EK-11780-UG-001

VAX-11/780
Hardware User's Guide

digital equipment corporation • maynard, massachusetts

First Edition, February 1979

The material in this manual is for informational purposes and is subject to change without notice. Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

CONTENTS (Cont)

FIGURES

FIGURES (Cont)

TABLES

CHAPTER 1
INTRODUCTION

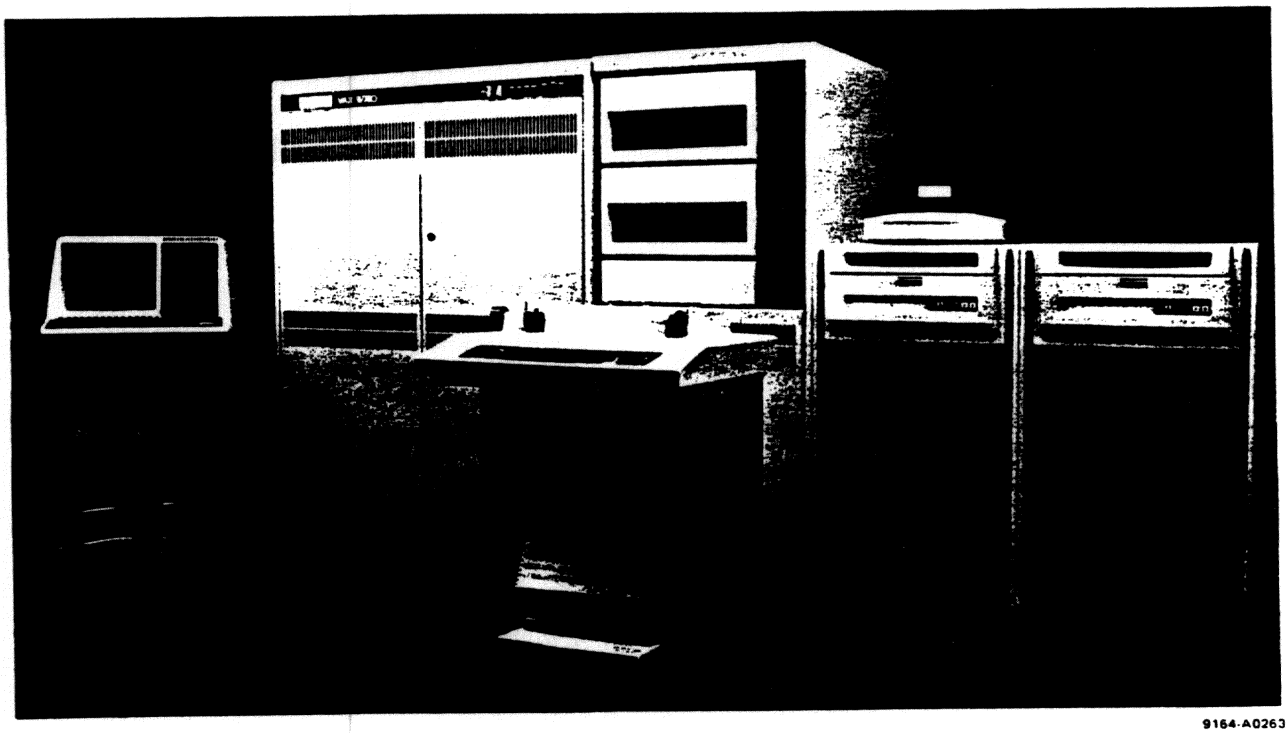
1.1 SCOPE

The *VAX-11/780 Hardware User's Guide* contains basic hardware operating and customer care information and procedures. A VAX-11/780 system is shown in Figure 1-1.

For detailed information on individual VAX-11/780 equipment, refer to the applicable equipment manual. For details on software operations, refer to the software user's guide. For details on the diagnostic programs, see the diagnostic user's manual.

The documentation related to the VAX-11/780 system can be divided into four categories.

1. System hardware manuals. The system hardware manuals are those manuals written to cover the basic (non-peripheral) VAX-11/780 system. The VAX-11/780 system hardware manuals are listed in Table 1-1.



9164-A0263

Figure 1-1 VAX-11/780 System Hardware

Table 1-1 VAX-11/780 System Hardware Manuals

Document Title	Control Number	Form
VAX-11/780 Power System Technical Description	EK-PS780-TD-001	Fiche
VAX-11/780 Installation Manual	EK-SI980-IN-001	Hard copy
DS780 Diagnostic System User's Guide	EK-DS780-UG-001	Hard copy
DS780 Diagnostic System Technical Description	EK-DS780-TD-001	Fiche
FP780 Floating Point Processor Technical Description	EK-FP780-TD-001	Fiche
REP05/REP06 Subsystem Technical Documentation	EK-REP06-TD-001	Fiche
VAX-11/780 Central Processor Technical Description	EK-KA780-TD-001	Fiche
VAX-11/780 Memory System Technical Description	EK-MS780-TD-001	Fiche
VAX-11/780 Architecture Handbook	EB07466	Hard copy

2. System peripheral manuals. A listing of a number of manuals related to VAX-11/780 peripheral equipment is contained in Table 1-2. This listing provides an entry level for additional information on the peripheral equipment.

3. VAX-11/780 software documentation (Table 1-3). The VAX/VMS software release distribution kit includes a software documentation set that contains programming and operating information associated with the VAX/VMS operating system. DIGITAL ships a complete software documentation set with each software system; however, the number and type of documents shipped to an installation depends on the license or licenses that the customer has purchased.

   All VAX/VMS customers receive the basic software documentation set consisting of the 28 documents packaged in Volumes 1 through 4. Customers receive optional documentation packaged in Volume 5 only when they purchase a special license or licenses for optional software.

4. Program listings. The VAX diagnostic programs are available on microfiche and are contained in the VAX-11/780 microfiche library.

Hard copy documents can be ordered from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532
Attn: Communications Services (NR2/M15)

For information concerning microfiche libraries, contact:

Digital Equipment Corporation
Technical Documentation
Micropublishing Group, PK3-2/T12
129 Parker Street
Maynard, MA 01754

Table 1-2 VAX-11/780 Peripheral Manuals

Document Title	Control Number	Form
LA35/LA35 DECwriter II User's Manual	EK-LA3635-OP-002	Hard copy
VT52 DECscope Maintenance Manual	EK-VT52-MM-001	Hard copy
RM03 Disk Subsystem User's Manual	EK-RM03-UG-001	Hard copy
RP05/RP06 DECdisk Storage Drive Technical Manual	ER-00012 67-01/51.20-01	Hard copy
TE15/TE10W/TE10N DECTAPE Transport Maintenance Manual	EK-OTE16-TM-001	Hard copy
RX8/RX11 Floppy Disk Maintenance Manual	EK-RX01-MM-002	Hard copy
LSI.PDP-03 User's Manual	EK-LSI11-TM-003	Hard copy
LP11/LSI1/LA11 Line Printer User's Manual	EK-LP11S-OP-001	Hard copy
CR11/CM11 Card Reader System Manual	EK-CR11-TM-004	Hard copy
LA180 DEC Printer Maintenance Manual	EK-LA180-MM-002	Hard copy
PDP-11 Peripherals Handbook	EB05961	Hard copy

Table 1-3 VAX-11 Software Documentation

Volume	Document Title	DIGITAL Order Number
Volume 1A System Reference	VAX/VMS Primer	AA-D030A-TE
	VAX/VMS Summary Description	AA-D022A-TE
	VAX/VMS Information Directory	AA-D016A-TE
	VAX/VMS Release Notes	AA-D015A-TE
	VAX-11 Software Installation Guide	AA-D021A-TE
	VAX/VMS System Services Reference Manual	AA-D018A-TE
Volume 1B System Reference	VAX/VMS Command Language User's Guide	AA-D023A-TE
	VAX-11 Linker Reference Manual	AA-D019A-TE

Table 1-3 VAX-11 Software Documentation (Continued)

DIGITAL Volume	Document Title	Order Number
	VAX-11 Symbolic Debugger Reference Manual	AA-D026A-TE
Volume 1C System Reference	VAX-11/RSX-11M Programmer's Reference Manual	AA-D020A-TE
	VAX-11/RSX-11M User's Guide	AA-D020A-TE
	VAX-11 MACRO Language Reference Manual	AA-D032A-TE
	VAX-11 MACRO User's Guide	AA-D033A-TE
Volume 2A System Procedures	VAX-11 Common Run-time Procedure Library Reference Manual	AA-D036A-TE
	VAX-11 Text Editing Reference Manual	AA-D029A-TE
Volume 2B System Procedures	VAX/VMS Operator's Guide	AA-D025A-TE
	VAX/VMS System Manager's Guide	AA-D927A-TE
	VAX/VMS System Messages and Recovery Procedures Manual	AA-D017A-TE
	VAX/VMS UETP User's Guide	AA-D643A-TE
	VAX-11 Disk Save and Compress User's Guide	AA-D739A-TE
Volume 3 VAX/VMS I/O	VAX/VMS I/O User's Guide	AA-D028A-TE
	Introduction to VAX-11 Record Management Services	AA-D028A-TE
	VAX-11 Record Management Services Reference Manual	AA-D031A-TE
	VAX-11 Record Management Services User's Guide	AA-D781A-TE
Volume 4 RMS-11/ SORT	IAS/RSX-11M RMS-11 MACRO Programmer's Reference Manual	AA-0002A-TC
	Introduction to RMS-11	AA-0001A-TC
34	RSX-11M/RMS-11 Utilities User's Guide	AA-D083A-TC
	PDP-11 SORT Reference Manual	AA-3341C-TC
Volume 5A Optional Software	VAX-11 FORTRAN IV-PLUS Language Reference Manual*	AA-D034A-TE

*Available under separate license.

Table 1-3 VAX-11 Software Documentation (Continued)

Volume	Document Title	DIGITAL Order Number
FORTRAN IV-PLUS	VAX-11 FORTRAN IV-PLUS User's Guide*	AA-D035A-TE
Volume 5B Optional Software	PDP-11 FORTRAN Language Reference Manual*	DEC-11-LFLRA-C-D
FORTRAN IV	PDP-11 FORTRAN Language Reference Manual Update Notice No: 1*	DEC-11-LFLRA-C-DN1
	IAS-RSX-11 FORTRAN IV User's Guide*	** DEC-11-LMFUA-D-D
Volume 5C Optional Software	PDP-11 COBOL Language Reference Manual*	AA-1749D-TC
COBOL	PDP-11 COBOL User's Guide*	** AA-1757C-TC
	PDP-11 COBOL Pocket Guide*	AA-1750C--C
Volume 5D	BASIC PLUS-2 Language Reference Manual*	AA-0153A-TK
BASIC PLUS-2	BASIC PLUS-2 RSX-11M/IAS User's Guide*	AA-0157A-TC
Volume 5E Optional Software	DECnet-VAX System Manager's Guide*	AA-D902A-TE
DECnet	DECnet-VAX User's Guide*	AA-D901A-TE
Volume 5F Optional Software	User's Guide to DATATRIEVE-AA*	AA-C742A-TC

*Available under separate license.

1.2 SYSTEM FEATURE SUMMARY

The VAX-11/780 system is a multi-user, multi-language, multi-programming, high-performance system. The VAX-11/780 combines a 32-bit architecture with a virtual memory operating system (VAX/VMS) and a memory management system.

The hardware (microcode) contains a native instruction set that includes integral floating-point, packed decimal arithmetic, and character and string instructions. The hardware also includes a compatibility mode instruction set that is a subset of the PDP-11/70. Programs can be executed concurrently in the native and compatibility modes.

Some of the instructions in the native set are direct counterparts to high-level language statements. The software system supports high-level languages that use these instructions to produce compiled code.

1.3 HARDWARE SYSTEM

The VAX-11/780 system (Figure 1-2) contains a central processor unit (CPU), a main memory subsystem, an I/O sub system, a console subsystem, and peripheral equipment.

The CPU operates on data as defined by the user program. (The user program is interpreted and controlled by the VAX/VMS operating system.) The CPU receives the program data and instructions via the main system interconnect (synchronous backplane interconnect). The CPU contains two microprogram instruction sets (VAX-11/780 native mode and PDP-11 compatibility mode). The CPU manipulated data is sent back to storage (main memory or peripheral devices) via the synchronous backplane interconnect (SBI).

The SBI is a part of the I/O subsystem that also includes the Massbus and the Unibus adapters. The Massbus connects the high-speed peripheral devices to the VAX-11/780. The Unibus connects all other peripheral devices to the VAX-11/780. The I/O subsystem performs a data buffer and transfer function.

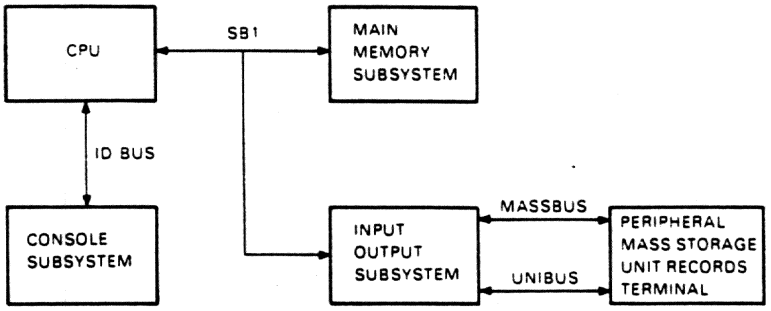


Figure 1-2 VAX-11/780 Overall Block Diagram

The user commands and programs are entered via the peripheral device. The user program data is transferred to the CPU under control of the VAX/VMS operating system. The VAX/VMS operating system is the system scheduler, the file manager, the translator, and the virtual memory manager. The VAX/VMS operating system is resident in the main memory subsystem.

All or parts of one or more user programs can also be stored in the main memory. Program data resident in the peripheral devices is accessed under the control of the VAX/VMS operating system.

The console subsystem is the operator's link to the CPU. The VAX-11/780 can be booted from and diagnostic programs can be run from the console terminal. Also, changes can be made to the microprogram instruction sets and to the VAX/VMS operating system from the console terminal.

1.3.1 Central Processor Unit (CPU)

The VAX-11/780 CPU is a high-speed microprogrammed, 32-bit computer that provides a full 32-bit operational capability (32-bit data and 32-bit address). Figure 1-3 is a functional block diagram of the CPU.

