. 4. Determines whether MOP V4.0 is to be used. By default, the firmware uses MOP V3.0, unless bits 3 or 4 in this field are set. Normally, if bit 4 is set, bit 0 will be set as well.
ECO Information	This field contains a counted string of ECO information that can be read through the SHOW VERSION console command. Typically, this field is used to summarize changes made through ECOs or to record the current ECO level of the DEC MicroServer.
Firmware ID	This field can be used to autoload a RAM version of the VAX firmware that the system will then use in preference to the ROM version. See Chapter 13 for an explanation of RAM-based firmware.

Table 16-2 (cont.)

Field	Content and Format
Part Number	This field is reserved for the part number of the unit. In the initial release of the product only the VARIANT and REVISION fields of that number are used. Both the VARIANT and REVISION fields contain two characters determined by the DEC standards for part numbers.
Software Usage	The remainder of the NVRAM is reserved for use by the communications software.

16.3 Clearing Fields in the NVRAM

Some fields in the NVRAM can be cleared when the DEC MicroServer is powered up while the DUMP switch is held in. These fields are:

1. Software ID
2. Service Password
3. Firmware Options

The Software ID and the Firmware Options fields will be cleared only if the NVRAM has not been initialized, that is, if the Initialization Flag and the Identification fields do not contain the correct information.

16.4 Initializing the NVRAM

Before units are sent to the customer, the NVRAM needs to be initialized. A suggested initialization procedure is:

1. Clear the Initialization Flag and the Error Log

2. Power up the DEC MicroServer

3. Set up the NVRAM

The following sections explain these steps in more detail.

16.4.1 Clear the Initialization Flag and Error Log

The initial contents of the NVRAM will be random. To prevent the unit from behaving strangely when powered up, it is essential to clear the Initialization Flag. This is probably best done during board testing procedures.

Also, clear the locations allocated to the Error Log so that they can be used during system testing.

16.4.2 Power Up the DEC MicroServer

Power up the DEC MicroServer while holding in the DUMP switch. This causes the firmware to clear the Software ID, Service Password, and Firmware Options fields.

16.4.3 Set Up the NVRAM

Initialize the contents of the NVRAM by using the following SPECIAL console commands:

- INIT_NVRAM
- ECO -- to enter any ECO information
- REVISION -- to set the revision field of the part number
- VARIANT -- to set the variant field of the part number
- SERIAL -- to set the unit's serial number
- OPTION 2 -- to disable use of the SPECIAL command from remote console.

16.4.4 Reinitializing an NVRAM

To reinitialize a unit, you can follow the same procedure, but a physical rather than remote console must be used. Alternatively, a program running on the VAX processor can be used. This program can use the ROM$DO_COMMAND service routine to execute the commands.

16.5 Access from the Remote Console

Once initialized, the only access to the NVRAM from the remote console is through these commands:

- SHOW VERSION
- SHOW ERRORS
- SHOW PASSWORD
- SHOW MCHECK
- SHOW SOFTWARE
- SHOW FIRMWARE
- SET PASSWORD
- SET SOFTWARE
- SET FIRMWARE
- CLEAR MCHECK
- CLEAR SOFTWARE
- CLEAR FIRMWARE

Any other access needs to be made through the SPECIAL command and a physical console.

16.6 Access from the MicroVAX Processor

Table 16-1 lists byte offsets in the NVRAM. However, the memory does not appear to the MicroVAX processor as sequential bytes. Each byte appears on a long word boundary, and so the offsets in Table 16-1 have to be treated as longword offsets.

This also means that the MicroVAX cannot directly read a multi-byte value from the NVRAM. Instead, it must read each byte in turn and build the value in some scratch area (such as a register).

PART IV

TROUBLESHOOTING

Chapter 17
Circuit Board Troubleshooting

17.1 Introduction

The DEC MicroServer has a number of diagnostic tests contained in the on-board ROM, which are run by the MicroVAX processor. These tests will discover any hardware failure on the board. The next page contains a list of the diagnostic tests: this list is in the order of execution. The full On-Board Test (OBT) is made up from these tests run in the sequence indicated.

The diagnostics start with testing hardware that is "closest" to the MicroVAX. That is, the system components requiring the minimum of other support components for correct operation. As the diagnostics progress, the sequence of subsystem testing is as follows:

1. The MicroVAX bus subsystem
2. The I_O bus subsystem
3. The SCC bus subsystem

The sections in this fault-finding chapter are presented in the order in which the diagnostics execute. Hence guides to solving problems in the earlier parts of the OBT will be found in the first section, whilst problems discovered in the Exerciser test will be covered in a later section.

The final two sections (Sections 17.7 and 17.8) provide a guide to useful locations to probe the major control signals on DEC MicroServer, a cross-reference between some of the more common error codes (provided by the OBT), and the section in which the debug of that type of error is covered.

All references to signals or areas of circuitry to fault diagnose are accompanied by a drawing reference. Each signal name has a postfix 'H' or 'L' which indicates the asserted sense of the signal. For example, the signal 'IOR1_IOBWR L' (which is sourced on drawing page IOR1) is asserted in the LOW sense and deasserted in the HIGH sense.

Note that in the sections where you are asked to use the console test commands, and test utilities such as XDT and TDT, the terminology **type** and **enter** have a particular meaning. **Type** means you simply type in the text shown and DO NOT press <RETURN> when you have finished. **Enter** means you type in the text shown and DO press <RETURN> when you have finished.

Table 17-1 lists the test numbers, the tests, and references to the sections in this chapter containing fault finding information.

Table 17-1: Test Numbers, Components Tested, and Fault-Finding References

Special Test Number (Hex)	Test	Fault-Finding Section
1	Firmware ROM CRC test	MicroVAX Bus Section 17.2
2	MicroVAX DRAM and display test	MicroVAX bus Section 17.2
3	MicroVAX CPU test	MicroVAX bus Section 17.2
4	VIC test	MicroVAX bus Section 17.2
5	NVRAM test	MicroVAX bus Section 17.2
6	Ethernet PROM test	MicroVAX bus Section 17.2
7	CSRs test	MicroVAX bus Section 17.2

Table 17-1 (cont.)

Special Test Number (Hex)	Test	Fault-Finding Section
8	10 ms timer interrupt test	MicroVAX bus Section 17.2
9	Watchdog timer test	MicroVAX bus Section 17.2
A	I/O DRAM and display test	I_O bus Section 17.3
B	Memory management register test	I_O bus Section 17.3
C	Memory management functional test	I_O bus Section 17.3
D	LANCE CSR test	I_O bus Section 17.3
E	LANCE internal loopback test	I_O bus Section 17.3
F	LANCE/SIA Ethernet loopback	I_O bus Section 17.3
10	MicroDMA controller test	SCC bus Section 17.4
11	TMS SRAM test	SCC bus Section 17.4

Table 17-1 (cont.)

Special Test Number (Hex)	Test	Fault-Finding Section
12	DUSCC test	SCC bus Section 17.4
13	Internal data test	SCC bus Section 17.4
14	Receive serial FIFOs test	SCC bus Section 17.4
15	Modem status test	SCC bus Section 17.4
16	Synchronous external data loopback test	SCC bus Section 17.4
17	TMS 32020 test	SCC bus Section 17.4
18	Exerciser test	Exerciser Section 17.5
1D	Manufacturing refresh logic test	MicroVAX bus Section 17.2

17.1.1 Equipment Level Required

The fault diagnosis described in the remainder of this chapter assumes you have at least a two-channel, high-speed oscilloscope. More in-depth diagnosis, as described in Section 17.5 for example, is performed best by a logic analyzer such as a Hewlett-Packard 1650 or 1631.

17.2 MicroVAX Bus and Basic System Faults

The title of each section indicates the fault discussed within that section.

17.2.1 At Power Up the LED is Blank

If the LED (including the decimal point) does not light, the most likely causes are:

- Some of the system power cables are not correctly inserted
- The power on the voltage rails is below specification

Do the following:

1. Check the multi-way power cable between the PSU and the main etch.
2. Check that all the smaller cables between the main etch and distribution panel are in place. These cables pass power to the distribution panel and display information to the LED.
3. Check that the 7-segment LED is firmly in its socket.
4. Check the DC levels on the Voltage rails:
   - +5.1 Volts - (+/- 5%)
   - +12.1 Volts - (+/- 5%)
   - -12.1 Volts - (+/- 5%)
5. If all these power rails are within specification, check if DCOK is asserted. Referring to drawing PSU1, DCOK can be monitored at E143 pin 5. If DCOK is not asserted, then suspect the PSU.

17.2.2 Processor Subsection

If the VCC rail is held at 5 V, and the decimal point on the display lights but the first element (top) LED still fails to light, suspect fundamental fault on processor bus somewhere. The

OBT code must be getting stopped at some point before the first LED update occurs.

If the DMR line to the MicroVAX is asserted (low), the MicroVAX will not be able to perform any instructions. Check DMR is high, E115 pin 18 (refer to drawing CPU1). DMR is not used in DEC MicroServer, so it is pulled high by a 10K resistor.

At power up or external reset to the MicroVAX, the MicroVAX performs a series of four EPS cycles. The EPS cycles are executed by the MicroVAX to find out whether the system comprises a floating point unit - DEC MicroServer doesn't, so there is no response to these four cycles. This is the only point in normal operation at which the MicroVAX will issue this EPS burst.

17.2.2.1 No EPS Activity on Power Up. - If there is no EPS activity from the MicroVAX at power up (drawing CPU1 shows EPS as E115 pin 23) then there is a fundamental fault. Perhaps the MicroVAX is faulty or the processor is not receiving a system clock.

17.2.2.2 Periodic EPS Activity on Power Up. - Possible causes of continuous EPS activity in order of likelihood are:

- Power on, RESET asserted
- HALT asserted
- Halt instruction (all zeroes) executed
- Machine check whilst in machine check
- SCB vector bits <1:0> = 11
- SCB vector bits <1:0> = 10

If EPS is being asserted periodically, the MicroVAX is being restarted due to one of the above reasons. The MicroVAX should only assert EPS during the power up and reset sequences.

1. Check if the Halt line is periodically being driven low
2. Check if the Watchdog timeout is occurring

If the DAL is suffering from any open or short circuits, the MicroVAX could read a HALT instruction off the bus (all 0's); this only needs to be a single byte. This could be reached from

an incorrect Jmp or Branch instruction from a previous incorrect ROM read. The Halt line can be driven low by the external hardware console, as can the Reset line.

Approximately 7 microseconds from power-up or reset, after the EPS cycles, the first address strobe should occur.

Possible situations at this point are:

1. Address strobe asserts, and remains asserted
2. Address strobe asserts, deasserting 25 microseconds later
3. Address strobe asserts, deasserting ~285 nanoseconds later
4. Address strobe asserts, deasserting ~485 nanoseconds later

- If case 1 is the situation, then neither RDY or ERR are being returned to the MicroVAX. Check to see if the ROMRD signal is asserting, showing the address to be decoded is correct. ROMRD is decoded by the UVDEC PAL (refer to drawing UVR2) and is asserted at E63 pin 14. Make sure that address strobe, UVBAS, is reaching E63 pin 3.
- If case 2 is the situation, then the decode did not function, but the Error logic saw the bus cycle start. The RDY logic did not see the ROMRD and therefore did not assert RDY. The PAL UVDEC (refer to drawing UVR2) returns ready for the ROM accesses. ROM accesses should be slow accesses and SLOWRDY should be returned to the MicroVAX. Check the PAL output at pin 12. If this is asserting then check the Ready delay chain on page UVR3. SLOWRDY is gated by E110, then delayed four times by four F175 elements clocked at 20 MHz. Check the output of each stage and the clock input (E136 pin 9). Ready is finally returned to the MicroVAX through the S09 E80 at pin 6.
- If case 3 is the situation, then the MicroVAX bus cycle is being terminated early. If the MicroVAX is accessing the ROM correctly then the cycle time should be 600 nanoseconds, not 400 nanoseconds. Check if the RDY line to the MicroVAX is permanently asserted or being asserted early (RDY should assert between <ASL+160 nanoseconds>, and <ASL+305 nanoseconds> for a single slip cycle). If the cycle is being terminated by a

FASTRDY, then it is possible that the decode circuitry is not decoding the ROM cycles correctly. A UVDRAM access would produce FASTRDY.

- If case 4 is the situation, then the MicroVAX is reading ROM, and should be executing the diagnostic startup. If this is the case, and the EPS cycles are repeating periodically, then it is possible that the ROMs have been either incorrectly blown, or they are inserted incorrectly. This is likely to produce a HALT.

17.2.3 MicroVIC Related Faults

If the diagnostics report a MicroVIC unsolicited interrupt fault then experience has shown that the most likely cause will be that the Daughter Board is either not connected at all, or the Daughter Board connector is not making good contact with one of the two boards. Push Daughter Board firmly down onto connector making sure of correct alignment with connector and main board.

When the daughter board is not connected to the main board, the X21 interrupt line into the MicroVIC floats in the asserted sense. Connecting the daughter board de-asserts the interrupt.

It is possible to debug the main etch up to (but not including) the SCC bus, without connecting the daughter board by tying this X21 interrupt to its deasserted sense, high. This can be achieved by a wire connecting pin 17 of the MicroVIC (E120) to a Vcc point.

17.2.4 UVDRAM Related Faults

The UVDRAM (MicroVAX DRAM) is controlled by the UVDYRC (Dynamic RAM controller) which performs RAS/CAS timing into the DRAM and also RDY/ERR handshake with the MicroVAX processor. The UVDYRC also performs refresh timing into the DRAM.

Possible faults in this circuitry are:

1. Bus-Timeouts whilst addressing UVDRAM
2. Stuck-at 0/1 and cross-coupling
3. UVDRAM parity reporting

4. Incorrect refresh operation

These four fault types are covered in the next four sub-sections.

17.2.4.1 Bus-Timeouts Whilst Addressing UVDRAM - If the MicroVAX cannot access ANY of the UVDRAM then a machine check will occur. The diagnostics will then attempt to write the machine check frame (containing information about how the machine check happened) to the UVDRAM which will cause another machine check. The MicroVAX will therefore see a double machine check and have an Unexpected System restart. The diagnostics will then continue trying to access the UVDRAM, causing further double machine checks. This looping action is useful when trying to discover the hardware fault and lets you probe the relevant signals continuously without looking for single-shot events. The signals to probe are identical to those detailed below for machine checks produced by failure to access single locations.

If, for some reason, the MicroVAX machine checks on a location in the UVDRAM but is still able to access the part of UVDRAM used to store the machine check frame a C100 error will be reported.

The OBT will report the address at which it thought the machine check occurred in register R6. This may not actually be the true machine check address. The reason for this is that the MicroVAX employs a 'pipelined' architecture which pre-fetches addresses whilst executing from the present address. The OBT will sometimes report the pre-fetched address. It is therefore a good idea to take a longword (4 hex) off the address in R6.

Hit the Break key on the hardware console. Set up a repeated read from the address specified in register 6 (minus 4). If register 6 contained 604 for example, you would type these two commands:

SPECIAL XDT
600/

This would perform a read of the above location, a machine check should occur - this will be reported as an **XDT TRAP**. A repeated read from this location can be performed by now typing **"R"** (for 'repeat').

Now check that the following are happening:

1. Address strobe at the UVDYRC - E88 pin 57.

2. UVRAMSEL - pin 15 off the F138 decoder E64 on drawing page UVR1 should be asserting during address strobe fall time, this is the chip select to the UVDYRC. This signal should also appear on pin 54 of the DYRC device E88.

3. Look to see if the DYRC is responding to the chip select. Pin 24 on the DYRC is the Ready output and should be asserting. If it is, then Ready is probably not getting back to the MicroVAX.

4. Look at pin 19 on the MicroVAX (E115), to see if Ready is reaching the MicroVAX.

17.2.4.2 Stuck-At 0/1 and Cross-Coupling - These errors will cover stuck at zero and stuck at one type faults (that is, shorted to Gnd or Vcc) which can affect either address or data, or cross-coupling between address or data lines which could mean that UVMEM<15> would take up the same value as UVMEM<16> on a bus operation just for example. Cross coupling would be as a result of a short between DAL lines.

An error of any of these types would be discovered in the UVDRAM test (test2). By going into **SPECIAL XDT** and performing a series of reads and writes, the exact nature of the error can be determined. Follow these steps:

1. Determine the address that the faulty access took place from, this will be found in register R2 or R7 in the error log depending on whether the OBT failed with error code 2800 or 2900.

2. Hit Break on the hardware console and enter the command:

   SPECIAL XDT

   This command will invoke the debugging tool, XDT. You can now read locations by postfixing an address with a backslash (/), and can write to locations by adding the write data to the same line and pressing RETURN.

3. Perform a write of all 'F's to the address in the error log. For example:

   810/12345678 FFFFFFFF

where 12345678 is the original contents of address location 12345678 hex and FFFFFFFF is to be the new contents.

4. Now read back the new contents. If they are anything other than all 'F's then there is a stuck-at-zero fault into this location.

5. Perform a write of all '0's to the above address. For example:-

   810/12345678 00000000

   Where 12345678 is the original contents of address location 12345678 hex and 00000000 is to be the new contents.

6. Now read back the new contents. If they are anything other than all '0's then there is a stuck at one fault into this location.

7. Write all '5's or all 'A's to the above location. Perform a read from the same location. If the answer is not all '5's or all 'A's respectively and the 'F's and '0's tests both passed, then there is a data coupling fault. Analysis of the read back data will show which of the data lines are coupling.

   For example if '55555555' was the write data and '55555557' was read back, then bit <1> could be coupling with one of its 'neighbour' bits <0> or bit <2>. Alternatively, it could be coupling with any of the other even (that is, HIGH) bits.

If a data corruption occurs due to a hardware fault, it is likely to be a 'stuck at' fault or a cross-coupling fault as detailed above. The MicroVAX operates a multiplexed address/data bus - UVDAL <31:00>.

There is one set of bus transceivers between the MicroVAX and the RAM SIPS which comprise the UVDRAM. These are E112, E106, E111 and E105. The outputs of these transceivers are a good place to look for the effects of a fault of the above type. A continuity tester will confirm a short to Vcc or GND or a cross-coupling fault.

It is unlikely that the fault will be on the UVDAL lines as these are used to access the ROMs at the start of the OBT to perform the ROM CRC check. More likely is a fault on the UVMEM lines or the UVRADD lines.

Check the enable and direction pins on all the transceivers. The enable, UVMEMCYC is decoded through the UVDEC PAL. Check that address strobe is present at the input to this PAL, and that UVADD<24,27> are in the correct sense at address strobe assertion time for the PAL to decode a UVDRAM memory access cycle.

A 'stuck at' or cross-coupling fault in the address will be slightly harder to find. UVDAL<0:19> are taken to the DYRC E88. The DYRC then performs some address multiplexing presenting nine bits of row address information on RADDUV <8:0> with RAS assertion, and then a further nine bits of column address with CAS assertion. These nine address lines are fed to the RAM SIPS through sections of the Octal Drivers E99 and E100.Continuity tests should therefore be performed both at the inputs and outputs of these drivers.

Check that all the enable pins to the Octal Drivers - E99 pins 1, 19, and E100 pin 19 - are low. Refer to drawing page UVR1 for details of these pins.

Also check that the CAS lines are being enabled to the memory devices. The signal WROK enables the octal driver E100 (refer to drawing UVR1) at pin 1. WROK is generated by the PAL UVERR (E78) on page UVR2.

17.2.4.3 UVDRAM Parity Errors - If an error occurs (ERR is asserted) in a UVDRAM access cycle, then a parity error is the likely cause assuming the error is not due to a bus timeout. If a parity error does occur, the DYRC (E88 drawing page UVR1) should still have produced RDY and the memory access will have completed. In this part of the DEC MicroServer circuitry, RDY is produced early because the MicroVAX requires RDY early relative to the data being available. Parity errors are reported to the MicroVAX in a subsequent bus cycle assuming parity is enabled.

Bit 14 in the DYRC CSR (Control and status register) controls WWP - write wrong parity. When WWP has been set and the parity enable bit, bit 13 in the DYRC CSR, has been set, and write operation followed by a read operation has been performed, a parity error will be forced. This should produce a machine check.

There are two kinds of parity errors that could be encountered whilst executing the OBT:

- Parity error occurs with WWP disabled, causing Machine checks.

- Parity error does not occur with WWP enabled

Sections 17.2.4.3.1 and 17.2.4.3.2 describe these two error types.

17.2.4.3.1 Machine Checks Due to UVDRAM Parity Errors

The first of these two errors could occur in any of the OBT tests that access the UVDRAM. A C100 error will be flagged which means that an unsolicited machine check has happened. Inspection of R11 would reveal bit 23 set, indicating a MicroVAX bus parity error. R6 would show the address from which the parity error was sourced. Again, due to the pipelined architecture of the MicroVAX, explained in section 2.4.2, you should take 4 (a longword) from this address to give the most probable source of the parity error.

Things to check:

1. Parity must be re-enabled in the DYRC CSR. After the DYRC has reported a parity error, it disables parity. Parity can be re-enabled by setting bit 13 in the DYRC CSR (address 24000000 accessed by word width).

2. Go into XDT from the hardware console and enable word-width transfers by entering the command:

   2S

3. Re-enable the parity reporting mechanism in the DYRC by writing 2000 to 24000000 (sets bits 13).

4. Set up an access to the address reported in R6 (Remember to subtract 4). This should result in a machine-check if the error is a hard one. After each machine check, the parity must be re-enabled as shown above.

5. It is not sufficient just to monitor the UVPERR<3:0> lines to find out which byte from the longword is causing the parity error because the UVPERR lines follow the UVMEM bus. The F280s E1, E8, E13 and E19 (refer to drawing UVR5) are the parity generating chips and work in an asynchronous manner.

   The equations for the UVERROR PAL E78 (refer to drawing UVR2) show that an error is flagged on pin 19 only when one of the UVPERR inputs (pins 5 - 8) are asserted. If pin 19 is still asserted at data strobe de-assert (rise) time, UVERROR will be asserted to the DYRC from E21 pin 9 through R91, the series resistor.

6. Use a logic analyzer to observe the ERR being asserted by E78 and determine which of the UVPERR lines sourced the error. If a logic analyzer is not available, you may be able to use an oscilloscope to look at the UVPERR lines individually to see which is asserted at data strobe time.

7. Make sure that the 4x2 MUX (E62 on page UVR5) is enabled.

8. Check if both the inputs to the MUX are in a steady state throughout the data valid time.

9. Look at power and ground at the F280s and the MUX.

10. Is the parity being stored in the DRAM correctly? Check continuity between the outputs of the F280s and the DRAM inputs (pins 29).

11. Is the parity being read out of the DRAM correctly? Check continuity between the DRAM outputs (pins 26) and the MUX inputs.

Another method of determining which byte of the longword sourced the parity error is to test each of the bytes individually. This can be done by starting XDT (as described above), and typing (no RETURN necessary):

S

This tells XDT to perform byte-wide accesses. If location 600 was producing a parity error when accessed as a longword, then locations 600, 601, 602 and 603 should be accessed byte by byte to determine which byte the parity error was in. This narrows down the area of circuitry to test as per the above instructions. Remember to re-enable parity error reporting (set CSR bit 13 in the DYRC) after each machine check.

NOTE

Parity errors are recorded even if parity reporting is not enabled in the DYRC CSR. The highest order bit will be set in the CSR if a parity error has been seen. Reading the CSR will reset this bit for the next CSR read. This bit is particularly useful when trying to detect parity errors in the IODRAM (refer to later section on IODRAM parity error diagnosis).

17.2.4.3.2 UVDRAM Parity Error Reporting Mechanism Fault

During the UVDRAM test, the MicroVAX sets the WWP bit in the CSR of the DYRC. This causes the DYRC to purposely write wrong parity into the DRAM. When the location is subsequently read, a Machine Check should occur due to the parity error. The OBT is expecting a machine check at this point. If it doesn't see one, it will flag a 2A00 error and bits 23 and 28 in R11 should be set.

You have to trace why the error reporting mechanism isn't working here. The circuitry to be fault-analysed is very much the same as the previous section. Follow these steps:

1. The error will have been detected in the UVDRAM test, so enter the following from the hardware console:

   SPECIAL TEST 2 1

   This causes the OBT to execute the UVDRAM test continually. Trace why the error is not getting back to the MicroVAX with an oscilloscope, logic probe, or logic analyzer.

2. The UVERROR PAL E78 (refer to drawing UVR2) should be sourcing an error.

   A logic probe on pin 19 of the PAL will show whether the error is ever detected. If the error is detected, then the problem is fairly easy to solve and just involves tracing the error signal from the PAL to E21 (same drawing). If this flip-flop is being clocked by data strobe correctly (pin 11), and pins 10 and 13 are held high, then the ERR should be then reported to the UVDYRC at pin 74. The DYRC should then relay the error to the MicroVAX directly from E88 pin 73 (refer to drawing page UVR1).

3. If the UVERR PAL isn't sourcing an error, then it is possible the parity generating logic on page UVR5 isn't working correctly. Check the UVPERR<0:3> outputs from the F280s to see if any are shorted high.

4. Check the DYRC to see if it is writing bad parity. To write bad parity UVVPAR, pin 73 on E88 will be de-asserted high. This will then be inverted by E69 (on page UVR5) and fed to the four inputs of the MUXs. Check E62 pins 14, 5, 11 and 2.

It is also possible to use XDT to set up a repeated read of the location to see why an error isn't being reported. If the reads

seem to be executing correctly then it is possible that bad parity was never written to the DRAM in the first place. Try setting WWP in the DYRC CSR from XDT (using a word access - command **2S**). Then perform a repeated write to a location in UVDRAM: you should see the DYRC attempting to write bad parity (E88 pin 73 as explained above).

17.2.4.4 Incorrect UVDRAM Refresh Operation -

NOTE

This section is equally applicable to the IODRAM.

The DYRC performs all the refresh operations necessary to maintain the data in the UVDRAM.

Pins 69 and 70 on E88 - the DYRC (refer to drawing UVR1) - select the refresh mode. Check that both these pins are held in the low state.

The DYRC performs refresh by issuing four RAS<3:0> assertions with no CAS assertions. The row address will increment for each of these RAS assertions. The DYRC provides a refresh-in-progress output RINPRG pin 67, to indicate to external circuitry that it is performing refresh and no DRAM access cycles should be performed. Check that RINPRG is asserting periodically (every 62.5 microseconds). If RINPRG is not asserting, then it is probable that the DRAM will not be refreshed.

17.3 I_O Bus Faults

17.3.1 I_O Bus Timeouts

When a device on the I_O bus is being addressed, and doesn't respond with IORDY within 25 microseconds of the MicroVAX starting its cycle, a bus timeout occurs. Devices on the I_O bus that potentially might not respond are the LANCE (E68), the ADMA (E36), the BDMA (E53), and the IODYRC (E9). All these devices should produce IORDY when the MicroVAX is addressing any of their internal registers.

