VAX-11/782
User's Guide

Order No. AA-M543A-TE

May 1982

This document describes the VAX-11/782 attached processor system.

REVISION/UPDATE INFORMATION:	This is a new document for this release.
SOFTWARE VERSION:	VAX/VMS Version 3.0

digital equipment corporation · maynard, massachusetts

First Printing, May 1982

Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	RSX
DEC/CMS	EduSystem	UNIBUS
DECnet	IAS	VAX
DECsystem-10	MASSBUS	VMS
DECSYSTEM-20	PDP	VT
DECUS	PDT	digital
DECwriter	RSTS	

ZK2215

HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6146 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575)

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
940 Belfast Road
Ottawa, Ontario K1G 4C2
Attn: A&SG Business Manager

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
A&SG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

CONTENTS

CONTENTS

FIGURES

PREFACE

MANUAL OBJECTIVES

This manual introduces the VAX-11/782 attached processor system, highlights its features, and explains how to set up, use, and maintain it.

INTENDED AUDIENCE

This manual is intended for all users of the VAX-11/782 system, but especially for system managers, system programmers, and operators.

All users should read Chapter 1. System managers and operators should read Chapters 2 and 3. Programmers should read Chapter 4. Those interested in a conceptual overview of VAX/VMS operating system support of the attached processor system should read Appendix A. The information in Appendix A assumes a detailed knowledge of VAX/VMS.

STRUCTURE OF THIS DOCUMENT

This manual has four chapters and one appendix.

- Chapter 1, Introduction, discusses multiprocessing terminology, configuration guidelines, performance characteristics, and VAX/VMS support.
- Chapter 2, Setting up the System, discusses installation and upgrade procedures.
- Chapter 3, System Management Considerations, discusses information of special concern to the system manager.
- Chapter 4, Programming Considerations, discusses information of special concern to programmers.
- Appendix A, VAX-11/782 System Overview, discusses system functions and changes made to VAX/VMS to support multiprocessing.

ASSOCIATED DOCUMENTS

This manual makes frequent reference to the VAX-11/780 Software Installation Guide.

CONVENTIONS USED IN THIS DOCUMENT

Unless otherwise noted, all numeric values are represented in decimal notation.

Unless otherwise specified, you terminate commands by pressing the RETURN key.

CHAPTER 1

INTRODUCTION

The VAX-11/782 is an attached processor system consisting of two VAX-11/780 central processing units (CPUs or processors) connected by MA780 shared memory.

This chapter provides an introduction to the VAX-11/782 system by discussing the following topics:

- The terminology used to classify the types of multiprocessing computer systems and, in particular, the classification of the VAX-11/782 as an "attached" processor system
- The VAX-11/782 system configuration guidelines and restrictions
- The performance of the VAX-11/782: the types of workload for which it is ideally suited and the increase in work throughput you can expect compared to the VAX-11/780
- VAX/VMS support of the VAX-11/782

1.1 MULTIPROCESSING TERMINOLOGY

A multiprocessing system consists of two or more central processing units (processors), each of which is capable of executing instructions simultaneously with the other(s) and of addressing a common pool of memory. For simplicity, this discussion is limited to a two-processor system.

A multiprocessing system can be classified on the basis of the relationship between its two processors. This relationship can be determined using the following two criteria:

- Coupling, that is, whether or not both processors execute a single copy of the operating system. If both processors execute a single copy of the operating system, the multiprocessing system is "tightly coupled." If each processor executes its own copy of the operating system, the multiprocessing system is "loosely coupled."
- Symmetry, that is, whether or not both processors have equal access to operating system code (and therefore to other system resources). If each processor can execute all of the operating system code, the multiprocessing system is "symmetrical." If one of the processors cannot execute all of the operating system code, the multiprocessing system is "asymmetrical."

A loosely coupled system is typically symmetrical, while a tightly coupled system is typically asymmetrical.

The VAX-11/782 is a tightly coupled, asymmetrical system. In a tightly coupled, asymmetrical system, the processor that can execute all the operating system code is called the "primary" processor (or "master"), while the processor that is restricted is called the "attached" processor (or "slave").

In a loosely coupled, symmetrical multiprocessing system, the processors are independent of each other. If one of the processors fails, the other processor is unaffected. This advantage is referred to as "high availability."

In a tightly coupled, asymmetrical multiprocessing system, the processors are not independent of each other. In the VAX-11/782 attached processor system, if the primary processor fails, the attached processor fails. If the attached processor fails for any reason other than powerfail, the primary processor fails. However, if a power failure occurs on the attached processor, the primary processor continues to run.

Though the tightly coupled, asymmetrical multiprocessing system does not have the advantage of "high availability," it does have the advantage of being suited for "dynamic load leveling." Dynamic load leveling is the capability of assigning a job to either processor on the basis of which processor is free to execute the job.

Appendix A, VAX-11/782 System Overview, describes in detail how the primary and attached processors function together in a tightly coupled, asymmetrical system.

1.2 VAX-11/782 SYSTEM CONFIGURATION

The VAX-11/782 consists of two VAX-11/780 processors, from two to four MA780 shared memory subsystems, the VAX/VMS operating system, and various peripheral devices. The VAX/VMS operating system is located in MA780 shared memory. Peripheral devices are connected to the primary processor.

Figure 1-1 depicts one of several possible configurations of the VAX-11/782 system.

The following list contains VAX-11/782 configuration guidelines and restrictions as they apply to processors, memory, and peripheral devices:

1. Processors

   - Both processors must be at the same hardware ECO revision level.
   - Both processors must be running the same version of microcode.
   - Both processors must have the same CPU options. For example, if one processor has an optional floating-point accelerator, then the other must have it too.
   - The Writeable Control Store (WCS) option is not supported.