For normal operation, data flow to and from the CPU is over the SBI via the SBI control. Data received on the SBI may be temporarily stored in the data cache. The VAX/VMS operating system instructions are decoded by the instruction buffer. The decode is used to control the operation of the writable diagnostic control store (WDCS) function.

The two microprogrammed instruction sets (VAX-11/780 native mode and PDP-11 compatibility mode) are contained in the WDCS. The basic instruction sets are contained in ROM but can be modified and added to in RAM. The changes are input via the console. The microprogram instruction set controls the operations in the data paths function. The data paths function contains a data paths section, an address section, an arithmetic section, and an exponent section. These sections are independent and can operate simultaneously.

The translation buffer is a two-way buffer that stores virtual to physical address translations for the process currently executing in the system. The virtual to physical location is performed by the CPU microcode.

If data from a physical address is called for that is not in the data cache, that data is then retrieved from memory. When the data has been operated upon (an instruction is completed) that data is transferred back to memory. For a more detailed description of virtual memory concepts, refer to Appendix A.

The floating-point accelerator is an option (Paragraph 1.3.5).

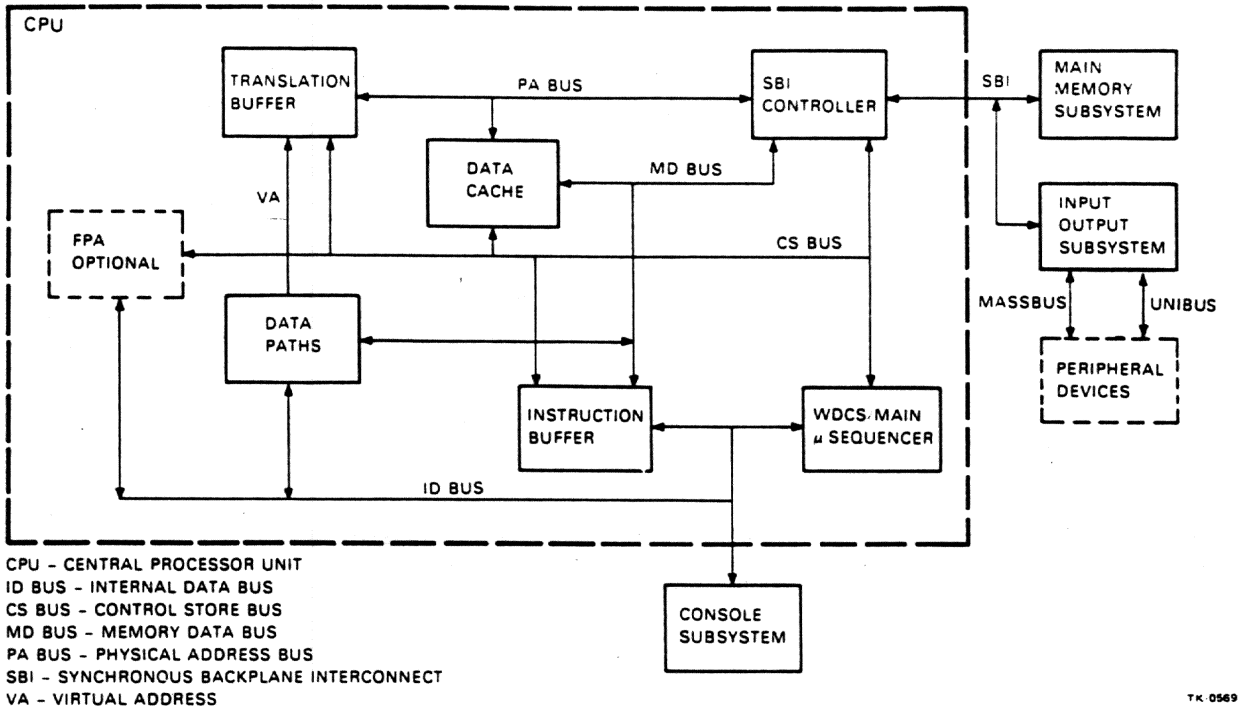


Figure 1-3 Central Processor Unit, Functional Block Diagram

1.3.2 I/O Subsystem

The I/O subsystem contains the SBI, the Massbus, and the Unibus. Figure 1-4 is a functional block diagram of the I/O subsystem and the peripheral devices.

1.3.2.1 SBI – The SBI is the system's internal backplane and bus that conveys addresses, data, and control information between the CPU, main memory, and peripheral devices. The SBI has a cycle time of 200 ns and can transfer 32 bits each cycle. Data transfers are made 64 bits at a time (two consecutive cycles). The maximum SBI transfer rate is 13.3 million bytes/s. The SBI provides a high-level throughput and reliability because it uses:

Time-division multiplexing
Distributed priority arbitration
Parity and protocol checking on every transfer
Transaction history recording

The protocol, or sequence in which operations occur on the SBI, is time-division multiplexed to increase the effective bandwidth of the bus. Time-division multiplexing means that the transactions for one transfer operation are interleaved with the transactions for another transfer operation. Thus, several operations can be in progress over the same period of time. For example, the CPU can ask a memory controller to read some data; the same memory controller might then transfer previously requested data to an I/O device before transferring the requested data to the CPU.

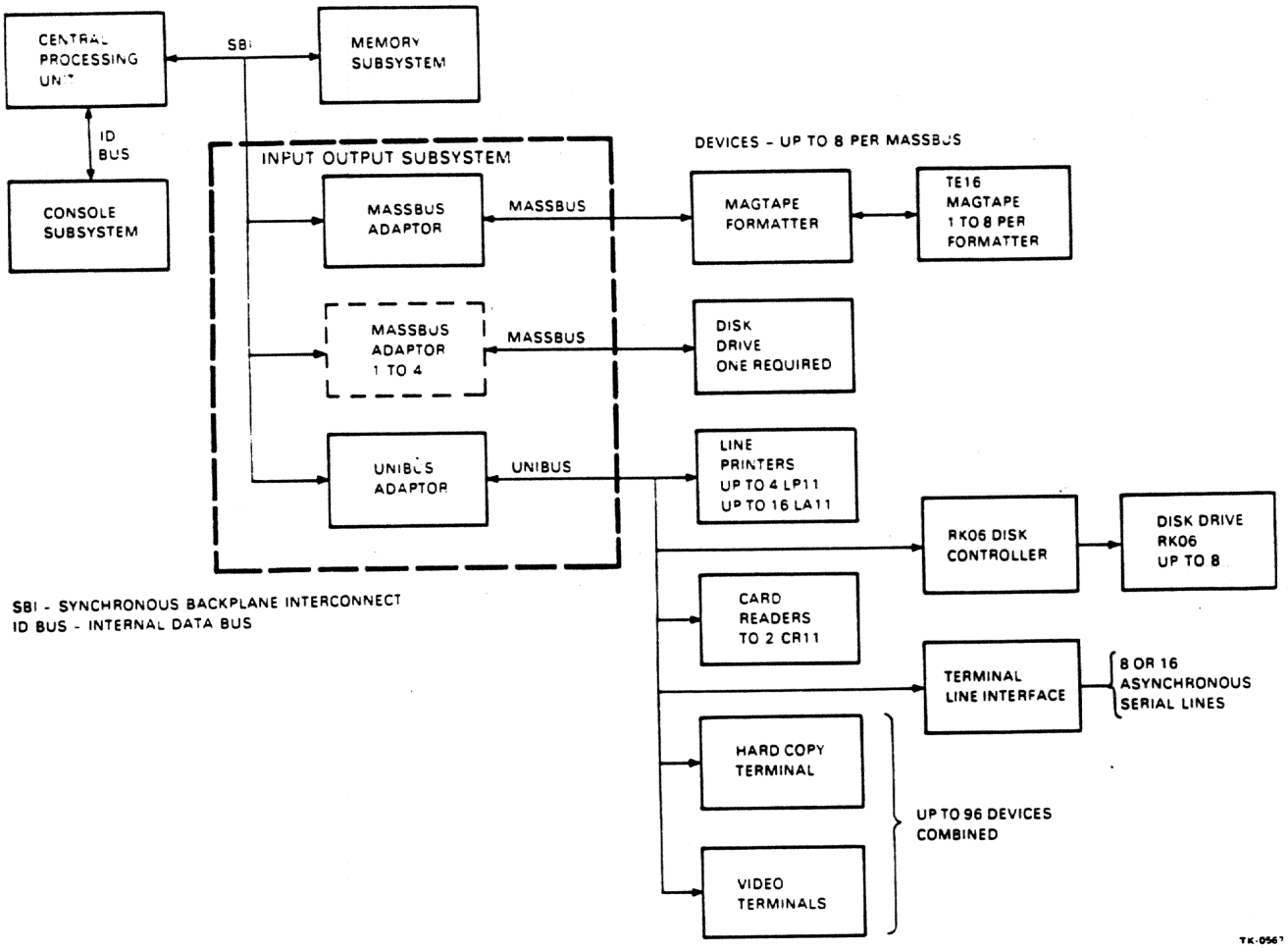


Figure 1-4 I/O Subsystem Peripheral Device, Functional Block Diagram

Arbitration on the SBI is distributed. Every unit on the SBI has its own arbitration line. Arbitration lines are ordered by priority. Every unit monitors all the arbitration lines in each cycle to determine if it will get the next cycle. Any unit on the SBI (except the CPU clock) can fail without causing a failure of the entire bus.

To ensure the integrity of the signals transmitted, the SBI includes several error checking and diagnostic mechanisms, such as:

Parity checking on data, addresses, and commands
Protocol checking in each interface
A history silo of the last 16 SBI cycles

1.3.2.2 Massbus – The processor interface for a Massbus peripheral is the Massbus adapter. The Massbus adapter performs control, arbitration, and buffering functions. Up to four Massbus adapters can be placed on the SBI.

Each Massbus adapter includes its own address translation map that permits scatter/gather disk transfers. In scatter/gather transfers, physically contiguous disk blocks can be read into or written from discontiguous blocks of memory. The translation map contains the addresses of the pages, which may be scattered throughout memory, from or to which the contiguous disk transfer takes place.

Each Massbus adapter includes a 32-byte silo data buffer. Data is assembled in 64-bit quadwords (plus parity) to make efficient use of the SBI bandwidth. On transfers from memory to a Massbus peripheral, the Massbus adapter anticipates upcoming Massbus data transfers by fetching the next 64 bits from memory before all of the previous data is transferred to the peripheral.

1.3.2.3 Unibus – All devices other than the high-speed disk drives and magnetic tape transports are connected to the Unibus, an asynchronous bi-directional bus. The Unibus is connected to the SBI through the Unibus adapter. The Unibus adapter does priority arbitration among devices on the Unibus.

The Unibus adapter provides access from the processor to the Unibus peripheral device registers and to Unibus memory by translating Unibus addresses, data, and interrupt requests to their SBI equivalents, and vice versa. To make the most efficient use of the SBI bandwidth, the Unibus adapter provides buffered direct memory access (DMA) data paths for up to 15 nonprocessor request (NPR) devices. Each of these channels has a 64-bit buffer (plus byte parity) for holding four 16-bit transfers to and from Unibus devices, The result is that only one SBI transfer (64 bits) is required for every four Unibus transfers. The maximum aggregate transfer rate through the buffered data paths is 1.5 million bytes/s. On SBI to Unibus transfers, the Unibus adapter anticipates upcoming Unibus requests by prefetching the next 64-bit quadword from memory as the last 16-bit word is transferred from the buffer to the Unibus. By the time the Unibus device requests the next word, the Unibus adapter has it ready to transfer.

Any number of unbuffered DMA transfers are handled by one direct data path. Every 8- or 16-bit transfer requires one 32-it transfer on the SBI. The maximum transfer rate through the direct data path is 500K bytes/s.

The Unibus adapter permits concurrent program interrupt and unbuffered and buffered data transfers. The aggregate throughput rate of the direct data path, plus the 15 buffered data paths, is 1.5 M bytes/s.

1.3.3 Main Memory Subsystem

The main memory subsystem contains one or two memory controllers. Each controller can handle from one to 16 MOS RAM arrays. Figure 1-5 is a functional block diagram of the main memory subsystem.

Main memory arrays are MOS RAM integrated circuits with a cycle time of 600 ns. A memory controller can access a maximum of 4,194,294 bytes (41M bytes). Two memory controllers can be connected to the SBI, yielding a maximum of 8M bytes of physical memory that can be available on the system. (The maximum total physical address space is 2^{29} or approximately 512M bytes.) The minimum required memory is 256K bytes.

A memory controller will buffer one command while it processes another to increase system throughput. Main memory can also be interleaved (where two memory controllers are each addressing the same amount of memory) to increase the available memory bandwidth. The memory system employs error checking and correction (ECC) that corrects all single bit errors and detects all double bit errors.

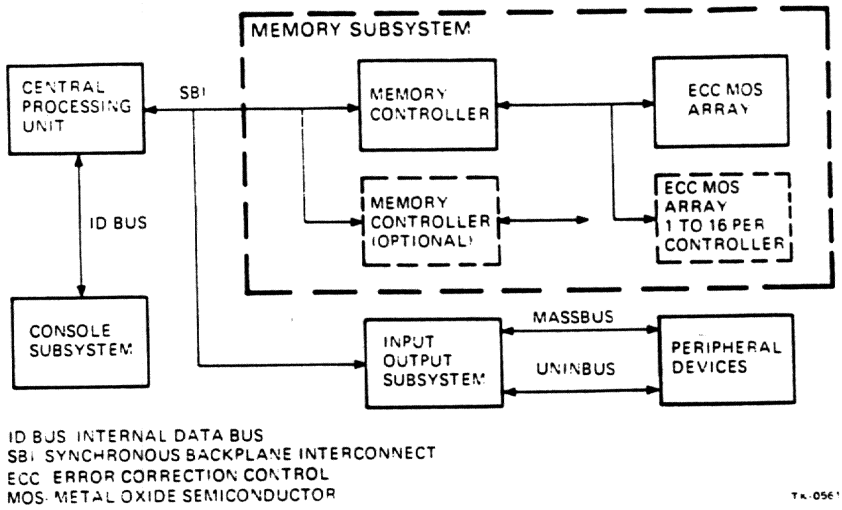


Figure 1-5 Main Memory Subsystem, Overall Block Diagram

1.3.4 Console Subsystem

The console subsystem contains a console interface, the VAX-11/780, the control panel, an LSI-11 microprocessor, a RX01 floppy disk, a serial line interface, and an LA36 terminal. A remote terminal and an interconnecting line unit are optional. Figure 1-6 is a functional block diagram of the console subsystem.