Special XDT can be used to set up a repeated access from the MicroVAX to any of the above devices. By invoking a repeated access, an oscilloscope or logic analyzer can be used to find out why the cycle is completing incorrectly, as follows:

1. Start XDT by entering:

   SPECIAL XDT

2. Set up the repeated access to the devices on the I_O bus. The base internal registers of the I_O bus devices reside at the following hex addresses:

   ADMA - 30000000

   BDMA - 32000000

   LANCE - 36000000

   IODYRC - 34000000

3. To access the ADMA, enter:

   30000000/

   This will cause the MicroVAX to perform a longword read access of the ADMA DGCTL register. After typing the backslash the contents of the register will appear if the ADMA responds. If the cycle does not complete correctly then an "**XDT Trap**" will occur.

   Entering **R** repeated read accesses will occur and the relevant bus control signals can be monitored (see below). When performing these repeated accesses, a Watchdog Timer link must be inserted at location W8 to disable the Watchdog timeout interrupts. Remember to take the link out again when the fault has been located and fixed. The other devices on the I_O bus can be accessed by using the same process as above but inserting the relevant address.

   Care must be taken when performing ADMA or BDMA register accesses. The MicroDMA register map contains several reserved locations and accesses performed to these registers can produce unpredictable results. These locations are documented earlier in this manual.

   The IODYRC must be accessed on a word basis. To limit the MicroVAX to word width accesses enter **2S** at the XDT prompt. Entering **4S** will revert the MicroVAX back to long-word accesses.

Once the repeated access has been started, use an oscilloscope to check that the following sequence of events is occurring:

a. UVAS and UVDS are being issued by the MicroVAX (E115 pins 30 and 29).

b. VAXR should be asserted periodically (once per access) by the RROBIN PAL - (E89 pin 14, drawing page IOB2).

   If VAXR is not being asserted then either the PAL is not functioning correctly, or the input conditions required at the PAL inputs are not occurring.

   A first check is that the PAL should be receiving a clock on its pin 11.

   From the PAL equations, given that the RROBIN PAL is in VAXEN state, VAXR will be issued if UVADD27, UVADD28, UVBAS and the latched version of UVBAS are all asserted (E89 pins 4, 5, 2 and 9).

   The signals on E89 pins 6, 7 or 8 should not be asserted. These are bus request signals from the other potential bus masters which should not be active at this time.

c. If VAXR is being asserted, then check that it appears on the input to the RRCRL PAL (E90 pin 13).

   Given that the PAL is receiving VAXR and the control signals UVBAS, UVBDS, and UVBWR (E90 pins 9, 10 and 11) are asserted then the PAL should issue the IOBUS control signals IOAS, IODS, IOADDEN, IOCRLEN and IODATEN (E90 pins 15, 14, 17, 18, and 19).

   VAXCRL and IOCRLEN are two signals generated by RROBIN PAL and RRCRL PALs respectively. They are used to enable and hold eight control lines from the MicroVAX bus onto the I_O bus. Probe pins 11 and 1 of E82 to verify that the latches in question are being enabled. Refer to Drawing IOB2.

d. As continuous reads/writes of one of the I_O bus devices are being performed, the corresponding chip select should be generated periodically.

   The chip selects for the LANCE, ADMA and BDMA can be found on the 138 decoder - E14 pins 7, 11 and 10 respectively. If the select is being generated check that address strobe is present on pin 4 (Drawing IOR2).

The chip select for IODYRC registers can be found on another 138 decoder: this one is E32 pin 9. (Drawing IOR1)

e. For any MicroVAX cycle to complete, the device being addressed must supply Ready or Error. Each of the I_O bus devices are able to return their own Ready signal through the IORDY line (for example, E9 pin 24 for the IODYRC, page IOR4).

   All the device Ready lines are tied together and are returned to the MicroVAX through some simple logic on page IOB2 and some more complex logic on page UVR3.

   If IOREADY is not being asserted on E42 pin 6, then check first that UVBDS is present on E42 page 12 and also VAXR is present on E42 pin 5.

   If IOREADY is being asserted and RDY is not getting back to the MicroVAX on E115 pin 19, then there must be some problem with the Ready delay logic on drawing page UVR3.

17.3.2 IODRAM Related Faults

The IODRAM is organized in much the same way as the UVDRAM in that it is the same width and depth and its access and refresh timing is also controlled by a DYRC chip (E9 page IOR1).

Test A (the IODRAM and display test) is the first test in the I_O bus section of the OBT. This will be the first attempt the MicroVAX makes to access the I_O bus in the OBT and will therefore be the first test of the I_O bus arbitration logic.

There are important differences in the way in which the UVDYRC and the IODYRC report parity errors back to the MicroVAX. These are covered in the section on IODRAM parity errors.

17.3.2.1 Stuck-At 0/1 and Cross-Coupling - The method described earlier for finding the short/open circuits or faulty devices in the UVDRAM can also be used here. That is, repeated reading/writing of specific data patterns to isolate faults in specific data or address lines.

The address/data path from the MicroVAX to the IODRAM is more complex than the path from the MicroVAX to the UVDRAM.

You should assume that the address and data are correctly presented on the UVDAL because this has already been tested earlier in the OBT.

Octal bus transceivers on page IOB1 gate data from the UVDAL onto the IODAL at data strobe time. A combination of UVBWR, UVBDS and IODATEN control the enabling and direction of these bus transceivers.

More octal transceivers on page IOR3 gate the data from IODAL onto IOMEM<0:31> with the assertion of IOMEMCYC. The IOMEM information then goes directly to the data inputs of the RAMSIPS.

At address time, the IODAL information is fed to the DYRC DAL inputs. The DYRC then presents half the full address on RADDIO at RAS assertion time, and the second half at CAS assertion time. The octal drivers on page IOR1 take the RADDIO address and drive them directly to the RAMSIPS on page IOR4 as IORADD<0:7>.

Octal drivers on the same page gate the UVADD information onto IODAL with the IOADDEN signal.

Given the above signal paths it should be possible to detect the open/short or faulty device with a continuity tester and a oscilloscope.

17.3.2.2 IODRAM Parity Reporting - When a parity error is discovered in the IODRAM, the IODYRC signals an error back to the MicroVAX in a similar manner to the UVDYRC reporting parity errors in the UVDRAM. However, I/O parity errors are reported to the MicroVAX through the PWRFL pin of the CPU, whereas UV parity errors are reported directly through the ERR pin of the CPU which in turn causes a machine check.

In XDT you can cause a parity error in the UVDRAM by writing wrong parity to a location after setting the WWP (write wrong parity) bit in the UVDYRC CSR. When the location is subsequently read, a machine check will occur.

However, the IODRAM error reporting mechanism is slightly different. It is still possible to instruct the IODYRC to write wrong parity, but when a read is executed from the corrupt location a machine check will not occur using XDT.

Inspection of the IODYRC's CSR will show that a parity error has been detected because the most significant bit will have been set.

There are two kinds of I/O parity errors that could be encountered whilst executing the OBT :

- Parity error occurs with WWP disabled, causing PWRFL interrupt
- Parity error doesn't occur with WWP enabled

Sections 17.3.2.2.1 and 17.3.2.2.2 describe these two error types in more detail.

17.3.2.2.1 IODRAM Parity Errors

The IODRAM Error reporting mechanism is much the same as the UVDRAM mechanism in that parity is generated on a per-byte basis by F280s. The IOERROR PAL - E26 is informed of incorrect parity through the IOPERR<3:0> lines (refer to drawing IOR2). The IOERR PAL will then assert its error output (pin 12) if IODS, and the relevant byte-mask is asserted and the cycle is a read cycle. Make sure that pin 11 of the PAL is held low (this is a requirement for the error line to be asserted).

The Error output of the PAL is latched externally by an F74 (E21, page IOR2) at IODS de-assertion time (that is, IODS rising edge). The output of the latch is fed to the DYRC (E9 pin 74) through R24. When the IODYRC detects this parity error, it will set the most significant bit in its CSR (destroyed after first read of CSR), and assert IOERR, E9 pin 25.

Referring now to drawing page IOB2, IOERR assertion and subsequent deassertion is used to clock a zero through E76 (pin 2 to pin 5). E76 pin 5, the latch output, is fed to the powerfail pin of the MicroVAX (E115 pin 8, drawing CPU1) through an LS08 (E147) which ORs the IOERRINT with true PWRFL. The MicroVAX will not necessarily machine check on an I/O parity error because of the interrupt priority level (IPL) of the pwrfl input. This is in contrast to UVDRAM parity errors which always cause machine checks as they are reported to the MicroVAX by assertion of UVERR.

If the OBT has reported an IODRAM parity error, the location at which the error was detected will be found in R6 (subtract 4 or perhaps 8 to give the true location after pre-fetching of MicroVAX has been taken into account).

An access to this location will not cause a machine check (see above), but will cause a PWRFL interrupt to the MicroVAX. After the IODYRC has detected a parity error, it sets the most significant bit in its CSR to report a parity error and disables the PEN (parity enable) bit in its CSR so that no further errors

will be reported until PEN has been re-set.

Check the following:

1. Firstly, parity must be re-enabled in the DYRC CSR. After the DYRC has reported a parity error, it disables parity. Parity can be re-enabled by setting bit 13 in the IODYRC CSR (address 34000000 accessed by word width).

2. Go into XDT from the hardware console and enable word-width transfers by entering:

   2S

3. Re-enable the parity reporting mechanism in the DYRC by writing 2000 to 34000000 (sets bit 13).

4. Set up an access to the address reported in R6 (Remember to subtract 4). Examine the CSR, if the highest order bit has been set, a parity error has been detected. After each time this ERR bit has been set, the parity must be re-enabled as shown above.

5. It is not sufficient just to monitor the UVPERR<3:0> lines to find out which byte from the longword is causing the parity error. The UVPERR lines only 'follow' the IOMEM bus, which may, or may not, be valid at any given time. The F280s E6, E5, E7 and E24 (refer to drawing IOR4) are the parity generation chips and work in an asynchronous manner.

   The equations for the IOERROR PAL E26 (refer to drawing IOR2) show that an error is flagged on pin 12 only when one of the IOPERR inputs (pins 5 - 8) is asserted at data strobe assertion time. If pin 12 is still asserted at data strobe rising-edge time, IOERROR will be asserted to the DYRC from E21 pin 5 through R24, the series resistor.

6. Use a logic analyzer to observe the ERR being asserted by E78 (the UVERR PAL) and determine which of the UVPERR lines sourced the error. It might also be possible to use an oscilloscope here to look at the UVPERR lines individually to see which is asserted at data strobe time.

7. Make sure that the 4x2 MUX - E12 on page IOR4 is enabled.

8. Check that both inputs to the MUX are in a steady state throughout the data valid time.

9. Look at power and ground at the F280s and the MUX.

10. Is the parity being stored in the DRAM correctly? Check continuity between the outputs of the F280s and the DRAM inputs (pins 29).

11. Is the parity being read out of the DRAM correctly? Check continuity between the DRAM outputs (pins 26) and the MUX inputs.

Another method of determining which byte of the longword sourced the parity error is to test each of the bytes individually. This can be done by typing the following to XDT:

S

This tells XDT to perform byte-wide accesses.

If location 10000600 was producing a parity error when accessed as a longword, then access locations 10000600, 10000601, 10000602 and 10000603 byte by byte to find the byte with the parity error. This narrows down the area of circuitry you have to test.

Remember to re-enable parity error reporting (set CSR bit 13 in the DYRC) after each error report.

17.3.2.2.2 IODRAM Parity Error Reporting Mechanism Fault

During the IODRAM test, the MicroVAX attempts to write bad parity to a location in IODRAM. If the parity error reporting mechanism is working correctly the MicroVAX should see a PWRFL interrupt. If the MicroVAX sees no interrupt then it knows that the mechanism is not working.

If the OBT sees such an error it will report with error code 4A00 - R11 should have bits 25 and 28 set. R7 contains the address at which the MicroVAX was trying to force a parity error (this is probably of little consequence as the most likely faulty area is in the error reporting logic which means that the error will not be address specific).

It is possible, using XDT, to set up the IODYRC to write bad parity to a specific location, then to perform a read from the same location and check to see if the DYRC reports an error. If the DYRC flags an error, then the path between the IODYRC and the MicroVAX must be suspected.

Perform the following steps:

1. Enable write wrong parity in the IODYRC. This is achieved by setting up XDT to perform word-width transfers (type 2S), then writing 6000 to location 34000000 - the IODYRC CSR.

2. Perform a read of the same location to ensure the DYRC has been set up correctly.

3. Perform a write to a location in IODRAM (starts at location 10000000).

   Referring to drawing page IOR1, the IOVPAR output of the DYRC (E9 pin 73) should be de-asserted to enable bad parity. This should be inverted correctly by E69 pin 11 to pin 10 (drawing page IOR4). Check that this signal reaches all the '0D' inputs to the 4X2 MUX (pins 14, 5, 11 and 2).

4. Assuming that bad parity has been written successfully to the IODRAM location, check to see that it is being read back out correctly. Set up a constant read access to the location from XDT and use an oscilloscope to monitor the error logic. Bad parity should have been written to all bytes of the longword in the longword write, so you should see IOPERR<3:0> all asserted low during the subsequent read cycle.

5. Trace the IOPERR lines back to the IOERR PAL which should assert its own error line from pin 12. E21 (on page IOR2) should then latch the error assertion on rising edge of IODS.

6. If this latch is flagging an error, make sure IOERRINT is being asserted (drawing IOB2) on E76 pin 5.

7. IOERRINT is sent to the MicroVAX through gate E147 on page CPU1 which ORs IOERRINT with 'true' PWRFL and sends the result to the MicroVAX on its PWRFL input, pin 8.

8. Remember to reset the 6000 to 2000 in the IODYRC's CSR to disable WWP.

17.3.2.3 Incorrect IODRAM Refresh Operation - The DYRC performs all the refresh operations necessary to maintain the data in the IODRAM.

Pins 69 and 70 on E9 - the DYRC (refer to drawing IOR1) - select the refresh mode. Check that both these pins are held in the low state.

The DYRC performs refresh by issuing four RAS<3:0> assertions with no CAS assertions. The row address will increment for each of these RAS assertions. The DYRC provides a refresh-in-progress output RINPRG pin 67, to indicate to external circuitry that it is performing refresh and no DRAM access cycles should be performed.

Check that RINPRG is asserting periodically (every 62.5 microseconds). If RINPRG is not asserting, then it is probable that the DRAM will not be refreshed.

When the I_O bus is idle, the DYRC refresh mechanism needs the IOCRL PAL, E90 (refer to drawing IOB2), to toggle between the IDLE and REFRESH states. The refresh state is decoded external to the PAL by a NAND gate (E110) and the REFRESH_L signal is fed to the EPS input of the IODYRC. The REFRESH_ signal should resemble a 10 MHz clock when the I/O bus is idle.

17.3.3 LANCE Test Faults

Tests 'E' and 'F' are the LANCE Internal and External loopback tests.

This section comprises two sub-sections:

- Section 17.3.3.1 describes a very basic fault which is frequently encountered.
- Section 17.3.3.2 explains how the LANCE acquires I_O bus-mastership and some of the possible faults that can occur whilst the device is DMA-ing data into or out of the IODRAM which contains the ring buffer structures.

17.3.3.1 Ethernet Loopback Not Fitted - If a fault of this type is detected, the display on the front of the DEC MicroServer box will show an 'E' denoting an Ethernet-related error. This is a common reason for 'E' being displayed on the LED when OBT has failed.

The DEC MicroServer is generally tested using an Ethernet loopback connector, although it will also perform successful loopback through an Ethernet transceiver cable connected to

various other DIGITAL products. The LANCE device will be unable to perform successful external loopback tests if neither is connected to the Ethernet port of the box. If a loopback is connected to the box, make sure it is connected firmly.

Check that the cable connecting the Ethernet port to the main board is connected properly.

17.3.3.2 LANCE Accessing IODRAM Incorrectly - Special tests 'E' and 'F' in the OBT are the first tests in which the bus-mastership of the I_O bus is passed to a device other than the MicroVAX. The LANCE is the first device to request I_O bus-mastership in the OBT.

Prior to test 'E', the MicroVAX has performed I_O bus cycles but there is no request/grant mechanism in force for these accesses: the arbiter (RROBIN PAL) simply decodes the address presented and completes the cycle when it returns to the VAXEN state. Meanwhile, the MicroVAX will have been performing slip cycles awaiting receipt of 'Ready'.

The LANCE issues a request and awaits receipt of a grant before commencing an I_O bus cycle.

Most errors encountered when there is a fault in the request/grant mechanism or a fault in the LANCE accessing the IODRAM will be reported as a LANCE 'Basic operations Error' - Error code 5002 or 5008, although an open circuit LNCG net has produced an Error code C900 - 'Machine check whilst handling a machine check'. Both these error types were reported from test 'E'.

It is not possible to set up a repeated access to help the debug of these kind of problems. One way to debug is to use a logic probe or logic analyzer to look for single events. In this way you could look to see if LNCR ever gets asserted and follow through the chain of events one signal at a time, although this would require a re-run of 'SPECIAL TEST E' (for example) for each pass. A logic analyzer would enable you to look at many signals in parallel each time, but might be a time consuming debug method if the fault turns out to be simple.

The following steps outline the request/grant mechanism for the LANCE and provide some pointers towards potential fault areas:

1. To make I_O bus arbitration possible, the state bits of the RROBIN PAL should be toggling. This indicates that the PAL is passing round the four internal states corresponding to potential VAX, LANCE, ADMA and BDMA

cycles. If the PAL is not passing through these states, the LANCE will not have an opportunity to access the I_O bus. The state bits of the PALL are present on E89 pins 16 and 17 (refer to drawing IOB2).

2. The LANCE requests I_O bus-mastership by asserting its HOLD pin (refer to drawing ENI1, E68 pin 17). This output is susceptible to uneven output and is smoothed by the comparator E135. Check the 2.5 V reference on pin 5 of this device.

3. LNCR is the output of this comparator, and is fed to one element of the Hexadecimal latch - E83 on drawing IOB2. This device dual-ranks all the requests from potential I_O bus masters except for the MicroVAX. Check that E83 is being enabled at pin 1, and receiving a clock on pin 11.

4. LNCR arrives at the bus arbiter, RROBIN PAL - E89, on its pin 6. Check that this PAL is also receiving a clock on pin 1, and enabled on pin 11.

5. When the RROBIN PAL is ready to service the LANCE's request it will issue IODMR on pin 15 which is sent to E90, RRCRL PAL, pin 4. RRCRL should respond with IODMG from its pin 16, which is sent back to E89 on pin 3. VAXR and IODMR are mutually exclusive, they should never be asserted together.

6. RROBIN responds to IODMG by asserting LNCG on pin 13 which is sent back to the LANCE through series resistor R97. LNCG, ADMAG and BDMAG are always gated by IODMG.

7. Now look for:

   IOAS at LANCE pin 18

   IORDY at LANCE pin 22

   IODS at LANCE pin 14.

17.4 SCC Bus Faults

17.4.1 SCC Bus Errors

If an Unexpected Machine Check (error code C100) occurs during one of the OBT routines that uses the SCC bus and register R6 contains an address on the SCC bus (that is, an eight figure hex address starting with either 30.. or 32..), then an SCC bus Error has occurred.

The SCC bus differs from the other two buses in that non-existent memory is explicitly decoded. The MicroVAX and I_O buses suffer bus timeouts if non-existent memory is addressed. The NXM on the SCC bus is decoded by the CHIPSEL PAL E84 pin 19 (refer to drawing DC01). NXM becomes SIERR through E80 (refer to drawing DC02) and is sent to both DMA devices at pins 130. The DMAs will then report the error back onto the I_O bus by asserting IOERR.

- If the address starts with 30.., then the SCC bus Error occurred whilst the MicroVAX was trying to access a device on the SCC bus through the ADMA.

- If the address starts with 32.., then the failed access was attempted through the BDMA. Section 17.4.1.1 describes ADMA accesses and Section 17.4.1.2 describes BDMA accesses.

17.4.1.1 SCC Bus Accesses Through the ADMA - To find the faulty area of circuitry, use the console command SHOW ERROR which in this instance will report an Unexpected Machine check. Then start XDT by entering:

SPECIAL XDT

Insert the Watchdog link at location W8, and set up a repeated cycle to the address which produces the machine check. This address is found in register R6 in the above error report. The repeated address cycle is started by entering:

30008004/

This should be the address found in R6 (minus 4) to account for pre-fetch - see Section 17.2.4.1 for details.

Now type:

R

The fault can now be located by scoping around the signals detailed above. If a Watchdog timeout occurs at the console, then the link W8 hasn't been inserted.

The fact that an unexpected machine check has occurred indicates that the MicroVAX has failed to receive a Ready signal back from the device being addressed. There are several means by which this could happen. The most likely reasons are:

1. The 'chip select' relating to the address specified is not being generated.

   All chip selects on the SCC bus are generated by E84 - the CHIPSEL PAL. If address strobe BIAS is not reaching pin 11 of this device, then no chip selects will be propagated.

2. Chip select is not reaching the device being addressed.

   Assuming CHIPSEL has generated the select signal, check that it does actually reach the device, that is, the net isn't open circuit.

3. Ready is not being produced.

   In the OBT, the MicroVAX can access only the TMS memory and the DUSCC registers on the SCC bus, through either the ADMA or the BDMA. Both the TMS memory ready and the DUSCC Ready eventually come back to the DMA devices through the octal driver E91 on drawing page DMA1 as signal DMA1_BIRDY. The input to the driver is called DMA1_SIRDY and is sourced on page DC02 from an S09 gate, E80. This gate ORs the readys from the TMS memory and the DUSCC, although these signals are not sourced directly from the devices themselves.

   DSPRDY comes from E138 pin 6 (refer to page SAS6), when both CRLEN and VAXRDY are true. VAXRDY comes from E118 pin 6 and will be true when the TMS is not itself performing a cycle, and CRLEN comes from the TMSCYC PAL E142 in response to the chip select DSPMEM on pin 7.

   The DUSCC ready isn't actually produced by the DUSCC itself. Looking back to page DC02, SIRDY is the OR function of DSPRDY and a signal called CSL0300. CSL0300 is produced from a pair of shift registers on the same page - E103 and E96 - and is generated from the chip select, DUSCCH, which is the input to E103 on pin 2.

Check that both the devices are being clocked at 20 MHz on pins 8 and that pins 1 are both high. The DUSCC chip select signals are initially produced by the CHIPSEL PAL on pins 20 and 17 for DUSCCs A and B respectively, and are inverted through the DUSCCHLD PAL E97.

4. Ready is not being passed back from the SCC bus to the I_O bus through the DMA devices.

17.4.1.2 SCC Bus Accesses Through the BDMA - The principal difference between SCC bus accesses through the ADMA and through the BDMA is that the BDMA has to request bus-mastership from the SCC bus arbiter before it can commence an SCC cycle. The ADMA however is the default SCC bus master and can commence cycles without involving the bus arbiter.

If a machine check occurs with the address identified as being through the BDMA, the points to check are exactly the same as those above (remembering that the address offsets will be different from those through the ADMA).

The sequence of events to check are given below. Set up a repeated access to the failing location using XDT as detailed in the previous section.

1. The BDMA requests bus-mastership by asserting its IDMR line. This is difficult to probe given the physical spacing of the DMA pins, so pick it up at R40 pin 2 (refer to drawing DMA2).

2. IDMR is subject to uneven output, so it is smoothed by comparator E135. Check that there is a reference voltage of 2.5 V at pin 7 of this device. Check BDMASIBR assertion at the output, pin 2 of the comparator. There should be continuity of this signal to pin 2 of the SCCARB PAL (E94, same drawing as above).

3. The arbiter PAL now issues a request to the ADMA by asserting SIDMR - pin 12.

4. The ADMA should respond with SIDMG, check pin 4 of the arbiter.

5. Finally, BDMASIBG is asserted by the arbiter to indicate bus grant has been given to the BDMA.

6. The BDMA is now free to start the cycle by issuing address strobe, receiving Ready, and completing the cycle with the transfer of data. These actions are exactly the same as described in the previous section covering ADMA accesses.

17.4.2 TMS SRAM Faults

In OBT test 11, the MicroVAX performs IO-ACCESS mode cycles through the ADMA to perform stuck-at and coupling tests on the TMS RAM.

In OBT test 17, the MicroVAX downloads TMS code from the on-board ROM to the TMS RAM in order that the TMS can perform the same tests on the SRAM.

Sections 17.4.2.1 and 17.4.2.2 describes the faults from the two tests.

17.4.2.1 SRAM Faults Detected by the MicroVAX - The highest level address decode on the SCC bus is performed by the CHIPSEL PAL - E84 (refer to drawing page E84). When the MicroVAX is attempting to perform an access to the TMS SRAM, the decode signal DSPMEM will only be asserted if the SCC address is in the correct range corresponding to SRAM space with address strobe asserted, and the TMS is not presently the SCC bus Master, and the TMS has been reset.

If a stuck-at or coupling fault has been detected, then it is highly probable that DSPMEM has been decoded correctly and that the HOLD/HOLDA handshake has taken place between CHIPSEL and the TMS processor. DSPMEM is used directly as the HOLD request to the TMS (pin 59), and HOLD ACKNOWLEDGE is produced from pin 45 of the TMS and synchronized through E117 on page SAS1.

HOLD ACKNOWLEDGE is an input to the TMSCYC PAL - E142. This PAL will enable the address and data isolation buffers (E122, E123, E127, E128 refer to drawing page SAS3) if it sees HOLDA and DSPMEM asserted. SCCADDEN (E142, pin 14) enables the address buffers, and SCBUFEN enables the data from E142 pin 12, in conjunction with DSPIDBE assertion at E138 pin 10.

If error codes 7800 or 7801 have been reported, then the test location will be found in R5. Set up a repeated access to this location with the data written, which can be found in R6. Remember to enable word transfers from the hardware console

before the constant access takes place by typing:

2S

Also make sure that the TMS is being held in the reset state so that accesses can be made to its SRAM. The TMS is reset by setting bit<5> in the system CSR which resides at address 27FFF808 hex.

Now the address and data paths can be traced to find the short/open circuit or faulty device which is corrupting the address or data.

Assuming the data reaches the SCC bus intact, the only buffers between the DMAs and the SRAM itself are the isolation buffers mentioned above. A combination of reads and writes with specific data patterns as outlined in the UVDRAM section (Section 17.2.4.2) will help isolate the address or data lines that are stuck-at or shorting.

17.4.2.2 TMS Internal RAM Fault - A fault of this nature would be detected in test 17 - the TMS test.

In this test, the MicroVAX downloads TMS code to the TMS SRAM. The TMS then executes this code which performs the function of testing its internal RAM.

If a fault occurs, then either the code downloaded incorrectly or there is a fault with the on-board RAM which would indicate that the TMS was faulty or there is a problem with the TMS executing out of its SRAM.