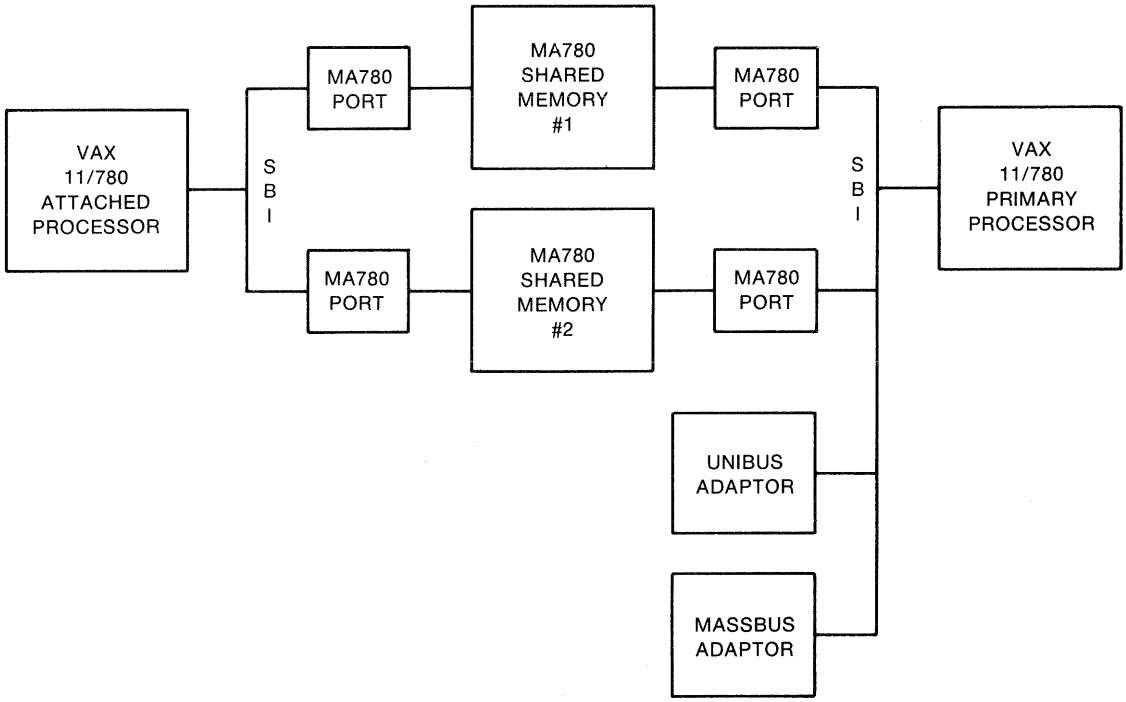

Figure 1-1: A Possible VAX-11/782 System Configuration

2. Memory

   - All active memory is MA780 shared memory. Local memory attached to either processor is ignored.
   - There may be from two to four MA780 shared memory subsystems, each one containing a maximum of 2 megabytes of memory.
   - The MA780 shared memory subsystems must be located at contiguous physical addresses, with the first MA780 starting at physical address 0.
   - Since all MA780 shared memory is used as main memory in the VAX-11/782 system, MA780 shared memory subsystems may not be used concurrently as interconnects for loosely coupled multiprocessor configurations. (Loosely coupled multiprocessor configurations using MA780 shared memory were implemented in Version 2.0 of VAX/VMS.)
   - Each MA780 subsystem must be configured at the same TR (Transfer Request) level on both processors.
   - Both the primary and attached processor must have 256 kilobytes of local (MS780) memory for diagnostic purposes. This memory must be configured at TR level 1.
   - Each MA780 shared memory subsystem should have the cache invalidation map option. This option reduces traffic on the Synchronous Backplane Interconnect (SBI) by reducing the number of cache invalidate requests sent to each processor. By keeping track of which locations in MA780 memory have been placed in the cache of each processor,

the option allows cache invalidate requests to be sent only to the processor(s) whose cache contains the location that has been invalidated.

3. Peripheral Devices

   - All active peripheral devices must be connected to the primary processor. Peripheral devices connected to the attached processor are ignored.
   - A console terminal must be connected to each processor.
   - RP07 disks must be operated at a throughput rate of 1.3 megabytes per second, rather than 2.2 megabytes per second.
   - The DR780 interface cannot be used.

1.3 PERFORMANCE CHARACTERISTICS

The performance of any computer system is limited by the available system resources. The major system resources are processing power (CPU cycles), memory, and I/O bandwith. A restriction in any one of these resources may degrade system performance (measured by either the time taken to complete a job or the terminal response time) regardless of how great the other resources may be.

In any computer system, one of the above three system resources is usually the limiting resource. Which system resource is the limiting resource depends on the system configuration (hardware and software) and on the nature of the system workload.

Figure 1-2 depicts the relationship among system resources, workload, and delay for a system in which processing (CPU) power is the limiting system resource. Delay can be interpreted as the time taken to complete a job or the terminal response time for interactive users.

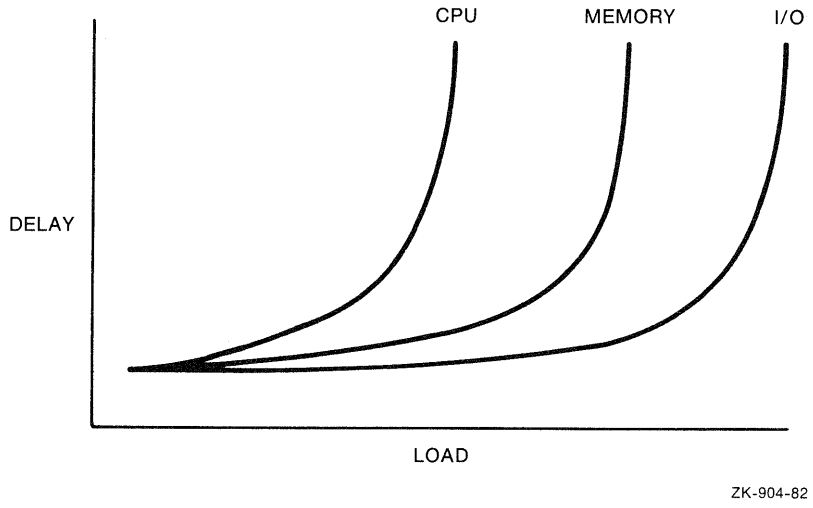

Figure 1-2: Limiting System Resource

In this system, as the load increases, delay increases along the CPU curve; therefore, processing power is the limiting system resource. Given a suitable workload, such a system would profit by the addition of an attached processor, whereas an increase in memory or I/O resources would not improve system performance.

Note that in another system where the limiting resource is memory, for example, adding an attached processor would not improve system performance.

If processing power is the limiting system resource in a single-processor VAX-11/780 system, upgrading the VAX-11/780 system to a VAX-11/782 system can result in a 60 to 80 percent increase in job throughput. However, because the attached processor functions as a computational workhorse, the performance of the VAX-11/782 depends greatly on the availability of suitable jobs for the attached processor to execute, as well as on a sufficient number of jobs to keep both processors busy.

An ideal workload for the VAX-11/782 is one that meets the following guidelines:

- The workload consists of multiple jobs (a "multistreamed" workload).

  There must be multiple processes ready to be executed in order to keep both processors busy. An application consisting of one large computational job would not run faster on the VAX-11/782 unless it could be broken down into multiple processes that could be executed simultaneously by both processors.

- The workload contains "compute-intensive" jobs.

  A compute-intensive job is one whose speed of completion is governed primarily by the amount of CPU time it gets, as opposed to an "I/O bound" job whose speed of completion depends primarily upon the speed in which its I/O requests are serviced.

To summarize, compute-intensive, multistreamed workloads benefit much more by the addition of an attached processor than those that are less compute-intensive or multistreamed. For example, it is possible to increase system throughput by running compute-intensive batch jobs concurrently with multiple I/O-intensive jobs (terminal users, for example). Since the compute-intensive batch job will be executed almost entirely by the attached processsor, leaving the primary processor free to execute terminal users' processes, the terminal users experience no delay as a result of the concurrent execution of the batch job.

1.4 VAX/VMS SUPPORT OF THE VAX-11/782

Minimal code changes were made in VAX/VMS to support the VAX-11/782 attached processor system. Approximately 17 pages of additional nonpaged memory accommodates the necessary code changes. This additional memory is not required when running a single-processor VAX-11/780 system.

When the attached processor is started by the START/CPU command, a number of hooks are inserted into some modules of the VAX/VMS executive to mark locations where control will pass to the new multiprocessing code that is loaded into nonpaged memory.

For example, hooks have been added to the following modules in the VAX/VMS executive:

- ASTDEL AST delivery and queueing
- BUGCHECK Bugcheck for both processors
- PAGEFAULT Translation buffer invalidates
- SCHED Process scheduling and rescheduling

As the VAX-11/782 is a tightly coupled, asymmetrical system, only the primary processor executes kernel-mode code. Kernel-mode-code components of VAX/VMS include the following:

- The I/O subsystem, including device drivers, device independent routines within the executive, and most system services (in particular the Queue I/O Request system service)
- The page fault handler and the swapper
- The scheduler and timer-related system services

Thus, the primary processor handles all I/O and page fault requests, and schedules all work on the the VAX-11/782 system.

Appendix A, VAX-11/782 System Overview, describes in more detail VAX/VMS support of the VAX-11/782 attached processor system.

CHAPTER 2

SETTING UP A VAX-11/782 SYSTEM

The VAX-11/782 attached processor system is supported by Version 3.0 of VAX/VMS. Therefore, to set up a VAX-11/782 system, you must either install or upgrade to VAX/VMS Version 3.0.

This chapter describes two procedures:

- How to upgrade a VAX-11/780 system running VAX/VMS Version 2.5 to a VAX-11/782 system running VAX/VMS Version 3.0
- How to install VAX/VMS Version 3.0 on a new VAX-11/782 system

Follow the first procedure if you have purchased the VAX-11/782 upgrade package; follow the second if you have purchased a new VAX-11/782 system.