Using the console terminal, the operator can examine and deposit data in memory locations or the processor registers; halt the processor; step through instruction streams; and boot the operating system.

The console also serves as a microdiagnostic monitor. The floppy disk provides basic diagnostic programs and data. The operator can load segments of diagnostic microcode from a floppy disk into the WDCS, controlling execution and reporting results.

The console is further used for updating the software with maintenance releases and for loading optional software products distributed on floppy disk.

An EIA serial line interface and modem can be added to the console to provide remote diagnosis and automated testing.

1.3.5 Floating-Point Accelerator Option

The floating-point accelerator is an optional high-speed processor extension. When included in the processor, the floating-point accelerator executes the addition, subtraction, multiplication, and division instructions that operate on single- and double-precision floating-point operands, including the special EMOD and POLY instructions in both single- and double-precision formats. Additionally, the floating-point accelerator enhances the performance of 32-bit integer multiply instructions.

The processor does not have to include the floating-point accelerator to execute floating-point operand instructions. The floating-point accelerator can be added or removed without changing any existing software.

When the floating-point accelerator is included in the processor, a floating-point operand register-to-register add instruction takes as little as 800 ns to execute. A register-to-register multiply instruction takes as little as 1 μs. The inner loop of the POLY instruction takes approximately 1 μs/degree of polynomial.

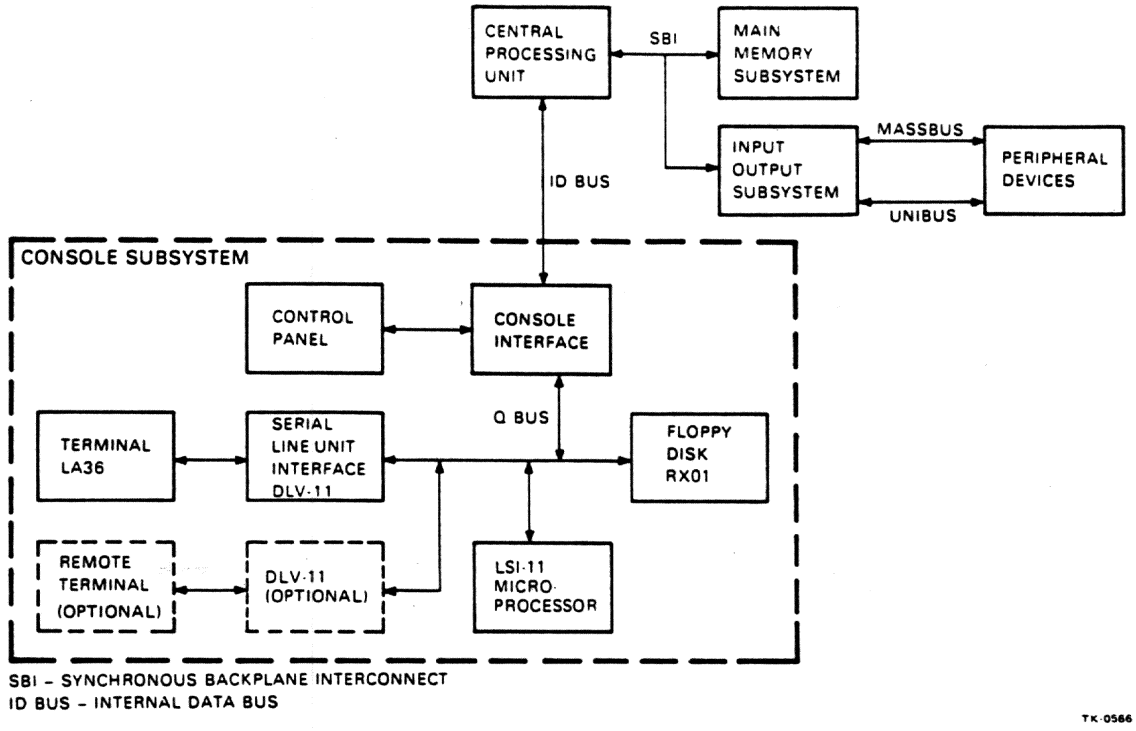


Figure 1-6 Console Subsystem, Overall Block Diagram

1.4 PERIPHERAL DEVICES

VAX-11/780 supports four types of peripheral devices:

1. Mass storage peripherals, such as disk and magnetic tape
2. Unit record peripherals, such as line printers and card readers
3. Terminals and terminal line interfaces
4. Interprocessor communications links.

Devices use either one of two types of data transfer techniques: direct memory access (DMA) or programmed interrupt request. The mass storage disk and magnetic tape devices and the interprocessor communications link are capable of DMA data transfers. The DMA devices are also called NPR because they can transfer large blocks of data to or from memory without processor intervention until the entire block is transferred.

The unit record peripherals and terminal interfaces are called programmed interrupt request devices. These devices transfer one or two bytes at a time to or from assigned locations in physical address space. Software then transfers the data to or from a buffer in physical memory.

1.4.1 Mass Storage Peripherals

The mass storage peripherals include three types of moving head disk drives and one type of magnetic tape transport:

1. The high-speed, large capacity RP05 and RP06 disk drives
2. The high-speed, high throughput RM03 disk drive
3. The medium capacity RK06 disk drive
4. The TE16 magnetic tape transport.

The RP06, RP05, and RM03 disk and the TE16 magnetic tape controllers are Massbus peripherals. It is possible to connect as many as four Massbus adapters to the central processor (via the SBI). Each Massbus can support up to eight logical devices. Each disk drive and magnetic tape formatter counts as a logical device. Each magnetic tape formatter can support as many as eight TE16 magnetic tape transports. For example, a Massbus can support up to eight RP06 disk drives or up to eight magnetic tape formatters. Disk drives and tape formatters can be mixed on the same Massbus.

1.4.1.1 Disks – The disk subsystems all provide high performance and reliability. They feature accurate servo positioning, error correction, and offset positioning recovery. Table 1-4 summarizes the capacities and speeds of the disk devices.

All disk drives use top loading removable media. The RM03, RP05, and RP06 disk drives can be mixed on the same Massbus controller.

Table 1-4 Disk Devices

Disk	RK06	RM03	RP05	RP06
Pack capacity	14 MB	67 MB	88 MB	176 MB
Peak transfer rate (/s)	538 KB	1200 KB	806 KB	806 KB
Avg seek time	38 ms	30 ms	28 ms	28 ms
Avg rotational latency	12.5 ms	8.3 ms	8.3 ms	8.3 ms

The Unibus accepts one RK611 disk controller. The controller can support up to eight RK06 disk drives. In small system configurations, where the RK06 is used as the systems device, two RK06 drives are required in the configuration.

To decrease the effective access time and increase throughput, the operating system's disk device drivers provide overlapped seeks for all disk units on a controller. All I/O transfers, including write checks, are preceded by a seek, except when the seek is explicitly inhibited by diagnostic software. On Massbus devices, seeks to any unit can be initiated at any time and do not require controller intervention. During seeks, the controller is free to perform a transfer on any unit other than the one on which the seek is active. If a data transfer was in progress at the time of completion, the driver processes the attention interrupts caused by seek completion when the controller is free.

1.4.1.2 Magnetic Tape – A magnetic tape transport holds one 9-track reel with a capacity of 40 million char/reel. The transport reads 45 in/s with an average transfer time of 14 μs/byte at 1600 bits/in.

The operating system's magnetic tape device driver supports the read reverse operation, which enables a program to request a sequential read of the block preceding the block at which the tape is positioned. Writing occurs only while the tape is moving forward.

1.4.2 Unit Record Peripherals

The operating system normally treats line printers and card readers as spooled sharable devices managed by multiple operator-controlled queues. The devices can also be allocated to individual programs.

The operating system's line printer handling includes line and page counting for job accounting. The user can specify carriage control as one line/record, FORTRAN conventions contained within the record itself; or general pre- and post-spacing (within the limits of the hardware capabilities).

The operating system's card reader driver interprets the encoded punched information using the American National Standard 8-bit card code. The driver uses a special punch outside the data representation to indicate end-of-file.

1.4.2.1 LP11 Line Printers – Up to four LP11 series line printers can be connected to the VAX-11/780 system. The LP11 series printers are impact-type, rotating drum, serial interface line printers. They feature full line buffering, a static eliminator, and a self-test capability.

All models are 132-column printers that can accept paper 4 to 16-3/4 inches wide with up to 6-part forms. They print 10 char/in horizontally, and 6 or 8 lines/in vertically (switch selectable). All models are available with either upper (64) or upper/lower (95) character sets (including numbers and symbols).

1.4.2.2 LA11 Line Printer – Up to 16 LA11 line printers can be connected to the system. The LA11 is an extremely low-cost, highly reliable parallel interface printer. The LA11 prints at speeds up to 180 char/s. The print set consists of the ASCII characters, including 95 upper- and lower-case letters, numbers, and symbols. Characters are printed using a 7 × 7 matrix with horizontal spacing of 10 char/in and vertical spacing of 6 lines/in.

Adjustable pin-feed tractors allow for a variable-form width of 3 to 14-7/8 inches (up to 132 columns). A forms length switch sets the top-of-form to any of 11 common lengths, with fine adjustment for accurate forms placement. The printer can accommodate multipart forms (with or without carbons) of up to six parts.

1.4.2.3 CR11 Card Reader – Up to two CR11 card readers can be connected to the system as programmed interrupt request devices. The CR11 reads up to 285 80-column punched cards/min. The card reader has a high tolerance for cards that have been nicked, warped, bent, or subjected to high humidity. The card reader uses a short card path, with only one card in the track at a time. It uses a vacuum pick mechanism and keeps cards from sticking together by blowing a stream of air through the bottom half-inch of cards in the input hopper. The input hopper holds up to 400 cards, and cards can be loaded and unloaded while the reader is operating.

1.4.3 Terminals and Interfaces

Up to 96 interactive terminals can be connected to the VAX-11/780 system. The operating system's terminal driver provides full-duplex handling for both hard copy and video terminals.

Programs can control terminal operations through the terminal driver. The terminal driver supports many special operating modes for terminal lines. A program can enable or disable the following modes by calling a system service.

1.4.3.1 LA36 Hard-Copy Terminal – The LA36 is an exceptionally reliable hard-copy terminal. The LA36 keyboard generates 128 ASCII characters, consisting of 95 upper- and lower-case printing characters and 33 control characters. The keyboard layout is similar to a standard office typewriter. In addition, the LA36 is available with optional special character sets, including various foreign language character sets.

Characters are printed using a 7 × 7 matrix with horizontal spacing of 10 char/in and vertical spacing of 6 lines/in. To ensure clear visibility of the printed line, the print head automatically retracts out of the way when not in operation. Adjustable pin-feed tractors allow for variable-form widths from 3 to 14-7/8 inches (up to 132 columns). The print mechanism will accommodate multipart forms (with or without carbons) of up to six parts.

The LA36 operates at speeds of 110, 150, or 300 baud (10, 15, or 30 char/s). Printable characters are stored in a buffer during the carriage return operation.

1.4.3.2 VT52 Video Terminal – The VT52 is an upper- and lower-case ASCII video terminal whose keyboard is similar to a typewriter keyboard. The operator can select or disable the audible response for keystrokes. A 3-key rollover feature ensures that keys typed in rapid sequence are received properly.

The VT52 display holds 24 lines of 80 characters. The VT52 has a variety of positioning functions for the cursor, and a variety of screen control functions. There are functions that move the cursor one position in any direction horizontally or vertically, tab the cursor, move the cursor to the top left-hand corner, and move it to any position on the screen. There are functions that scroll the screen up or down, completely erase the screen, and erase the display from the cursor position to the end of the line or screen.

1.4.3.3 DZ11 Terminal Line Interface – The DZ11 is a serial line multiplexer whose character formats and operating speeds are programmable on a per line basis. A DZ11 connects the Unibus with up to a maximum of 8 or 16 asynchronous serial lines. Each line can run at any one of 15 speeds.

Local operation with EIA terminals is possible at speeds up to 9600 baud. Remote dial-up terminals can operate full duplex at speeds up to 300 baud using Bell 103 or 113 modems, or up to 1200 baud using the Bell 212 modem.

1.4.4 Interprocessor Communications Link

The DMC11 communications link is designed for high-performance point-to-point interprocessor connection based on the DIGITAL data communications message protocol (DDCMP). The DMC11 provides local or remote interconnection of two computers over a serial synchronous link. Both computers can include the DMC11 and DECnet software, or both computers can include the DMC11 and implement their own communications software. For remote operations, a DMC11 can also communicate with a different type of synchronous interface provided that the remote system has implemented the DDCMP protocol.

1.5 THE VAX SOFTWARE SYSTEM

The VAX-11/780 software system includes the operating system, file and record management facilities, I/O drivers, the applications migration executive batch capabilities, on-line diagnostics and error log utilities, and the supported languages and utility programs.

NOTE

A description of the virtual memory concept is contained in Appendix B.

1.5.1 The VAX Virtual Memory Operating System (VAX/VMS)

Virtual memory is an interaction between hardware and software that logically extends the physical memory of a system onto disk space. From the programmer's viewpoint, the secondary storage locations are treated as though they were physical memory.

The size of virtual memory depends on the amount of physical memory available plus the amount of disk storage used for nonresident data and code. This design provides some obvious advantages.

With memory the user can write and execute arbitrarily large programs without worrying about addressing limitations. The VAX/VMS provides more virtual memory addresses than physical addresses. The system hardware and software cooperate to translate the virtual memory requirements of the user's program into whatever physical memory is available on the system. The VAX-11/780 physical memory configuration can change without requiring program redesign or relinking. The programmer (user) never has to structure his program, although he may, at his option, to achieve maximum efficiency and performance.

Virtual memory allows large programs (in excess of physical memory) to be executed. The program's virtual addresses are translated into the available physical memory.

The paging algorithm used by the VAX/VMS memory management leads to more predictable application performance. The system manager can manipulate the working set limit of individual users (the maximum amount of physical memory in which a particular job can execute) to achieve the desired trade-off between the performance of a single program, and the performance of the entire system in a multiuser environment.

The paging and swapping algorithms allow more programs to execute than the available physical memory would allow if all programs were totally resident. In addition to providing a larger addressing space, virtual memory systems need only hold a portion of a program in physical memory at one time. Efficient memory management enables the system to map several programs into memory at once.