17.5 Exerciser Faults

Test number 18 in the OBT is the Exerciser test, which aims to test the system as though it were being used in the field. Previous tests in the OBT have acted on parts of the system in isolation. The Exerciser test runs the whole system as a unit. Depending on your loopback configuration, this means that all four synchronous lines are performing external loopback at 64 Kbits/s, the LANCE is performing internal loopback, and the MicroVAX (when free) is performing background tasks.

Test 18 is the most severe of the tests in the OBT, in terms of the number of devices involved, and consequently has the heaviest drain on the power supply.

It is usually very difficult to diagnose a hardware problem just by looking at the error code produced by the OBT. With so much of the circuitry functional in the Exerciser test it is just about impossible to immediately implicate one faulty device or open/short circuit.

At this point in the OBT, most of DEC MicroServer's functionality has been tested in isolation, but in order that the system should function as a whole, there are several mechanisms that are 'fired up' for the first time, and it would seem reasonable that these could perhaps be responsible for an OBT error.

Hardware functions tested in the Exerciser test ONLY are:

- TMS accesses to DUSCC A/B
- TMS accesses to ADMA/BDMA
- WINDOW mechanism
- TMS interrupts to MicroVIC

These functions are discussed in more detail in the following sections.

17.5.1 TMS Related Faults

The Exerciser test is the first test in which the TMS performs accesses on the SCC bus which requires the TMS request/grant mechanism in the SCC bus arbiter to be working. If the OBT reports an error, such as one of those listed below, this might indicate that the TMS is unable to make SCC bus accesses and therefore cannot perform functions such as configuring DMA registers, configuring DUSCC registers, or WINDOWing through the BDMA to the IODRAM.

- B309 TMS dumped
- B30A TMS failed to update I'm alive counter
- B300 TMS failed to boot
- B400 Channel 0 failed to initialise
- B500 Channel 1 failed to initialise
- B600 Channel 2 failed to initialise

- B700 Channel 3 failed to initialise

The above errors indicate that the bus grant or WINDOW mechanisms might not be functional. If an error is found almost immediately in the Exerciser test, then it would be a good idea to use the following quick checks. If the Exerciser runs for a while (in the order of seconds), then it is very unlikely that the source of the error is the TMS not being able to get onto the SCC bus (unless there is an intermittent fault) because the TMS is constantly performing SCC bus accesses in the Exerciser.

Here are some quick checks if the OBT has reported the above types of error:

1. Hit Break on the hardware console.

2. Go into XDT by entering:

   SPECIAL XDT

3. A TMS debugging tool - TDT - exists so that it is possible to test circuitry around the TMS by executing code in the TMS rather than the MicroVAX. This is helpful in debugging situations where, for example, the MicroVAX can access the DUSCC registers but the TMS cannot. This could be due to the TMS not being able to get SCC bus grant for example.

   To invoke TDT, enter:

   V

   Now, remember that the TMS works with 16 bit addresses and the MicroVAX works with 32 bit addresses. For example, the TMS will see the base DUSCC A register at address E100 whilst the MicroVAX would access 1C200 plus 30000000 (the MicroVAX accesses the DUSCC A through the ADMA) making 3001C200 in total, that is, 1C200 = 2* E100.

   Type **E100/** to make the TMS perform a read of the Base register of the DUSCC, and type **R** to set up a repeated cycle so that signals can be probed.

4. Look for TMS bus request at the SCC arbiter E94 pin 3 (refer to drawing DMA2).

   If the request does not appear, then probe the source of the signal - VAXEMUL PAL E144 pin 21 (refer to drawing SAS1). If there is no activity here, check that the VAXEMUL PAL is being clocked - pin 1 - and that the

ranking device is being clocked - E151 pin 11. Check that E151 is enabled - pin 1. Pin 17 of E142 should be asserting to indicate an SCC transfer as a result of E142 pin 11 asserting which happens when the TMS wants to perform an external cycle.

5. Assuming that DSPSIBR is asserting, look for TMS bus grant - E94 pin 14. If this is not happening, look for the arbiter requesting the bus from the default bus Master, the ADMA. SIDMR should assert - pin 12 - and the ADMA flags its grant on pin 4. Check that the arbiter SCCARB is being enabled and clocked - pins 11 and 1 respectively.

   Is the TMS never getting the bus because the BDMA has locked up the bus? Look for BDMASIBR permanently asserted.

6. Assuming the TMS attains bus grant - DSPSIBG - does the bus grant get back to the TMS? Check E144 pin 10 (refer to drawing SAS1) for the grant getting back to VAXEMUL.

   As a response to the grant, VAXEMUL will forward DSPAEN, SIAS, DSPIDBE, DSPIDS and TMSRDY. Look for these asserting. The order in which they assert can be determined from the PAL equations. If the VAXEMUL sequencer appears not to be working correctly a logic analyzer can be used to monitor the above signal assertions against the internal state which is produced on pins 17, 18, 19, and 20 of the device.

If the failure is of the 'failure to update I'm alive counter' type, then the TMS will be attempting to use the WINDOW mechanism of the DMA to perform a write operation through the BDMA to the IODRAM where the counter resides.

Invoke TDT again by typing:

V

Set up a constant access of the IODRAM from the TMS.

In order to perform Window accesses through the BDMA, Channel 3 of that device has to be 'programmed' to point to the correct address space on the I_O bus, the window mask has to be set-up to accept the TMS addresses, and the channel control register for channel 3 has to be enabled.

Now do the following in XDT:

1. Write 414 to address 32000000 (DGCTL)
2. Write 7FFE to 32000110 (DCWM3)
3. Write 208B1 to address 32000100 (DCCTL3)
4. Write 10000000 to 32000114 (DCUPA3) to point to the base address in IODRAM.

Now, write a recognizable pattern (for example, 12345678) into the base location of IODRAM from XDT. If the TMS is able to perform Window cycles correctly, you should be able to read the same data from the TMS through TDT.

Start TDT by entering:

V

Perform a read from location 8000. This is the base address of Channel 3 window space. The DMACYC PAL should decode this address as window space, and the BDMA should subsequently perform a fetch from location 10000000 on the I_O bus.

If an error is reported, it is likely that either the DMACYC PAL is not functioning correctly, or the BDMA registers have not been initialised correctly.

If the cycle is completing, but fetching the wrong data, then the address is being translated incorrectly somewhere. Make sure the DMACYC PAL is decoding window space. Look to see the BDMA requesting bus-mastership of the I_O bus. It might be necessary to look at the addresses on the buses at address strobe time if everything else appears to be correct.

If cycles are completing without error, then a repeated access can help in scoping signals.

WIND should be asserted at DMACYC PAL E85 pin 17 (refer to page DC01) indicating a Window cycle in progress.

If this is not happening, check that address strobe BIAS is reaching pin 13 which will enable the PAL to decode the address on pins 1 to 11. DSPSIBG should be asserted on pin 14.

If WIND is asserting, this should, in turn, lead to BDMA PAL E113 (page DUS1) pin 18 RX3ITR asserting. If this signal is asserting, check that this request reaches the BDMA at E53 pin 110.

When the BDMA's channel priority mechanism decides that it is time to service this request, it looks at the address on the SCCDAL, decides that this is a WINDOW cycle and performs a write cycle on the I_O bus.

Look for the BDMA requesting bus Grant from the Round Robin arbiter. The request is ranked at E83 pin 18 (refer to drawing IOB2). If E83 is clocked and enabled, the request will reach RROBIN PAL E89 at pin 8. After the IODMR/IODMG handshake with the IOCRL PAL E90, bus grant BDMAG should be sourced on E89 pin 19.

After the fetch from the IODRAM, the BDMA will conclude the cycle on the SCC bus acting as bus slave supplying Ready back to the TMS.

17.5.2 Channel Related Faults

Channels 0, 1, 2, and Channel 3 Transmit, all operate the same DMA mechanism: the DUSCC requests a DMA transfer, whether it be for RX or TX data. The DMA request is serviced by the ADMA for channels 0 and 1 and the BDMA for channel 2, DMAing information from IODRAM to DUSCC A/B for TXDMA requests and from DUSCC A/B to IODRAM for RXDMA requests.

Channel 3 RX is a special case however (for reasons specified earlier in this manual), in that the TMS services the DUSCC DMA requests instead of the BDMA servicing them. The TMS services two Receive DMAs and then performs a WINDOW write operation with the word of data into IODRAM.

It is possible to tell, in some circumstances, whether a problem is just related to the one channel which the OBT reported as sourcing the error, or perhaps two channels are affected or even all four channels. Running SPECIAL TEST 1C, with additional flags set, enables you to switch off any of the channels in the Exerciser test.

For example:

- **SPECIAL TEST 1C 3**

  Disables channels 0 and 1.

- **SPECIAL TEST 1C C**

  Disables channels 2 and 3.

- **SPECIAL TEST 1C 8**

  Disables channel 3.

The diagnostics error document explains the format of the flag field and details some other options which may be useful in certain situations.

The first two of the above commands might help in isolating the fault to A or B DMA device or DUSCC A or B.

The third command might help to trace the fault to the 'pseudo-DMA' mechanism operated by the TMS purely for channel 3 receive as mentioned above.

Each DMA receive or transmit operation works with a byte of data. This is because the DUSCC has a byte-wide data bus.

17.5.2.1 Channel Faults Excluding Channel 3 Receive - This section does not aim to give step-by-step fault diagnosis instructions as presented in some of the earlier sections. The aim of this section is to detail the DMA request mechanism, how the request is serviced by the DMA device, and how the data is fetched from/transferred to the IODRAM ring buffers.

The following steps describe a DMA transfer on channel 0 receive. The DMA mechanism is equally applicable on all other channels, except channel 3 receive which is explained in the Section 17.5.2.2. Remember that channels 2 and 3 will use DUSCC B and BDMA.

1. The DUSCC signals a receive-DMA request when its internal FIFOs contain information to be DMA'd. DUSCC A, E107, will assert RX0DMR on pin 37 (refer to drawing DUS2). This request is passed to the ADMAREQ PAL, E108, (refer to page DUS1) on pin 13. This PAL will in turn assert RX0ITR on pin 12.

2. RX0ITR is fed direct to the ADMA device. The ADMA will service the request after a delay dependent upon requests pending, and the DMA's internal channel prioritizing.

3. There is no specific DMA acknowledge sourced from the DMA. The DMA is initialised with addresses by the TMS firmware with which to perform DMA transfers to/from for each channel, receive and transmit. External logic must then decode this address generated by the DMA, and generate the acknowledge to forward to the DUSCC, which

in turn must prepare to receive/present the data to be DMA'd.

4. Going back to the Channel 0 Receive DMA, the ADMA receives the ITR on pin 111 of E36 (refer to drawing DMA1). The DMA services the request by performing a read cycle from a location in DMA acknowledge space (that is, address range 1E800 to 1EBFF for RX0).

5. The ADMA does not have to request the SCC bus-mastership from the arbiter, it is the default SCC bus master. The DMACYC PAL, E85 (refer to drawing DC01), performs the address decode and will assert DACK on pin 16 at BIAS-assert time. The PAL qualifies the acknowledge by specifying the channel to be acknowledged. In the case of Channel 0 Receive again, the PAL does this by driving pins 12, 15 and 18 low (DMACYC PAL equations explain fully the relevance of these signals).

6. The F138 decoder, E98 (refer to page DC01), recognizes the DACK and the three-bit code, and produces the channel specific DMA acknowledge signal to go to DUSCC A in this case. E98 pin 15 will therefore be asserted.

7. The DUSCC will then produce the data to be DMA'd, and the SCC bus cycle will complete.

8. Once the DMA has received the data, it initiates a write cycle on the I_O bus to the IODRAM to store the receive data in the channel 0 receive ring buffers. You should therefore see the ADMA requesting bus-mastership of the I_O bus from the RROBIN PAL - E89 (refer to page IOB2). ADMA grant should be given after a suitable delay, and the write cycle completed.

The other DMA channels operate in exactly the same manner, except that the transmit channels source data from the IODRAM, and the DMA performs a write operation to the DUSCC. Each channel also has a different acknowledge address.

17.5.2.2 Channel 3 Receive Faults - The DMA action on channel 3 receive is driven directly by the TMS firmware. There is no DMA activity from either the ADMA or BDMA on this channel.

Here are the order of events for the channel 3 receive mechanism:

1. Refer to drawing SAS1 to see that the RX3DMR interrupt from DUSCC B is fed directly to the TMS on pin 58. This input, BIO, is effectively a maskable interrupt pin

which is polled by the TMS firmware to see whether there is a channel 3 receive DMA request pending.

2. If RX3DMR is asserted, the TMS will perform a read cycle to DMA acknowledge space for CH3 RX. The address range for this operation is 1FC00 to 1FFFF.

3. The DMACYC PAL, E85 on page DC02, decodes the cycle as a DMA acknowledge and asserts DACK, pin 16. It also drives pin 18 high, and 12 and 15 low, so that the F138 (E98) can perform further decode on these three bits to assert RXDMA3 (pin 9) back to DUSCC B.

4. DUSSC B will then recognize the acknowledge and present the data to be read (or DMA'd as it would expect). The TMS is bus Master for this cycle, and reads the data - one byte.

5. The TMS will perform two of these cycles, collecting two bytes to form a word, which it then WINDOWs into the IODRAM.

   The BDMA may perform 'packing' - this means that it could store four or more bytes before commencing a longword cycle on the I_O bus. This improves the efficiency of the bus, by performing one access instead of four.

   The WINDOW mechanism is covered in a previous section on TMS related faults - Section 17.5.2.

17.6 System Test

One fault that has been encountered whilst running the system test is that the OBT passed successfully, but the system software failed to download from the System test MicroVAX. The reason for this is that a software identity had been set up previously in the NVRAM, and that software is not resident on the test MicroVAX so could not be down-loaded.

17.7 Useful Test Points

This section can be used as a quick reference to the major control signal nets and the most convenient points to probe them.

The DEC MicroServer uses many surface mount devices to which it is difficult to attach oscilloscope probe clips or logic analyzer probes (this is particularly true of the DMA devices and also of the DYRCs - though to a lesser extent).

MicroVAX BUS

EPS	E115-23	ROMRD	E57-20
UVBAS	R113-2	UVRDY	R184-2
UVBDS	R159-2	UVERR	R183-2

I__O BUS

IOAS	E90-15	IORDY	R27-1
IODS	E90-14	IOERR	R41-2
VAXR	E89-14	BDMAR	E83-18
LNCR	E83-3	BDMAG	E89-19
LNCG	E89-13	VAXCRL	E89-18
ADMAR	E83-17	IOCRLEN	E90-18
ADMAG	E89-12	IODATEN	E90-19
LANCE	R11-2	ADMA	R12-2
BDMA	R10-2		

SCC BUS

BDMASIBR	E94-2	SIDMR	E94-12
BDMASIBG	E94-13	SIDMG	E94-4
DSPSIBR	E94-3	BIAS	E84-11
DSPSIBG	E94-14	BIDS	E87-22
DACK	E85-16	WIND	E84-1
IREGA	R161-2	DUSCCA_SEL	E84-20
IREGB	R160-2	DUSCCB_SEL	E84-17
NXM	E84-19	RSTDSP	E84-14

17.8 Error Code Cross-Reference

Here is a list of error codes with cross references to the relevant sections in this chapter containing further information.

Error Code	Section
2B00	17.2.4.1
0800	17.2.2
1900	17.2.3
2800 / 2900	17.2.4.2
C600	17.2.4.3.1
2A00	17.2.4.3.2
4B00	17.3.1
4800 / 4900	17.3.2.1
5002 / 5008	17.3.3.2
7800 / 7801	17.4.2.1
B700	17.5.2.1
B600	17.5.2.1
B500	17.5.2.1
B400	17.5.2.1
B300	17.5.1
B309	17.5.2.1
B30A	17.5.2.1
C100	17.2.4.1 or 17.3.1 or 17.4.1 (dependent upon contents of R6)

PART V

APPENDIXES

Appendix A
IC Descriptions

A.1 Introduction

This appendix contains the pin diagrams and pin listings for each of the major ICs used in the DEC MicroServer:

- MicroVAX (see Section A.2)
- LANCE (see Section A.3)
- DYRC (see Section A.4)
- DMA (see Section A.5)
- VIC (see Section A.6)
- DUSCC (see Section A.7)
- TMS32020 (see Section A.8)
- Synchronous support logic (see Section A.9)
- Memory chips (see Section A.10)

A.2 MicroVAX

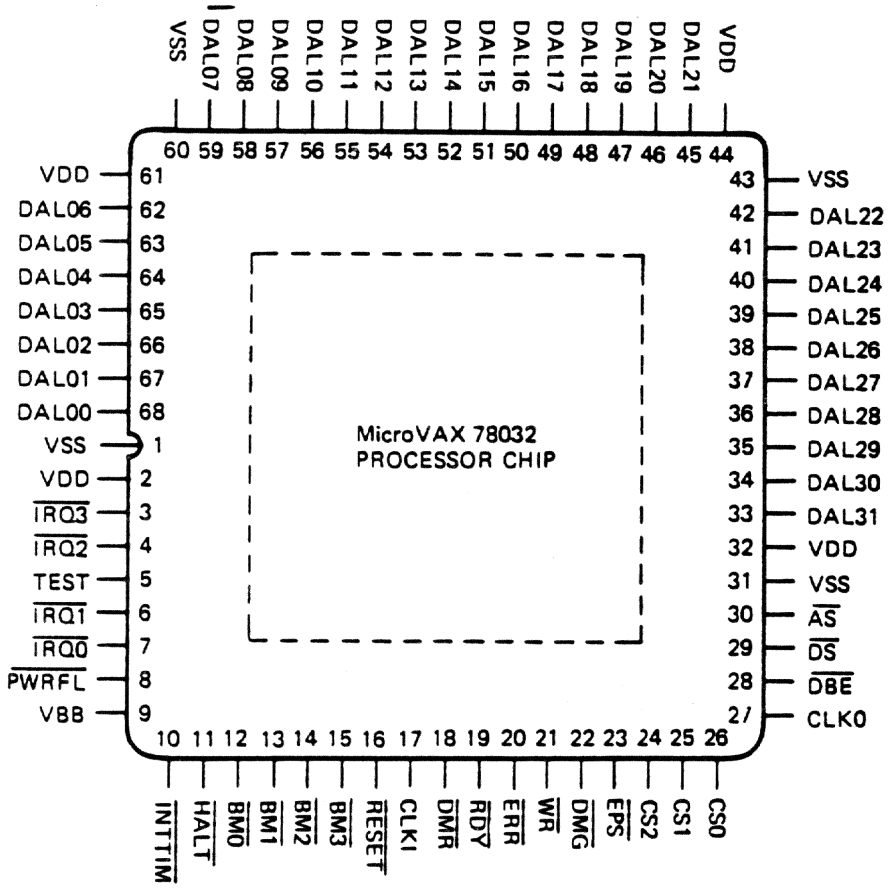


Figure A-1: The MicroVAX Chip

A.3 LANCE

Figure A-2: The AM7990 LANCE Chip

A.4 DYRC

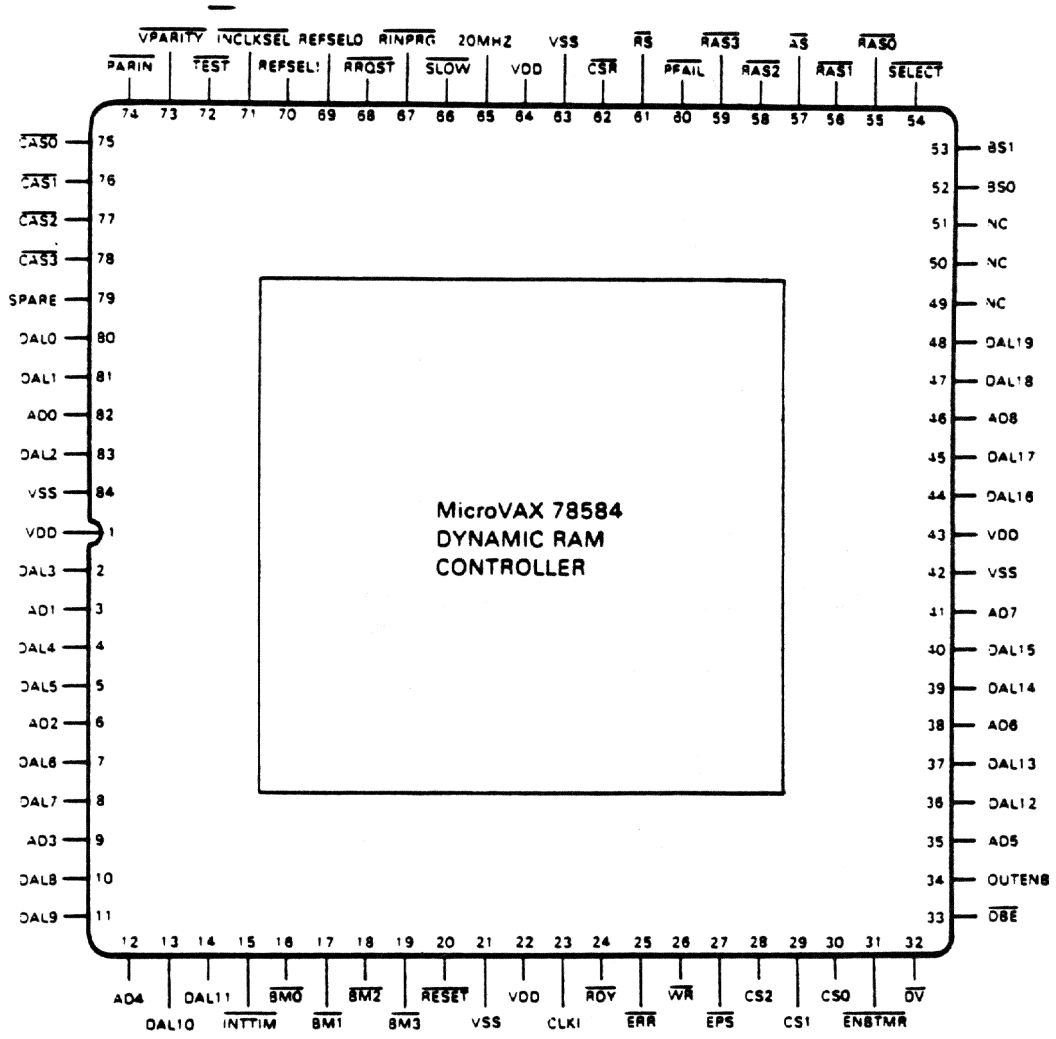


Figure A-3: The MicroVAX Dynamic RAM Controller (DYRC) Chip

A.5 DMA

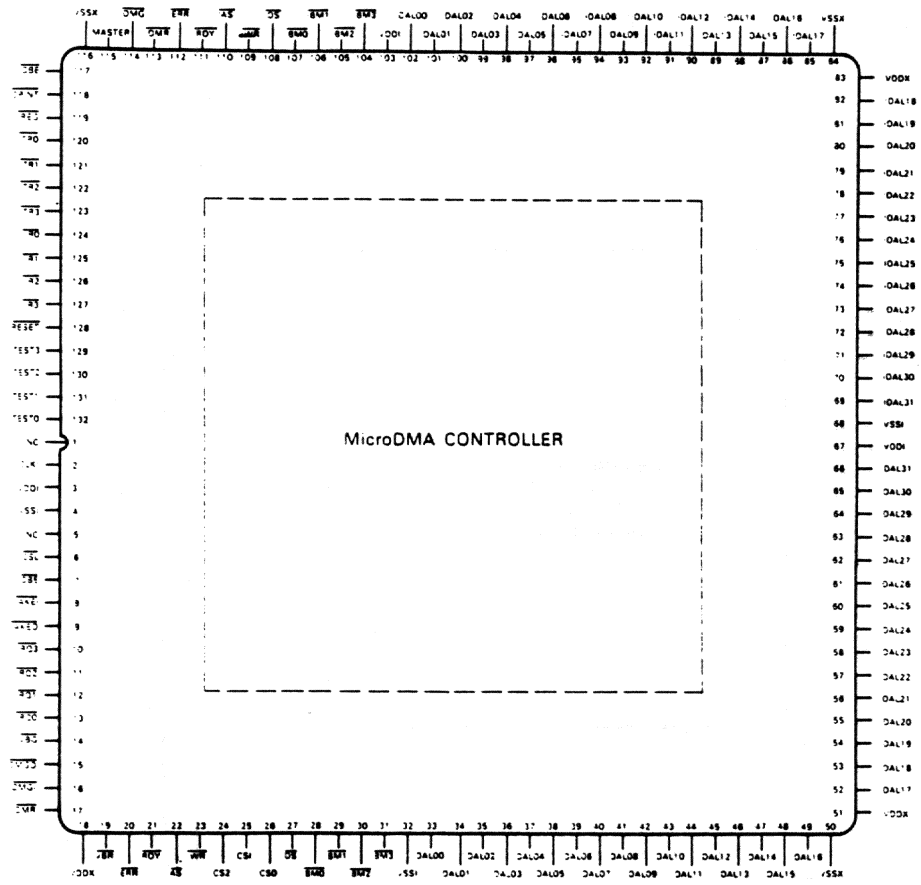


Figure A-4: The MicroDMA Chip

A.6 VIC

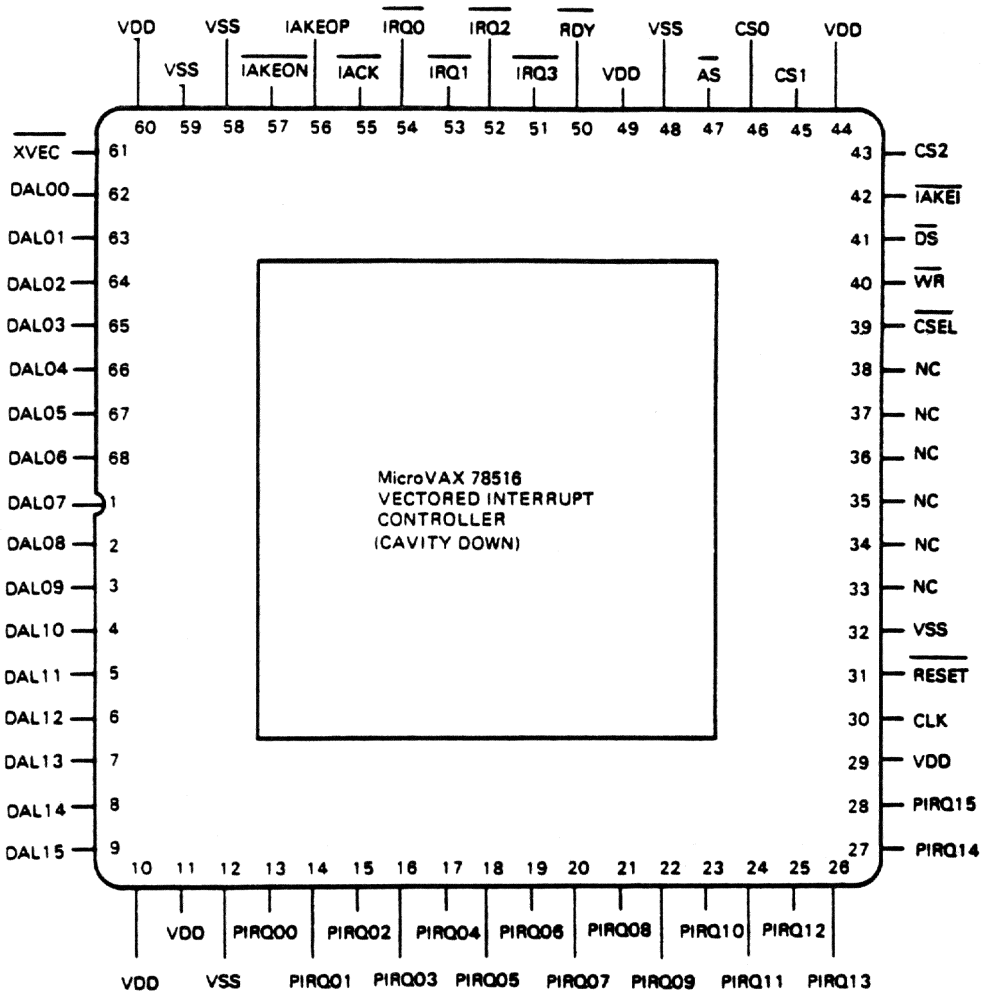
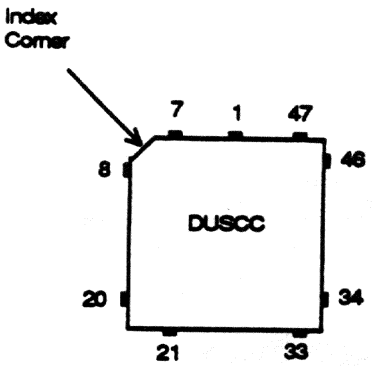