VAX/VMS system maintenance updates are applied to the VAX-11/782 attached processor system in the same way as they are applied to the single-processor VAX-11/780 system. Refer to Chapter 5 in the VAX-11/780 Software Installation Guide for a description of how to apply system maintenance updates.

The procedures described in this chapter assume the following:

- The required VAX-11/782 hardware has already been installed and configured. You should keep a record of certain memory configuration information: in particular, the number and type of memory controllers, the processor TR levels at which the controllers are configured, and the amount of memory on each controller.
- You are familiar with the normal VAX/VMS installation and upgrade procedures, which are described in detail in the VAX-11/780 Software Installation Guide.

2.1 UPGRADING TO VAX/VMS VERSION 3.0

The procedure described in this section describes how to upgrade a VAX-11/780 system running VAX/VMS Version 2.5 to a VAX-11/782 system running VAX/VMS Version 3.0.

This procedure consists of the following steps, each of which is fully described in a following section:

1. Upgrade VAX/VMS Version 2.5 to Version 3.0 by performing the upgrade procedure for the single-processor VAX-11/780 system. This procedure is described in Chapter 4 of the VAX-11/780 Software Installation Guide.

2. Build multiprocessing console floppy disks for the primary and attached processors.

3. Reboot the system as an attached processor system.

4. Edit SYSTARTUP.COM.

If your VAX-11/780 system is already running VAX/VMS Version 3.0 or higher, you need not perform step 1 in the above procedure. In this case, only perform steps 2, 3, and 4.

2.1.1 Performing the Upgrade Procedure

Perform the upgrade procedure described in Chapter 4 of the VAX-11/780 Software Installation Guide.

If your system has at least 512 kilobytes of local (MS780) memory configured at TR level 1, you can use your current version of the VAX-11/780 console floppy disk when you perform the upgrade procedure. However, if your system does not have 512 kilobytes of local memory configured at TR level 1, you must use a VAX-11/780 console floppy disk with a part number of AS-E633R-DE or higher (in a higher part number, the letter R would be replaced with a subsequent letter of the alphabet such as S, T, and so on).

A console floppy disk with a part number of AS-E633R-DE or higher allows for the booting of the system from local (MS780) memory or from MA780 shared memory. If 512 kilobytes of local memory is configured at TR level 1, the system is booted from local memory; otherwise, the system is booted from MA780 shared memory. Note, however, that the diagnostic supervisor is always booted using local memory.

During the upgrade procedure, the system will be a single-processor VAX-11/780 system because VAX/VMS Version 2.5 does not support the VAX-11/782.

During the upgrade procedure, you create a new console floppy disk and a new system disk. The final steps of the upgrade procedure are to shut down the system and then reboot it. After rebooting, the system is running VAX/VMS Version 3.0 as a single-processor VAX-11/780 system.

The next step is to build multiprocessing console floppy disks. This process is described in the next section.

2.1.2 Building Multiprocessing Console Floppy Disks

Each processor in the VAX-11/782 system must have its own console floppy disk. Multiprocessing console floppy disks allow for the booting of the VAX-11/782 attached processor system by means of several boot command procedures. These boot command procedures cause MA780 shared memory rather than local memory to be used, and they set the memory configuration registers to ensure that MA780 shared memory is configured at the low physical addresses (beginning at 0) and local memory at the higher addresses.

In addition, each multiprocessing console floppy disk contains a "Reset Memory" command procedure RMEM.COM, which is specific to the processor and the processor's memory configuration. The RMEM.COM command procedure reconfigures local memory to start at physical

address 0 and MA780 shared memory to start at adjacent higher physical addresses. Thus, after executing RMEM.COM, you can boot the VAX-11/782 system as a single-processor VAX-11/780 system, if you should so desire, by using a standard VAX-11/780 console floppy disk.

To build multiprocessing console floppy disks, you execute the interactive command procedure SYS$UPDATE:BOOTBLDR.COM. This command procedure first creates a new console floppy disk for the primary processor and then one for the attached processor. The procedure executes interactively; it prompts you for information about the memory configuration of the system.

The following sections describe how to obtain information about the memory configuration of the system and how to execute BOOTBLDR.COM.

2.1.2.1 Determining the Memory Configuration - For the purposes of executing BOOTBLDR.COM, you must determine the following information about each memory controller on the system:

- The TR level at which it is configured
- Its type: either MS780 or MA780
- The amount of memory it holds (in increments of .25 megabytes)

You will need the above information for both the primary and attached processors. You can obtain this information by asking a DIGITAL Field Service Representative for the information or by following the procedure in this section. If you already know the memory configuration of the system, proceed to Section 2.1.2.2.

This section describes how to obtain this information first for the primary processor and then for the attached processor. The procedure is the same in both cases, with the following exceptions:

- For the primary processor, perform the procedure at the primary processor's console terminal; for the attached processor, at the attached processor's console terminal.
- You need to determine the TR level and memory amount of each MA780 controller only once (for the primary processor), since an MA780 memory controller must be configured at the same TR level on both processors. Note however that obtaining this information twice (for both processors) allows you to check that MA780 memory has been configured correctly (is at the same TR level) on both processors.

To determine the memory configuration of the primary processor, perform the following steps:

1. Put the console terminal of the primary processor in console mode (if it is not already in console mode) by entering CTRL/P on the console terminal keyboard. (The console terminal is in console mode when pressing the RETURN key displays the console mode prompt >>>.) Then issue the HALT command.
2. Issue the following EXAMINE command:

   ```
   EXAMINE 20002000
   ```

   Interpret the value displayed by this command to determine information about the type of device connected to TR1 (TR level 1):

A. If the two rightmost digits are either 08 or 09, an MS780 memory controller containing 64 kilobyte memory boards is configured at TR1.

To determine the amount of memory contained in this controller, perform the following steps:

- Determine the number of 64 kilobyte memory boards held by the controller by further interpreting the displayed value as follows: determine the value (in decimal) of bits 9 through 12; then add 1 to the result.
- Calculate the total memory on this controller by multiplying the number of memory boards by 64 kilobytes.
- Convert the kilobyte value to a megabyte value by dividing by 1024 kilobytes.

B. If the two rightmost digits are either 10 or 11, an MS780 memory controller containing 0.25 megabyte (or 256 kilobyte) memory boards is configured at TR1.

To determine the amount of memory contained in this controller, perform the following steps:

- Determine the number of 0.25 megabyte memory boards held by the controller by further interpreting the displayed value as follows: determine the value (in decimal) of bits 11 through 14; then add 1 to the result.
- Calculate the total memory on this controller by multiplying the number of memory boards by 0.25 megabytes.

C. If the two rightmost digits are in the range 40 through 43, an MA780 memory controller containing 0.25 megabyte (or 256 kilobyte) memory boards is configured at TR1.

To determine the amount of memory contained in this controller, perform the following steps:

- Add the value 0000000C (hexadecimal) to the examined address. The examined address for TR1 is 20002000. Adding 0000000C yields 2000200C.
- Examine the result of the above addition at the console terminal. For example, for TR1, issue the command EXAMINE 2000200C.
- Inspect the value displayed by this command.
- Extract the fifth digit from the right.
- Add 1 to this digit to obtain the number of memory boards.
- Multiply the result by 0.25 megabytes to obtain the total memory in megabytes.

D. If the two rightmost digits are in the range 28 through 2B hexadecimal, a Unibus Adapter (UBA) is configured at TR1. You should know at which TR level each UBA is configured on your system (some systems may have more than one UBA).

3. Issue the following EXAMINE command:

```
EXAMINE 20004000
```

Interpret the value displayed by this command to determine information about the type of device connected to TR2. The procedure is exactly the same as that described above for TR1.

4. Issue the following EXAMINE command:

```
EXAMINE 20006000
```

Interpret the value displayed by this command to determine information about the type of device connected to TR3. The procedure is exactly the same as that described above for TR1.