The paging algorithm minimizes disk overhead. During execution, pages that have been removed from the working set remain temporarily in memory. When pages are needed by a program, the pager first checks to determine if they are in memory. If so, the pager can bring them back into the working set without reading them back from the disk. In addition, the pager keeps track of modified pages so that only those pages that have been changed are ever written to disk.

System management capabilities give users the ability to control program behavior. A user can establish page cluster sizes that minimize disk accesses. In addition, a user can lock pages in the working set. A privileged user can lock pages in physical memory, and can lock an entire working set in physical memory (never to be paged or swapped out) to maximize program performance.

Methods by which the VAX/VMS operating system benefits from a virtual memory system are developed in the following paragraphs.

1.5.2 The VAX-11/780 User Environment

1.5.2.1 System Management – VAX/VMS provides tools for a system manager and a system operator to control the configuration and operation of the system to realize its most effective use. The system manager authorizes system users, determines quotas to control virtual memory usage, and defines privileges and priority to each user. At run time, the system operator can initiate and terminate sessions, monitor system operations, load and unload volumes, and perform backup and other on-line maintenance activities.

1.5.2.2 User Access – To use the system, each user logs in by typing an identifying user name, account number (optional), and a password. If the user is authorized, access is granted automatically and the resources of the system are available for use.

VAX/VMS supports two interactive interfaces between the user and the system: the native mode and the compatibility mode.

The native set is a set of commands that a user types to initiate and control system operations (Appendix H). The language makes it easy for new users to become acquainted with the system because it prompts for all information needed to complete a command operation. In addition, the command language provides a help file for those who are not familiar with command formats. Native commands are usable in both the interactive and the batch environments.

For those users already familiar with RSX-11M operation, VAX/VMS also supports the RSX-11M command language (compatibility mode). Users may choose either command language.

1.5.2.3 Processes – When a user logs onto the VAX/VMS, a process (job) is created. In VAX/VMS the concept of a process is separated into two basic aspects: the image that executes, and that virtual address space and control information (referred to as process context) required for image execution.

For example, system and user programs exist under VAX/VMS as image files. An image file is the result of linking one or more object modules together. The image file is loaded by the image activator (a system service) and executes in the context of the process.

The process executes all of the images needed to perform the user's requests. If the user chooses to edit, assemble, and link a source program, for example, the process executes three images: the editor, the assembler, and the linker.

A VAX/VMS process can execute any image. Furthermore, a VAX/VMS process remains to execute subsequent images when the current or initial image is exited.

1.5.3 Virtual Address Space

Virtual address space is defined as all possible virtual addresses that an image executing in the context of a process can use (to identify instructions and data). The available virtual address space of this system is an array of bytes labeled 0 through $2^{32}-1$ or 4.3 gigabytes.

Figure 1-7 illustrates how the VAX-11/780's 4 gigabytes of virtual address space are allocated between the VAX/VMS and each process.

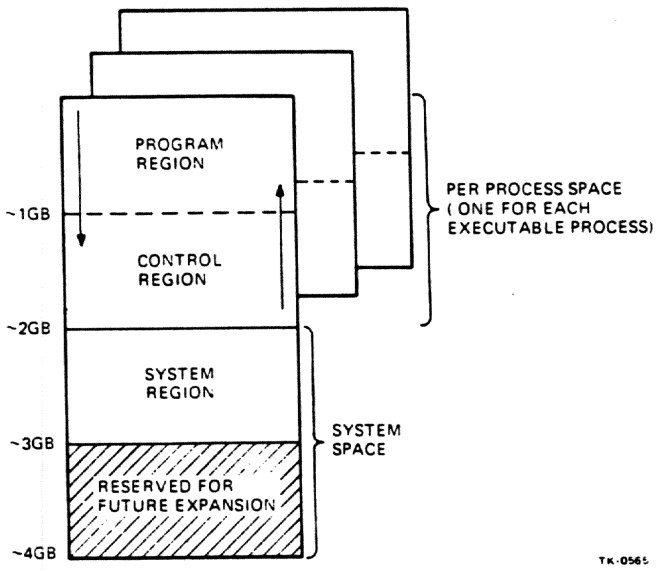


Figure 1-7 Virtual Address Space

The virtual address space is divided into four regions, each approximately 1 gigabyte long. The top 1 gigabyte region is reserved for future system expansion. Addresses in the system region are used by the VAX/VMS software.

Process virtual address space (addresses available for user code) comprises the lower 2 gigabytes (the control region and the program region). This address space contains information maintained by the system on behalf of the process (context information and user stacks–one per access mode per process), and the image currently being executed.

1.5.4 Memory Management

Memory management involves both the VAX-11/780 hardware and the VAX/VMS software with control of the pages of a process (through image activation and paging) and the protection of a process.

1.5.4.1 Pages – In the VAX-11/780 system, a process's virtual memory is subdivided into pages of 512 contiguous bytes each. This number makes efficient use of memory and it is the size of a physical block on a disk.

1.5.4.2 Image Activator – The image activator is responsible for making an image capable of running in the context of the requesting process. The image activator locates the image and sets up the proper control information.

1.5.4.3 Paging – Paging is the transparent moving of pages to and from secondary storage as they are needed. As a process executes, only those pages required for execution (i.e., the process's working set) need reside in physical memory. Remaining pages of the process can reside on secondary storage (e.g., disk) and be brought into physical memory when needed.

A maximum working set size for each process is set by the system manager so that individual processes will not use too much physical memory and degrade the execution of other processes. If a process is time-critical, the system manager may assign a large maximum working set size. Parts or all of a working set of a given process can be locked into memory to gain maximum performance.

The paging facility maintains page tables on the status and locations of all virtual pages of processes in the system. The system page table (always in physical memory) maps individual process page tables (which may or may not be in physical memory).

If a page is not in physical memory, the process page table entry denotes where the page is on disk. The translation buffer speeds page table lookup procedures (i.e., CPU performance) by storing the most frequently referenced page table entries in a high-speed memory.

The VAX operating system does process-oriented paging, i.e., it discards the oldest pages from a process when it is necessary to make room for new pages required by that process. The process pages only against itself. Individual processes doing a great deal of paging do not significantly degrade the performance of other active processes in the system.

Software caches in physical memory, called the free page list (for unmodified pages) and the modified page list (for modified pages), are reserved for the temporary storage of pages removed from processes' working sets. As (unmodified) newly removed pages are added to the bottom of the free page list, older pages work their way to the top and the space is eventually reused. As (modified) newly removed pages are added to the bottom of the modified page list, older pages work their way to the top and are eventually written to a paging file on disk. If a process needs a page that happens to be on one of these lists, the page is simply reallocated to the process, rather than reading the page back into memory from secondary storage.

The system handles all of the virtual to physical translations automatically for the user. The user can, if desired, control the paging structure for his process. He can set a page cluster size at link-time, thereby causing multiple pages to be mapped from disk to memory to minimize disk accesses. The user can lock pages in his working set. Privileged users can lock pages in memory, or lock the entire working set in memory, to gain maximum program performance.

1.5.4.4 Protection - Four hierarchical access modes provide protection for the system (Figure 1-8). These modes, from most to least privileged, are: kernel, executive, supervisor, and user.

Each page table entry contains a code specifying, for each of the four modes, the type of access (read only, read write, and no access) allowed to the associated memory page. If a less privileged mode can access a page, then a more privileged mode can also access the page.

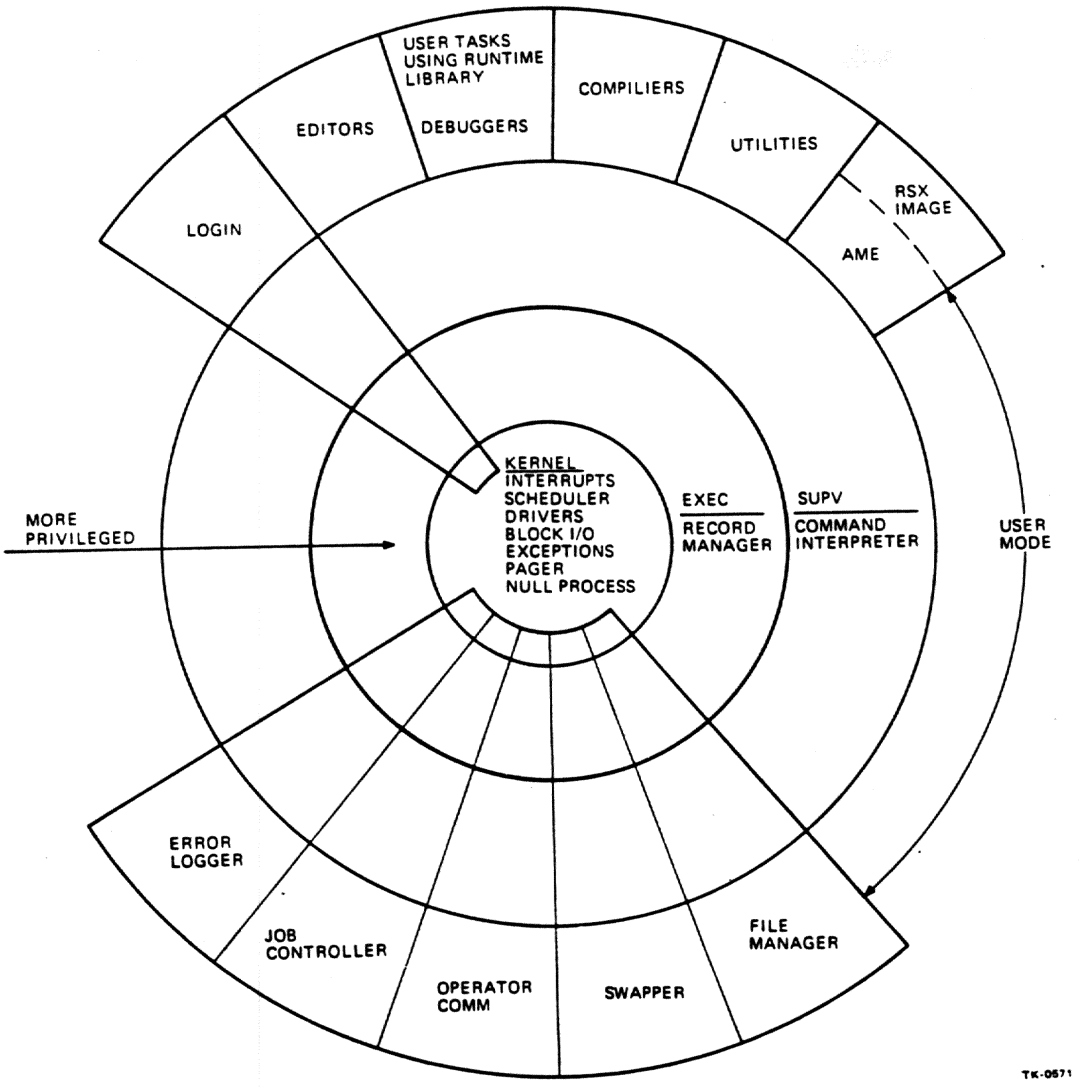


Figure 1-8 VAX's Use of Hierarchical Access Modes

In VAX/VMS, the system services (record management services, I/O drivers, etc.) are part of every process's virtual address space. However, individual pages are protected at different access modes. For example, the I/O drivers, scheduler, and record management service execute in kernel mode; the record manager executes in executive and kernel modes; the command language interpreters execute in supervisor mode; user software and utility programs usually execute in user mode (although the system manager can assign privileges to users allowing them to protect pages at higher levels).

The access modes only determine access to pages (they do not affect the virtual space in which programs are mapped). By assigning appropriate access modes to pages of a process, the operating system is protected from users who might inadvertently try to destroy operating system information.

1.5.5 Interprocess Communication

VAX/VMS provides a number of methods by which processes can communicate with one another.

- Implicit communication using shared files or shared memory (the run-time library is an example of system code that is shared).

  NOTE

  Memory data can be shared between native and compatibility mode programs, but code cannot because the instruction sets are different.

- Cooperating processes can synchronize using common event flags.

- Message communication is allowed through mailboxes. A mailbox is a buffer in virtual memory that is treated as if it were a record-oriented I/O device. Mailboxes are accessed through the operating system's I/O system, and processes can communicate by reading and writing messages through a mailbox. This method provides more protection, but requires system overhead.

1.5.6 Scheduling and Swapping

The VAX/VMS scheduler and swapper are responsible for ensuring that (in a multi-programming environment) processes receive processor time that is appropriate for their priority (which is controlled by assignment) and their ability to execute (which is controlled by system events).

The scheduler determines the order in which processes are executed. It uses fixed priorities for time-critical applications and recomputed priorities for interactive and batch processes.

VAX/VMS scheduling is priority-ordered, pre-emptive, round-robin/time-slicing, and event-driven. The VAX operating system defines 32 levels of priority. Any number of processes can execute at each level.

Of the 32 software levels, levels 16 through 31 are reserved for time-critical processes; once a priority is assigned to a process (either at login time, by the system manager, or by privilege system service), the system does not modify it.

Each process assigned to levels 0 through 16 is assigned a base priority; the process never drops below its base, and may even receive a temporary boost. The actual priority at which it is scheduled is recomputed by the system (based on the type of process and its execution history) to achieve the best possible balance among all processes. For example, the scheduler tries to maintain a balance between interactive and computer-bound processes; it favors interactive processes (and processes doing I/O) for faster response.

The scheduler always selects the highest priority, executable, resident process, and schedules by round-robin within the same priority level. The scheduler guarantees execution (for some minimum amount of time) for any process that it brings into memory from disk.

Events are occurrences that cause the status of a process to change. When the status changes, the process is moved to an appropriate priority queue to be scheduled for execution.

Swapping involves the moving of a process's entire working set in and out of memory in a multiuser environment. Maintenance of the balance set (processes in physical memory) is handled by the swapper, whose function is to keep the system scheduler supplied with executable processes by removing and adding processes to the balance set.

1.5.7 File and Record Management Facilities

1.5.7.1 Files-11 – The VAX/VMS file manager (Files-11) provides a facility for the access, creation, extension, and deletion of files. Designed into the file system is a scheme for volume and file protection that allows the owner of a file to deny all access or certain kinds of access to all users, others in the group, the system, and the owner.