Figure A-5: The MicroVAX Vectored Interrupt Controller (VIC) Chip

A.7 DUSCC

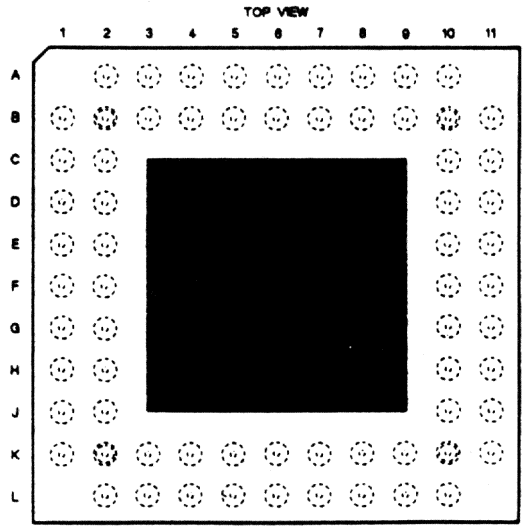


Pin	Function
1	IACKN
2	A3
3	A2
4	A1
5	RTxDAKNB/GPI1BN
6	IRQN
7	NC
8	RESETN
9	RTSBN/SYNOUTBN
10	TRxCB
11	RTxCB
12	DCDBN/SYNIBN
13	NC
14	RxDB
15	TxDB
16	TxDAKBN/GPI2BN
17	RTxDRQBN/GPO1BN
18	TxDRQBN/GPO2BN/ RTSBN
19	CTSBN/LCBN
20	D7
21	D6
22	D5
23	D4
24	DTACKN
25	DTCN
26	GND

Pin	Function
27	CSN
28	R/WN
29	DONEN
30	D3
31	D2
32	D1
33	D0
34	NC
35	CTSAN/LACAN
36	TxDRQAN/GPO2AN/ RTSAN
37	RTxDRQAN/GPO1AN
38	TxDAKAN/GPI2AN
39	TxDA
40	RxDA
41	NC
42	DCDAN/SYNIAN
43	RTxCA
44	TRxCA
45	RTSAN/SYNOUTAN
46	X2/DCN
47	X1/CLK
48	RTxDAKAN/GPI1AN
49	A6
50	A5
51	A4
52	V_{DD}

Figure A-6: The Dual Universal Serial Communications Controller (DUSCC) Chip

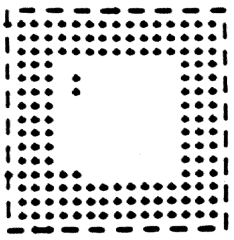
A.8 TMS32020



PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL	PIN	SIGNAL
A2	D8	B6	D15	E1	D2	H10	$\overline{STRB}$	K9	A14
A3	D10	B7	$\overline{BIO}$	E2	D1	H11	R/$\overline{W}$	K10	$\overline{DS}$
A4	D12	B8	READY	E10	$\overline{HOLDA}$	J1	DR	K11	V_{SS}
A5	D14	B9	CLKR	E11	DX	J2	FSR	L2	V_{SS}
A6	V_{CC}	B10	V_{CC}	F1	DO	J10	$\overline{PS}$	L3	A2
A7	$\overline{HOLD}$	B11	$\overline{IACK}$	F2	$\overline{SYNC}$	J11	$\overline{IS}$	L4	A4
A8	$\overline{RS}$	C1	D6	F10	FSX	K1	A0	L5	A6
A9	CLKX	C2	D5	F11	X2/CLKIN	K2	A1	L6	V_{CC}
A10	V_{CC}	C10	$\overline{MSC}$	G1	$\overline{INT0}$	K3	A3	L7	A9
B1	V_{SS}	C11	CLKOUT1	G2	$\overline{INT1}$	K4	A5	L8	A11
B2	D7	D1	D4	G10	X1	K5	A7	L9	A13
B3	D9	D2	D3	G11	$\overline{BR}$	K6	A8	L10	A15
B4	D11	D10	CLKOUT2	H1	$\overline{INT2}$	K7	A10		
B5	D13	D11	XF	H2	V_{CC}	K8	A12		

Figure A-7: The TMS32020 Chip

A.9 Synchronous Support Logic



Signal Name	Pin Number
DSPDATA<15> H	30
DSPDATA<14> H	44
DSPDATA<13> H	29
DSPDATA<12> H	14
DSPDATA<11> H	43
DSPDATA<10> H	28
DSPDATA<9> H	13
DSPDATA<8> H	26
DSPDATA<7> H	11
DSPDATA<6> H	25
DSPDATA<5> H	23
DSPDATA<4> H	8
DSPDATA<3> H	7
DSPDATA<2> H	6
DSPDATA<1> H	4
DSPDATA<0> H	3
GAADDR H	24
DSPSTRB L	39
GARDY H	18
RESET L	10
CS L	41
DSPWR L	27
X21INT L	2
CLKI H	22
ENABLETST L	45
CLRDISABLE L	36
TSTCLR	21
OPDISABLE L	50
VDD	46
VDD	72
VDD	95
VDD	130
VDD	107
VDD	144
VDD	73
VDD	15
VDD	12
VDD	38
VDD	19
VDD	1
VSS	48
VSS	60
VSS	78
VSS	90
VSS	103
VSS	105
VSS	108
VSS	110
VSS	97
VSS	85
VSS	67
VSS	55
VSS	42
VSS	40
VSS	37
VSS	35
RTS_CH<3> L	119
RTS_CH<2> L	104
RTS_CH<1> L	133
RTS_CH<0> L	118
DMA_REQUEST_CH<3> L	124
DMA_REQUEST_CH<2> L	109
DMA_REQUEST_CH<1> L	138
DMA_REQUEST_CH<0> L	123
EOF_CH<3> L	141
EOF_CH<2> L	140
EOF_CH<1> L	125
EOF_CH<0> L	139
CCITT_103_CH<3> H	79
CCITT_103_CH<2> H	87
CCITT_103_CH<1> H	86
CCITT_103_CH<0> H	93
CCITT_104_CH<3> H	92
CCITT_104_CH<2> H	91
CCITT_104_CH<1> H	99
CCITT_104_CH<0> H	98
CCITT_105_CH<3> L	143
CCITT_105_CH<2> L	127
CCITT_105_CH<1> L	112
CCITT_105_CH<0> L	111
CCITT_109_CH<3> L	114
CCITT_109_CH<2> L	113
CCITT_109_CH<1> L	129
CCITT_109_CH<0> L	128
CCITT_113_CH<3> L	75
CCITT_113_CH<2> L	74
CCITT_113_CH<1> L	81
CCITT_113_CH<0> L	80
CCITT_114_CH<3> L	62
CCITT_114_CH<2> L	61
CCITT_114_CH<1> L	69
CCITT_114_CH<0> L	68
CCITT_115_CH<3> L	51
CCITT_115_CH<2> L	57
CCITT_115_CH<1> L	56
CCITT_115_CH<0> L	63
FDATA<7> H	17
FDATA<6> H	16
FDATA<5> H	34
FDATA<4> H	33
FDATA<3> H	32
FDATA<2> H	31
FDATA<1> H	47
FDATA<0> H	54
FADD<14> H	88
FADD<13> H	82
FADD<12> H	83
FADD<11> H	84
FADD<10> H	76
FADD<9> H	77
FADD<8> H	70
FADD<7> H	71
FADD<6> H	64
FADD<5> H	65
FADD<4> H	66
FADD<3> H	58
FADD<2> H	59
FADD<1> H	52
FADD<0> H	53
FRD L	89
FWR L	94
RXDA_CH<3> H	115
RXDA_CH<2> H	100
RXDA_CH<1> H	101
RXDA_CH<0> H	102
TXDA_CH<3> H	132
TXDA_CH<2> H	117
TXDA_CH<1> H	131
TXDA_CH<0> H	116
RXCLK_CH<3> L	106
RXCLK_CH<2> L	135
RXCLK_CH<1> L	120
RXCLK_CH<0> L	134
TXCLK_CH<3> L	137
TXCLK_CH<2> L	122
TXCLK_CH<1> L	136
TXCLK_CH<0> L	121

Figure A-8: Synchronous Support Logic Pins

A.10 Memory Chips

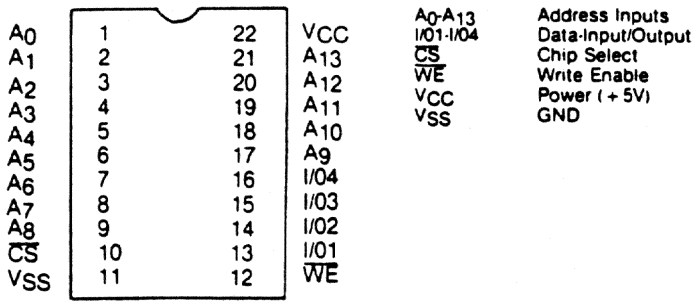


Figure A-9: The RAM Chip Pins

Appendix B
Memory Maps

B.1 Introduction

This appendix contains the important memory maps of the DEC MicroServer system:

- MicroVAX (see Section B.2)
- TMS 32020 (see Section B.3)
- MicroVAX window to the I/O subsystem (see Section B.4)
- TMS I/O address space (see Section B.5)
- DMA register offsets (see Section B.6)
- LANCE register offsets (see Section B.7)
- DUSCC register offsets (see Section B.8)
- Synchronous support logic registers (see Section B.9)
- ROM callable routine addresses (see Section B.10)

B.2 MicroVAX Memory Map

Table B-1 shows the MicroVAX memory map. Note that this map refers only to addresses generated by the MicroVAX. Also note these other restrictions:

- The DMA controllers can only access the Buffer RAM bus. Other accesses cause an NXM timeout. This prevents lock-ups caused by the TMS trying to access itself through the DMA controllers.
- The LANCE can only access the Buffer RAM. The most significant address byte of the LANCE addresses is forced to D0 (hex) in the hardware.

- The MicroVAX is the only device that can access the DMA controllers' global registers, the LANCE registers, the DYRC registers, and the DMA's channel registers.
- The MicroVAX's window through the DMA controllers to the SCC bus uses a different addressing scheme to that used locally on the SCC bus. Section B.4 contains the MicroVAX's I/O window memory map.

See Chapter 4 for details of MicroVAX operations.

Table B-1: MicroVAX Memory Map

Address (Hex)	Description	Notes
X8000000-XFFFFFFF	External memory (can be accessed using the console port)	Processor bus peripherals are all disabled. Delayed "ready" returned if CONSOLE asserted
	I_O Bus Devices	
0XXXXXXX and 2XXXXXXX		Nonexistent memory: attempting to access causes a bus timeout
36000008-37FFFFFF	Aliased LANCE registers	Do not use
36000004	LANCE RAP register	Word-wide, longword aligned
36000000	LANCE RDP register	Word-wide, longword aligned

Table B-1 (cont.)

Address (Hex)	Description	Notes
34000008-35FFFFFF	Aliased Buffer RAM DYRC registers	Do not use
34000004	FAR in the Buffer RAM's DYRC	Word-wide, longword aligned
34000000	CSR in the Buffer RAM's DYRC	Word-wide, longword aligned
33000000-33FFFFFF	Aliased DMA B space	Do not use
32000200-32FFFFFF	DMA B space	MicroVAX window to the SCC bus (see DMA and TMS memory maps for details)
32000000-320001FF	DMA B registers (see Section B.6 for the offset of each register)	
31000000-31FFFFFF	Aliased DMA A space	Do not use
30000200-30FFFFFF	DMA A space	MicroVAX window to the SCC bus (see DMA and TMS memory maps for details)
30000000-300001FF	DMA A registers (see Section B.6 for the offset of each register)	

Table B-1 (cont.)

Address (Hex)	Description	Notes
28000000-2FFFFFFF	Reserved space (external)	Returns delayed "ready" when CONSOLE asserted
	MicroVAX Bus Devices	
27FFF810-27FFFFFF	Aliased register addresses	Do not use
27FFF80C	Writing any value to this address generates the ERRCLR signal. This clears the PRTNERR and IOERRSRC bits in the System CSR	Write cycles only
27FFF808	System CSR	Byte-/word-wide, longword aligned
27FFF804	Display register	Byte-wide, longword aligned, write cycles only
27FFF800	Watchdog timer update	Byte-wide, longword aligned, write cycles only
27000000-27FFF7FF	Unused	Nonexistent memory
26000080-26FFFFFF	Aliased VIC registers	Do not use

Table B-1 (cont.)

Address (Hex)	Description	Notes
26000000-2600007F	VIC registers (see Section B.11 for individual register offsets)	Each register is word-wide and longword aligned
24000008-25FFFFFF	Aliased DYRC registers	Do not use
24000004	FAR for the System RAM's DYRC	Word-wide, longword aligned
24000000	CSR for the System RAM's DYRC	Word-wide, longword aligned
22008000-23FFFFFF	Aliased NVRAM space	Do not use
22000000-22007FFF	NVRAM	8K bytes. Each byte is aligned on a longword
21000080-21FFFFFF	Aliased Ethernet PROM	Do not use
21000000-2100007F	Ethernet PROM	Each location is byte wide and longword aligned
20080000-20FFFFFF	Aliased Firmware ROM	Do not use

Table B-1 (cont.)

Address (Hex)	Description	Notes
20040000-2007FFFF	Firmware ROM	256K bytes or an aliased submultiple, longword-wide
20000000-2003FFFF	Aliased Firmware ROM	Do not use
	I_O Bus Memory	
12000000-1FFFFFFF	Reserved memory space	Nonexistent memory on the I_O bus
10100000-11FFFFFF	Aliased Buffer RAM	Do not use
10000000-100FFFFF	Buffer RAM	1 Megabyte of RAM with per-byte parity
	MicroVAX Bus Memory	
08000000-0FFFFFFF	External reserved memory space	Returns delayed "ready" if CONSOLE asserted
02000000-07FFFFFF	Reserved memory space	Nonexistent memory
00100000-01FFFFFF	Aliased System RAM	Do not use
00000000-000FFFFF	System RAM	1 Megabyte of RAM with per-byte parity

B.3 TMS 32020 Memory Map

Table B-2 shows the TMS processor's memory map. Note that this is a word oriented machine, so each of the addresses in the table refers to a 16-bit value. See Chapter 6 for details of TMS operations and the functions of the I/O subsystem.

Table B-2: TMS Processor's Data Memory Map

Address (Hex)	Description	Notes
0000-03FF	On-chip data memory	
0400-3FFF	Reserved	Expansion memory
4000-7FFF	Data memory	Unused
8000-BFFF	DMA B port 3 window access	Mainly provides TMS processor access to the Synchronous I/O Control Block and the ring buffers. Also acts as data path between I/O port 3 receiver and the Buffer RAM
C000-DFFF		Unused
E000-E0FF	DMA A registers	Each register is 32 bits long and so uses two addresses. See Section B.6 and Chapter 8
E100-E1FF	DUSCC A registers	Each register is one byte long and is word aligned

Table B-2 (cont.)

Address (Hex)	Description	Notes
E200-E2FF	DMA B registers	Each register is 32 bits long and so uses two addresses. See Section B.6 and Chapter 8
E300-E3FF	DUSCC B registers	Each register is one byte long and is word aligned
E400-EFFF	Reserved	Unused
F000-F1FF	DMA A channel 0 window	
F200-F3FF	DMA A channel 1 window	
F400-F5FF	DMA A channel 2 window	
F600-F7FF	DMA A channel 3 window	
F800-F9FF	DMA B channel 0 window	
FA00-FBFF	DMA B channel 1 window	
FC00-FDFF	DMA B channel 2 window	

Table B-2 (cont.)

Address (Hex)	Description	Notes
FE00-FFFF	DMA B channel 3 DMA acknowledge	

B.4 MicroVAX Window to the I/O Subsystem

The MicroVAX address map contains two windows to the synchronous side of the DEC MicroServer (one window provided by each DMA device). Table B-3 shows the offsets of devices and registers in these windows.

Table B-3: MicroVAX Window to the I/O Subsystem

Offset (Hex)	Description	Notes
00000-007FF	Reserved	
00800-0081F	TMS32020 I/O space	Aliased every 1F bytes up to 8FF (*)
01000-07FFF	Reserved	Unused
08000-0FFFF	TMS RAM	(*)
10000-1C1FF	Unused	
1C200-1C3FF	DUSCC A registers	
1C400-1C5FF	Reserved	Unused

Table B-3 (cont.)

Offset (Hex)	Description	Notes
1C600-1C7FF	DUSCC B	Each register byte-wide, and word aligned
1C800-1CFFF	Reserved	Unused
1E000-1E3FF	Channel 0 TX DMA acknowledge(+)	
1E400-1E7FF	Channel 1 TX DMA acknowledge(+)	
1E800-1EBFF	Channel 0 RX DMA acknowledge(+)	
1EC00-1EFFF	Channel 1 RX DMA acknowledge(+)	
1F000-1F3FF	Channel 2 TX DMA acknowledge(+)	
1F400-1F7FF	Channel 3 TX DMA acknowledge(+)	
1F800-1FBFF	Channel 2 RX DMA acknowledge(+)	

Table B-3 (cont.)

Offset (Hex)	Description	Notes
1FC00-1FFFF	Reserved	

* The MicroVAX can access these parts of the system when the TMS processor is held in the RESET state. To do this, the MicroVAX sets the RSTDSP field in the System CSR.

+ Writing to any of these offsets generates the appropriate DMA acknowledge signal to one of the DUSCCs. This, in turn, enables data transfer on that I/O port.

B.5 TMS32020 I/O Space

Table B-4 lists the offsets in the TMS 32020 I/O space, and shows the items accessed at each address. Different items are accessed at each address for read and write operations. Chapter 6 describes TMS operations and the register contents in detail.

Table B-4: TMS 32020 I/O Space

I/O Address (Hex)	Item Accessed	
	Read Operation	**Write Operation**
0	MDSR0	IOCR0
1	MDSR1	MDCR1
2	MDSR2	MDCR2
3	SSL data register	SSL data register

Table B-4 (cont.)

	Read Operation	Write Operation
4		Set TMSINT
5		Reset TMSINT
6		Turn on Window Access Block
7		Reset EOF flag for Port 0
8 to A	Reserved	
B	SSL address register	SSL address register
C to F	Aliases 0 to 7 (described above)	Aliases 0 to 7 (described above)

B.6 DMA Register Offsets

The MicroVAX and TMS 32020 memory maps have two areas that contain DMA registers (one area for each DMA device). Table B-5 shows the offset in these blocks of each register in the MicroVAX memory map. Table B-6 shows the offsets used in the TMS memory map.

For details of the register contents, see Chapter 8.

Table B-5: DMA Register Offsets in the MicroVAX Memory Map

	Offset (Hex)	Register Name
000 to 0038 -- Global Registers		
	000	DGCTL
	004	DSBR
	008	DGBR
	00C to 01F	Reserved
	020	DGTRN
	024	DGPA
	028	DGTEMP
	02C to 037	Reserved
	038	DGBC
	03C to 03F	Reserved
040 to 07F -- Channel 0 Registers		
	040	DCCTL
	044 to 047	Reserved(*)
	048	DCINT
	04C	DCIOBA/DCIDS
	050	DCIBC/DCWM
	054	DCBO/DCUPA
	058	DCSPTE/DCIDD

Table B-5 (cont.)

	Offset (Hex)	Register Name
	05C to 05F	Reserved(*)
	060	DCCSV
	064	DCIOA
	068	DCBC
	06C	DCPTE
	070	DCPA
	074 to 07F	Reserved(*)
080 to 0BF -- Channel 1 Registers		
	080	DCCTL
	084 to 087	Reserved(*)
	088	DCINT
	08C	DCIOBA/DCIDS
	090	DCIBC/DCWM
	094	DCBO/DCUPA
	098	DCSPTE/DCIDD
	09C to 09F	Reserved(*)
	0A0	DCCSV
	0A4	DCIOA
	0A8	DCBC

Table B-5 (cont.)

	Offset (Hex)	Register Name
	0AC	DCPTE
	0B0	DCPA
0C0 to 0FF -- Channel 2 Registers		
	0C0	DCCTL
	0C4 to 0C7	Reserved(*)
	0C8	DCINT
	0CC	DCIOBA/DCIDS
	0D0	DCIBC/DCWM
	0D4	DCBO/DCUPA
	0D8	DCSPTE/DCIDD
	0DC to 0DF	Reserved(*)
	0E0	DCCSV
	0E4	DCIOA
	0E8	DCBC
	0EC	DCPTE
	0F0	DCPA
	0F4 to 0FF	Reserved(*)
100 to 13F -- Channel 3 Registers		
	100	DCCTL

Table B-5 (cont.)

	Offset (Hex)	Register Name
	104 to 107	Reserved(*)
	108	DCINT
	10C	DCIOBA/DCIDS
	110	DCIBC/DCWM
	114	DCBO/DCUPA
	118	DCSPTE/DCIDD
	11C to 11F	Reserved(*)
	120	DCCSV
	124	DCIOA
	128	DCBC
	12C	DCPTE
	130	DCPA
	134 to 13F	Reserved(*)

* Do not try to access these reserved areas of the DMA device's memory map. If you do, the device locks up and the system has to be restarted.

Table B-6: DMA Register Offsets in the TMS 32020 Memory Map

	Offset (Hex)	Register Name
000 to 01F -- Global Registers		
	000	DGCTL_L
	001	DGCTL_H
	002	DSBR_L
	003	DSBR_H
	004	DGBR_L
	005	DGBR_H
	006 to 00F	Reserved
	010	DGTRN_L
	011	DGTRN_H
	012	DGPA_L
	013	DGPA_H
	014	DGTEMP_L
	015	DGTEMP_H
	016 to 01B	Reserved
	01C	DGBC_L
	01D	DGBC_H
	01E and 01F	Reserved

Table B-6 (cont.)

	Offset (Hex)	Register Name
020 to 03F -- Channel 0 Registers		
	020	DCCTL_L
	021	DCCTL_H
	022 and 023	Reserved(*)
	024	DCINT_L
	025	DCINT_H
	026	DCIOBA_L/DCIDS_L
	027	DCIOBA_H/DCIDS_H
	028	DCIBC_L/DCWM_L
	029	DCIBC_H/DCWM_H
	02A	DCBO_L/DCUPA_L
	02B	DCBO_H/DCUPA_H
	02C	DCSPTE_L/DCIDD_L
	02D	DCSPTE_H/DCIDD_H
	02E and 02F	Reserved(*)
	030	DCCSV_L
	031	DCCSV_H
	032	DCIOA_L
	033	DCIOA_H
	034	DCBC_L

Table B-6 (cont.)

	Offset (Hex)	Register Name
	035	DCBC_H
	036	DCPTE_L
	037	DCPTE_H
	038	DCPA_L
	039	DCPA_H
	03A to 03F	Reserved(*)
040 to 05F -- Channel 1 Registers		
	040	DCCTL_L
	041	DCCTL_H
	042 and 043	Reserved(*)
	044	DCINT_L
	045	DCINT_H
	046	DCIOBA_L/DCIDS_L
	047	DCIOBA_H/DCIDS_H
	048	DCIBC_L/DCWM_L
	049	DCIBC_H/DCWM_H
	04A	DCBO_L/DCUPA_L
	04B	DCBO_H/DCUPA_H
	04C	DCSPTE_L/DCIDD_L
	04D	DCSPTE_H/DCIDD_H

Table B-6 (cont.)

	Offset (Hex)	Register Name
	04E and 04F	Reserved(*)
	050	DCCSV_L
	051	DCCSV_H
	052	DCIOA_L
	053	DCIOA_H
	054	DCBC_L
	055	DCBC_H
	056	DCPTE_L
	057	DCPTE_H
	058	DCPA_L
	059	DCPA_H
	05A to 05F	Reserved(*)
060 to 07F -- Channel 2 Registers		
	060	DCCTL_L
	061	DCCTL_H
	062 and 063	Reserved(*)
	064	DCINT_L
	065	DCINT_H
	066	DCIOBA_L/DCIDS_L
	067	DCIOBA_H/DCIDS_H

Table B-6 (cont.)

	Offset (Hex)	Register Name
	068	DCIBC_L/DCWM_L
	069	DCIBC_H/DCWM_H
	06A	DCBO_L/DCUPA_L
	06B	DCBO_H/DCUPA_H
	06C	DCSPTE_L/DCIDD_L
	06D	DCSPTE_H/DCIDD_H
	06E and 06F	Reserved(*)
	070	DCCSV_L
	071	DCCSV_H
	072	DCIOA_L
	073	DCIOA_H
	074	DCBC_L
	075	DCBC_H
	076	DCPTE_L
	077	DCPTE_H
	078	DCPA_L
	079	DCPA_H
	07A to 07F	Reserved(*)
080 to 09F -- Channel 3 Registers		
	080	DCCTL_L

Table B-6 (cont.)