5. Issue the following EXAMINE command:

```
EXAMINE 20008000
```

Interpret the value displayed by this command to determine information about the type of device connected to TR4. The procedure is exactly the same as that described above for TR1.

6. Issue the following EXAMINE command:

```
EXAMINE 2000A000
```

Interpret the value displayed by this command to determine information about the type of device connected to TR5. The procedure is exactly the same as that described above for TR1.

7. Issue the following EXAMINE command:

```
EXAMINE 2000C000
```

Interpret the value displayed by this command to determine information about the type of device connected to TR6. The procedure is exactly the same as that described above for TR1.

Note that if the rightmost two digits of the value displayed by any one of the above six EXAMINE commands is not in the ranges 8 to 11, 40 to 43, or 28 to 2B hexadecimal, neither a memory controller nor a Unibus Adapter (UBA) is configured at the TR level corresponding to the examined address. In this case, what is configured at that particular TR level is not of concern, and you need not further interpret the displayed value. Further, if a microcode error results when you examine any of the addresses, simply assume that a device is not configured at the TR level corresponding to the examined address.

Whereas a memory controller may be configured at any TR level from 1 through 6, a UBA may be configured at any TR level from 1 through 15. Further, there may be more than one UBA configured on the system. Therefore, you should examine the following additional locations, which correspond to TR levels 7 through 15, respectively, to determine

whether a UBA is configured at any of these TR levels: 2000E000, 20010000, 20012000, 20014000, 20016000, 20018000, 2001A000, 2001C000, and 2001E000. Remember that if the two rightmost digits of the displayed value are in the range 28 through 2B, a UBA is configured at the TR level corresponding to the examined address.

You have now determined the memory configuration for the primary processor. To determine the memory configuration of the attached processor, repeat the above procedure at the attached processor's console terminal. That is, put the terminal in console mode, issue the six EXAMINE commands (for TR levels 1 through 6), and interpret the displayed values.

Again, you need not obtain information about MA780 memory a second time, since information about MA780 memory is identical for both the primary and attached processors. Thus, you need not examine one of the six addresses if an examination of that address at the primary processor's console terminal revealed an MA780 memory controller.

You have now determined the memory configuration of the system. You have all the information needed to execute BOOTBLDR.COM. Proceed to the next section.

2.1.2.2 Executing BOOTBLDR.COM - BOOTBLDR.COM is an interactive command procedure that builds console floppy disks for the primary and attached processors. You must execute BOOTBLDR.COM when you initially set up your VAX-11/782 system and whenever you change its memory configuration. The multiprocessing console floppy disks created by BOOTBLDR.COM can only be used in a system with the same memory configuration as the memory configuration of that system for which they were created.

BOOTBLDR.COM requests that you enter information, and it gives you instructions about what to do next. As it executes, it displays messages that indicate what is taking place.

This section discusses those parts of the command procedure over which you have control. That is, it discusses how to respond to requests for information. If you want more information, you can read the command procedure itself by issuing the following command at the console terminal:

```
$ TYPE SYS$UPDATE:BOOTBLDR.COM
```

Note that the console floppy disks you are creating initialize the starting physical addresses of all memory on the system. If you enter incorrect memory amounts when executing BOOTBLDR.COM, these starting physical addresses will be incorrect. A machine check will result if VAX/VMS references an incorrect physical address.

To execute BOOTBLDR.COM, issue the following command:

```
$ @SYS$UPDATE:BOOTBLDR
```

The procedure prompts as follows:

```
Enter memory type (MA780, MS780, or <RETURN> to end):
```

Respond by entering MS780 if you are entering information about an MS780 memory controller that is configured on the primary processor. Respond by entering MA780 if you are entering information about an MA780 memory controller that is configured on the primary processor. Respond by simply pressing the RETURN key if you have already entered information about all controllers (MS780 and MA780) configured on the

primary processor. Do not abbreviate or add suffixes to your response; for example, do not abbreviate MS780 as MS.

The procedure then prompts as follows:

```
Enter TR level (1 through 6):
```

Respond by entering the number of the TR level at which the memory controller (MS780 or MA780) you entered in response to the previous prompt is configured. Enter only a number. Note that if you simply pressed the RETURN key in response to the previous question, this prompt does not appear.

The procedure then prompts as follows:

```
Enter amount of memory for this controller in .25 megabyte
increments (for example, for 512 kilobytes, enter .5):
```

Respond by entering the amount of memory configured at this TR level. Enter only a number; that is, do not enter a suffix such as Mbytes or megabytes. Note that memory must be present in increments of .25 megabytes.

This sequence of three requests is repeated until you press RETURN (and nothing else) in response to the first request. You should press RETURN after you have mentioned all memory controllers connected to the primary processor.

The procedure then displays the following message:

```
The Unibus Adapter (UBA) is assumed to be at TR level x.
```

The letter x represents a number from 1 through 15. BOOTBLDR.COM derives the number represented by the letter x by adding one to the number of the highest TR level at which an MA780 memory controller is configured. BOOTBLDR.COM uses the TR level of the UBA in the creation of boot command procedures (such as DM0BOO.CMD) for unibus devices (such as RK06 and RK07 disk drives).

The procedure then prompts as follows:

```
Enter the TR level of the UBA (enter RETURN for the default):
```

If a UBA is configured at the TR level displayed in the preceding message, simply press RETURN; do not enter a number. On the other hand, if a UBA is not configured at the TR level displayed in the preceding message, enter the TR level at which the UBA is configured; enter only a number and then press RETURN.

Note that a system may have more than one UBA and that BOOTBLDR.COM can create boot command procedures for use on only one UBA. In a system with more than one UBA, you must select the UBA for which you want boot command procedures created. In this way, boot command procedures for devices on that UBA (but not for devices on the other UBA) will be created.

The procedure continues by prompting as follows:

```
Enter the name of the default boot command procedure:
```

VAX/VMS supplies a number of default boot command procedures to enable you to boot the system from various devices. In general, the file name of the default boot command procedure you should choose has as its first three characters the device name of the device on which you expect the system disk to reside; the remaining characters in the file name are BOO.CMD. Read Sections 3.2, 3.2.1, and 3.2.2 in the

VAX-11/780 Software Installation Guide to determine which default boot command procedure to select. Then respond to the request by entering its file name.

The procedure then prompts as follows:

```
Enter the name of the floppy disk drive you want to use

(format is ddu:):
```

Respond by entering the device name of the floppy disk drive that you want to use. Only one floppy disk drive is used throughout the procedure. If you use the floppy disk drive in the console subsystem (in the CPU cabinet), enter the name CS1:. If you use an RX02 floppy disk drive, enter the name DYx:, where x is the unit number printed on the drive itself (this number is usually 0 or 1).

Note that you must enter the device name in the format ddu: (not ddcu:). That is, do not specify a controller designation. For example, specify CS1:, not CSA1:. For more information about device naming conventions, see Section 2.3 in the VAX-11/780 Software Installation Guide.

The procedure then instructs you to insert the original floppy disk in the drive you mentioned. The original floppy disk is a VAX-11/780 console floppy disk provided by DIGITAL, order number AS-E633x-DE where x is a letter of the alphabet. The original floppy disk is your current console floppy disk if you are running a single-processor VAX-11/780 system. If you do not have one, order one directly from DIGITAL. Do not insert a VAX-11/782 console floppy disk or a console floppy disk created by the upgrade procedure.

The procedure then prompts as follows:

```
Ready to continue? (YES or NO):
```

Respond by entering YES.

The procedure then instructs you to remove the original floppy disk from the drive.

After you remove the original floppy disk from the drive, the procedure instructs you to insert a scratch floppy disk in the drive. A scratch floppy disk is a floppy disk whose contents are unimportant to you.

The procedure warns you that the scratch disk will be initialized and prompts as follows:

```
Ready to continue? (YES or NO):
```

Respond by entering YES.

After several minutes, the procedure issues a message to the effect that the multiprocessing console floppy disk for the primary processor has been created.