Additional file management features include:

Alphabetically ordered directories for faster search,

Multi-volume files,

Duplication of critical information (index file header and home block) for improved reliability (loss of this information is fatal to the entire volume),

Improved allocation/deallocation of storage space,

Efficient handling of large files (theoretically 2^{32} blocks in one file, 2.2×10^{12} bytes) and large disks (also 2^{32} blocks),

Dynamic bad block handling,

File protection via user identification codes and passwords and by read/write protection.

1.5.7.2 RMS/VAX – Record management services (RMS) is a set of general-purpose file handling capabilities that allow user written application programs to create, access, and maintain data files and records within the files.

1.5.8 Network Communication (DECnet)

VAX/VMS supports DECnet Phase 2.

Access to network facilities is device-independent. A user who desires can exert control over operations to obtain error reports or notification of broken connections (interrupt messages, inbound connections). Full LOGIN protection applies to all network access.

1.5.9 Batch Facilities

VAX/VMS supports an extensive batch facility that utilizes the same command languages as the interactive portion of the system.

Batch facilities under VAX/VMS include the following.

- Multistream Batch. Jobs may be submitted to batch streams from the interactive environment (using a terminal command), from another batch job, or by any program using a system call. Batch jobs are queued, and a time may be specified after which a batch job will execute. The operator controls the number of batch jobs that can run concurrently. Whenever that limit is exceeded, the remaining jobs are held in a queue until a batch stream becomes available.

- Job Control. The method chosen to submit a batch job does not affect the structure of the application program. The user can control the flow of the batch job by specifying what error conditions cause the job to stop, and by suspending execution ($MOUNT command) until an operator loads a device.

- Conditional Branching. The user can specify in a batch stream where (in the stream) program control should transfer on command errors or errors returned by the programs that are executed in the stream.

- Spooled Printing. Printer output is generally spooled to a file-structured volume first, unless the (privileged) program allocates a printer. Print jobs (one or more files to be printed together) can be submitted from a terminal, from another batch job, from any program using a system call, and automatically at the end of the batch job. Multiple print queues are also available. The operator defines the number of queues and the user indicates in the print command which one is to be used.

- Spooled Input. Input spooling is available from the card reader.

- Operator Control. The operator may control batch jobs and print jobs, and the execution of the users' jobs (i.e., the operator can kill a job or change its priority.)

- Accounting. Batch accounting includes user and account identification, connect time, CPU usage, I/O usage, and page listing count.

In addition to the batch capabilities described, command procedures are supported by the command language. A command procedure is a file that contains a sequence of system operating commands. It is implemented as an indirect command file and can be initiated from an active terminal or from within a batch stream. Command procedures are most often used as an easy method of invoking a series of commands required frequently.

Batch command files and command procedures are completely compatible. Command procedures are executed immediately, within the context of the initiating process and from an active terminal (and, therefore, may request and receive user responses during processing). Batch jobs are queued and executed as detached processes under control of the system.

1.5.10 Languages and Utilities

VAX/VMS supports the following language compilers: FORTRAN IV-PLUS/VAX, MACRO/VAX, BASIC-PLUS-2, PDP-11 COBOL, MACRO-11, and FORTRAN 14.

FORTRAN IV-PLUS/VAX and MACRO/VAX generate code that uses the hardware and software capability of the VAX native mode.

BASIC-PLUS-2, PDP-11 COBOL, MACRO-11, and FORTRAN 14 produce code only for compatibility mode.

CHAPTER 2
OPERATION

2.1 GENERAL

This chapter contains a description of the VAX-11/780 CPU cabinet controls and indicators and the bootstrap operations/procedures. The controls and indicators for the console terminal (LA36) and the peripheral equipment are described in Chapter 5.

2.2 CPU CABINET CONTROLS AND INDICATORS

Figure 2-1 shows the CPU cabinet controls and indicators. With the cabinet doors closed, only the console control panel controls and indicators are visible. Access to the other CPU cabinet controls and indicators is unnecessary except for specific test and maintenance operations.

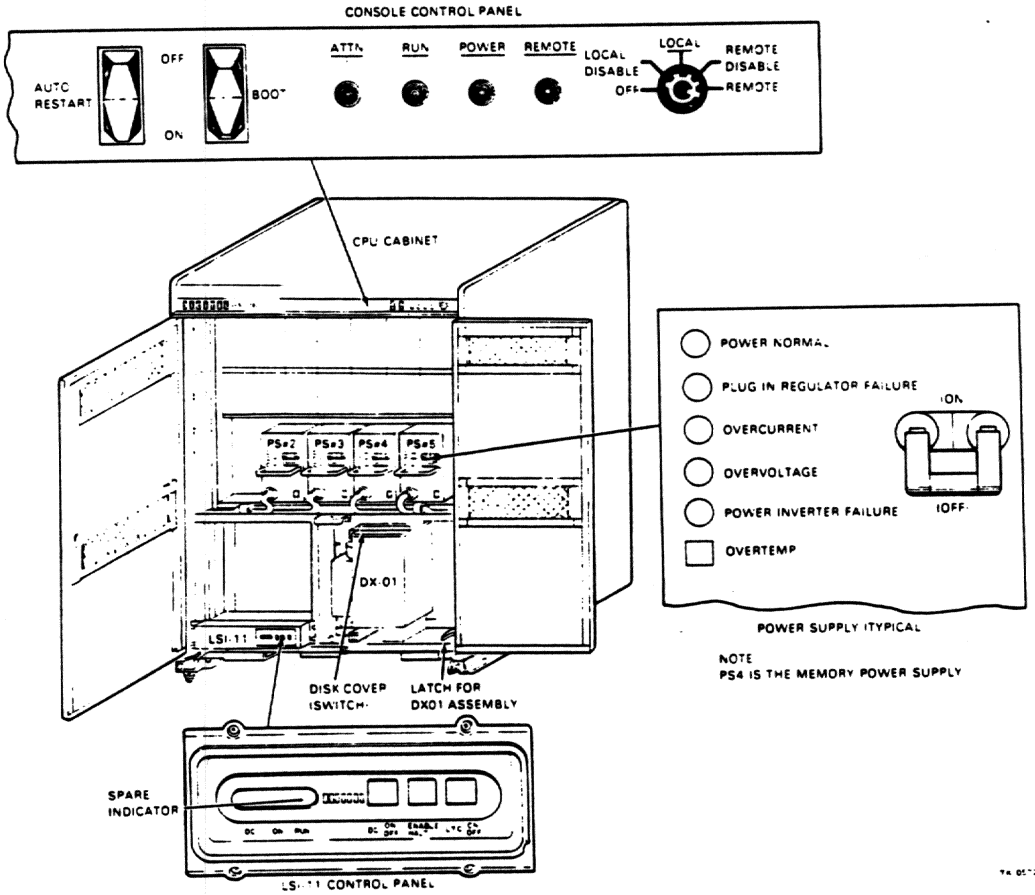


Figure 2-1 VAX-11/780 Controls and Indicators

2.2.1 Console Control Panel

The console control panel has a limited number of controls and indicators. Operations normally associated with a console control panel (e.g., examine, halt, etc.) are performed through the console software package using the console command language. The console control panel is connected to the console via the console interface board. The console control panel controls and indicators are described in Table 2-1.

2.2.2 Power Supplies

There are four or five identical power supplies in the CPU cabinet. The power supply marked PS4 (Figure 2-1) is the memory power supply. The other power supplies provide power to the remaining logic cards (located above the power supply section). All power supplies are normally left in the ON (circut breaker up) condition. The power supply controls and indicators are described in Table 2-2.

2.2.3 LSI-11

The LSI-11 is set to a run condition and is not touched unless maintenance operations are required. The LSI-11 controls and indicators are described in Table 2-3.

Table 2-1 Console Control Panel Controls and Indicators

Control/Indicator	Function
AUTO RESTART rocker toggle switch	When in the ON (down) position, the VAX-11/780 CPU is restarted automatically following a power failure or error halt.
BOOT rocker pushbutton switch	When pressed, the operating system is bootstrapped. When the bootstrap operation is completed, the console is set to the program I/O mode of operation.
ATTN indicator (red)	When lit, indicates that the CPU is halted.
RUN indicator (green)	When lit, indicates that the CPU is strobing interrupts.
POWER indicator (green)	When lit, indicates that the +5 V power supply is on.
REMOTE indicator (red)	When lit, indicates that remote access is enabled. When extinguished, indicates that remote access is disabled.
Five position key rotary selection switch	This is an off/on select switch.
OFF position	In this position, the VAX-11/780 is turned off.
LOCAL DISABLE position	In this position, remote access is disabled and console I/O mode of operation is inhibited.
LOCAL position	In this position, remote access is disabled, but console I/O mode of operation is not inhibited.
REMOTE DISABLE position	In this position, remote access is enabled and console I/O mode of operation is not inhibited.
REMOTE position	In this position, remote access is enabled and the console I/O is not inhibited.

Table 2-2 Power Supply Controls and Indicators

Control/Indicator	Function
POWER NORMAL (green)	When lit, indicates that the power supply is operating normally.
PLUG IN REGULATOR FAILURE (red)	When lit, indicates that there is a malfunction in the plug-in regulator in that power supply. The POWER NORMAL indicator should extinguish.
OVERCURRENT (red)	When lit, indicates that an overcurrent condition exists. The POWER NORMAL indicator should be extinguished.
OVERVOLTAGE (red)	When lit, indicates that an over voltage condition exists. The POWER NORMAL indicator should be extinguished.
POWER INVERTER FAILURE (red)	When lit, indicates that a failure has occurred in the inverter. The POWER NORMAL indicator should be extinguished.
OVERTEMP circuit breaker	When tripped (extended), indicates that an overtemperature condition exists. The POWER NORMAL indicator should be extinguished.
on/off circuit breaker	In the on position the power supply is operational if the POWER NORMAL indicator is lit.

Table 2-3 LSI-11 Controls and Indicators

Control/Indicator	Function
DC ON LED indicator	Illumintes when the DC ON/OFF toggle switch is set to ON and proper dc output voltages are being produced by the LSI-11. If either the +5 V or +12 V output from the LSI-11 is faulty, the DC ON indicator does not illuminate.
RUN LED indicator	Illuminates when the LSI-11 processor is in the run state (refer to ENABLE/HALT).
SPARE LED indicator	Not used.
DC ON/OFF Two-position toggle switch	When set to ON, enables the dc outputs. The DC ON indicator illuminates if the dc output voltages are of proper values. When set to OFF, the dc outputs are disabled and the DC ON indicator is extinguished.
ENABLE/HALT Two-position toggle switch	When set to ENABLE, the B HALT L line is not asserted and the processor is in the run mode (RUN indicator illuminated). When set to HALT, the B HALT L line is asserted allowing the processor to execute console ODT microcode (RUN indicator extinguished).
LTC ON/OFF Two-position toggle switch	When set to ON, enables the generation of the Line Time Clock (LTC) BEVNT L signal. When set to OFF, disables the line time clock.
AC ON/OFF (rear panel) Two-position toggle switch	When set to ON, applies ac power to the LSI-11. When set to OFF, removes ac power from the LSI-11.
FUSE (rear panel) 5 A fast-slow	Protects LSI-11 from excessive current.

2.2.4 RX01

The RX01 contains one switch. This switch is connected to the disk access door. If the door is not closed the disk cannot be run. To insert and remove disks from the DX01, the DX01 must be swung out on its hinges. The DX01 is held in place by a pressure latch on the mounting plate.

2.3 BOOTSTRAPPING

The bootstrapping of the VAX-11/780 is an automatic operation performed when one of the listed conditions exists within the VAX-11/780. Conditions for bootstrapping are as follows.

1. The console/CPU power is applied with the control panel AUTO/RESTART switch in the ON (down) position.

2. The VAX-11/780 has been on and the control panel BOOT switch is pressed.

3. The VAX-11/780 is operational and the CPU enters either a time out/odd address or an unused vector trap.

4. The VAX-11/780 is operational and either the CPU or the clock stops.

Some bootstrap operations require little operator intervention while others require no operator intervention. Conditions 1 and 2 are the only bootstrapping sequences that require operator action.

The description of the series of events during each of these sequences is contained in Appendix B.

2.3.1 Bootstrap Terms

A number of terms for data files/programs are used in describing the bootstrap sequence. These terms are described in the following.

1. Console Boot – The console boot is a program resident in the LSI-11 ROM. This program is initiated automatically every time the VAX-11/780 is turned on and every time the console program crashes (fault). This program loads the console program from the RX01 floppy disk (disk ZZESZAB-7.0) to the LSI-11 RAM.

2. Console Program – The console program is loaded into the LSI-11 RAM by the console boot. This program contains the console command language interpreter and the VAX-11/780 bootstrap functions.

3. WCS/ECO File – The Writable Control Store (WCS) Engineering Change Order (ECO) file is contained on the RX01 floppy disk. This file must be loaded into the CPU WCS before any instruction set processor (ISP) level software can execute.

4. VAX-11/780 Primary Boot Command File – The VAX-11/780 primary boot command file is contained on the RX01 floppy disk. This file contains the console commands for loading and starting the primary loader program. These console commands are executed by the indirect command file processor in the console program. There can be more than one primary boot indirect command file. The primary boot file allows designation to the unit or drive number.

5. VAX-11/780 Primary Loader – The VAX-11/780 primary loader is VAX/VMS operating system level program that loads a secondary loader program from the system load device (e.g., RK06). The parameters for the system load device are contained in the primary boot indirect command file. The primary loader file is contained on the RX01 disk and is loaded into main memory and started by the primary boot command file.

6. Restart Referee – The restart referee is a ROM operating system level program that is initiated by the VAX-11/780 console program during an automatic restart sequence. The restart referee determines whether a restart shall be a cold or a warm restart.

7. Memory Sizer – The memory sizer is a ROM operating system level program that is initiated by the primary boot file during a cold restart or a boot sequence. The memory sizer locates a contiguous block of 64K in memory and sends the starting address to the VAX-11/780 primary boot. The primary boot then loads the primary loader in this location.

8. VAX-11/780 Secondary Loader – The VAX-11/780 secondary loader is an operating system level program that loads the VAX/VMS or diagnostic supervisor from the system load device (e.g., RK06). The secondary loader is a file stored on the system disk (Files 11 boot block) or tape (ANSI boot record). The secondary loader is loaded and started by the primary loader.

2.3.2 Console (LSI-11), CPU/Memory Bootstrap Functional Areas

Within the VAX-11/780 there are two bootstrap functional areas or groups, the console subsystem functional area and the CPU/memory subsystems functional area.