	Offset (Hex)	Register Name
	081	DCCTL_H
	082 and 083	Reserved(*)
	084	DCINT_L
	085	DCINT_H
	086	DCIOBA_L/DCIDS_L
	087	DCIOBA_H/DCIDS_H
	088	DCIBC_L/DCWM_L
	089	DCIBC_H/DCWM_H
	08A	DCBO_L/DCUPA_L
	08B	DCBO_H/DCUPA_H
	08C	DCSPTE_L/DCIDD_L
	08D	DCSPTE_H/DCIDD_H
	08E and 08F	Reserved(*)
	090	DCCSV_L
	091	DCCSV_H
	092	DCIOA_L
	093	DCIOA_H
	094	DCBC_L
	095	DCBC_H
	096	DCPTE_L
	097	DCPTE_H

Table B-6 (cont.)

Offset (Hex)	Register Name
098	DCPA_L
099	DCPA_H
09A to 09F	Reserved(*)

* Do not try to access these areas of the DMA devices memory map. If you do, the device locks up and the system has to be restarted.

DMA registers are 32 bits long, and so each uses two locations in the TMS memory map. The "_L" suffix to each register name indicates that a location contains the least significant part of the register. "_H" similarly marks the most significant part.

B.7 LANCE Register Offsets

The LANCE has four control and status registers. These registers are accessed through two addressable ports in the MicroVAX memory:

- MicroVAX address 36000004 for the LANCE RAP (address port).
- MicroVAX address 36000000 for the LANCE RDP (data port).

Both ports are read/write.

The CSRs are read from or written to in a two step operation:

1. The address of the CSR is written into the address port (RAP) during a bus slave operation.

2. During a subsequent bus slave operation, the data to be read from (or written to) the data port (RDP) is read from (or written to) the selected CSR.

Table B-7 describes the bits in the LANCE RAP port. Chapter 5 describes the four LANCE registers and how they are used.

Table B-7: Fields in the LANCE Register Address Port

Bit Field	Field Name	Contents
<15:2>		Reserved. When writing to the port, this field is ignored. When reading the port, the field always contains zero.
<1:0>	CSR address	Contains the number of the CSR to be read or written.

B.8 DUSCC Register Offsets

DUSCC registers are directly addressable by both the MicroVAX (through the DMA windows) and the TMS 32020 (the registers are part of the TMS's code/data space). Table B-8 describes the offsets used to access the registers. Chapter 7 describes the use of the DUSCCs in detail.

NOTE

A0 of the SCC address bus is not used, therefore the DUSCC registers appear on word boundaries. Valid data is transferred on the lower byte. A1 of the SCC address bus is connected to A0 of the TMS bus so that the TMS can access devices correctly.

In the table, "_A" appears at the end of register names for Channel A. Similarly, "_B" appears after those for Channel B. Register names with no suffix are global registers.

Table B-8: DUSCC Register Offsets

Offset (Hex)	Register Name	Function
00	CMR1_A	Channel Mode Register 1
01	CMR2_A	Channel Mode Register 2
02	S1R_A	SYN 1/Secondary Address Register 1
03	S2R_A	SYN 2/Secondary Address Register 2
04	TPR_A	Transmitter Parameter Register
05	TTR_A	Transmitter Timing Register
06	RPR_A	Receiver Parameter Register
07	RTR_A	Receiver Timing Register
08	CTPRM_A	Counter/Timer Preset Register High

Table B-8 (cont.)

Offset (Hex)	Register Name	Function
09	CTPRL_A	Counter/Timer Preset Register Low
0A	CTCR_A	Counter/Timer Control Register
0B	DMR_A	Output and Miscellaneous Register
0C	CTH_A	Counter/Timer High
0D	CTL_A	Counter/Timer Low
0E	PCR_A	Pin Configuration Register
0F	CCR_A	Channel Command Register
10 to 13	TXFIFO_A	Transmitter FIFO
14 to 17	RXFIFO_A	Receiver FIFO
18	RSR_A	Receiver Status Register
19	TRSR_A	Transmitter and Receiver Status Register

Table B-8 (cont.)

Offset (Hex)	Register Name	Function
1A	ICTSR_A	Input And Counter/Timer Status Register
1B	GSR	General Status Register(*)
1C	IER_A	Interrupt Enable Register
1E	IVR	Interrupt Vector Register - Modified
1F	ICR	Interrupt Control Register
20	CMR1_B	Channel Mode Register 1
21	CMR2_B	Channel Mode Register 2
22	S1R_B	SYN 1/Secondary Address Register 1
23	S2R_B	SYN 2/Secondary Address Register 2
24	TPR_B	Transmitter Parameter Register

Table B-8 (cont.)

Offset (Hex)	Register Name	Function
25	TTR_B	Transmitter Timing Register
26	RPR_B	Receiver Parameter Register
27	RTR_B	Receiver Timing Register
28	CTPRM_B	Counter/Timer Preset Register High
29	CTPRL_B	Counter/Timer Preset Register Low
2A	CTCR_B	Counter/Timer Control Register
2B	OMR_B	Output and Miscellaneous Register
2C	CTH_B	Counter/Timer High
2D	CTL_B	Counter/Timer Low
2E	PCR_B	Pin Configuration Register
2F	CCR_B	Channel Command Register

Table B-8 (cont.)

Offset (Hex)	Register Name	Function
30 to 33	TXFIFO_B	Transmitter FIFO
34 to 37	RXFIFO_B	Receiver FIFO
38	RSR_B	Receiver Status Register
39	TRSR_B	Transmitter and Receiver Status Register
3A	ICTSR_B	Input and Counter/Timer Status Register
3B	GSR	General Status Register(*)
3C	IER_B	Interrupt Enable Register
3E	IVRM	Interrupt Vector Register - Unmodified

*The GSR is accessible though both offsets 1B and 3B.

B.9 Synchronous Support Logic

The TMS 32020 accesses the SSL through two locations in the TMS I/O address space (see Section B.5 for details). These addresses correspond to two ports on the SSL:

- TMS I/O address 3 - the SSL data port
- TMS I/O address B - the SSL address port

Reading from, and writing to, SSL registers is similar to the procedure with the LANCE CSRs:

- To read a register, the TMS puts the number of the register into the SSL address port and then reads the data from the SSL data port.
- To write to a register, the TMS puts the number of the register into the SSL address port, and then writes the data into the SSL data register.

Table B-9 lists SSL register numbers. Chapter 7 contains details of the SSL registers.

Table B-9: SSL Register Identification

Register Number (Hex)	Register Name
0	DBIDR
1, 2, and 3	Not used
4	DBCR0
5	DBCR1
6	DBCR2

Table B-9 (cont.)

Register Number (Hex)	Register Name
7	DBCR3
8 to F	Reserved for future use

B.10 Firmware System Support Routines

The DEC MicroServer firmware provides a number of software callable routines. These routines allow the system software to obtain information about the DEC MicroServer system, and to alter how the DEC MicroServer behaves.

Table B-10 lists the addresses of these routines in the Firmware ROM. Chapter 13 describes the DEC MicroServer's firmware functions in detail.

Table B-10: Firmware System Support Routines

Routine Address (Hex)	Routine Name	Effect
20040008	ROM$CALL_BOOT	Reboots the DEC MicroServer. Control is returned to the new system image.
2004000C	ROM$CALL_DUMP	Dumps and reboots the system.

Table B-10 (cont.)

Routine Address (Hex)	Routine Name	Effect
20040010	ROM$MOP_FN	Processes console carrier messages received by the running operating system. The routine processes the MOP packet handed to it, and returns to the caller after creating a reply packet for the host.
20040014	ROM$MACHINE_CHECK	Called after a machine check, either while the console code is running or from the operating system. Logs the machine check in NVRAM and returns to the calling code.
20040018	ROM$GET_PARAMS	Reads information from the NVRAM. Returns 2 longwords: R0 is the first longword parameter, R1 is the second longword parameter.
2004001C	ROM$DO_COMMAND	Processes a character string as if it were a console command.
20040020	ROM$LOAD_SYS	Reboots the DEC MicroServer. Control is returned to the new system image. Loads a specified system, without calling self-test.

Table B-10 (cont.)

Routine Address (Hex)	Routine Name	Effect
20040024	ROM$SET_TEXT	Sets up a state variable when called by the running system containing the "state" of the software. The variable is the address of a buffer containing the text.
20040028	ROM$PROVIDE_IMAGE	Allows loading of a RAM version of the firmware. It saves the entry address for later use. The RAM image can be loaded at any address in main memory, but must have calculated its checksum before use.
2004002C	ROM$GET_VERSION	Returns the DEC MicroServer hardware version number in R0. High word is variant number, low word is revision level.

Appendix C
Device Registers

C.1 Introduction

This appendix is a reference listing of all the on-chip and system registers described in the main body of the manual. Each register is illustrated and is accompanied by a listing of the individual fields.

The registers appear in the same order as in the main part of the book. Each section contains a cross reference to the chapter that contains a detailed description of its registers.

Each section contains the registers for a specific device:

- DYRC registers (see Section C.2)
- System and display registers (see Section C.3)
- LANCE registers (see Section C.4)
- Synchronous control and status registers (see Section C.5)
- DUSCC registers (see Section C.6)
- SSL registers (see Section C.7)
- MicroDMA registers (see Section C.8)
- VIC registers (see Section C.9)

C.2 DYRC Registers

See Chapter 4 for details of the DYRC registers.

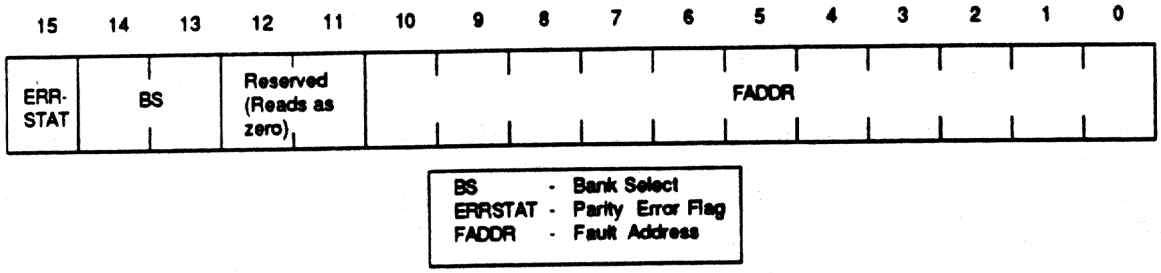


Figure C-1: DYRC CSR

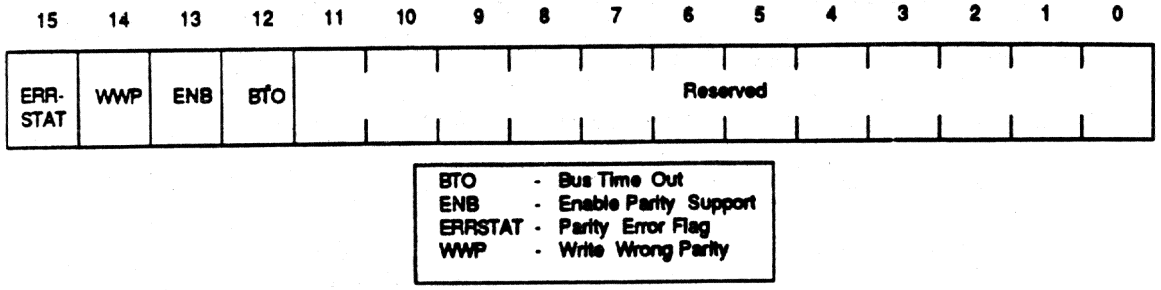
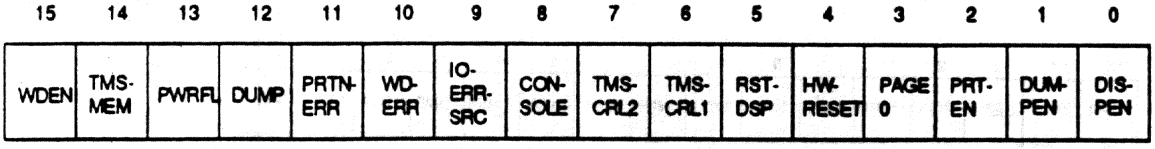


Figure C-2: DYRC FAR

C.3 System and Display Registers

See Chapter 4 for details of the system and display registers.



CONSOLE	Console Port Status	PRTNERR	Partition Error
DISPEN	Display Enable	PWRFL	Power Failure
DUMP	Dump Switch Status	RSTDSP	TMS Processor Reset
DUMPEN	Enable Dump Switch	TMSCRL1	TMS Processor Control 1
HWRESET	Hardware Reset	TMSCRL2	TMS Processor Control 2
IOERRSRC	I_O Bus Error Source	TMSMEM	TMS RAM Size
PAGE0	TMS Processor Page	WDEN	Watchdog Timer Enable
PRTEN	Enable Partition Logic	WDERR	Watchdog Timer Expired

Figure C-3: System CSR

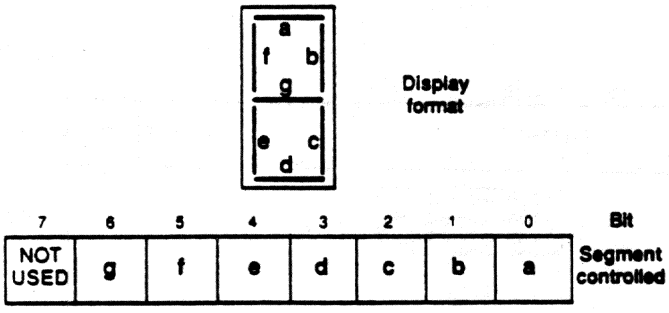


Figure C-4: Display Register

C.4 LANCE Registers

See Chapter 5 for details of the LANCE registers.

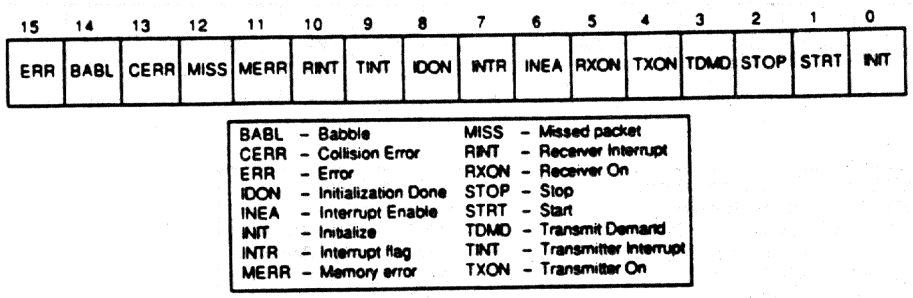


Figure C-5: LANCE CSR0

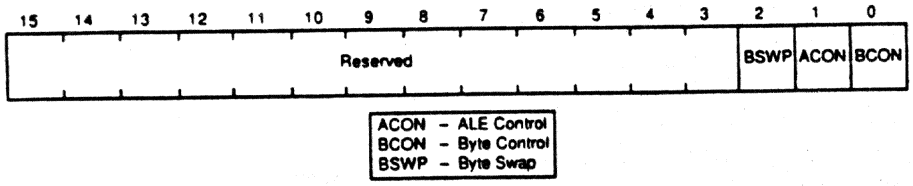


Figure C-6: LANCE CSR03

C.5 Synchronous Control and Status Registers

See Chapter 6 for details of the synchronous control and status registers.

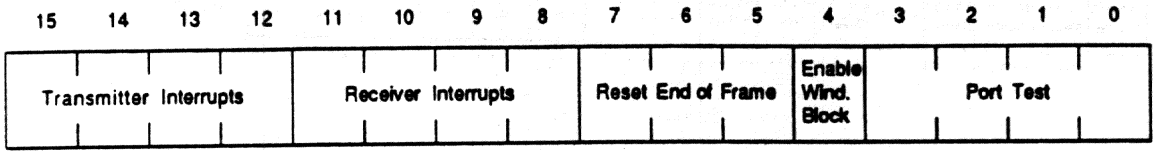


Figure C-7: IOCR0

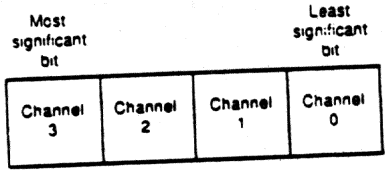


Figure C-8: IOCR0: Allocation of Bits

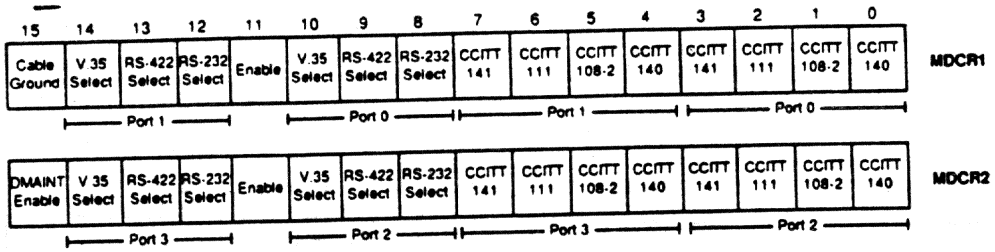


Figure C-9: MDCR1 and MDCR2

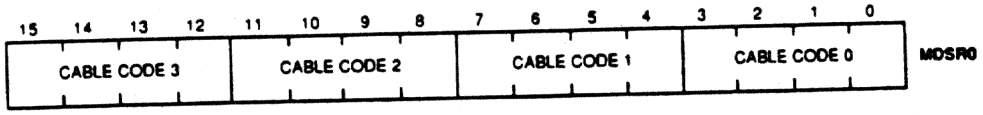


Figure C-10: MDSR0

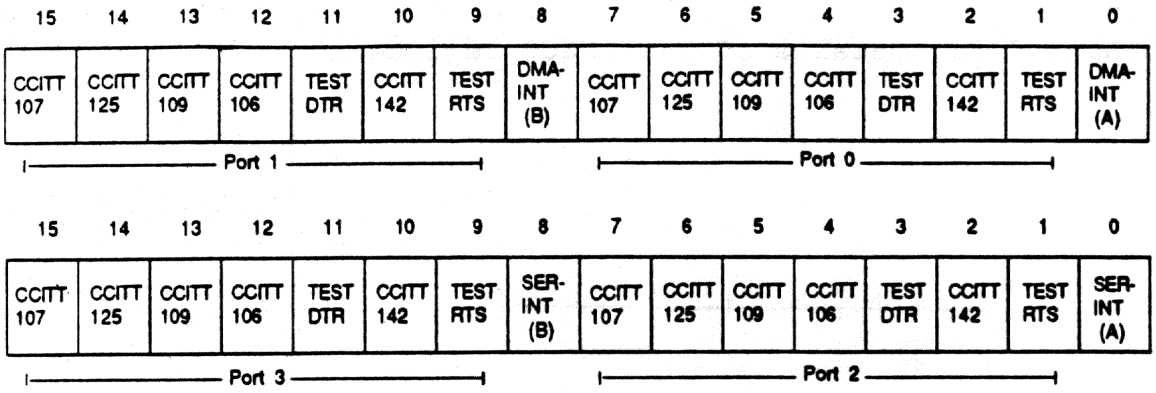


Figure C-11: MDSR1 and MDSR2

C.6 DUSCC Registers

See Chapter 7 for details of the DUSCC registers.

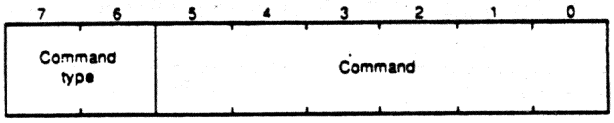


Figure C-12: DUSCC CCR

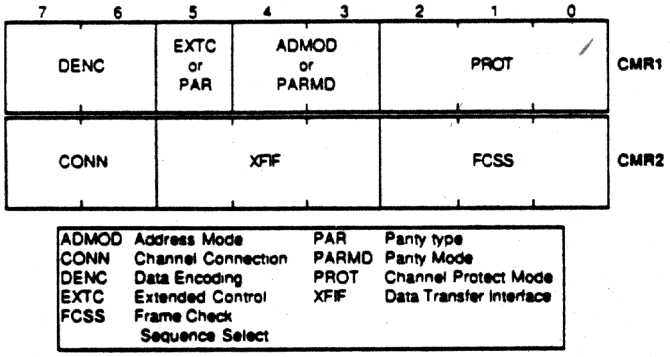


Figure C-13: DUSCC CMR1 and CMR2

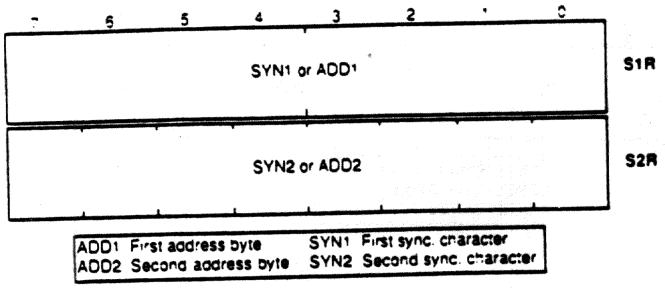


Figure C-14: DUSCC S1R and S2R

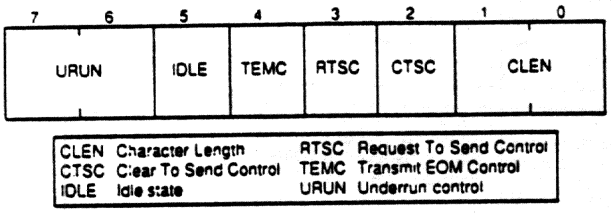


Figure C-15: DUSCC TPR

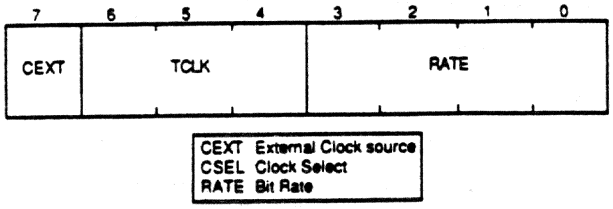


Figure C-16: DUSCC TTR

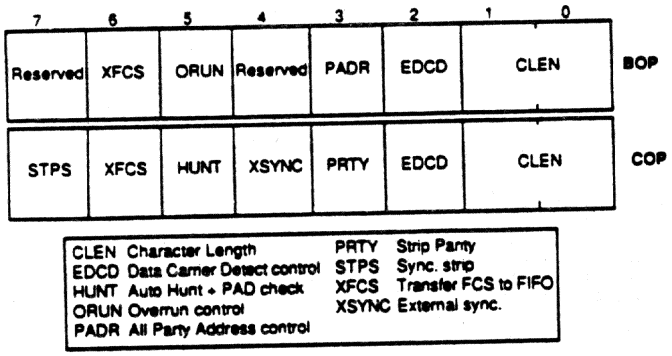


Figure C-17: DUSCC RPR (BOP and COP)

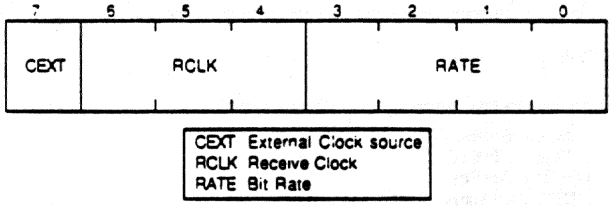


Figure C-18: DUSCC RTR

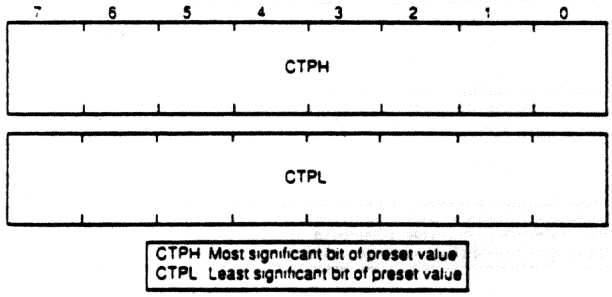


Figure C-19: DUSCC CTPRL and CTPRH

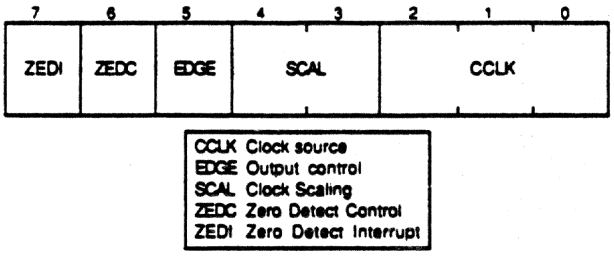


Figure C-20: DUSCC CTCR

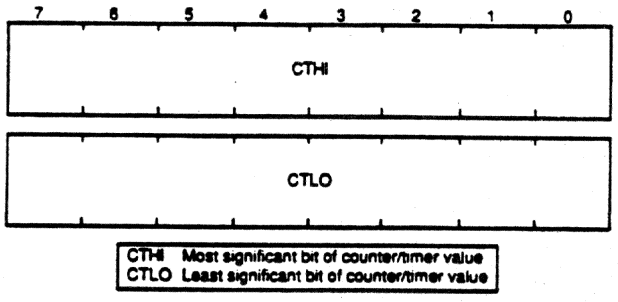


Figure C-21: DUSCC CTH and CTL

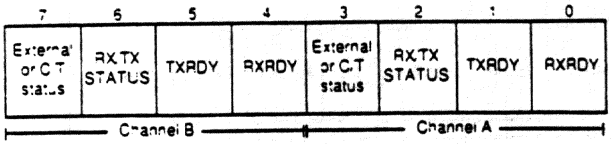


Figure C-22: DUSCC GSR

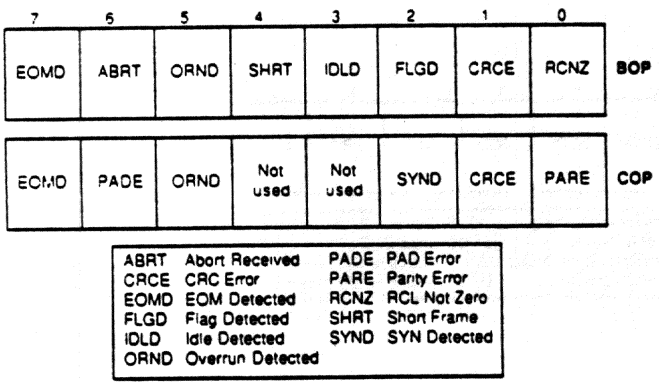


Figure C-23: DUSCC RSR (BOP and COP)

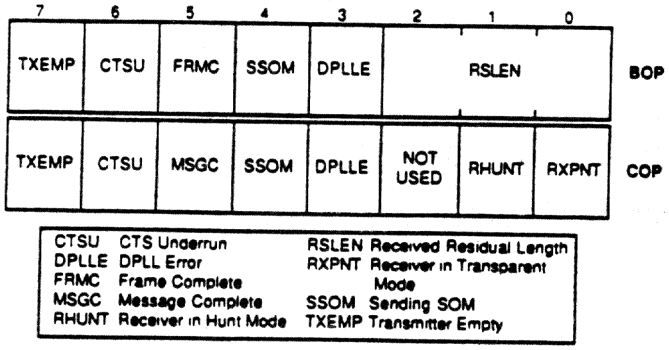


Figure C-24: DUSCC TRSR (BOP and COP)

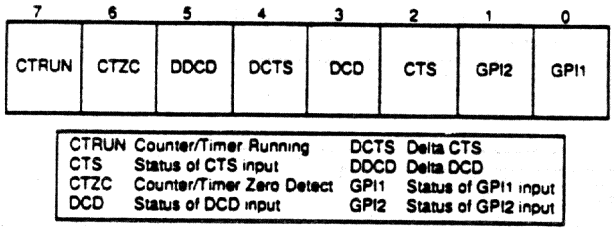


Figure C-25: DUSCC ICTSR

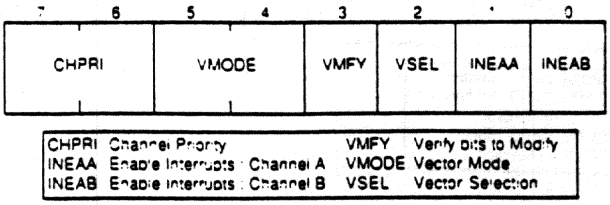


Figure C-26: DUSCC ICR

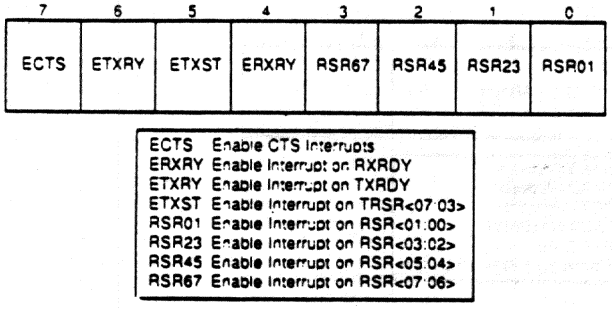


Figure C-27: DUSCC IER

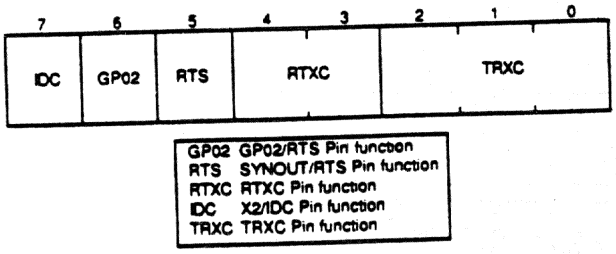


Figure C-28: DUSCC PCR

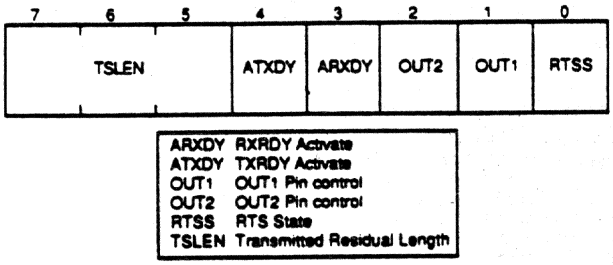


Figure C-29: DUSCC OMR

C.7 SSL Registers

See Chapter 7 for details of the SSL registers.