The procedure then reminds you that MA780 memory must be identical on both processors. For this reason, the procedure will not prompt for the TR level(s) at which MA780 memory is configured on the attached processor. You have already provided the necessary information about MA780 memory.

The procedure then mentions that MS780 (local) memory on the attached processor may be different than MS780 memory on the primary processor. That is, MS780 memory on the attached processor may be configured at

different TR levels; further, there may be more or less MS780 memory on the attached processor.

The procedure then prompts as follows:

```
Is an MS780 (local) memory controller configured at TR level 1?
(YES or NO):
```

Respond by entering YES if there is; NO if there is not.

If you answer YES, the procedure prompts as follows:

```
Enter amount of memory for this controller in .25 megabyte
increments (for example, for 512 kilobytes, enter .5):
```

Respond by entering the amount of memory. Enter only a number; that is, do not enter a suffix such as Mbytes or megabytes.

The procedure then prompts as follows but substitutes for n the number of the next TR level at which MA780 memory is not configured:

```
Is an MS780 (local) memory controller configured at TR level n?
(YES or NO):
```

Respond appropriately to this prompt and to the remaining prompts that ask whether MS780 memory is configured at particular TR levels (TR levels at which MA780 memory is not configured). Remember that when you answer YES at any time, the procedure then asks how much memory is configured at that particular TR level.

After you have entered the necessary memory configuration information, the procedure instructs you to insert a scratch floppy disk in the previously named floppy disk drive and then prompts as follows:

```
Ready to continue? (YES or NO):
```

Respond by entering YES.

The procedure then creates a multiprocessing console floppy disk for the attached processor, issues some messages, and exits.

You now have multiprocessing console floppy disks for the primary and attached processors. Be sure to label them correctly: ATTACHED for the attached processor's console floppy disk; PRIMARY for the primary processor's console floppy disk. They are not interchangeable.

You should also indicate on the label the machine for which the multiprocessing console floppy disks are intended. These floppy disks can only be used in a VAX-11/782 system whose memory configuration is identical to the memory configuration you described when you created the floppy disks using BOOTBLDR.COM. These floppy disks cannot be used in a single-processor VAX-11/780 system or in a VAX-11/782 system with a different memory configuration.

2.1.3 Shutting Down the System

After you have created console floppy disks for the primary and attached processors, shut down the system by issuing the following command:

```
$ @SYS$MANAGER:SHUTDOWN
```

This command invokes the system shutdown command procedure, which shuts down the system in an orderly fashion.

Ensure that both processors are halted and that the >>> prompt appears on both console terminals.

2.1.4 Booting the VAX-11/782 System

With the system shut down and both processors halted, boot the system in the following manner:

1. Insert the primary processor's console floppy disk in the primary processor's console floppy disk drive.
2. Insert the attached processor's console floppy disk in the attached processor's console floppy disk drive.
3. Issue the BOOT command on the primary processor's console terminal.
4. Log in using the SYSTEM account.
5. Issue the DCL command START/CPU on the primary processor's console terminal.
6. Issue the BOOT command on the attached processor's console terminal.

VAX/VMS Version 3.0 is now running on the VAX-11/782 system. You should now follow the procedure for editing SYSTARTUP.COM. This procedure is described in Section 2.3.

2.2 INSTALLING VAX/VMS VERSION 3.0

This section describes how to install VAX/VMS Version 3.0 on a VAX-11/782 system. Follow this installation procedure if you have purchased a new VAX-11/782 system.

The installation procedure consists of the following steps, each of which is fully described in the reference given:

1. Follow the procedure for installing VAX/VMS Version 3.0 on a single-processor VAX-11/780 system. This procedure is fully documented in Chapter 2 of the VAX-11/780 Software Installation Guide. The following are the steps in this procedure, listed in the order in which they are performed:

   - Boot stand-alone BACKUP.
   - Restore the required save set from the distribution kit to the target disk.
   - Boot the target disk. This step restores the optional save sets from the distribution kit to the target disk. You must use a VAX-11/780 console floppy disk with a part number of AS-E633R-DE or higher (in a higher part number, the letter R would be replaced with a subsequent letter of the alphabet such as S, T, and so on).

   After performing each step of the procedure described in Chapter 2 of the VAX-11/780 Software Installation Guide, a bootable system disk has been created, and the system is running VAX/VMS Version 3.0.

2. Log in using the system manager's account (USERNAME of SYSTEM; PASSWORD of MANAGER).

3. Follow the procedure for building multiprocessing console floppy disks. This procedure is fully described in Sections 2.1.2, 2.1.2.1, and 2.1.2.2 in this manual.

4. Follow the procedure for shutting down the system. This procedure is described in Section 2.1.3 in this manual.

5. Follow the procedure for booting the VAX-11/782 system. This procedure is described in Section 2.1.4 in this manual.

6. Follow the procedure for editing SYSTARTUP.COM. This procedure is described in Section 2.3 in this manual.

7. Follow the procedures for performing site-specific modifications to VAX/VMS. These procedures are described in Chapter 3 of the VAX-11/780 Software Installation Guide.

2.3 EDITING SYSTARTUP.COM

When VAX/VMS Version 3.0 is running on the VAX-11/782 system, you should edit the site-specific start-up command procedure SYS$MANAGER:SYSTARTUP.COM to allow automatic restart of the attached processor following a system shutdown.

To accomplish this, edit the command procedure SYS$MANAGER:SYSTARTUP.COM to include the following commands:

```
$ START/CPU
$ WRITE SYS$OUTPUT "You can boot the attached processor now."
```

On a cold start, you can boot the attached processor when the message "You can boot the attached processor now" appears on the primary processor's console terminal. To boot the attached processor, you can issue the BOOT command at the attached processor's console terminal, or you can press the BOOT button on the attached processor's console panel.

CHAPTER 3

SYSTEM MANAGEMENT CONSIDERATIONS

This chapter contains information of special concern to the system manager of a VAX-11/782 system.

3.1 DCL COMMANDS FOR MULTIPROCESSING

The system manager uses three DCL commands to start, stop, and display the status of the attached processor. These commands are START/CPU, STOP/CPU, and SHOW/CPU.

CMKRNL privilege is required to execute any one of these commands. The system manager's account typically has CMKRNL privilege.

The START/CPU and STOP/CPU commands can change the state of the attached processor, while the SHOW/CPU simply displays its current state. The six possible states of the attached processor (INITIALIZE, IDLE, BUSY, EXECUTE, DROP, and STOP) are described in detail in Section A.3, while Figure A-1 depicts the transition path between these states.

3.1.1 START/CPU

The system manager issues the DCL command START/CPU to start the attached processor.

If the attached processor is in the STOP state, the START/CPU command causes the attached processor to enter the INITIALIZE state.

If the attached processor is in the INITIALIZE, IDLE, BUSY, EXECUTE, or DROP states, the START/CPU command has no effect. In this case, the attached processor is already running (the IDLE, BUSY, EXECUTE, or DROP states) or is initializing itself to prepare to run (the INITIALIZE state).

3.1.2 STOP/CPU

The system manager issues the DCL command STOP/CPU to stop the attached processor. If the attached processor is already in the STOP state, the STOP/CPU command has no effect.

If the attached processor is not executing a process when the STOP/CPU command is issued, the attached processor simply enters the STOP state. If it is executing a process, the attached processor returns the process to the primary processor before entering the STOP state.

While the attached processor is in the STOP state, the primary processor will not schedule work for or make any requests of the attached processor.

3.1.3 SHOW/CPU

The system manager uses the DCL command SHOW/CPU to display the current state of the attached processor. The current state of the attached processor is one of the six possible states mentioned in Section 3.1. For example:

```
$ SHOW/CPU

        Attached processor is in the EXECUTE state.
```

3.2 MAINTAINING AND MONITORING THE SYSTEM

The VAX-11/782 system requires less maintenance than two VAX-11/780 systems, but slightly more than a single VAX-11/780 system.