2.3.2.1 Console (LSI-11) Bootstrap – The LSI-11 bootstrap consists of the following.

1. With the power-on sequence, the console ROM bootstrap program is started (operator action).

2. A series of LSI-11 tests (from the console ROM) is executed.

3. The console program is loaded from the RX01 disk to the LSI-11 memory.

4. The console program is initiated.

5. The console loads the WCS ECO file from the DX01 disk to the WCS portion of the CPU subsystem.

6. If the AUTO RESTART switch is ON, the CPU bootstrap is initiated.

7. If the AUTO RESTART switch is OFF, the console is held in the console I/O mode of operation (awaiting operator input).

2.3.2.2 CPU/Memory Bootstrap – The CPU/memory bootstrap consists of the following.

1. In power-on sequence, the CPU initialization microcode is executed.

2. The CPU waits for the start of a console boot sequence. The console boot can be initiated by one of the following.

   a. Console BOOT command entered (by the operator)

   b. Console BOOT switch pressed (by the operator)

   c. An auto-restart sequence initiated (AUTO RESTART switch is ON and a warm restart attempt fails).

3. When any one of the preceding conditions occurs, the console loads a bootstrap into CPU memory from the RX01 (program B).

4. The console starts the CPU in the bootstrap just loaded.

5. The console enters the program I/O mode of operation.

2.3.3 Performing a Bootstrap

To perform a VAX-11/780 bootstrap operation from a cold start, perform the following steps.

1. Insert the bootstrap floppy disk (ZZESZAB) into the RX01 disk drive.

2. Place the AUTO RESTART toggle switch in the ON (down) position.

3. Place the five-position rotary selector switch on the control panel to the LOCAL position.

4. The following is printed out at the console terminal.

```
CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000
INIT SEQ DONE
HALTED AT 00000000

(RELOADING WCS)
LOAD DONE. 00003200 BYTES LOADED
VER: PCS=01 WCS=03-10 FPLA=03 CON=PX03-08
```

5. The console enters the program I/O mode of operation and the bootstrap operation is completed.

CHAPTER 3
CONSOLE OPERATOR/PROGRAM COMMUNICATION

3.1 GENERAL

This chapter contains an introduction to the console subsystem and a detailed description of the console command language. The console subsystem is the operator interface for initializing, for establishing system controls, and for troubleshooting the VAX-11/780 System.

The console subsystem includes the following hardware:

1. An LA36 terminal
2. An RX01 floppy disk drive
3. An LSI-11 microcomputer
4. A console interface
5. A VAX-11/780 control panel
6. An optional remote terminal interface.

The primary interface between the operator and the VAX-11/780 is the LA36 terminal (operator terminal or terminal).

Integral to the console subsystem is a software interface package. This software interface package provides the means of communications between operator (at the operator terminal) and the console subsystem, and between the console subsystem and the VAX-11/780 CPU.

3.2 OPERATOR/PROGRAM COMMUNICATION

Communication between the operator and the console subsystem is through one of the software-based languages (e.g., console command language or diagnostic supervisor language). The console command language is described in this chapter. The diagnostic supervisor language is described in Chapter 4. Both of these languages are English-language based.

3.3 CONSOLE SUBSYSTEM OPERATIONAL MODES

The console subsystem operates in either one of two operational modes, the console I/O mode or the program I/O mode.

In the console I/O mode, operator inputs (e.g., console commands) to the terminal are interpreted by the LSI-11 based software interface package. The identified command is initiated (e.g., boot, halt, etc.). The software interface is loaded into the LSI-11 memory from the floppy disk or main disk during the boot operation.

In the program I/O mode, the inputs of the operator to the terminal are transferred to the CPU (on a one-to-one or character-by-character basis). The data is handled by the VAX-11/780 level software (operating system).

3.3.1 Equivalent Console Panel Functions

Some of the functions performed under the console I/O mode of operation are equivalent to (are performed in lieu of) functions performed by a standard console panel. These include operating system program control, CPU clock control, and the entry and display of memory data.

3.3.1.1 Program Control – The console is used to initialize the CPU (the console command language initialize command). The console can be used to initiate and to terminate operating system instruction execution. The console is used for bootstrapping the VAX-11/780. Bootstrapping is accomplished by loading VAX-11/780 main memory with a bootstrap program from the console floppy disk and then initiating the program.

3.3.1.2 Memory Data Display/Modification – The console can be used for the display and modification of data (address location) contained in main memory, I/O registers, general registers, and internal registers. The address locations can be accessed (read from and written to) in the following quantities.

1. Main memory: byte, word, longword, and quadword.
2. General registers (R0 through R13, SP, and PC) and processor registers: byte, word, longword.
3. I/O registers: byte, word, longword (dependent upon register data length).

3.3.1.3 Clock Control – The CPU clock rate can be controlled via the console. Console clock rate control is used for hardware and software troubleshooting. There are three console clock rate control modes:

1. Single instruction step
2. Single SBI bus cycle step
3. Single SBI time state step.

In the single instruction step mode, the operating system executes one instruction at a time. The CPU enters the halt state after the instruction execution.

In the SBI bus single cycle step mode, the CPU clock is stopped when SBI time state 1 (T1) is asserted (set). T1 remains set until another cycle is initiated via the console.

In the single SBI time state step mode, the CPU holds in one of the time states (T1, T2, T3, or T4) until a control signal is received from the console. The CPU then advances to and holds in the next time state.

3.3.2 Console Control Functions

Console control functions are provided for the following:

1. Controlling numeric radices of console input
2. Controlling console addressing modes
3. Controlling console data length
4. Displaying console and CPU status
5. Repeating commands
6. Linking commands.

3.3.2.1 Default Settings - The console is used to specify defaults for the addressing modes, define the radix of numeric inputs and outputs, and define the data length of addressable memory. The default options are set via the SET DEFAULT command. Default settings can be overriden within the context of a command by the use of a qualifier or changed via the SET DEFAULT command.

1. The default addressing modes are virtual, physical, or register addressing (general and internal).

2. The default radices for console numeric inputs and outputs are octal and hexadecimal.

3. The defaults for addressable memory data lengths are byte, word, longword, and quadword.

4. The defaults for power up are as follows:

   Radix = hexadecimal
   Data length = longword
   Addressing = Physical.

3.3.2.2 Status - The status of the console and CPU subsystems can be displayed on the terminal. The CPU status includes the stop/run state of the ISP level software and current clock step mode. The console status includes the current default settings, the console status, and the number of terminal fill characters. The status is displayed for each boot cycle, for each return to the console program, and for each console command language show command.

3.3.2.3 Command Linking and Repeating - A group of console commands can be linked into a single executable list. The linked commands are entered into a queue. The commands in the queue can be executed once or repeatedly (under the control of the operator). Single commands or linked commands can be repeated. Once initiated, the repeating continues until terminated by the operator. These features are aids to troubleshooting.

3.4 CONSOLE SOFTWARE

The console software package is made up of a series of modules. The console software package is contained in the 4K word ROM (microroutine) and 4K word RAM of the LSI-11. The console software provides for console control, operator interface, microdiagnostic execution, and console support function.

The console is in the null or wait loop the majority of the time. This null loop consists of a series of tests and conditional branches. When any of the branch conditions are met (flag for the branch is set), the program branches to the signified routine.

3.5 CONSOLE COMMAND LANGUAGE SYMBOLS AND DEFINITIONS

A series of terms/symbols are used to describe the console command language. A list of these terms/symbols and their definitions is contained in Table 3-1.

NOTE

These same terms and symbols are applicable to the diagnostic supervisor command language.

Table 3-1 Console Command Language Symbol Definition

Term/Symbol	Definition
/	Delimits a command from its qualifiers.
+	Represents the default address when used as an address argument in an examine or deposit command. (The default address is the sum of the last address used plus the current data length in bytes.)
,	Used as a separator within a list.
*	Can be used as an address argument in an examine or deposit command to represent the last address referenced.
:	Used as a separator within a command line.
!	Indicates the exclusive OR operation (i.e., selection of parameters within a command line). For example, <A> ! <B> means either <A> or <B> but not both is to be selected.
>>>	Console command language level input prompt character, requesting operator response.
DS>	Diagnostic supervision command language level input prompt, requesting operator response.
MIC >	Microdebugger input prompt, requesting operator response.
<<<	Link mode input prompt, requesting operator response.
$	VAX/VMS input prompt, requesting user response.
()	Indicates that one of the syntactic units of the expression are to be selected.
[ ]	Indicates the part of an expression that is optional e.g., NEXT [<blank> <count>] indicates that the count argument for the NEXT command is optional.
< >	Denotes a category name or label e.g., <address> indicates that a valid address is to be inserted as part of the command.
<blank>	Represents one or more blank spaces or tabs.
<count>	Represents a numeric count argument.
<XYZ - list>	Indicates one or more occurrences from the category indicated by XYZ.
<address>	Represents an address argument.
<data>	Represents a numeric argument.
<qualifier>	Represents a command modifier or default.
<input prompt>	Represents the console's input prompt string.
<CR>	Represents a console terminal carriage return. Used to terminate each command line.
<LF>	Represents a console terminal line feed.

3.5.1 Notation Examples

The following are examples of notation within a command line.

EXAMINE [</qualifier - list>] [<blank> <address>]

For each examine command there are two options.

1. A list of one or more qualifiers.

2. An address argument. If an address is specified, it must be preceded by one or more spaces or tabs.

The following are valid examine commands:

EXAMINE	No qualifiers are entered. The examine qualifiers are then entered under the set default command.
EXAMINE/BYTE/VIRTUAL	Word and address qualifiers entered but the address is controlled by default entered under the set default command.
EXAMINE <space> 1234 or EXAMINE <tab> 1234	No qualifiers selected (therefore, the qualifiers are controlled by the default entry entered under the set default command), a memory address argument is entered.
EXAMINE/WORD <space> 1234 or EXAMINE/WORD <space> <tab> 1234	Word qualifier entered (address by default) a memory argument is entered.

3.5.2 Command Abbreviations

Console command words may be abbreviated. The minimum abbreviation for each command word is specified in parentheses in each console command description.

Example

>>>EXAMINE/VIRTUAL/BYTE 1234

may be abbreviated to:

>>>E/V/B 1234

If an incorrect command entry is given, the console software responds by repeating the command and stating that it is incorrect.

Example

>>>HLP

>>> ? 'HLP' is incorrect.

NOTE

Use the @ ABBREV. HLP console command (console I/O mode) to get a listing of the abbreviations applicable to your system. A sample printout of the abbreviation file is contained in Appendix D.

3.6 CONSOLE COMMANDS

The console commands are English-language entries or their accepted abbreviations entered by the operator at the console terminal. These commands can only be entered when the console is in the console I/O mode of operation. When a valid command is entered, the indicated sequence of events is initiated. When that set of events is completed, another console command can be entered.

The console commands are described in the following paragraphs. These descriptions include, as applicable, the following items.

1. Syntax – the structure of the command.
2. Command description – a brief description of the command operations, general restrictions, and available options.
3. Response – a description of the console program response to the command.

The terms and symbols used in the descriptions are defined in Table 3-1.

NOTE

Every console command (command line) must be terminated with a carriage return (<CR>).

3.6.1 Boot Command (B)

Syntax – BOOT [<device name>] <CR>

Where the optional <device name> has the format: DDn, where DD is two-letter device mnemonic (e.g., DB for RPOX), and n is a one-digit unit number.

Description – The boot command initiates a VAX-11/780 system bootstrap sequence. The command can support bootstrap operations from or for a set of VAX-11/780 system devices.

When an optional <device name> is not given with the boot command, the console performs the boot sequence for the device preset under the default listing by executing the boot program (indirect command file) named DEFBOO. CMD.

When an optional <device name> is given with the command, the console executes boot program DDNBOO. CMD (indirect command file) where DDN is the device name given in the command.

Bootstraps (from devices other than the system default device) are performed by indirect command files containing the console commands necessary to boot from the device named.

Response – The status of the CPU and CPU controls is printed on the console terminal at the completion of a boot operation.

```
173000G
  CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM RAD=HEX,
  ADD=PHYS,DAT=LONG,FILL=00,REL=00000000
  INIT SEQ DONE
  HALTED AT 00000000

  (RELOADING WCS)
  LOAD DONE, 00003200 BYTES LOADED
  VER: PCS=01 WCS=03-10 FPLA=03 CON=PX03-08
```

or

>>>BOOT RPO

With this command the indirect command file RPOBOO. CMD. is booted. The console enters the program I/O mode after executing the command file.

3.6.2 Continue Command (C)

Syntax - CONTINUE <CR>

Description - The continue command causes the CPU to begin operating system execution at the address currently contained in the CPU program counter (PC). CPU initialization is not performed. The console subsystem enters the program I/O mode after the continue command is executed.

Response

>>>CONTINUE

$

3.6.3 Deposit Command (D)

Syntax - DEPOSIT [<qualifier - list>] <blank> <address> <blank> <data> <CR>

Where the qualifiers are /BYTE, /WORD, /LONG, /QUAD, /NEXT, /VIRTUAL, /PHYSICAL, /IDBUS, /VBUS, /INTERNAL, /GENERAL (refer to Table 3-4 for a description of these qualifiers).

The deposit command writes (deposits) <data> into the specified <address>. The address space used depends on the qualifiers specified with the command. If no qualifiers are used, the data and address defaults (entered via the SET DEFAULT command) are used to determine the address space.

Response

>>>DEPOSIT 10AD 310014DA
>>>

The <address> argument may be either a number identifying a location in one of the address options or one of the symbolic addresses. The symbolic address symbols and a description of them are contained in Table 3-2.

Example

Using symbolic addresses with the examine and deposit commands -

>>>E SP	Examines stack pointer
>>>D@ <data>	Deposits <data> to the location specified by the contents of the stack pointer.

Table 3-2 Symbolic Addresses - Used with the Deposit and Examine Commands

Symbol	Definition
PSL	Deposits to or examines the processor status longword.
PC	Deposits to or examines the program counter (general register F).
SP	Deposits to or examines the stack pointer (general register E).
+	Deposits to or examines the location immediately following the last location referenced. For physical and virtual references the location referenced is the last address plus n where n = (1 for byte, 2 for word, 4 for longword, 8 for quadword). For all other address spaces, n is equal to 1.
-	Deposits to or examines the location immediately preceding the last location referenced.
*	Deposits to or examines the location last referenced.
@	Deposits to or examines the address represented by the last data examined or deposited.