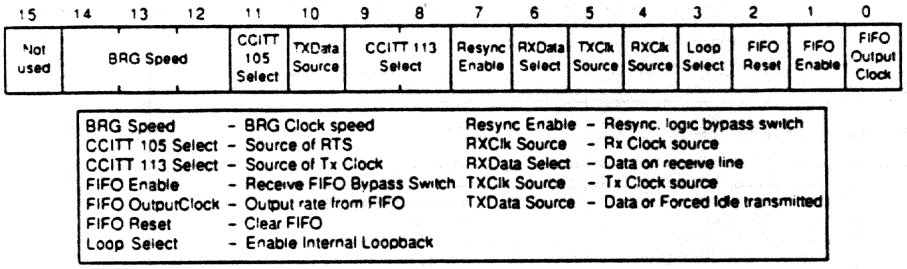


Figure C-30: SSL DBCR

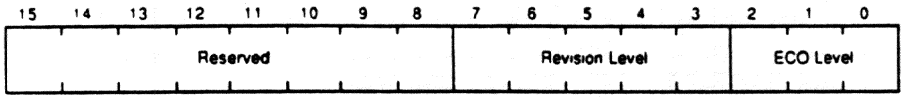


Figure C-31: SSL DBIDR

C.8 MicroDMA Registers

See Chapter 8 for details of the MicroDMA registers.

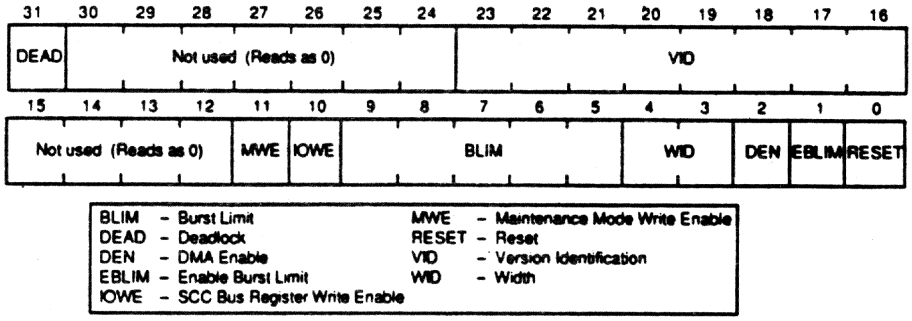


Figure C-32: MicroDMA DGCTL

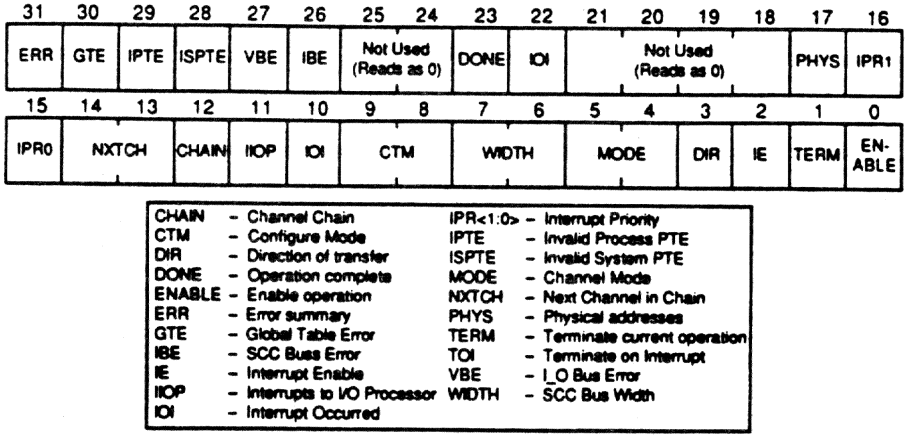


Figure C-33: MicroDMA DCCTL

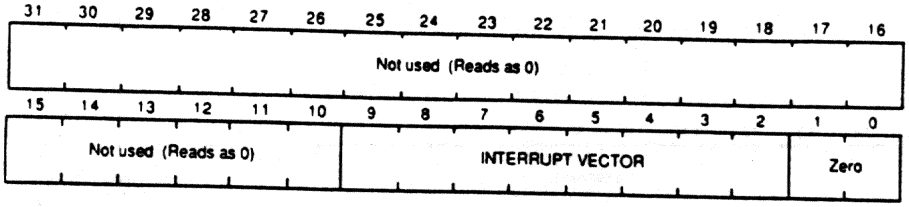


Figure C-34: MicroDMA DCINT

C.9 VIC Registers

See Chapter 10 for details of the VIC registers.

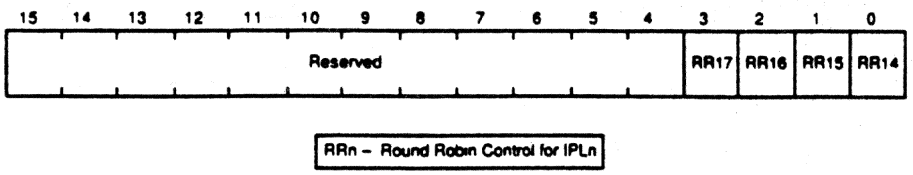


Figure C-35: VIC ROBIN

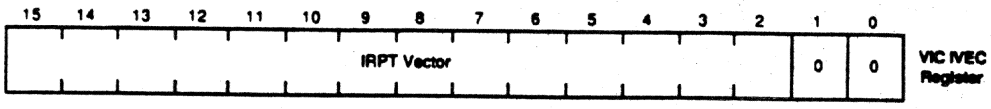


Figure C-36: VIC IVEC

Appendix D
Electrical and Environmental Specifications

D.1 Introduction

This appendix contains a summary of the DEC MicroServer's electrical and environmental specifications. The topics covered are:

- AC supply
- Internal DC supply
- Electrical characteristics of the synchronous interfaces
- General environmental specifications
- Ventilation space
- Protecting the unit against damage

D.2 AC Supply

The DEC MicroServer is manufactured in two variants:

- **DEMSA-AA**

  Set for 120 V mains supply.

- **DEMSA-AB**

  Set for 240 V mains supply.

Table D-1 sets out the specifications for the AC supply for both types of unit.

Table D-1: Electrical Specifications

Specification	120 V Supply	240 V Supply
Nominal voltage	100-120 V~	220-240 V~
Voltage range	88 to 128 V	176 to 256 V
Number of supply phases	1	1
Nominal line frequency	60 Hz	50-60 Hz
Frequency range	47 to 63 Hz	47 to 63 Hz
Maximum current consumption	6 A	4 A
Maximum input power consumption	320 W	320 W

D.3 Internal DC Supply

The DEC MicroServer uses three DC voltages, and Table D-2 shows how much current is needed at each of these voltages.

Table D-2: General Power Requirements

Voltage	Tolerance	Typical Current	Maximum Current
+5 V	+/- 5%	11.0 A	16.0 A
+12 V	+/- 5%	1.9 A	3.0 A
-12 V	+/- 5%	0.5 A	1.0 A

A 196 W module (part number: 54-15086-02) provides these voltages. The module has maximum ratings of:

- o +5 V at 20 A
- o +12 V at 7 A
- o -12 V at 1 A

The DC power is supplied through the J2 (12-18944-00) connector at the back of the power supply module. Voltages are associated with the following pins (when viewed from the back, pin 1 is to the right):

+5 V	Pins 6, 7, 8, 9
+12 V	Pin 5
-12 V	Pin 4
POK	Pin 3
DCOK	Pin 1
N/C	Pin 2
Returns	Pins 10 through 16

DC power output connects through a power cable to the logic module header connector (12-18944-08).

D.4 Electrical Characteristics of the Synchronous Interfaces

Table D-3 shows the supply voltages used by each of the synchronous interfaces. There are values for both the transmitters (line drivers) and receivers.

Also note these points:

- o The 10 V supplies are derived from the appropriate 12 V supplies through resistors and voltage regulators.
- o The External Logic Interface provides the following voltages on these pins:

  -12 V -- Pins 1, 2

+12 V -- Pins 5, 6

+5 V -- Pins 11, 37, 38, 39, 40

- The DEC MicroServer does not meet the slew rate requirements for the V.10 electrical interface because of the wide range of data rates offered.

Table D-3: Interface Electrical Characteristics

Interface	Type	Device	Power Supply
RS-232-C RS-423-A V.28 V.10	Drivers	5170	+10/-10 V
RS-422-A V.11 X.27	Drivers	26LS31	+5 V
V.35	Drivers	75113	+5 V
RS-232-C RS-423-A RS-422-A V.28 V.11 V.10 X.27	Receivers	26LS32-3 5181	+5 V
V.35	Receivers	26LS32-B	+5 V

D.5 General Environmental Specifications

Table D-4 sets out the environment in which the DEC MicroServer can be operated and stored.

Table D-4: Environmental Specifications

Specification	Operating	Non-Operating
Maximum altitude	2438 m (8000 ft)	4877 m (16000 ft)
Temperature range(*)	5 to 50 °C (41 to 122 °F)	See Note 1
Temperature change rate	20 °C/hour (68 °F/hour)	See Note 1
Relative humidity	10% to 95% See Note 2	10% to 95% See Note 2

* Reduce the maximum temperature specification by 1.8 °C for each 1000 m (1 °F for each 1000 ft) above sea level.

Notes:

1. If you store the DEC MicroServer for longer than 60 days, it should be in a place that meets the DEC MicroServer operating specifications. If you store the DEC MicroServer for less than 60 days, it should be in a place that is between -40 °C (-40 °F) and 66 °C (151 °F) with relative humidity below 95%.

2. While working or being stored, the DEC MicroServer must always be in a NON-CONDENSING environment.

D.6 Ventilation Space

When in operation, there must always be at least 10 cm (4 inches) of free space on either side of the unit to allow for correct ventilation.

To help prevent dust entering the ventilation slots, keep the DEC MicroServer at least 46 cm (18 inches) above floor level.

D.7 Protecting the DEC MicroServer Against Damage

Follow these guidelines to prevent damage to the DEC MicroServer:

DO NOT put the DEC MicroServer:

- Where it will get bumped
- Near strong sources of electromagnetic radiation (for example, photocopiers)
- Near heaters
- In direct sunlight

DO NOT:

- Lean anything against the DEC MicroServer
- Put anything on top of the DEC MicroServer
- Put drinks or other liquids on or near the DEC MicroServer

Minimize damage from static discharge by:

- Laying antistatic carpets or mats
- Using antistatic spray
- Positioning the DEC MicroServer well away from busy office corridors
- Keeping the relative humidity greater than 40%

Keep the area round the DEC MicroServer:

- Clean
- Free from dust, ash, and other abrasive materials

Appendix E
Console Commands

E.1 Introduction

In everyday use, the DEC MicroServer does not need a console. However, one becomes useful when testing a newly manufactured unit, or when trying to trace a fault in an existing unit or network.

The DEC MicroServer has two sorts of console:

- Remote Console
- Manufacturing Console

This appendix shows you:

- How to connect both types of console
- The facilities available from the console
- Some example uses of the console

Before that, however, there's a brief description of the consoles themselves.

E.2 The Console Devices

The DEC MicroServer does not have an extra communciations port that a console terminal can be attached to. As the unit does not usually require a console, this is no real problem until there's a situation where such a terminal becomes necessary.

With no dedicated physical port, other ways have to be used to connect a console:

1. Creating a link between a terminal on a host system and the DEC MicroServer
2. Attaching a terminal to the External Logic Interface on the main circuit board

The first is called a **remote console**. It is the only type of console available to customers, Field Service, and support operations. The second can be attached only if the unit is dismantled and so is reserved for manufacturing use.

The following sections detail some of the major features of the DEC MicroServer's two types of console and of the facilities each provides.

E.2.1 The Remote Console

The remote console uses the console carrier facilities of the Maintenance and Operations Protocol (MOP) in DECnet. This enables a user of a terminal on a host system to create a logical connection to a DEC MicroServer using the Ethernet. Once this connection is created, the terminal behaves as if it were a real console.

Figure E-1 shows a remote connection.

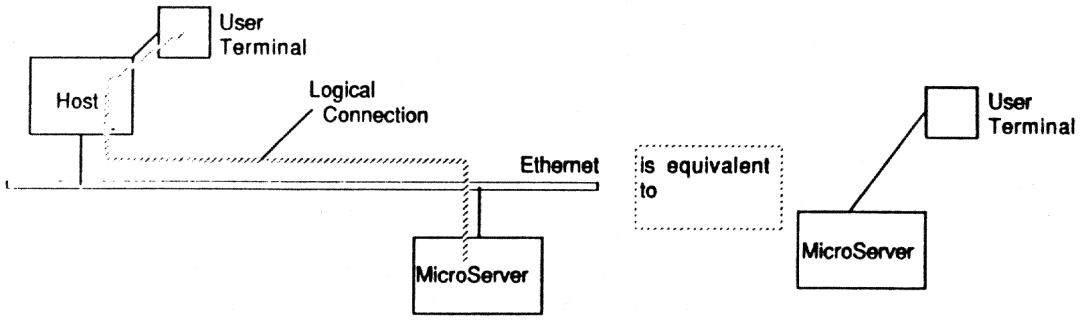


Figure E-1: The Remote Console Creates a Logical Connection to the DEC DEC MicroServer

Typically, to create a remote console connection simply involves

running a program on the host system, as Section E.3 shows.

Although the route between the terminal and the DEC MicroServer could be long, the remote console has the advantage that any terminal logged in to a host system can act as a console. This saves terminals having to be set aside for consoles and also allows someone to start tracing a problem as soon as they see it (rather than having to go to the dedicated terminal).

E.2.2 The Manufacturing Console

A console terminal can be connected to the External Logic Interface on the main circuit board using a specially designed console board. The board is designed around the DC 319 DLART device and provides a connection point for an RS-232-C terminal.

Of course, using this board means that the system box must be opened. Hence, this facility is not available outside manufacturing.

Using the console board means that a dedicated terminal has to be used, losing the flexibilty of the remote console. However, the person using a manufacturing console usually is working on a unit at board level, with dedicated equipment.

E.2.3 Facilities Available

There is a set of commands that you can use on either type of console. They have the same syntax and produce the same results. However, by using the manufacturing console some extra commands become available. These provide the more detailed testing and interrogation facilities that manufacturing and repair engineers need.

In the rest of this appendix, we'll highlight those commands that can only be used from the manufacturing console.

E.3 Connecting a Console

Follow the instructions in either Section E.3.1 or Section E.3.2 to attach a remote or manufacturing console.

E.3.1 Attaching a Remote Console

You can use terminals connected to either VAX/VMS or ULTRIX systems as a remote console. Unfortunately, the connection procedure is different for each. So, use one of Sections E.3.1.1 or E.3.1.2 as appropriate.

E.3.1.1 Connecting from a VAX/VMS System - To connect from a load host that runs VAX/VMS:

1. Log in
2. Start NCP:

   $ RUN SYS$SYSTEM:NCP

3. Enter the CONNECT command:

   NCP>CONNECT NODE node-name

   Replace node-name with the node name of the DEC MicroServer.

   If the unit has a service password assigned to it, use this form of the CONNECT command:

   NCP>CONNECT NODE node-name SERVICE PASSWORD password

   Replace node-name with the node name of the DEC MicroServer, and password with the appropriate service password.

Once the connection is complete, the following prompt appears on your terminal:

>>>

E.3.1.2 Connecting from an ULTRIX System - To connect from a load host that runs ULTRIX:

1. Log in
2. Enter the ccr command:

   % ccr -n node-name

Replace node-name with the node name of the DEC MicroServer.

If there is a service password to protect the console, use this form of the command:

% ccr -n node-name -p password

Replace node-name with the node name of the DEC MicroServer, and replace password with the appropriate service password.

When the connection is complete, the following prompt appears on your terminal:

>>>

E.3.2 Attaching a Manufacturing Console

To attach the manufacturing console:

1. Detach the power cable
2. Remove the plastic skins from the metal case
3. Open the metal case using the appropriate tool
4. Attach the console board to the External Logic Interface
5. Attach a terminal to the RS-232-C connector at the back of the console board
6. Attach the power cable and wait for the DEC MicroServer to power up
7. Press the BREAK key on the terminal, and the following prompt appears: >>>

E.4 Summary of Console Commands

Table E-1 is a summary of the commands available on the console. Sections E.4.1 to E.4.13 contain more details about each of the commands.

Table E-1: Summary of the Console Commands

Command Name	Description
BOOT	Boots the system by running the OBT and asking for a load image
CLEAR	Clears information in the NVRAM
DEPOSIT	Changes the contents of locations in the System RAM, Buffer RAM, and MicroVAX registers
EXAMINE	Displays the contents of specified locations in the System RAM, the Buffer RAM, and MicroVAX registers
DUMP	Causes the system to dump and reboot
HALT	Stops the system software or any load/dump in progress
HELP	Displays information on the syntax of the console commands
INITIALIZE	Resets the system and RAM memories
LOAD	Loads a new system image using Software ID
SET	Sets parameters in the NVRAM and the characteristics of the console terminal
SHOW	Displays information held in the NVRAM

Table E-1 (cont.)

Command Name	Description
SPECIAL	Executes one of the facilities available to manufacturing only This command is available from the manufacturing console only
START	Start execution of the system from a given PC
TEST	Do a loopback test on one or more synchronous ports

E.4.1 BOOT

You use the BOOT command to reload the system image. The command does the following:

1. Runs the OBT
2. Issues a load request on the Ethernet
3. Receives a new image from a load host
4. Starts execution of that image

E.4.2 CLEAR

As chapter 16 shows, the DEC MicroServer stores some system related information in the NVRAM, such as:

- Communications counters
- Console password

- Software ID
- Firmware ID

You can remove any of these items by using the appropriate CLEAR command:

CLEAR item

Replace item with one of the following:

COUNTERS	to return all the counters to zero
PASSWORD	to remove any console password
SOFTWARE	to remove any Software ID
FIRMWARE	to remove any Firmware ID

NOTE

The NVRAM also contains error information that can be cleared only by using the SPECIAL command, as Section E.4.11 explains.

E.4.3 DEPOSIT and EXAMINE

Occasionally, you may need to read or write specific values into one of the RAMs or one of the registers. For example, this could be useful when trying to locate a memory fault. You can use the DEPOSIT and EXAMINE commands to do these operations.

The EXAMINE command displays the contents of a location or register, and looks like this:

EXAMINE [addr-type][data-size][repeat] address

The DEPOSIT command writes data into a location, and looks like this:

DEPOSIT [addr-type][data-size][repeat] address data

In these commands:

addr-type	is the type of location to read or modify (see Section E.4.3.1)
data-size	is the size of the data to read or modify (see Section E.4.3.2)

repeat — specifies whether the access is to be repeated until the console BREAK key is pressed (see Section E.4.3.3)

address — is the start address of the data item to read or modify (see Section E.4.3.4)

data — is the information to be written to the location (see Section E.4.3.5)

The following sections explain each of these items and give some examples of the commands.

E.4.3.1 Address Type (addr-type) - The address type is a qualifier that defines the type of location you're accessing. Table E-2 shows the qualifiers you can use and explains what each means.

Table E-2: Address Type Qualifiers for the DEPOSIT and EXAMINE Commands

Qualifier	Meaning
/G	The address parameter identifies one of the MicroVAX general registers.
/I	The address parameter identifies one of the other MicroVAX registers, the System CSR, or one of the DYRC registers.
/P	The address parameter is a physical address.
/V	The address parameter is a virtual address.
If you do not specify any of these qualifiers, /P is used as the default.	

E.4.3.2 Size of the Data Item (data-size) - The data size qualifier sets out how many bytes are to be read or written.

Table E-3 lists the qualifiers you can use, and what each means.

Table E-3: Data Size Qualifiers for the EXAMINE and DEPOSIT Commands

Qualifier	Meaning and Use
/B	One byte is to be read or written
/W	One word -- that is, two bytes -- is to be read or written
/L	One longword -- that is, four bytes -- is to be read or written
If you do not supply one of these qualifiers, the firmware uses /L as the default.	

E.4.3.3 Repeated Access (repeat) - When locating an intermittent fault, or when debugging revised firmware, it is useful to be able to read or modify a location repeatedly. For example, a location may get corrupted, and to help find the cause it can help to read that location repeatedly.

If you add the /R qualifier, the command is repeated until you press the BREAK key on the console terminal.

E.4.3.4 Location Address (address) - This parameter is the hexadecimal address of the location to be read or modified. The addr-type qualifier determies the type of location to be accessed:

1. Virtual memory address
2. Physical memory address
3. General MicroVAX register
4. Special MicroVAX register, System CSR, or DYRC register

In the last case, the number determines the precise register that is accessed, as Table E-4 shows.

Table E-4: Address Values for Special MicroVAX, System CSR, and DYRC Registers

Address Value	Register Accessed
000	Kernel Stack Pointer
001	Executive Stack Pointer
002	Supervisor Stack Pointer
003	User Stack Pointer
004	Interrupt Stack Pointer
005 to 007	Not used
008	P0 Base Register
009	P0 Length Register
00A	P1 Base Register
00B	P1 Length Register
00C	System Base Register
00D	System Length Register
00E and 0F	Not used
010	Process Control Block Base Register
011	System Control Block Base Register
012	Interrupt Priority Level Register
013	AST Level Register
014	Software Interrupt Request Register
015	Software Interrupt Summary Register
016 and 017	Not used

Table E-4 (cont.)

Address Value	Register Accessed
018	Interval Clock Control Register
019 to 037	Not used
038	Memory Management Enable Register
039	Translation Buffer Invalidate Single Register
03A	Translation Buffer Invalidate All Register
03B to 03D	Not used
03E	System Identification Register
03F	Translation Buffer Check Register
040 to 09F	Not used
100	Program Status Longword (PSL)
101	System CSR
102	CSR from System RAM's DYRC
103	FAR from System RAM's DYRC
104	CSR from Buffer RAM's DYRC
105	FAR from Buffer RAM's DYRC

E.4.3.5 Data to Deposit (data) - This parameter is the hexadecimal value to be deposited in the given location.

E.4.3.6 How to Use the Commands - If you're unfamiliar with using these commands, use these guidelines to help get them correct:

1. First enter the appropriate address type qualifier
2. Enter the data size qualifier
3. Add the repeat qualifier if necessary
4. Enter a space followed by the appropriate address expression
5. If you're using the DEPOSIT command, enter another space followed by the data value

E.4.3.7 Notes on Using the Commands - Bear the following in mind when using the DEPOSIT and EXAMINE commands:

1. When using the commands from a remote console, you can access the System RAM and Buffer RAM only. This restriction also applies if you're using the manufacturing console on a running system.
2. When using the manufacturing console, you can access any location or register once the system is halted. However, such accesses may cause a machine check (for instance, if you try to access nonexistent memory locations).
3. If the system dumps after you've used the DEPOSIT command and before the next normal reboot, the resultant dump file will indicate that memory had been modified.
4. The console firmware saves a number of the MicroVAX registers so that it can run. So, accessing these registers actually refers to the saved copies and not to the registers themselves. A modified version of one of these registers does not come into effect until you use the START command.
5. If you use the /W or /B qualifiers with a 32-bit register, the least significant part of the value is accessed.

E.4.3.8 Example Commands - Here are some examples of the DEPOSIT and EXAMINE commands.

To read one byte of data in the Buffer RAM at physical address 100ABCDE, use:

```
EXAMINE/P/B 100ABCDE
```

To change the value in MicroVAX general register R0 to the hex number 1234, use:

```
DEPOSIT/G/L 0 1234
```

To display the contents of the System CSR, use

```
EXAMINE/I/W 101
```

E.4.4 DUMP

Occasionally, you may need to dump the contents of the system for later examination. To do this from a console, enter the DUMP command, which causes the following to occur:

1. The DEC MicroServer sends a dump request over the Ethernet and waits for a reply.
2. When a host answers the request, the DEC MicroServer sends it the contents of memory (including the NVRAM).
3. When the dump is complete, the DEC MicroServer runs the OBT and reboots.