Because the VAX-11/782 has two processors, error log entries (written to the error log file) and bugcheck codes (written to the console log) contain information that identifies which processor originated the error log entry or bugcheck code. Sections 3.2.1 and 3.2.2 discuss these topics.

You can also use the Monitor Utility (MONITOR) to monitor the performance of the system. Section 3.2.3 discusses its use.

3.2.1 Error Log Entries

Each error log entry includes the value of a System Identification Register (SID). The SID uniquely identifies each processor. Therefore, if you know the SID of each processor, you can determine which processor made which error log entry.

To determine the SID of a processor, perform the following steps:

1. Put the processor's console terminal in console mode (if it is not already in console mode) by entering CTRL/P. The console terminal is in console mode when entering RETURN causes the prompt >>> to be displayed.

2. Issue the command HALT.

3. Issue the command EXAMINE/I 3E. The value displayed by this command is the processor's SID.

4. Issue the command CONTINUE if the processor was running before you issued the HALT command and you want it to continue running.

3.2.2 Bugcheck Codes

The codes for bugchecks initiated by the attached processor begin with the letters MP, while the codes for bugchecks initiated by the primary processor are the same as for a single processor VAX-11/780 system.

3.2.3 Monitoring System Performance

The Monitor Utility (MONITOR) enables you to obtain information about system performance. You should refer to the VAX-11 Utilities Reference Manual for a detailed description of the Monitor Utility.

This section describes how to use the Monitor Utility to display information about the amount of time spent by the primary and attached processors in the following seven processor modes:

- Interrupt Stack
- Kernel Mode
- Executive Mode
- Supervisor Mode
- User Mode
- Compatibility Mode
- Idle Time

To invoke the Monitor Utility, issue the DCL command MONITOR. You will be prompted for a class name. Enter the class name MODES to specify that you want to monitor the TIME IN PROCESSOR MODES class.

You can also issue several qualifiers. The relevant qualifiers here are /CPU, /NOCPU, /ALL, and /PERCENT. If you specify the /CPU qualifier (the default), the amount of time spent by each of the two processors in each of the seven modes (14 items of information) is displayed. If you specify the /NOCPU qualifier, the amount of time spent by both processors in each of the seven modes (7 items of information) is displayed.

For example, the following command displays the amount of time, measured in clock ticks (there are 100 clock ticks per second), currently being spent by both the primary and attached processors in each of the seven processor modes:

```
$ MONITOR MODES/CPU
```

Specifying the /ALL qualifier expands the display to include all available statistics, that is, time spent by both processors in each of the seven modes (1) at the current time, (2) on the average since MONITOR was invoked, (3) at a maximum since MONITOR was invoked, and (4) at a minimum since MONITOR was invoked. For example:

```
$ MONITOR MODES/CPU/ALL
```

Specifying the /PERCENT qualifier allows you to read the statistics as a percentage of the total time rather than as a number of clock ticks. For example:

```
$ MONITOR MODES/CPU/ALL/PERCENT
```

These examples are not exhaustive. You can use various combinations of the qualifiers mentioned in this section, as well as some of the other statistics qualifiers discussed in the VAX-11 Utilities Reference Manual.

CHAPTER 4

PROGRAMMING CONSIDERATIONS

This chapter discusses programming considerations of concern when using the VAX-11/782 system. All programmers who use the VAX-11/782 system should read this chapter.

Almost all programs that run on the single-processor VAX-11/780 will run on the VAX-11/782 without modification. However, some programs may require modification. The sections in this chapter describe those characteristics of a program that require careful consideration and may indicate that modification of that program is necessary.

In particular, if an existing program has one or more of these characteristics, you should determine whether it requires modification to run on the VAX-11/782. Similarly, if you are writing a new program to run on the VAX-11/782, you should be aware of these program characteristics to ensure that you incorporate them correctly in the program.

4.1 WRITEABLE GLOBAL SECTIONS

A writeable global section is an area of memory that may be accessed (read and modified) by more than one process.

On a single-processor VAX-11/780 system, access to a global section by more than one process is automatically synchronized by virtue of two facts:

- Only the currently executing process can access the global section.
- Only one process can be the currently executing process.

However, in the two-processor VAX-11/782 system, two processes can execute concurrently, one on each processor. As a result, it is possible that both currently executing processes will simultaneously access the same location(s) in a writeable global section. If such access occurs, information may be lost.

In the VAX-11/782 system, when writing an application program using writeable global sections, you must use one of the following methods to ensure synchronized access to the global sections by multiple processes:

- Use interlocked instructions instead of ordinary instructions to control access to the writeable global section. The following are the seven interlocked instructions:
  - BBCCI Branch on Bit Clear and Clear, Interlocked
  - BBSSI Branch on Bit Set and Set, Interlocked
  - ADAWI Add Aligned Word, Interlocked
  - INSQTI Insert into Queue Tail, Interlocked
  - INSQHI Insert into Queue Head, Interlocked
  - REMQTI Remove from Queue Tail, Interlocked
  - REMQHI Remove from Queue Head, Interlocked
- Use VAX/VMS system services to control access to the writeable global section.
- Use code that executes in kernel mode to control access to the writeable global section. Since only the primary processor executes kernel-mode code, synchronization is ensured.

You should check existing programs that use writeable global sections to ensure that proper synchronization techniques are in place. You must review the program code itself; do not rely on testing alone since an instance of simultaneous access by two processes to a location in a writeable global section is bound to be rare.

If queue instructions were used to control access to writeable global sections, ensure that these were interlocked queue instructions. Interlocked queue instructions were first available with Version 2.0 of VAX/VMS; programs designed before this time are more likely to need revision than those designed after Version 2.0.

In fact, most application programs do not use writeable global sections, and most of those that do use them have followed the synchronization methods presented in this section. In particular, programs designed to run on the loosely coupled multiprocessing configuration that uses the MA780 as a system interconnect should have used the proper synchronization methods; these programs will run on the VAX-11/782 without error.

4.2 SYNCHRONIZATION USING PROCESS PRIORITY

In some applications (usually real-time applications), a number of processes are used to perform a series of tasks. In such applications, the sequence in which a process executes may be controlled or synchronized by means of process priority.

The basic method of synchronization by priority involves executing the process with the highest priority while preventing all other processes from executing. Because the VAX-11/782 has two processors, it is impossible to prevent all other processes from executing.

Further, synchronization by process priority fails in the VAX-11/782 system because scheduling is preemptive for the primary processor but not preemptive for the attached processor. As a result, the primary processor will always be executing the highest priority process. At the same time, however, the attached processor may not be executing

the next highest priority process. In fact, the attached processor may continue to execute a lower priority process even though a higher priority process is computable and ready to execute.

Thus, application programs that use the method of synchronization by process priority must be modified before they will run correctly on the VAX-11/782 system.

4.3 EXECUTIVE-MODE, USER-WRITTEN SYSTEM SERVICES

Some application programs call user-written system services that execute in executive mode and require access to information in a Process Control Block (PCB).

Such programs may have referenced the system-wide location SCH$GL_CURPCB in the VAX/VMS executive in order to acquire the address of the current process's PCB.

However, since the VAX-11/782 system has two processors, there may be two current processes -- one running on each processor. As a result, you must check all references to the location SCH$GL_CURPCB to ensure that the correct address will be obtained by all such references.

If a process is running on the primary processor when it makes a reference to SCH$GL_CURPCB, the PCB address obtained is correct. A process is guaranteed to be running on the primary processor if the reference to SCH$GL_CURPCB is made while the process is executing in kernel mode, since the primary processor executes all kernel-mode code.

On the other hand, if a process makes a reference to SCH$GL_CURPCB while it is running on the attached processor, the PCB address obtained is not correct. A process may be running on the attached processor if the reference to SCH$GL_CURPCB is made while the process is running in any mode other than kernel mode.