3.6.4 Examine Command (E)

Syntax - EXAMINE [<qualifier - list>] [<blank> <address>] <CR>

Where the qualifiers are /BYTE, /WORD, /LONG, /QUAD, /NEXT, /VIRTUAL, /PHYSICAL, /IDBUS, /VBUS, /INTERNAL, /GENERAL (refer to Table 3-4 for a description of these qualifiers).

Description - The examine command reads and displays the contents of the specified <address>. If no <address> is specified, the contents of the address defined by the address default (entered via the SET DEFAULT command) are displayed.

The <address> argument may be one of the symbolic addresses (Table 3-2).

Response - Sample command entries and console responses

```
>>>EXAMINE/PHYSICAL 1234 (console prompt, operator command entry)

  P 00001234 EF89 (console response to command)
>>>EXAMINE/VIRTUAL 1234 (console prompt, operator command entry)

  P 0005634 4567 (console response to command)
(The translated physical address is displayed for virtual examines.)

>>>EXAMINE/G 0 (console prompt, operator command entry)

  G 00000000 5432 (console response to command).
>>>EXAMINE/BYTE 1234

  P 00001234 78
>>>EXAMINE/WORD 1234

  P 00001234 5678
>>>EXAMINE/LONG

  P 00001236 FFFF1234
>>>
```

3.6.5 Initialize Command (I)

Syntax - INITIALIZE <CR>

Description - This command causes the CPU system to be initialized - set to a specified starting condition.

Response

```
>>>INITIALIZE

  INIT SEQ DONE

>>>
```

3.6.6 Halt Command (H)

Syntax - HALT <CR>

Description - The halt command stops the CPU after the CPU completes execution of the instruction that is in process when the halt command is entered.

Response

```
>>>HALT

  HALTED AT 00002436

>>>
```

The number given is the location at which the process is halted.

3.6.7 Help Command

Syntax - HELP <CR>

Description - The console opens and prints the contents of an indirect command file (help file). The help file contains a description of all console commands.

Response

```
>>>HELP

!  CONSOLE HELP FILE REV.3 13-SEP-1977
!  TO STOP PRINTING, TYPE ↑C
!  FOR ABBREVIATION RULE, TYPE ...
   .
   .
   .
   .

!  <END-OF-CONSOLE.HLP>

<@EOF>

<@EXIT>

>>>
```

Refer to Appendix C for a sample printout of the console help file.

3.6.8 Abbreviation Help Command

Syntax – @ ABBREV.HLP <CR>

Description – The console opens and prints the contents of an indirect command file (abbreviation help file). The abbreviation help file contains a list of abbreviations and rules for the console command language.

Response

```
>>> @ABBREV.HLP

!VAX-11/780 CONSOLE ABBREVIATION RULES REV-5 8-NOV-77
!  THIS FILE SHOWS THE SHORTEST UNIQUE COMMAND STRINGS THAT WILL BE
!  RECOGNIZED AS THE CONSOLE COMMAND LISTED.
!
!   COMMAND - SHORTEST ABBREVIATION RECOGNIZED
!  'EXAMINE <ADDRESS>' - 'E <ADDRESS>'

   .
   .
   .
   .

!  /IDBUS                    /ID
!  /WCS                      /WC
!  /NEXT:<NUMBER>            /N:< NUMBER >
!  /COMMAND                  /C
!  /START:<ADDRESS>          /S:< ADDRESS >
!  <END-OF-ABBREV.HLP>

>>>
```

Refer to Appendix D for a sample printout of the console abbreviation help file.

3.6.9 Error Help Command

Syntax – @ ERROR.HLP <CR>

Description – The console opens and prints the contents of an indirect command file (error help file). The error help file contains a list of all error messages for the console command language.

Response

```
>>>@ERROR.HLP

!VAX-11/780 ERROR MESSAGE HELP FILE REV-2 8-NOV-77
!
!  ?'<TEXT-STRING>' IS INCOMPLETE
!       THE <TEXT-STRING> IS NOT A COMPLETE CONSOLE COMMAND
!  ?'<TEXT-STRING>' IS INCORRECT
!       THE <TEXT-STRING> IS NOT RECOGNIZED AS PART OF A COMMAND
```

```
!  ?UNEXPECTED TRAP
!  MOUNT CONSOLE FLOPPY, THEN TYPE ↑C
!       CONSOLE TRAPPED TO AN UNUSED VECTOR. CONSOLE REBOOTS WHEN ↑C
!  TYPED
!  ?Q-BLKD
!       CONSOLE'S TERMINAL OUTPUT QUEUE IS BLOCKED. CONSOLE WILL REBOOT.
!  <END-OF-ERROR.HLP>

>>>
```

Refer to Appendix E for a sample printout of the console error help file.

3.6.10 Load Command

Syntax - LOAD [<qualifier list>] <blank> <file specification> <CR>

where the <qualifier list> is the storage location for the file being loaded: and where the <file specification> is the identity of the file to be loaded.

Description - The load command is used to read or transfer file data from the console floppy disk to main memory, or to Writable Control Store (WCS). If no qualifier is entered, physical main memory, location 0, is the default location (entered via the SET DEFAULT command) to begin loading. The load defaults are described in Table 3-3.

Table 3-3 Load Command Qualifiers

Qualifier	Definition
/START:<address>	This qualifier specifies a starting address for the file loaded. The console starts loading at address 0 if a qualifier is not given.
/WCS	The WCS is loaded with the specified file.
/PHYSICAL	The physical main memory is loaded with the specified file.

Response

```
>>>LOAD DS.EXE/START:FE00
        LOAD DONE. 000(000BYTE)LOADED

>>>
```

The diagnostic supervisor file is loaded into main memory beginning at location FE00.

3.6.11 Quad Clear Command (Q Clear)

Syntax - QCLEAR <blank> <physical address> <CR>

Description - The quad clear command clears the quadword at the specified <address>. The command is used to clear an uncorrectable ECC error.

Response

```
>>>QCLEAR 4444

>>>
```

The <address> is always interpreted as a physical main memory location. The <address> given is forced to a quadword boundary by unconditionally clearing the three low-order address bits.

3.6.12 Repeat Command (R)

Syntax - REPEAT <console command> <CR>

Description - With a repeat command the specified console command is repeatedly executed until terminated by a CTRL C (↑C). Any console command, except the repeat command, may be specified.

Response

```
>>>REPEAT EXAMINE 100

    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000
    P       00000100        00000000

>>>
```

The repeat response is dependent on the command specified to be repeated.

3.6.13 Set Default Command (SE)

Syntax - SET <blank> DEFAULT [<blank> <default option>] <CR>

Description - The set default command is used to set the defaults for address, data length, and radix. The <default options> and their definitions are contained in Table 3-4. The console applies the defaults from those set under this command when a console command does not contain a qualifier.

Table 3-4 Set Default Command Options

Option Group	Item	Specification
Address defaults	Virtual	Sets the default to the specified address.
	Physical	
	General	
	Internal	
	ID bus	
	V bus	
Data defaults	Byte	Sets the default to the specified data length.
	Word	
	Long	
	Quad	
Radix defaults	Hex	Sets the default for the termi-nal I/O to the specified radix.
	Octal	

Response
>>>SET DEFAULT PHYSICAL, BYTE, HEX

>>>

3.6.14 Set Step Command

Syntax - SET <blank> STEP [<blank> <step option>] <CR>

Description - The set step command sets the CPU clock mode. The CPU clock modes (<step option>) are defined and listed in Table 3-5.

Table 3-5 Set Step Command Options

Option Group	Item	Specification
Step	Instruction	Sets the CPU clock to the single instruction step mode.
	Bus	Sets CPU clock to the single Synchronous Backplane Interconn ect (SBI) cycle step mode.
	State	Sets CPU clock to the single SBI time state step mode

Response
>>>SET STEP BUS

>>>

3.6.15 Set Terminal Fill Command

Syntax - SET <blank> TERMINAL FILL <count> <CR>

Description - The set terminal fill command sets the number (<count>) of blanks that are transmitted to the console terminal after a carriage return (<CR>) or a line feed (<LF>).

Response
>>>SET TERMINAL FILL:4

>>>

3.6.16 Set Terminal Program Command

Syntax - SET TERMINAL PROGRAM

Description - The set terminal program command places the console terminal in the console I/O mode of operation.

Response
>>>SET TERMINAL PROGRAM

>>>

3.6.17 Set Clock Command

Syntax - SET <blank> CLOCK [<blank> (SLOW! FAST! NORMAL)] <CR>

Description - The set clock command sets the CPU clock to the frequency specified by the command argument (FAST, SLOW, NORMAL). The frequencies are as follows:

Fast = 10.525 MHz

Slow = 8.925 MHZ

Normal (or no argument) = 10.0 MHz

Response
>>>SET CLOCK SLOW

>>>

3.6.18 Set SOMM Command

Syntax - SET <blank> SOMM <CR>

Description - The set SOMM command sets the Stop On Microbreak Match (SOMM) enable (on the console interface board). The CPU clock is stopped if SOMM is enabled and the contents of the microbreak match register (ID register 21), match the contents of the CPU micro-PC.

Response
>>>SET SOMM

>>>

3.6.19 Set Relocation Command

Syntax - SET RELOCATION: <data> <CR>

Description - The set relocation command deposits data into the console's relocation register. The value <data> set in the relocation register is added to the effective address of all virtual and physical memory examines and deposits.

Response
>>> SET RELOCATION: 46342422

>>> EXAMINE 200 -*

* (with the relocation set to 1000 an examine 200 command examines the contents of physical memory location 1200)

P 00001200 49

>>>

3.6.20 Show Command (SH)

Syntax - SHOW <CR>

Description - With the show command the console terminal displays the following:

a. the default settings for data length, address type, and radix of address and data inputs and outputs,

b. the terminal fill character count,

c. the CPU status including the run halt state and current clock mode setting.

Response

```
>>>SHOW

        CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
        RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000

>>>
```

3.6.21 Start Command(s)

There are two formats used with the start command.

Syntax - START <blank> <address> <CR>
START/WCS <blank> <address> <CR>

Description - The start command of the first format performs the equivalent of the following sequence of console commands:

1. Initialize the CPU,
2. Deposit the <address> data in the PC,
3. Issue a continue instruction to the CPU.

The start command of the second format performs the equivalent of the following sequence of commands:

1. Deposit the <address> data in the micro-PC,
2. Start the CPU clock in the normal mode of operation.

Response

```
>>>START 100
```

or

```
>>>START/WCS FF
```

3.6.22 Next Command

Syntax - NEXT [<blank> <count>] <CR>

Description – With the next command the CPU clock is stepped the number of times indicated by <count>. The type of step performed by the clock is determined by the previous set step command. If a next command is issued while the CPU is in normal clock mode, it defaults to single instruction step mode for the duration <count> of the command.

The console enters program I/O mode immediately prior to performing the step command. The console I/O mode is re-entered as soon as the step command is complete.

Response – Step-dependent responses are displayed on the console terminal after the completion of each of the count steps as specified in the following.

a. Single Instruction Mode (set step instruction).

```
>>>SET STEP INSTRUCTION

>>>ST 100

     HALTED AT 00000102

>>>NEXT

     HALTED AT 00000104
     HALTED AT 00000106
```

b. Single Bus Cycle Mode (set step bus)

```
>>>SET STEP BUS

>>>
     CPU CLK STOP
     CPTO UPC=OOFF

>>>NEXT

     CPTO UPC-OOFF
     CPTO UPC-OOFF
     CPTO UPC-OOFF
     CPTO UPC-OOFF
     CPTO UPC-OOFF

>>>NEXT 2

     CPTO UPC-OOFF
     CPTO UPC-OOFF
```

c. Single Time State Mode (set step state)

```
>>>SET STEP STATE

>>>NEXT 6
```

```
CPT1
CPT2
CPT3 APC=0000
CPT0 UPC=00FF
CPT1
CPT2

>>>SET STEP NORMAL

>>>
```

If a <count> is not specified in the command, one step is performed, and then the console enters the space bar step mode. For each depression of the space bar there is one execution of the step option currently enabled (i.e., step bus, step time, step instruction).

If the next command has an argument the space bar step mode is not enabled. For example, for a NEXT 2 input, two steps are executed and the console then prompts for another command.

To exit from the space bar step mode, enter any character except a space.

3.6.23 Test Command (T)

Syntax - TEST [/com] <CR>

Description - The test command enables the microdiagnostic monitor program. Microdiagnostic execution begins immediately if no /com qualifier is entered. If microdiagnostic testing is completed successfully (i.e., no errors detected), the console I/O mode is re-entered. A microdiagnostic floppy must be loaded in the RX01.

With a [/COM] qualifier the microdiagnostic monitor enters its command mode (MIC>) and waits for an operator command.

Response

```
>>>TEST

ZZ-ESKAB U8.2
01,02,03,04,
NO. OF WCS MODULES = 0002
05,06,07,08,09,0A,0B,0C,0D,0E,0F,10,11,12,13,14,15,16,17,
18,19,1A,1B,1C,1D,1E,1F,20,21,22,23,24,25,26,27,28,29,2A,2B,2C,2D,
2E,2F,30,31,32,33,34,35,36,37,38,39,3A,3B,3C,3D,
END PASS 0001
MOUNT FLOPPY ZZ-ESZAD 7 TYPE "DI"
MIC>DI
3E,
# MEM CTRLS= 00000001
3F,40,
4K CHIP 00000608
41,42,
CPU TR= 00000010
43,44,45,46,47,48,49,4A,4B,4C,4D,
CTRL 1 MAX ADR+1= 00040000
4E,
CTRL 1 MAX ADR+1= 00040000
```

```
4F.50.
NO FPA

END PASS 0001

>>>
```

At the end of running the first microdiagnostic test program (ZZ-ESKAB) the operator is instructed to mount the floppy disk containing the second diagnostic test. A microdiagnostic prompt (MIC>) appears. The second hits a trap and responds with a microdiagnostic prompt. The operator returns (RET) to the console command program and the console automatically performs a boot operation.

3.6.24 Unjam Command (U)

Syntax - UNJAM <CR>

Description - This command initiates a clearing (unjam) of the SBI.

Response

```
>>>UNJAM

>>>
```

3.6.25 Wait Command (WA)

Syntax - WAIT <blank> DONE <CR>

Description - The wait command is executed from an indirect command file. When executed, further execution of the command file is suspended until one of the following occurs:

a. A DONE signal is received from a program running in the CPU. On receipt of DONE, the console resumes execution of the command file.

b. The console prints <@ EXIT> and aborts execution of the remainder of the console command file if the CPU halts and a DONE signal is not received.

c. The operator enters a CTRL C. The console aborts execution of the remainder of the command file.