While the dump is in progress, some status messages will appear on the console terminal. However, you cannot enter any more commands until the dump completes (or fails).

E.4.5 HALT

The HALT command stops the communications software. Any system load or dump in progress when the command is issued is stopped.

Use the HALT command before using any of the following commands (otherwise the results of these commands could be unpredictable):

- BOOT
- DUMP

- LOAD
- TEST

E.4.6 HELP

There are two levels of help information available. By simply entering:

```
HELP
```

A list of the customer usable subset of commands appears on your terminal, as shown in Figure E-2.

```
>>>help

Commands -

Boot
DUmp
HAlt
Help [Test]
SEt Password
SHow {Password,STate}
TEst keyword [/keyword_qualifiers...]

CAPITALS - required letters, lowercase - optional letters
[...] - Optional, {...} - Select One

>>>
```

Figure E-2: The Help Information for the User Subset of Commands

As you can see, the command list contains details of the syntax of all the commands (except TEST). For information on the syntax of that command, enter:

```
HELP TEST
```

To get a complete listing of all the commands and their syntax, enter:

```
HELP MORE
```

E.4.7 INITIALIZE

The INITIALIZE command resets the DEC MicroServer to an initial state as it would be at power up but before software is loaded. In particular, INITIALIZE:

1. Resets all I/O devices
2. Clears all RAM space (except that reserved for the firmware)
3. Sets all internal registers to their default values

E.4.8 LOAD

Use the LOAD command to load (but not start) a new system image. A load host is requested and the image loaded in the normal way.

If you want to load a specific image, add its Software ID as a parameter to the LOAD comand. For example:

```
LOAD TEST1
```

This is particularly useful when you want to load a test package.

To start the newly loaded image, use the START command.

E.4.9 SET

SET commands are mainly used to establish values for certain fields in the NVRAM. For example, one of the commands sets up the console password. There are other commands you can use to tell the firmware what sort of terminal is being used as the console.

Generally, the SET command looks like this:

```
SET topic[qualifier] [parameter]
```

where:

topic — identifies what value is being set, and is one of:

- PASSWORD
- SOFTWARE
- FIRMWARE
- HARDCOPY
- SCOPE

qualifier — is a qualifier for the command. Not all the commands use qualifiers.

parameter — is the value to be set up. Again, not all the commands have a parameter.

The following sections deal with each of these commands in more detail.

E.4.9.1 PASSWORD - Use the SET PASSWORD command to create a console password. Use a command like this:

```
SET PASSWORD password
```

Replace password with up to 16 hexadecimal digits.

For example:

```
SET PASSWORD FEFEFEFEFEFEFEFE
```

To change a password, simply use another SET PASSWORD command.

NOTE

If you omit the password from the command, the firmware prompts for it and then asks for confirmation (when you type the password in again). Your reponse to each prompt is not echoed.

E.4.9.2 SOFTWARE - Use the SET SOFTWARE command to set up a Software ID to be used when the system is booted. The DEC

MicroServer includes the Software ID in the LOAD REQUEST it sends on the Ethernet. This is used by the load host to determine the image to send to the DEC MicroServer.

The basic form of the command looks like this:

SET SOFTWARE software-id

Replace software-id with the appropriate Software ID. You can use up to 16 characters each of which can be a letter, a digit, a dollar sign ($), or an underscore (_).

For example:

SET SOFTWARE router

The other form of the command uses qualifiers instead of a parameter:

SET SOFTWARE/qualifier

Replace /qualifier with one of the following:

/NONE	To indicate that no Software ID is to be used
/OPERATING	To indicate that the standard operating system is to be loaded
/MAINTENANCE	To indicate that the maintenance operating system is to be loaded

These qualifiers provide a shorthand, standard way of specifying generic systems to be loaded.

E.4.9.3 FIRMWARE - A separate Software ID is used to load a RAM version of the VAX Firmware. If specified, then the alternate version of the firmware is loaded before sending the LOAD REQUEST for the software.

To set up a "Firmware ID", use this command:

SET FIRMWARE firmware-id

Replace firmware-id with the appropriate software ID. You can use up to 16 characters, each of which can be a letter, a digit, a dollar sign ($), or an underscore (_).

For example:

```
SET FIRMWARE consoletest
```

To change the Firmware ID, simply use another SET FIRMWARE command. To remove the ID, use the appopriate CLEAR command.

E.4.9.4 HARDCOPY and SCOPE - The console firmware usually assumes that it is communicating with a VDU type of terminal. However, the way that the DELETE key is handled on a VDU differs from that on a hardcopy terminal.

So, you need to tell the console firmware what sort of terminal you have.

If you are using a hardcopy terminal, use this command:

```
SET HARDCOPY
```

If you move to using a VDU type of terminal, use this command:

```
SET SCOPE
```

SET SCOPE can be useful if you use a hardcopy terminal for part of a console session. When you revert to using the VDU terminal, you use the SET SCOPE command.

E.4.10 SHOW

To display the contents of various fields in the NVRAM, use the appropriate SHOW command. The command looks like this:

```
SHOW topic
```

Replace topic with one of the following:

ADDRESS	Displays the Ethernet hardware address of the DEC MicroServer (the address stored in the Ethernet PROM).
COUNTERS	Displays a list of statistics for the Ethernet port. This list includes general information (such as how many bytes have been sent and received) as well as error information (such as the number of receive failures).
ERRORS	Displays the contents of the Error Log. This can

be particularly useful when first examining a unit returned from the field. Figure E-3 shows an example display.

HALT — Displays the halt code when the system software stopped running.

MCHECK — Displays information on the last machine check that occurred in the DEC MicroServer. This information is also useful when examining a returned unit. Figure E-4 shows an example display.

PASSWORD — Displays the current service password as a 16-digit hexadecimal number. If there is no service password, the number 0000000000000000 is displayed.

SOFTWARE — Displays the Software ID field. If none has been defined, the string "*NONE*" appears.

FIRMWARE — Displays the Software ID used to load a RAM version of the firmware. If none has been defined, the string "*NONE*" appears.

STATE — Displays the current system state. This can be one of the following:

- Halted
- Requesting Boot
- Booting
- Deferrring Boot
- Requesting Dump
- Dumping
- Running

VERSION — Displays the system revision number, the system code number, and the version number of the DEC MicroServer hardware.

ALL — Displays all of the previous items one after the other. The items appear in the same order as shown here.

SHOW ERRORS and SHOW MCHECK are useful when fault finding on new units or on those returned from the field. Figures E-3 and E-4 give examples of both commands. Note that in Figure E-4, the DYRC register information appears only if the registers were accessible when the machine check occurred.

```
)))show error

Node: 08-00-2B-04-4D-8A, OBT V3.9, Current Boot# 115

   PREVIOUS ERRORS                          CODE  COUNT  BOOT#
Channel 0 DUSCC data error                  8600    4    109
Channel 2 exerciser fault                   8601    1    111
Ethernet external loopback fault            6804    1    112
Channel 1 exerciser fault                   8501    1    113

   LAST ERROR
Ethernet external loopback fault            6804         114

      R0       R1       R2       R3       R4       R5
00000000 00006804 000FF26C 0000A2F3 36000000 00000021
      R6       R7       R8       R9       R10      R11
00000000 00000000 00000000 00000000 0000000B 8000080F
```

Figure E-3: Example Display from the SHOW ERRORS Command

```
>>>show mcheck

MACHINE CHECK LOG
-----------------

0000000C
00000080
30000004
02080007
20044BE9
041F0005

MAIN DYRC CSR 1000, FAR 1800    IO DYRC CSR 1000, FAR 1800

NO PARITY ERRORS

>>>
```

Figure E-4: Example Display from the SHOW MCHECK Command

E.4.11 SPECIAL

The SPECIAL command provides access to those facilities available from the manufacturing console only (although this can be changed for manufacturing repair purposes).

The command looks like this:

SPECIAL special-command [parameters]

This is the only command that does not have an abbreviation. Table E-5 lists all the commands that can appear in place of special-command. The table also shows which of these takes parameters.

Table E-5: The SPECIAL Commands

Command	Meaning
CLEAR_PARAM b	Clears bit b in the LES parameter field of NVRAM.
DUMP xxx yyy	Displays the contents of yyy bytes of memory starting from address xxx.
DUMP_NVRAM xxx yyy	Displays the contents of yyy bytes of NVRAM starting from offset xxx.
ECO	Prompts you to enter new ECO text into NVRAM (up to 64 ASCII characters)
HARDWARE	This command does the same as REVISION.
HELP	Displays a list of the SPECIAL commands
INIT_NVRAM	Initializes the NVRAM.
NAX	Starts the NAX responder code. This, together with complementary software in another system, can be used for Ethernet testing. For example, the other machine could be a PDP-11 system running LUNAX.

Table E-5 (cont.)

Command	Meaning
OPTION b [n]	Changes bit b of the Firmware Options field in NVRAM to the value of n (either 1 or 0). The available bits are: 0. SNAP SAP processing for MOP messages 1. Load RAM versions of the VAX firmware 2. Use of SPECIAL command from remote console 3. Use of MOP V3.1 4. Use of MOP V4.0 To enable any of these options, set the appropriate bit to 1. See Chapter 16 for more information on how to set bits 3 and 4. NOTE If you set bit 2 to a remote console to use the SPECIAL command, make sure the bit is cleared before the unit is sent to the field.
SERIAL	Sets the serial number field in the NVRAM
SET_PARAM b	Sets bit b in the LES parameter field of NVRAM
TEST test c	Runs one of the OBT's manufacturing tests. See Section 12.3 for details of how to use the tests and the values of the parameters.

Table E-5 (cont.)

Command	Meaning
UNLOAD	Unloads the RAM version of the console firmware. If the system is running, it is allowed to continue and future calls to the firmware will use the ROM version instead of that in RAM. If the system is not running, control is returned to the ROM-based version of the console firmware.
REVISION	Prompts for the hardware revision level to appear in the SHOW VERSION command. For example, entering a value of B for this command will cause the first line of the SHOW VERSION command to look like this: Hardware Revision: 02. B01
VARIANT	Enters the 'variant' number of the Microserver into NVRAM.
VERSION	This command does the same as REVISION.
XDT	Starts the XDT debugging tool. Section F.5 gives details of how to use XDT.

E.4.12 START

Use the START command to start execution of a system. Typically, you use this command after a LOAD command to start the newly loaded system.

The command looks like this:

```
START [start-addr]
```

Use start-addr to state the address program where execution is to begin. Be sure to specify the address in hex.

If you do not include start-addr, execution starts at the address in the PC. Note, that the LOAD command automatically sets the PC to the correct start address.

E.4.13 TEST

Use the TEST command to exercise the synchronous communications ports, their modem control lines, and the adapter cables. Another form of the command lets you continuously exercise the complete DEC MicroServer.

The first format of the command is:

```
>>>TEST type channel /qualifier
```

where:

type determines the sort of test you want to do. Use one of the following words:

```
DATA_LOOPBACK
MODEM_SIGNAL
CABLE_TYPE
```

channel defines which of the synchronous lines the test is to run on. You can use one or more of the following:

```
/CH0
/CH1
/CH2
/CH3
/ALL
```

/qualifier is a qualifier that can be added to the TEST DATA_LOOPBACK command to specify the sort of test you want to do. If none are specified, the default is the 50-way loopback test. You can use one of the following values:

```
/INTERNAL
/MANUAL
/LOCAL
/REMOTE
```

The command to test the DEC MicroServer continuously is:

```
>>>TEST CONTINUOUS
```

The following sections show what each test does, and give examples of their use.

E.4.13.1 Data Loopback Tests - The data loopback tests use a connector in the communications path. This loopback connector joins the transmit and receive paths so that all information sent is returned. The tests send test data along the line and wait for the looped data to return. The returned data is compared with what was transmitted. Differences between the two copies of the data indicate that an error on the path is corrupting the data.

Figure E-5 shows the points where information can be looped back. You use different forms of the TEST DATA_LOOPBACK command for each point, as the figure indicates.

For example, to loopback through the local modem on channel 1, you use either:

```
>>>TEST DATA_LOOPBACK/CH1/LOCAL
```

or:

```
>>>TEST DATA_LOOPBACK/CH1/MANUAL
```

Use the first command if the modem recognizes the Local Loop modem signal. Use the second if you have to set the modem into loopback mode manually (for example, by pressing a switch on the modem).

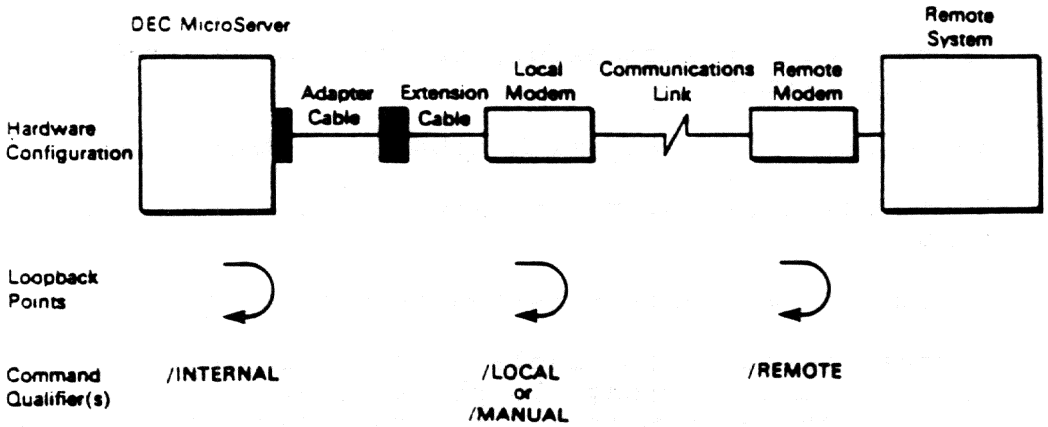


Figure E-5: Loopback Points on a Communications Path

E.4.13.2 Modem Signal Tests - The modem signal tests operate on the modem signals between the DEC MicroServer and the local modem. The signals can be looped back either at the DEC MicroServer's synchronous port or at the end of the adapter cable (see Figure E-6).

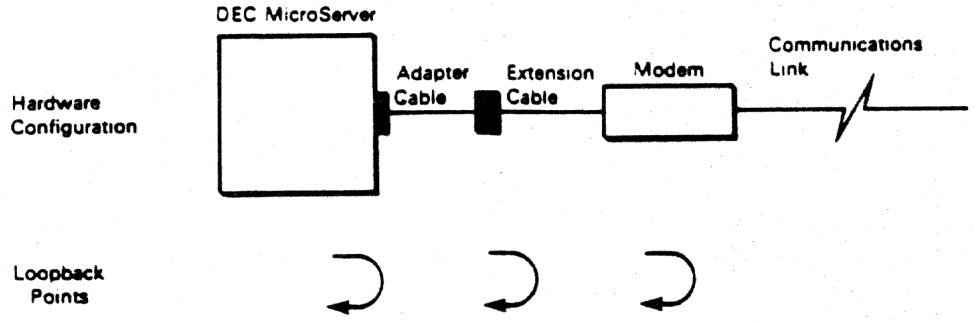


Figure E-6: Loopback Points that Can Be Used with the Modem Signal Test

To do a modem signal test you:

1. Attach a loopback connector to one of the loop points.
2. Enter:

   >>>TEST MODEM_SIGNAL/CHn

   Replace n by the number of the line you want to test (0, 1, 2, or 3).
3. If the test is successful, the following message appears on your terminal:

   Successful Modem Signal Loopback on Channel n
4. If the test fails, the following message appears on your terminal:

   Modem Signal Loopback Failed on Channel n

One use of these tests is to isolate a faulty adapter cable. First, attach the 50-way loopback connector to the DEC MicroServer's synchronous port and run the test. Then repeat the test, but this time with an appropriate loopback connector on the end of the adapter cable. If there is a faulty adapter cable, the first test passes but the second test fails.

E.4.13.3 Cable Type Tests - There is a range of adapter cables available for the DEC MicroServer. These provide easy connection to various interface standards. Each type of cable contains a unique identifier.

Using the cable type test, you can find out the type of cables attached to the DEC MicroServer.

To find out the cable attached to a particular line, make sure there is no loopback connector on the port, and then use:

```
>>>TEST CABLE_TYPE/CHn
```

To find out all the cables attached to the DEC MicroServer, make sure there are no loopback connectors on any port, and then use:

```
>>>TEST CABLE_TYPE/ALL
```

For each cable, the test displays a message like this:

type Cable on Channel n

If, however, any port has a 50-way loopback connector on it, the test displays a message like this:

H3199 Loopback on Channel n

If the test cannot determine the type of cable, it displays a message like this:

Unsupported Cable on Channel n, Cable Code xxxx

If this message appears for a supported type of cable, check that the cable is correctly connected to the socket, and then try the test again. If the message appears again, the adapter cable is probably faulty.

E.4.13.4 Continuosly Running the OBT - The DEC MicroServer's internal tests are usually run during power up or when the system is being relaoded. You can, however, run them continuously to soak test a DEC MicroServer unit. To do this you:

1. Attach any loopback connectors you want to use to the DEC MicroServer's synchronous ports

2. Enter:

```
>>>TEST CONTINUOUS
```

The unit will now continuously run the internal test program. To reload the system, power down the DEC MicroServer, wait three seconds and power it up again.

E.5 XDT

XDT is a basic hardware debugging tool, designed to run with the minimum working hardware. XDT helps isolate faults that have even prevented the on-board self-test running.

XDT runs if these components are working:

- MicroVAX processor
- Firmware ROM
- Manufacturing console

To use the TMS 32020 features of XDT, the following also have to be working:

- TMS 32020
- The DMA controllers can be used in IO-ACCESS mode to load the TMS RAM

XDT uses no RAM on the MicroVAX side of the DMA controllers, keeping its state information in registers.

NOTE

XDT does not preserve the state of any registers when it starts, so a dump taken after using XDT may contain misleading information.

E.5.1 Starting XDT

You can use XDT only from the manufacturing console. To run the tool, use the following command:

```
SPECIAL XDT
```

E.5.2 XDT Commands

Table E-6 lists all the commands you can use with XDT. Specify all location addresses in hexadecimal.

Table E-6: XDT Commands

Command	Description
xxx/	Open and display location xxx
xxx=	Open location xxx without displaying it
<RETURN>	Close current location
xxx<RETURN>	Update and close current location
<LINEFEED>	Open next location
xxx<LINEFEED>	Update location xxx and open next location
^	Open previous location
xxx^	Update location xxx and open previous location
.	Refers to the current location
xxxG	Start executing code from location xxx
H	Halt TMS executing code.
I	Interpret subsequent addresses as TMS 32020 I/O addresses.

Table E-6 (cont.)

Command	Description
xxxJ	Start the TMS 32020 executing code at address xxx.
L	Load program file from console. See Section F.5.3 for the format of the file.
P	Interpret subsequent addresses as locations in the TMS 32020 Program space.
R	Repeat last store/fetch instruction indefinitely.
nS	Set data size to n, where n is 1, 2, or 4 (bytes).
T	Fill the TMS RAM with TRAP instructions.
V	Interpret subsequent addresses as addresses in the TMS 32020 data space.
W	Interpret subsequent addresses as a VAX address to use in the DMA controller as a base address for TMS 32020 window operations.
<CTRL/Z>	Exit to self test. Use this after any of the I, P, V or W commands to return address interpretation to the MicroVAX.
<DELETE>	DELETE can be used to correct errors when entering hexadecimal numbers.

E.5.3 Load File Format

The L command reads a program file from the console device. This file must be in MOSTEK hexadecimal format, as follows.

Each line of the file has this format (all characters, apart from the initial semicolon, are ASCII hexadecimal characters - 0 to 9, and A to F):

```
; bc aaaa dd dd dd dd.....  dd cccc
```

Each line contains these components:

bc — Data byte count (two hexadecimal characters)

aaaa — Load address for the first byte in this line (four hexadecimal characters)

dd — Data (two hexadecimal characters at a time) for a byte

cccc — Checksum: the sum of all the bytes in the line, that is, bc + aaaa + dd + ... + dd (four hexadecimal characters)

The last line in the file should have:

- bc = 00
- aaaa indicate the number of lines in the whole file
- No dd fields
- No cccc field

While reading the file, the console firmware runs a 30 second timeout. If no character is received by the end of 30 seconds, the load is aborted.

After the file loads, a message indicates if there are any errors.

Note that XON/XOFF are not used by the console XDT firmware, because the firmware is quick enough to receive the load file at 9600 bits/s without losing any characters.

Appendix F
Dump file format

F.1 Dump File Structure

The address space in the DEC MicroServer is a number of distinct areas. These areas do not always lie next to one another and there are large gaps in the address space. The dump file contains just those areas that are used, together with information that enables a dump analyzer to reconstruct the original address space.

The extra information is held in the first two pages (Pages 0 and 1) of the dump file. Page 0 contains a set of descriptors, one for each area of the address space in the file. Page 1 contains copies of the MicroVAX registers. These are included for compatability with the old types of dump analyzer. The remainder of the dump file contains the address space segments.

The following sections explain the content of the dump file in more detail. Section F.3 explains how the descriptors are used, and shows what they contain. Section F.4 shows the content of the second page of the dump file.

Before that, however, Section F.2 lists the areas of the MicroVAX address space that are included in the dump file.

F.2 Address Areas Included in the Dump File

The dump file includes the following:

- System RAM
- Buffer RAM
- TMS RAM
- NVRAM
- Ethernet PROM

- Firmware ROM
- DMA controller registers (both devices)
- LANCE registers
- DUSCC registers
- DYRC registers (both devices)
- System CSR
- TMS 32020 I/O space
- VIC registers

F.3 The Area Descriptors

The first page of the dump contains the descriptors -- one for each of the memory areas in the dump file. These descriptors contain information to tell a dump analyzer where each segment is, how large it is, the sort of information it contains, and so on. From this, the analyzer can reconstruct the original address space.

Each descriptor consists of four longwords, as Figure F-1 shows. Table F-1 explains the content of each field in the descriptor.

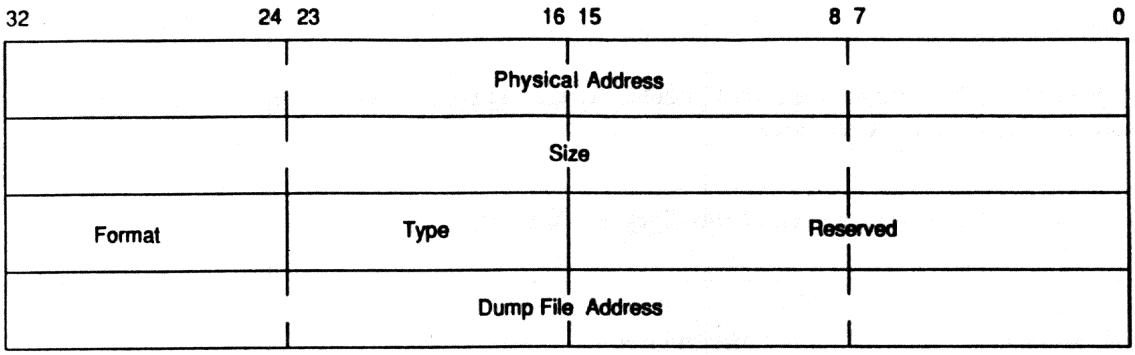


Figure F-1: Structure of a Descriptor in the Dump File

Table F-1: Fields in a Dump File Descriptor

Field Name	Description
Physical address	The start address of the area in the MicroVAX physical memory map.
Size	The number of bytes in the area.
Type	The type of the area. See Section G.3.1.
Format	The format of the area. See Section G.3.2.
Dump file address	The location of the area in the dump file.

F.3.1 Types of Area

The Type field in the descriptor tells the analyzer what sort of information the area contains. This field contains a name of the form:

USV$K_name

Table F-2 lists the values that can appear in place of name, and explains what each means.

Table F-2: Contents of the Type Field

Type Name	Meaning
TYPE_MAIN_MEMORY	System RAM
TYPE_IO_MEMORY	Buffer RAM
TYPE_ENET_ROM	Ethernet PROM
TYPE_DYRC	DYRC registers for both devices
TYPE_DMA	DMA controller registers
TYPE_DUSCC	DUSCC registers
TYPE_LANCE	LANCE registers
TYPE_VIC	VIC registers
TYPE_CSR	System CSR
TYPE_NVRAM	NVRAM

Table F-2 (cont.)

Type Name	Meaning
TYPE_TMS_MEMORY	TMS RAM, including the TMS 32020 context
TYPE_TMS_IOSPACE	TMS I/O space
TYPE_WATCHDOG	State of the Watchdog timer
TYPE_ROM	Firmware ROM
TYPE_DISPLAY	Display register

F.3.2 Formats of a Dumped Area

The Format field of a descriptor tells the analyzer how the information in the dump file is stored. Like the Type field, this field contains a name of the form:

USV$K_name

Table F-3 lists the values that can appear in place of name and explains what each means.

Table F-3: Contents of the Descriptor Format Field

Format Name	Meaning
FORMAT_NORMAL	Unformatted dump
FORMAT_PACKED_WORDS	Words on longword boundaries are packed into contiguous words in the dump file

Table F-3 (cont.)

Format Name	Meaning
FORMAT_PACKED_BYTES	Bytes on longword boundaries are packed into contiguous bytes in the dump file
FORMAT_WORD_CSRS	Word CSRs not necessarily on consecutive longword boundaries are packed into contiguous words in the dump file
FORMAT_BYTE_CSRS	Byte CSRs not necessarily on consecutive longword boundaries are packed into contiguous bytes in the dump file

F.4 The Register Dump

The second page of the dump contains the MicroVAX registers in place when the dump was taken. These are included to maintain compatability with earlier dump analyzers. However, more recent analyzers should use the descriptors and the contents of the System RAM to collect register information.

This part of the dump file contains the following:

Stack pointers:

- Kernel
- Executive
- Supervisor
- User
- Interrupt

General registers:

- 0 to 11

Pointers:

- Argument

Frame
Stack

Program counter

Program status

P0 base

P0 length

P1 base

P1 length

System base

System length

Process control block base

System control block base

AST level

Software interrupt summary

Interval clock control

Interval count

Time of year

Appendix G
System Test

G.1 Introduction

System test is one of the last stages in a DEC MicroServer's manufacture. The test makes sure that each unit produced can:

- Boot correctly
- Loop data through all communications ports
- Dump correctly

In addition, the test prints the Ethernet address label that's attached to the back of the unit.

You direct this testing through a test station. This appendix shows how to do this and how to use the extra facilities of that station.

G.2 The Test Station

Figure G-1 shows the components of the test station and how a DEC MicroServer is attached to it.