To solve the problem, you should replace the location SCH$GL_CURPCB with the new location CTL$GL_PCB in all programs that reference the location SCH$GL_CURPCB while running in an access mode other than kernel mode. The new location CTL$GL_PCB is process-specific, not system-wide like SCH$GL_CURPCB; it always contains the address of the current process's PCB.

The system-wide location SCH$GL_CURPCB may still be used if a process is executing in kernel mode when it makes the reference to SCH$GL_CURPCB.

When writing programs to run on the VAX-11/782, you should use the location CTL$GL_PCB to get the address of the current process's PCB. CTL$GL_PCB contains the address of the current process's PCB, whether that process is running on the primary processor or the attached processor.

In addition, the new location EXE$GL_MP may be used to determine whether or not the system you are using is running as a VAX-11/782 attached processor system. If the contents of EXE$GL_MP is 0, the system is not running as a VAX-11/782 attached processor system. If the system is running as a VAX-11/782 attached processor system, the value of EXE$GL_MP is the starting virtual address of the new multiprocessing code in nonpaged memory.

In general, references to all locations in the VAX/VMS executive should be made while executing in kernel mode. This can be accomplished either by changing all non-kernel-mode references to such locations to kernel-mode references or by calling the appropriate system service to make the reference. For example, to access the VAX/VMS executive location EXE$GQ_SYSTIME, you can use the $GETTIM system service.

4.4 ACCESSING PHYSICAL ADDRESSES USING $CRMPSC

The Create and Map Section ($CRMPSC) system service allows a privileged process (one with PFNMAP privilege) to map a portion of its virtual address space to specific physical addresses. Typically, the physical addresses of interest are those of the device registers in the I/O space of a processor.

Because the VAX-11/782 has two processors, one of which has no devices connected to it, the physical address space is not the same on both processors.

Therefore, VAX/VMS locks any process using the PFNMAP privilege onto the primary processor. If such a process were to execute on the attached processor, a machine check would probably result.

APPENDIX A

VAX-11/782 SYSTEM OVERVIEW

This appendix provides an overview of VAX-11/782 system functions and explains changes made to VAX/VMS to support the attached processor.

The information in this chapter is conceptual in nature and is intended for those who want to know more about how the VAX/VMS operating system supports the VAX-11/782 system.

A.1 SYSTEM INITIALIZATION

The first step in the initialization of the VAX-11/782 system is the bootstrapping of the primary processor using a command procedure on the primary processor's console floppy disk. This command procedure differs in two major respects from the command procedure used in a single-processor VAX-11/780 system:

- It directs the primary bootstrap program VMB to use MA780 shared memory rather than local memory.
- It sets the memory configuration registers to ensure that:
  - The first MA780 shared memory starts at physical address 0.
  - Additional MA780s are placed at contiguous physical addresses.
  - Local (MS780) memory starts at a physical address higher than the highest shared memory address.

After the primary processor is booted, the attached processor is started by the DCL command START/CPU. This command has the following effects:

- It allocates a section of nonpaged memory for the new multiprocessing code and loads this code into the allocated memory.
- It inserts hooks in the normal VAX/VMS operating system code; these hooks cause control to transfer to the multiprocessing logic.
- It modifies a location in the Restart Parameter Block (RPB) where the attached processor will begin execution. Before modification, this location contains an instruction that jumps to itself. After modification, it contains a pointer to the address of initialization code for the attached processor. This initialization code is part of the multiprocessing code that is loaded into nonpaged memory.

- The System Control Block (SCB) for the primary processor is modified to contain the virtual addresses of exception or interrupt service routines that differ from those used in a single-processor VAX-11/780 system. For example, areas of the SCB are modified to use the new scheduling code and the MA780 interrupt mechanism.

- An SCB is created for the attached processor.

A.2 PROCESSOR COMMUNICATION

To maintain the integrity of the VAX/VMS operating system in a VAX-11/782 system, only the primary processor is permitted to execute kernel-mode and interrupt code. In this way, vital operating system data structures that are manipulated by kernel-mode and interrupt code are protected from simultaneous modification by both processors. This arrangement eliminates the need to synchronize access to these data structures.

The primary and attached processors communicate using the interrupt handling mechanism of VAX/VMS in conjunction with the hardware interrupt mechanism of the MA780 shared memory subsystem.

For example, the primary processor interrupts the attached processor when any one of the following events occurs:

- The primary processor invalidates a system space address in its translation buffer.

- The primary processor receives an AST for the process currently executing on the attached processor.

- The primary processor requests a bugcheck.

Similarly, the attached processor interrupts the primary processor when any one of the following events occurs:

- The attached processor requests a reschedule event.

- The attached processor detects an event (such as an error) that must be logged.

- The attached processor requests a bugcheck.

A.3 ATTACHED PROCESSOR STATES

A location in the loaded multiprocessing code contains information on the current state of the attached processor. This location, called the "state variable," indicates the attached processor is in one of six possible states: INITIALIZE, IDLE, BUSY, EXECUTE, DROP, and STOP.

The transition path from one state to another is "owned" by either the primary or the attached processor. Only the processor that "owns" a particular transition path can cause a transition along that path. Thus, when the attached processor is in a particular state, only the processor that "owns" the transition path from that state can alter the state variable.

Figure A-1 shows the six attached processor states, the transition path from one state to another, and the "owner" of each transition

path. A transition path contains the letter P if the primary processor controls the transition along that path or the letter A if the attached processor does.

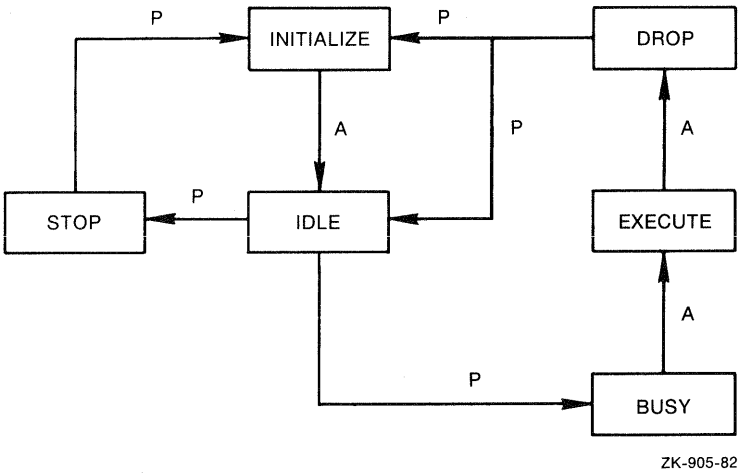

Figure A-1: Attached Processor States

The attached processor is set to the INITIALIZE state when the multiprocessing code is loaded by the DCL command START/CPU.

After the attached processor has executed its initialization code, it changes the state variable to IDLE.

The primary processor checks the state variable whenever it performs a scheduling operation. If the state variable is set to IDLE, the primary processor schedules a process to run on the attached processor. When the primary processor schedules a process to run on the attached processor, it sets the state variable to BUSY.

The attached processor continuously checks the state variable when the state variable is set to IDLE. When the attached processor finds the state variable set to BUSY, it sets the state variable to EXECUTE and begins executing the process.

The BUSY and EXECUTE states, though similar, must be unique so that special conditions such as powerfail can be handled correctly. If powerfail occurs on the attached processor when the state variable is set to BUSY, the attached processor simply halts. However, if the state variable is set to EXECUTE in this situation, the attached processor first saves the context of the current process (by executing a Save Process Context instruction) and then halts.

The attached processor executes the process until one of the following conditions arises:

- The process exhausts its quantum of CPU time.
- The process requires operating system services that are executed in kernel mode.

When the attached processor stops executing a process for one of the above reasons, it takes the following action:

1. It executes a SVPCTX instruction.
2. It sets the state variable to DROP.
3. It interrupts the primary processor.

When the primary processor receives the interrupt sent to it from the attached processor, it takes the following action:

1. It places its own current process in the appropriate scheduling queue.
2. It places the attached processor's current process in the appropriate scheduling queue.
3. It sets the state variable to IDLE.
4. It performs a scheduling operation. Because the state variable is set to IDLE, the primary processor schedules a process to run on the attached processor.
5. It sets the state variable to BUSY.