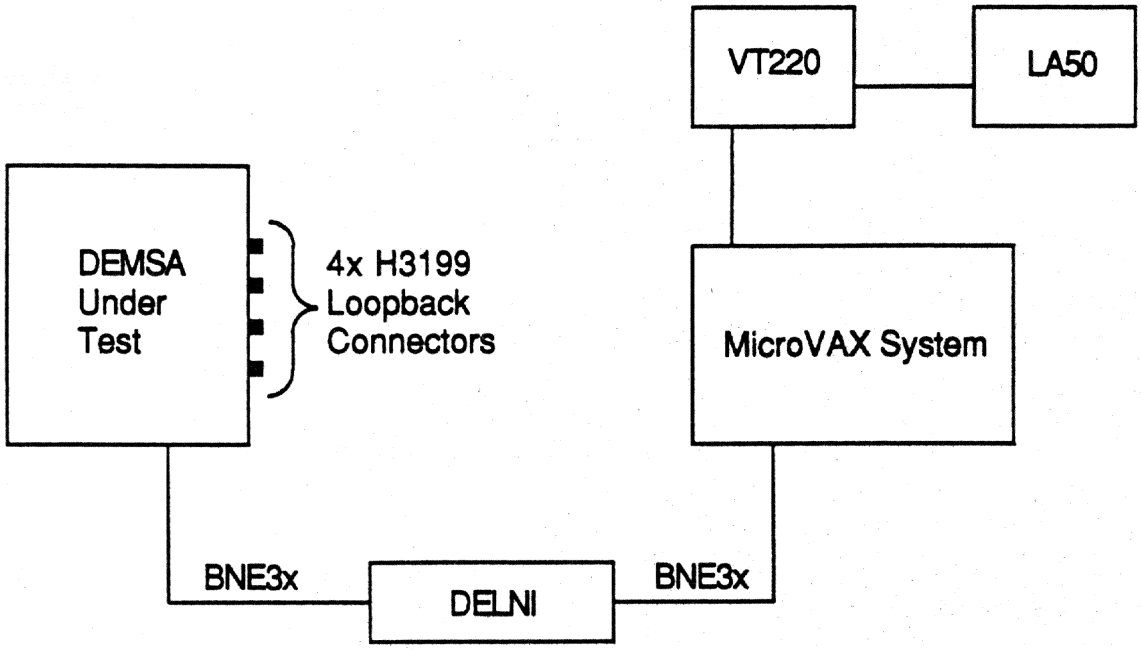


Figure G-1: The Test Station with DEC MicroServer

The MicroVAX system is made up of:

- MicroVAX II with the MS650 memory option
- DEQNA interface to the DELNI
- TK50 tape cartridge
- 2 RD53 disk units

The system runs V4.5 (or later) of MicroVMS together with the full routing license key for DECnet-VAX.

G.3 Using the Test Station

You can do three things from the test station:

- Test DEC MicroServer units

- Modify parts of the DEC MicroServer's NVRAM
- Control the test station

The testing is largely automated, but you still need to:

- Attach each unit to the DELNI
- Attach the synchronous loopback connectors
- Press the DUMP switch when necessary

Section G.5 explains the testing in more detail.

You use commands to modify the NVRAM and to control the test station. Section G.7 describes these commands and shows you how to use them.

G.4 Starting the Test Station

You start the test station by turning the power on or by pressing the RESET button (if the MicroVAX is already powered up). This starts an initialization sequence:

1. Loads and checks the test station software
2. Enables automatic login of the terminal

When this is complete, press the RETURN key to automatically login and start testing DEC MicroServer units.

If either step in the initialization fails:

1. Automatic login is not enabled
2. The system runs as an ordinary MicroVMS system

In this case, log in to the SYSTEM account and correct the error preventing startup.

NOTE

The checking process calculates checksums and compares them with expected values in one of the software files. When correcting an error, you may need to recalculate the expected values. Do this by excuting the command procedure MSASYS$:GENCHECK.COM.

G.5 Testing DEC MicroServer Units

Once you've started the station, you can start testing DEC MicroServer units. As we've already seen, the test is largely automated, but you still need to attach cables and press the DUMP switch.

The software goes through a number of steps and uses the test station's terminal to tell you which step it's on. If the station needs you to do something before completing a step, it tells you what's needed and waits. Once you've completed the action, simply press the RETURN or the DO key on the terminal. If you need help at any stage, press the HELP or PF2 keys.

Excluding the time for your actions, the test takes around five minutes for each DEC MicroServer. If the test works correctly, an Ethernet address label is printed. Otherwise, an error code is printed.

The point reached in the test sequence is indicated by the information in the 'Step Name' field on screen. As the test progresses through each of its steps, the Step Name part of the screen changes accordingly.

When you have to do something, the software indicates you should act by an on-screen prompt. Whenever one of these prompts appears, carry out the action, and then press RETURN or DO.

Table G-1 summarizes the steps in the test process, and what (if anything) you have to do for each step.

Table G-1: Steps in the System Test

	Test Description	What to Do
1	Get ready to test the new unit	1. Attach H3199 loopback connectors to all synchronous ports 2. Attach one end of the power cord to the DEC MicroServer

Table G-1 (cont.)

	Test Description	What to Do
2	Apply power and load software. Then determine Ethernet address and initialize NVRAM.	1. Attach the Ethernet cable to the appropriate port 2. Connect the DEC MicroServer to the mains supply
3	Check all the communications ports: 1. Use DECnet loop messages to check the DEC MicroServer can send and receive Ethernet messages 2. Use internal loopback on all four synchronous ports 3. Use external loopback -- through the H3199 connectors -- on all four synchronous ports	None

Table G-1 (cont.)

	Test Description	What to Do
4	Check that the DUMP switch works correctly: 1. Check that there are no events logged that could indicate failure of the DEC MicroServer 2. Wait for the DUMP switch to be pressed 3. Wait up to 60 seconds for the DEC MicroServer to request a dump	When prompted: 1. Press the DUMP switch 2. Press RETURN or DO on the terminal keyboard
5	Print Ethernet label and complete test	1. Remove the printed label and stick to the indicated place on the back panel 2. Remove the loopback connectors 3. Disconnect all cables from the DEC MicroServer 4. Remove the DEC MicroServer from the test station

Table G-1 (cont.)

	Test Description	What to Do
6	Fault Sequence. If any of the previous steps fail: 1. Print the error code 2. Ask operator to attach this to the unit	1. Remove the label and attach to the DEC MicroServer 2. Remove the loopback connectors 3. Disconnect all cables from the DEC MicroServer 4. Return the faulty unit to the appropriate engineer

G.6 Error Codes

If the system test fails, it prints an error code on the test station's printer. Table G-2 lists all the error codes that it can print together with an explanation of each.

Table G-2: System Test Error Codes

Error Code	Meaning
BADEVT	Error indicated by network event
DMPSW	No dump requested (possible faulty dump switch)
ELOOP0	External loopback error on channel 0
ELOOP1	External loopback error on channel 1

Table G-2 (cont.)

Error Code	Meaning
ELOOP2	External loopback error on channel 2
ELOOP3	External loopback error on channel 3
ETHLOOP	Ethernet loopback error
ILOOP0	Internal loopback error on channel 0
ILOOP1	Internal loopback error on channel 1
ILOOP2	Internal loopback error on channel 2
ILOOP3	Internal loopback error on channel 3
LINIT	Error loading NVRAM initialization program
LRQST	No load request after 2 minutes (possible OBT failure)
LRTR	Error loading DECrouter image
RELOAD	MicroServer reloaded software during testing
SETUP-E	Error setting up external loopback
SETUP-I	Error setting up internal loopback

G.7 Other Facilities of the Test Station

The test station recognizes a set of commands that allow you to:

1. Control the operation of the test station
2. Modify the information put in the NVRAM by its initialization program.

You can't enter these commands when the station is testing a unit. Instead, you put the station into command mode.

To do this, wait until the station is asking for Step 1 in the test sequence to be completed and then:

1. Press CTRL/Z or F10 on the terminal keyboard
2. When prompted, enter the command mode password

Once in command mode, the following prompt appears:

```
MSASYS>
```

Table G-3 lists the commands you can enter and provides a brief description of each.

Table G-3: Test Station Commands

Command Format	Function
Test Station Control Commands	

Table G-3 (cont.)

Command Format	Function
EXIT	Leave the system test software. The VAX/VMS prompt appears on the screen, with the terminal being logged into the MSASYS account. This command allows you to change files and similar operations without shutting down the whole station. To restart the system test software, enter this command: MSASYS
HELP	Provides help on the commands available and how to use them. You can add the name of a command to get more detailed information.
INITIALIZE	Reinitializes the test station. To do this, the station is automatically shut down and immediately rebooted. On restart, the system test screen is displayed.
START	Start testing DEC MicroServer units. This leaves command mode and the test display reappears.

Table G-3 (cont.)

Command Format	Function
STOP	Shut down the test station. To restart the system either: o Press the RESET button on the MicroVAX II or o Power up the MicroVAX II
SET CHECKSUM	Recalculates the check sums of all the files in the system test software. This creates a new version of the file CHECKSUM.DAT. Always use this command after using SET ECO, SET REVISION, or SET VARIANT.
SHOW VERSION	Displays the version numbers of all the software components in system test. When reporting a problem, always quote these numbers.
NVRAM Initialization Commands	
SET ECO eco-text	Changes the value of the ECO string put into the NVRAM when it's initialized. When you enter this command, the system displays a list of the new initialization values and prompts for confirmation. When you answer the prompt, the station applies the change.

Table G-3 (cont.)

Command Format	Function
SET REVISION rev-code	Changes the value of the revision string put into the NVRAM when it is initialized. The string has up to four characters: one or two letters followed by two digits. When you enter this command, the system displays a list of the new initialization values and prompts for confirmation. When you answer the prompt, the station applies the change.
SET VARIANT var-code	Changes the variant string to be stored in the NVRAM when it's initialized. The variant is one or two alphanumeric characters. When you enter this command, the system displays a list of the new initialization values and prompts for confirmation. When you answer the prompt, the station applies the change.
SHOW NVRAM	Shows the current values of the ECO, REVISION, and VARIANT fields that will be used to initialize NVRAM.
After using any of the SET ECO, SET REVISION, or SET VARIANT commands, ALWAYS use the SET CHECKSUM command. As distributed, the initial values of the NVRAM fields is: o ECO -- "INVALID" o REVISION -- "A01" o VARIANT -- "00"	

G.8 Managing the Test Station

Once started, the system test software runs until the system is shut down, or the power fails. Any internal errors that the software detects will cause the test station to shut itself down.

All significant events are sent to an event log file kept in the EVTLOG subdirectory of the MSASYS account. This file maintains a record of any errors recorded while the test station is in use.

G.9 Installing the Software

You use the standard VAX/VMS installation procedure (VMSINSTAL) to install the test station software. The procedure copies the files over the Easynet and puts them in the appropriate places on the MicroVAX system.

Refer to the separately supplied installation notes for details of where the files are kept and how to install them.

Appendix H
Adapter Cable Interfaces

H.1 Introduction

There are five adapter cables that select the appropriate signals from the 50-way socket and present them in the correct form for particular interfaces.

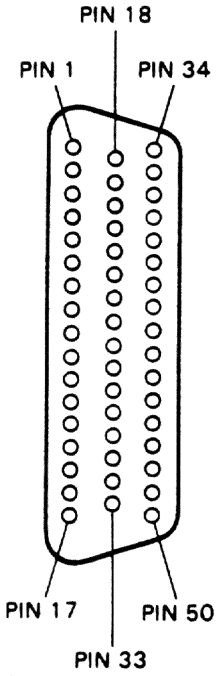
Sections H.2 to H.6 show the wiring at each end of the adaptor cables for these interfaces:

- V.24/RS-232-C
- RS-423
- V.35
- RS-422
- X.21

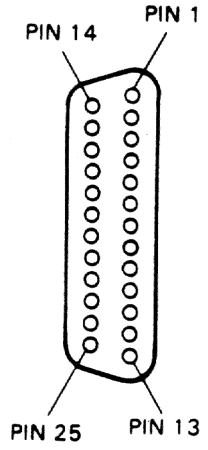
In addition, Section H.7 shows the wiring of the V.24 to RS-232-C adapter connector. (This connector allows the V.24/RS-232-C cable to be used in RS-232-C environments.)

H.2 V.24/RS-232-C

50-WAY PINS	SIGNAL NAME	25-WAY PINS
1	CODE GROUND	*
2	CODE 0	
3	CODE 1	*
4	CODE 2	
5	CODE 3	
8	TX DATA	2
11	RX DATA (A)	3
12	RX DATA (B)	#
13	LOCAL LOOP	18
15	TEST I	25
16	REM.LOOP	21
17	RI	22
18	RX CLOCK (A)	17
19	RX CLOCK (B)	#
20	TX CLOCK (A)	15
21	TX CLOCK (B)	#
22	CLOCK	24
33	DTR	20
34	DSR (A)	6
35	DSR (B)	#
36	RTS	4
37	DCD/I (A)	8
38	DCD/I (B)	#
39	CTS (A)	5
40	CTS (B)	#
41	DCE GROUND	#
44	DTE GROUND	7,#
50	SPEED SEL	23



50-WAY D-TYPE CONNECTOR (FEMALE)



25-WAY D-TYPE CONNECTOR (MALE)

* – CONNECTED TOGETHER

– CONNECTED TO DCE GROUND

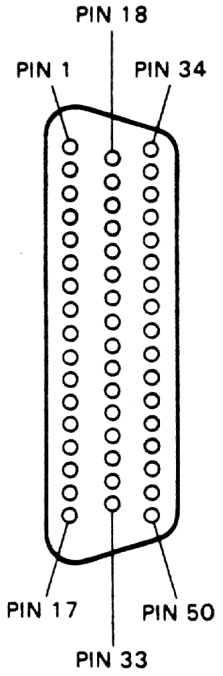
(A),(B) – WIRES A AND B OF A TWISTED PAIR

H.3 RS-423

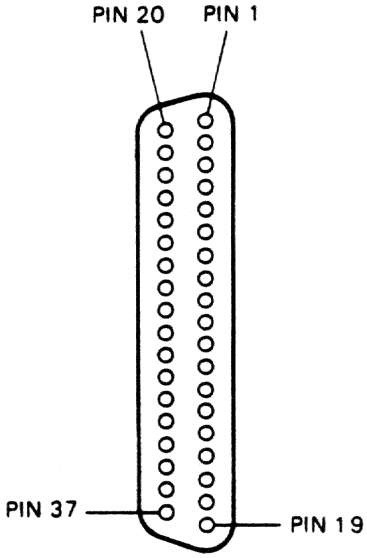
50-WAY PINS	SIGNAL NAME	37-WAY PINS
1	CODE GROUND	*
2	CODE 0	
3	CODE 1	*
4	CODE 2	
5	CODE 3	
8	TX DATA	4
11	RX DATA (A)	6
12	RX DATA (B)	24
13	LOCAL LOOP	10
15	TEST I	18
16	REM.LOOP	14
17	RI	15
18	RX CLOCK (A)	8
19	RX CLOCK (B)	26
20	TX CLOCK (A)	5
21	TX CLOCK (B)	23
22	CLOCK	17
33	DTR	12
34	DSR (A)	11
35	DSR (B)	29
36	RTS	7
37	DCD/I (A)	13
38	DCD/I (B)	31
39	CTS (A)	9
40	CTS (B)	27
41	DCE GROUND	20
44	DTE GROUND	19, 22, 25, 30, 35, 37
50	SPEED SEL	16

* - CONNECTED TOGETHER

(A), (B) – WIRES A AND B OF A TWISTED PAIR



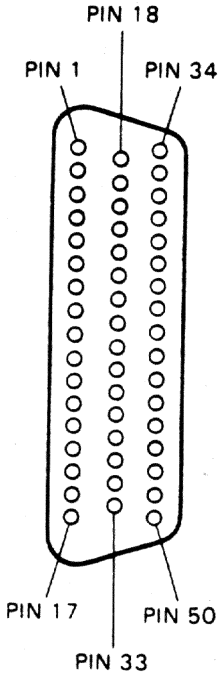
50-WAY D-TYPE CONNECTOR (FEMALE)



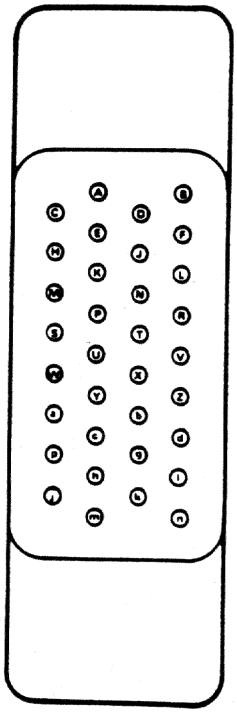
37-WAY D-TYPE CONNECTOR (MALE)

H.4 V.35

50-WAY PINS	SIGNAL NAME	34-WAY PINS
1	CODE GROUND	*
2	CODE 0	*
3	CODE 1	
4	CODE 2	
5	CODE 3	
17	RI	J
23	V.35 TX CLOCK (A)	Y
24	V.35 TX CLOCK (B)	a
25	V.35 CLOCK (A)	U
26	V.35 CLOCK (B)	W
27	V.35 RX DATA (A)	R
28	V.35 RX DATA (B)	T
29	V.35 TX DATA (A)	P
30	V.35 TX DATA (B)	S
31	V.35 RX CLOCK (A)	V
32	V.35 RX CLOCK (B)	X
33	DTR	H
34	DSR (A)	E
35	DSR (B)	#
36	RTS	C
37	DCD/I (A)	F
38	DCD/I (B)	#
39	CTS (A)	D
40	CTS (B)	#
41	DCE GROUND	#
44	DTE GROUND	B,#



50-WAY D-TYPE CONNECTOR (FEMALE)



34-WAY SQUARE CONNECTOR (MALE)

* CONNECTED TOGETHER

\# CONNECTED TO DCE GROUND

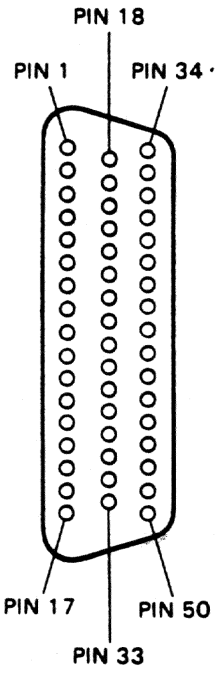
(A), (B) – WIRES A AND B OF A TWISTED PAIR

H.5 RS-422

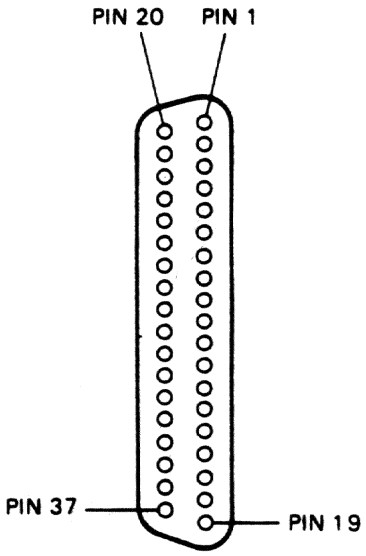
50-WAY PINS	SIGNAL NAME	37-WAY PINS
1	CODE GROUND	*
2	CODE 0	
3	CODE 1	
4	CODE 2	*
5	CODE 3	
6	TX DATA (A)	4
7	TX DATA (B)	22
9	RTS/C (A)	7
10	RTS/C (B)	25
11	RX DATA (A)	6
12	RX DATA (B)	24
13	LOCAL LOOP	10
15	TEST I	18
16	REM.LOOP	14
17	RI	15
18	RX CLOCK (A)	8
19	RX CLOCK (B)	26
20	TX CLOCK (A)	5
21	TX CLOCK (B)	23
34	DSR (A)	11
35	DSR (B)	29
37	DCD/I (A)	13
38	DCD/I (B)	31
39	CTS (A)	9
40	CTS (B)	27
41	DCE GROUND	20
44	DTE GROUND	19, 37
45	DTR (A)	12
46	DTR (B)	30
47	CLOCK (A)	17
48	CLOCK (B)	35
50	SPEED SEL	16

* – CONNECTED TOGETHER

(A),(B) – WIRES A AND B OF A TWISTED PAIR

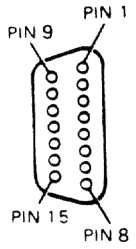


50-WAY D-TYPE CONNECTOR (FEMALE)



37-WAY D-TYPE CONNECTOR (MALE)

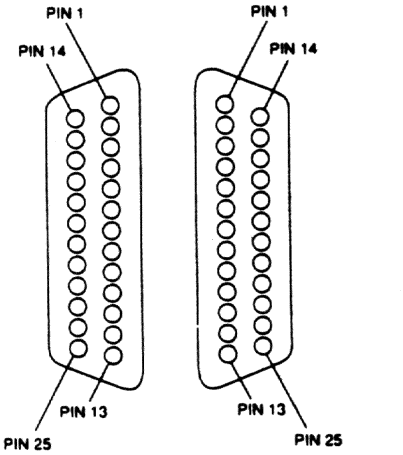
H.6 X.21



Pin	Signal Name	Pin	Signal Name
2	TX DATA A	9	TX DATA B
3	CONTROL A	10	CONTROL B
4	RX DATA A	11	RX DATA B
5	INDICATION A	12	INDICATION B
6	SIGNAL TIMING A	13	SIGNAL TIMING B
7	BYTE TIMING A	14	BYTE TIMING B
8	DTE GROUND DRAIN WIRE	SHELL	SHIELD BRAID

H.7 V.24 to RS-232-C Adapter Connector

25-WAY MALE	SIGNAL NAME	25-WAY FEMALE
1	not connected	
2	TX DATA	2
3	RX DATA	3
4	RTS	4
5	CTS	5
6	DSR	6
7	GROUND	7
8	DCD	8
9	not connected	
10	not connected	
11	not connected	
12	not connected	
13	not connected	
14	not connected	
15	TX CLOCK	15
16	not connected	
17	RX CLOCK	17
18	not connected	
19	not connected	
20	DTR	20
21	not connected	
22	RI	22
23	not connected	
24	CLOCK	24
25	TEST IND	25



25-WAY D-TYPE CONNECTOR (MALE)

25-WAY D-TYPE CONNECTOR (FEMALE)

Appendix I
Related Documentation

I.1 Introduction

This appendix provides a list of DEC MicroServer related documentation. This includes:

- The DEC MicroServer engineering specifications.
- The data sheets and technical descriptions for the major on-board components.
- The documentation sets of the DEC MicroServer software products.
- The rest of the DEC MicroServer documentation set.
- The algorithms used in the OBT

I.2 DEC MicroServer Engineering Specifications

The following DIGITAL internal engineering specifications provide further information on the DEC MicroServer's hardware and OBT:

- DEC MicroServer Main Logic Board Engioneering Specification, Revision 0.1, July 1988, Document Number: K-SP-5418093-0-0.
- DEC MicroServer Serial Assist Daughter Board Functional Specification, Version V1.1, 16 February 1988, Document Number: K-SP-5418100-00.
- DEC MicroServer Engineering Specification (Distribution Panel), Revision 0.1, 8 July 1988, Document Number: K-SP-5418095-0-0.
- Guide to MicroServer OBT Diagnosis Exerciser Errors, Revision OBT V3.17, April 1988.

I.3 MicroVAX II Processor, MicroDMA, VIC, and DYRC

These four devices are all part of the MicroVAX chip set. You can get functional, interface, and electrical information on these devices from:

- DIGITAL Semiconductor Data Book, Pub: Digital Equipment Corporation.

NOTE

The contents of this data book are confidential and proprietary to DIGITAL. The book is available only on restricted distribution.

I.4 TMS 32020

The following books contain more information on the TMS 32020, its architecture, the device characteristics, and programing information.

- The TMS32020 User's Guide, Pub: Texas Instruments.
- TMS32020 Data Sheet, Pub: Texas Instruments.

I.5 LANCE

The following publication contains information on the LANCE, its architecture, device characteristics, and programming:

- The AM7990 Local Area Network Controller for Ethernet (LANCE) Data Sheet, Pub: Advanced Micro Devices.

I.6 DUSCC

The following contains information on the DUSSC, its architecture, device characteristics, and programming:

- The SCN68562 Dual-channel Universal Serial Communications Controller (DUSCC) Data Sheet, Pub: Signetics Corporation.

I.7 DEC MicroServer Software Products Documentation

I.7.1 DECrouter 2000

The documents in the DECrouter 2000 software kit are:

- Routing and Networking Overview
- DECrouter 2000 Installation Procedures
- DECrouter 2000 Management Guide
- DECrouter 2000 Problem Solving Guide

I.7.2 X25router 2000

The documents in the X25router software kit are:

- Routing and Networking Overview
- X25router Installation Procedures
- X25router Management Guide
- X25router Problem Solving Guide

I.7.3 DECnet/SNA Gateway

The documents in the DECnet/SNA Gateway software kit are:

- DECnet/SNA Gateway Installation Guide
- DECnet/SNA Gateway Problem Determination Guide
- DECnet/SNA VMS Gateway Management Installation Guide
- DECnet/SNA VMS Gateway Management Guide

I.8 DEC MicroServer Documentation

Other books related to the DEC MicroServer hardware are:

- Installing the DEC MicroServer
- DEC MicroServer Systems Configuration Card
- DEC MicroServer Pocket Service Guide (DIGITAL internal only)
- DEC MicroServer Field Maintenance Print Set (DIGITAL internal only)
- DEC MicroServer Error Document (DIGITAL internal only)

All of these are published by DIGITAL Equipment Corporation.

I.9 On-Board Test Algorithms

The following papers describe the algorithm that the OBT uses to check the System RAM and the Buffer RAM:

- Functional Testing of Semiconductor Random Access Memories by M. S. Abadir and H. K. Reghbati, from ACM Computing Surveys, Vol 15 No. 3, Pub: Association of Computing Machinery.
- Efficient Algorithms for Testing Semiconductor Random Access Memories by R. Nair, S. M. Thatte, and J. A. Abraham, from IEEE Transactions on Computing, Vol C-27 No. 6 (June), Pub:IEEE.

Notes

The following pages are left blank for you to add notes to this book. You can leave the notes here, or move them into the appropriate chapter of the book.

-D-

READER'S COMMENTS

What do you think of this manual? Your comments and suggestions will help us to improve the quality and usefulness of our publications.

Please rate this manual:

	Poor				Excellent
Accuracy	1	2	3	4	5
Readability	1	2	3	4	5
Examples	1	2	3	4	5
Completeness	1	2	3	4	5

Did you find errors in this manual? If so, please specify the error(s) and page number(s).

_____

_____

_____

_____

General comments:

_____

_____

_____

_____

Suggestions for improvement:

_____

_____

_____

_____

Name _____ Date _____

Title _____ Department _____

Company _____ Street _____

City _____ State/Country _____ Zip Code _____

Digital Equipment Company Limited
Wide Area Communications Environment
PO Box 121 READING
Berkshire RG2 0TU
England