Once the state variable is set to IDLE, the state transition path, as diagrammed in Figure A-1, has come full circle. However, an additional state exists, the STOP state, that is not part of the normal transition path.

The primary processor sets the state variable to STOP when either one of the following occurs:

- The system manager (or other user with CMKRNL privilege) issues the DCL command STOP/CPU.
- The primary processor requests a bugcheck.

The primary processor sets the state variable from STOP to INITIALIZE when the DCL command START/CPU is issued.

When the attached processor is in the INITIALIZE or STOP states, the primary processor does not schedule work for or request any action from the attached processor.

A.4 SCHEDULING

The primary processor schedules all work on the system, both for itself and for the attached processor.

The scheduling algorithm used by the primary processor is basically the same as that used in a single-processor VAX-11/780 system. However, the following changes to the basic VAX/VMS scheduling algorithm are unique to the VAX-11/782 system:

- The primary processor always schedules a process to run on the attached processor before it schedules a process for itself to execute.
- The primary processor schedules a process for the attached processor only if that process does not require immediate execution in kernel mode.

- Scheduling is preemptive on the primary processor, but it is not preemptive on the attached processor. Thus, if the attached processor is busy executing a job while another job of higher priority becomes computable, the primary processor does not interrupt the attached processor in order to schedule the higher priority job for it.

- If the primary processor cannot schedule a computable process for the attached processor to execute because all computable processes require execution in kernel mode, the primary processor executes a process itself. However, in this situation, when the primary processor stops executing the process in kernel mode, an AST (Asynchronous System Trap) interrupt is generated. The primary processor processes this AST interrupt by performing a rescheduling operation. As a result, the primary processor schedules the process it was just executing for the attached processor.

  Note that if there is only one executable process, the AST interrupt is not issued. As a result, the primary processor executes the process even though the attached processor is idle and the process is not executing in kernel mode. In this way, AST and rescheduling overhead is avoided.

A.5 EXCEPTION HANDLING

Exception handling refers to the servicing of any of three types of exceptions: faults, traps, or aborts.

If an exception causes a transition to kernel mode, its service routine is executed by the primary processor, no matter which processor the exception occurred on.

In general, when an exception occurs on the attached processor, the attached processor interrupts the primary processor. The primary processor handles the exception and reschedules another process for the attached processor.

The following sections discuss some of the exceptions whose service routines are part of the multiprocessing code that is executed by the primary processor.

A.5.1 AST Delivery

The delivery of an AST (Asynchronous System Trap) to a process actually involves a software interrupt at IPL 2. All interrupt code is executed by the primary processor because it involves modification of the system data structures and thus requires synchronization.

When an AST is delivered for a process running on the attached processor, the primary processor interrupts the attached processor. The attached processor returns the process to the primary processor for delivery of the AST by means of a rescheduling request.

A.5.2 Quantum End

The System Generation (SYSGEN) parameter QUANTUM defines both the maximum amount of CPU time a process can receive before it is rescheduled and the minimum amount of time a process must remain in memory before it can be swapped out.

Quantum is enforced on both the primary and attached processors; that is, both processors check and deduct time from a process's quantum. The size of a process's quantum is the same regardless of which processor is executing the process.

A.5.3 Powerfail

When a fluctuation or drop in operating voltage occurs, the CPU generates a powerfail interrupt at IPL 30. The location of the powerfail interrupt service routine is found in the System Control Block (SCB). Since each processor has its own SCB, power failure can be handled differently for each processor.

If a power failure occurs on the primary processor, the primary processor executes the usual VAX-11/780 powerfail recovery sequence. The attached processor simply waits until the primary processor restarts.

If a power failure occurs on the attached processor, two different sequences of events occur depending on whether it is executing a process at the time of the power failure:

- If the attached processor is executing a process when the power failure occurs, the attached processor returns the current process to the primary processor by means of a rescheduling request and then enters the INITIALIZE state.
- If the attached processor is not executing a process when the power failure occurs, the attached processor interrupts the primary processor and then enters the INITIALIZE state.

The primary processor continues to execute processes whether or not the attached processor regains power.

When power is restored to the attached processor, the attached processor executes its initialization code in the same way as it does during a cold start.

A.5.4 Bugcheck

When either processor detects an internal inconsistency such as a corrupted data structure or an unexpected exception, it declares a bugcheck. If the system can continue to run, a nonfatal bugcheck is declared with a resultant error log entry. If the system cannot continue to run, a fatal bugcheck is declared, and the system shuts itself down in a controlled fashion.

In the case of a fatal bugcheck, a predetermined shutdown procedure occurs. The attached processor requests a fatal bugcheck by interrupting the primary processor. The following events occur no matter which processor requests the fatal bugcheck:

1. The primary processor interrupts the attached processor.
2. If the attached processor is currently executing a process, it executes a Save Process Context (SVPCTX) instruction.
3. The attached processor interrupts the primary processor.
4. The attached processor halts.

5. If the automatic restart button on the attached processor's console panel is set to the ON position, the attached processor executes the command procedure DEFBOO.CMD on its console floppy disk. This command procedure causes the attached processor to execute a self-jump instruction located in the Restart Parameter Block (RPB).

6. When the primary processor receives the interrupt from the attached processor, it writes the system dump file.

7. If the System Generation (SYSGEN) parameter BUGREBOOT is set and if the automatic restart button on the primary processor's console panel is set to the ON position, the primary processor begins the automatic reboot procedure; otherwise, it halts.

The automatic reboot procedure in the VAX-11/782 system allows both processors to reboot without human intervention.

A.5.5 Machine Check

A machine check is an exception that is reported when the CPU or an external adaptor detects an internal error. Machine checks that occur while the processor is executing in kernel or executive modes cause bugchecks. Machine checks in supervisor or user mode are handled by the appropriate exception-handling routine.

If a machine check occurs on either the primary or attached processor, control passes to a machine-check exception handler, which is executed by the primary processor.

If the handler determines that a recovery from the machine-check exception is possible, an entry is recorded in the error log file. If the handler determines that recovery is not possible, the machine-check exception handler causes a fatal bugcheck exception to be generated. The fatal bugcheck exception causes shutdown of the system, as described in Section A.5.4.

A.5.6 Error Logging

The error-logging subsystem is used to record device errors, processor-detected conditions, and other system events such as volume mounts and system start ups.

The same errors that are logged in a single-processor VAX-11/780 system are logged in the VAX-11/782 system. However, in the VAX-11/782 system, both the primary and attached processors can originate error log entries. Each error log entry contains the System Identification Register (SID), which identifies which processor originated the entry.

Though both the primary and attached processors can originate error log entries, only the primary processor can actually log the entries. The primary processor logs an entry by writing it in one of two error log buffers, whichever one is the current buffer. The attached processor writes error log entries in a temporary buffer, which the primary processor then copies to the current buffer.

Another routine executed by the primary processor detects when the current buffer is full and awakens the ERRFMT process. The ERRFMT process then writes the contents of the buffer to the error log file [SYSERR]ERRLOG.SYS.

A.5.7 Automatic Restart

Automatic restart of both processors after system failure is supported by VAX/VMS. The appropriate command procedures on the console floppy disks of both processors are executed providing that the automatic restart buttons on the console panels of both processors are set to the ON position.

In addition, to enable the automatic restart of the attached processor, the DCL command START/CPU must appear in the site-specific start-up command procedure SYS$MANAGER:SYSTARTUP.COM.

INDEX

VAX-11/782
User's Guide
AA-M543A-TE

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here

Cut Along Dotted Line

VAX-11/782
User's Guide
AA-M543A-TE

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

— — — Do Not Tear - Fold Here and Tape —

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

— — — Do Not Tear - Fold Here —

Cut Along Dotted Line