Response

```
WAIT DONE
↑C

>>>
```

3.6.26 Indirect (@) Command

Syntax - @ <filename> <CR>

Description - This command causes the console to open the file specified by <filename> and begin executing console commands from the file. Execution continues until one of the following events occurs:

a. A WAIT DONE command is read from the file (refer to wait command).

b. The end of the indirect file is reached. The console prints <@EOF> and an <input prompt>.

c. The operator enters a CTRL C (↑C). Execution of the indirect file is aborted.

Response

```
>>> @ ABBREV.HELP

VAX-11/780 CONSOLE

.
.
.
<@EOF>

>>>
```

An indirect command file may also be entered using the following sequence. Note the lack of a prompt signal once this routine has been entered.

```
>>>@ESCAA

HALT     ! HALT THE CPU

         CPU HALTED
INIT     ! INITIALIZE

         INIT SEQ DONE
UNJAM    ! AND CLEAR THE SBI

LO ESCAA.EXE/ST:200

         LOAD DONE, 0000B400 BYTES LOADED
!        RH780 MASSBUS DIAGNOSTIC

LO ASSAA.EXE/ST:FE00

         LOAD DONE, 0000D000 BYTED LOADED
START 10000

<@EOF>
<@EXIT>

DIAGNOSTIC SUPERVISOR, ZZ-ESSAA-4.02-417 21-JUL-1978 01:33:44.92

DS>
>>>H
         HALTED AT 00016B04
>>>
```

3.7 CONSOLE COMMANDS PERFORMED WITH CPU RUNNING

Most console commands have an interaction with the CPU's running and, therefore, require that the CPU be halted before command execution. However, some console commands do not require interaction with the CPU and, therefore, may be executed with the CPU running. These commands include:

SHOW
HELP
The SET commands
EXAMINE/V BUS
HALT
CLEAR
WAIT

If any other console command is entered while the CPU is running, the command is rejected and the following error message is displayed on the console terminal:

```
? CPU NOT IN CONSOLE WAIT LOOP, COMMAND ABORTED

>>>
```

3.8 COMMENTS WITHIN COMMANDS

Comments, proceded by an exclamation mark (!), may be used in any command line. When the console detects an exclamation mark, all remaining text in that command line is ignored. A comment may begin in any character position within a command line, including the first. (The ! is considered a part of the comment.)

Example

```
>>> ! THIS IS A VALID COMMENT

>>>
```

This is a comment following directly after the console prompt

```
>>> EXAMINE 1234 ! THIS IS ALSO A COMMENT

>>>
```

This is a comment made after a command - prior to the carriage return.

3.9 CONTROL AND SPECIAL CHARACTERS

There are a number of control characters and special characters that cause an action to occur within the VAX-11/780 when entered independently. These form an integral part of the console command language. A list of the control and special characters is contained in Table 3-6.

3.10 COMMAND-QUALIFIERS AND DEFAULTS

Qualifiers are entered as part of a command line and further define the basic console command (e.g., address type, data length). Defaults are supplied automatically by the console software. Defaults are supplied for those commands where the option for entering a qualifier is not exercised by the operator. The defaults can be changed by using the set default console command.

The following paragraph describes the qualifiers and defaults available for use as address and data arguments: qualifiers and defaults for address type, data length, and numeric argument defaults: as well as a qualifier to operate on multiple sequential addresses.

Table 3-6 Control/Special Character Descriptions

Character	Description
CTRL C (↑C)	When a CTRL C is typed any repetitive console operation is halted. This includes the following: a. Executions resulting from a repeat command b. Successive operations resulting from a /next qualifier c. Delays resulting from a wait command d. Successive steps resulting from a next command e. Aborts further execution of an indirect command file after current instruction is completed. The console responds with an <input prompt>.
CTRL O (↑O)	When a CTRL O is entered, the console terminal output is suppressed or enabled (on a toggle basis). Console terminal output is always enabled at the next console terminal <input prompt>.
CTRL U (↑U)	When a CTRL U is typed before a line terminator, all characters typed since the last line terminator are deleted from temporary storage. The console echoes: U <CR> <LF>
DELETE	Typing DELETE (the DELETE key) deletes the last character typed on an input line. Only characters typed since the last line terminator can be deleted. Several characters can be deleted in a sequence by typing successive deletes. The first delete echoes as a backslash (\) followed by the character which has been deleted. Subsequent deletes cause only the deleted character to be echoed. The next character typed that is not a delete causes another backslash (\) to be printed, followed by the new character.
CTRL P (↑P)	When a CTRL P is entered the console is placed in the console I/O mode. (unless the console mode switch is in the LOCAL/DISABLE or REMOTE/DISABLE position).
CARRIAGE RETURN (CR)	When a carriage return is entered a console command line is terminated.

3.10.1 Qualifiers for Address Type

Qualifiers for address type are used within a command line. These qualifiers specify the type of address argument as virtual, physical, or register address. The qualifiers for the respective types are:

```
/VIRTUAL
/PHYSICAL
/IDBUS
/VBUS
/GENERAL
/INTERNAL
```

Example

To examine virtual address 1234, type:

```
>>>EXAMINE/VIRTUAL 8000E3FC
```

The system responds by displaying the contents of that address

```
P 0002E3FC 0000ABCD

>>>
```

Some register addresses have mnemonic names that are unique and unambiguous; therefore, the /GENERAL qualifier need not be specified when mnemonic addresses such as PC or SP are referenced.

Example
To examine the PC, the operator enters either of the following statements:

```
>>>EXAMINE/GENERAL PC
```

The system responds by displaying the contents of the PC

```
    G 0000000F 00000001

>>>
```

or

```
>>> E PC
```

The system responds by displaying the contents of the PC

```
    G 0000000F 00000001

>>>
```

If a console command references a nonexistent virtual address, the execution of that command is aborted. An error message is then typed out on the console terminal.

3.10.2 Defaults for Address Types

The console applies a default for address type to any command that requires an address argument and does not contain an address qualifier. The default is set by using the set default command.

Example
The command:

```
>>> SET DEFAULT VIRTUAL

>>>
```

After executing this command the console defaults to virtual addressing for any console command that requires an address argument, but does not have an address qualifier.

With virtual default set, the command EXAMINE 1234 results in the contents of virtual address 1234 being typed on the terminal.

3.10.3 Qualifiers for Data Length

Qualifiers for data length are used within a command line to specify the length of the data quantity associated with the command. The qualifiers for data length are:

```
BYTE
WORD
LONG
QUAD
```

Example

The command:

```
>>>EXAMINE/BYTE 1233

    P 00001233 FF

>>>
```

The data (FF) in the byte at address at 1233 is typed at the terminal.

Since the general and processor registers are longword quantities, all register references default to longword data length, regardless of the current setting of the data length default.

3.10.4 Defaults for Data Length

The console applies a default for data length to any command that references data and does not contain a data length qualifier. The default can be set through the set default command.

Example

The command:

```
>>>SET DEFAULT WORD

>>>
```

results in the console defaulting to word data length.

The command:

```
>>>EXAMINE 1234
```

The system responds by displaying the contents of the word that has its first byte at address 1234.

```
    P 00001234 FFFF

>>>
```

All general and processor registers are longword quantities. Therefore, all register references default to longword data length, regardless of the current setting of the data length default.

3.10.5 Local Radix Override

The current default radix can be overridden by including a local radix override as a prefix to any numeric argument. A local radix override is one of the following:

%O (percent sign O) for octal arguments

```
>>>EXAMINE %07654

     P 00000FAC FFFF

>>>
```

%X (percent sign X) for hexadecimal arguments

```
>>>EXAMINE %X7654

     P 00000073124 177777

>>>
```

A local radix override must appear as the two leftmost characters of the numeric argument it modifies, and it must not be separated by spaces or tabs from that argument.

Defaults For Radix – The console allows setting the radix default for address and for data arguments.

Example
The command:

```
>>>SET DEFAULT OCTAL

>>>
```

The default radix for address and data arguments is set to OCTAL.

3.10.6 Default Address Facility

Each time an examine or deposit command is executed, the address of the memory location following the location referenced by the command is computed. The address of the next memory location is the <default address>. An examine command that does not specify an address references the next address by default. The <default address> is computed as follows:

<default address> = <address used by last examine or deposit command> + n

where n = * 1 for byte references
2 for word references
4 for longword references
8 for quadword references

The following example shows a sequence of console commands and the value taken by the computed default <address>. Note that the next address is data length dependent, since a byte reference advances the default address by 1, while a longword reference advances the default address by 4.

Example (all numbers are octal)

```
Command                          Default address after execution

>>>EXAMINE/BYTE 1234

     P 00001234 78               1235
>>>EXAMINE/WORD (uses default address)

     P 00001235 3456             1237
>>>EXAMINE/LONG (uses default address)

     P 00001237 FFFFFF12         1243
>>>EXAMINE R0

     G 00000000 00000010         R1
>>>EXAMINE

     G 00000001 00002000         R2
>>>EXAMINE R11

     G 0000000B 04000000         R12
>>>EXAMINE

     G 0000000C 00001C00         R13
>>>EXAMINE PC

     G 0000000F 00000001         R0
>>>EXAMINE

     G 00000000 00000010         R1

>>>
```

Note that the <default address> following a PC reference is R0.

3.10.6.1 Specifying Defaults for Address - The symbol (+) can be used as an address argument in a deposit or examine command to represent the <default address>. This symbol permits depositing to (or examining) successive locations without typing the address argument for commands after the first deposit.

Example
To load a program starting at address 123456, the following deposit commands can be used:

```
>>>DEPOSIT     123456          987654

>>>DEPOSIT     +               12345

>>>DEPOSIT     +               6543

>>>
```

Each deposit command, after the first, writes the <data> into the next successive memory location. This same technique can be used with the examine command.

```
>>>EXAMINE 12345

                P   00012345   00987654
>>>EX +

                P   00012349   00012345
>>>EX +

                P   0001234D   00006543
>>>
```

3.10.6.2 Last Address Notation – The last address argument referenced (virtual, physical, register) by an examine or deposit command is denoted by an asterisk (*). The <last address> may be used as an argument to an examine or deposit command by typing an asterisk in place of the address argument.

Example
The command:

```
>>>EXAMINE 1234

      P 00001234 12345678
```

The content of location 1234 is typed out.

If the next command issued is:

```
>>>DEPOSIT * FFFF
```

The number 7356 is loaded into location 1234.

```
>>>EXAMINE *

      P 00001234 0000FFFF

>>>
```

Examines and deposits of general and processor registers replace the <last address> with the register address. Mnemonic register names are translated into register addresses by the console.

3.10.7 NEXT Qualifier
Syntax - Command/NEXT [:<COUNT>]

The <count> argument specifies the number of additional executions of the command to be performed after the initial execution of the command. The default value for <count> is one.

Example
The command:

```
>>>EXAMINE/BYTE 12345/NEXT:2

      P  00012345   54
      P  00012346   76
      P  00012347   98
```

is evaluated by the console as follows:

1. Initially the console applies default values.

2. The command (with applied defaults) is executed. The contents of location 12345 are typed out, and the <default address> is updated to 12346.

3. The qualifier /NEXT is now evaluated. The console repeats the operation specified by the command the number of times indicated by the <count> argument. Each execution of the command uses the <default address> for its address argument and updates the <default address> afterwards. In this example, locations 12345 and 12346 are typed out and the final value of the <default address> is 12347.

Example
If the command:

```
>>>EXAMINE/NEXT:2

P  00012348  01234500
P  0001234C  00654300
P  00012350  FFFFFF00
```

is now entered (following the command in the previous example) the contents of locations 1234E, 1234C, and 1235O are typed out. The default is longword. Since the examine command does not contain an address argument, the initial execution of the command uses the current <default address> (which was 12347 following the command in the previous example).

When the qualifier /NEXT is used to examine or deposit multiple general registers, the next register after PC is R0.

Example
The command:

```
>>>EXAMINE/NEXT:5 R13

G  0000000D  F0F0F0F0
G  0000000E  8C000000
G  0000000F  00000001
G  00000000  00000010
G  00000001  00002000
G  00000002  00000800

>>>
```

results in the contents of R13, SP, PC, R0, R1, and R2, being typed out (in that order).

```
>>>EXAMINE/BYTE 12345/NEXT:2

P  00012345  54
P  00012346  76
P  00012347  98
```

```
>>>EXAMINE/WORD 12345/NEXT:2

      P  00012345  7654
      P  00012347  0098
      P  00012349  2345

>>>
```

The preceding commands are used to examine the contents of the same location.

3.11 COMMAND REPEAT FACILITY

The command repeat facility is primarily a maintenance aid. Commands are executed repeatedly so that CPU logic involved with the commands can be tested with an oscilloscope. The following paragraphs describe the repeat facility command.

3.11.1 Repeating Commands

Example

```
>>>REPEAT EXAMINE 1234

      P  00001234  0000FFFF
      P  00001234  0000FFFF
      P  00001234  0000FFFF
      P  00001234  0000FFFF
      P  00001234  0000FFFF
      P  00001234  0000FFFF
      P  00001234↑   C0000FFFF
```

results in the content of location 1234 being repeatedly examined and displayed.

Once initiated, repeated execution continues until terminated by the entry of a CTRL C (↑C) on the console keyboard.

3.12 COMMAND LINK FACILITY

The console's command link facility provides a means for multiple commands to be linked into a single executable list of commands. Once a list of linked commands is constructed, the list can be executed once, or executed repeatedly.

3.12.1 Link Command Operation

The link command puts the console in the link mode. The desired commands are then entered. The link command is entered only at the beginning of the command string (i.e., the first command line). Commands to be linked are entered one-per-line, with each command line terminated by a <CR>. The console then returns a link mode input prompt (<<<) requesting the next command. The link operation is terminated by entering a CTRL C (↑C) on the console.

As the command string is entered, the commands are stored on a dedicated sector on the floppy disk (RX01). When the command string is executed, the string is treated as an indirect command file (i.e., command retrieved, parsed, executed, and the next command retrieved, etc.).

Once the input of a list of linked commands has been terminated, no further commands can be added to the commands already linked. The linked commands are executed by entering a perform command. The console does not execute commands being linked until a perform command is issued.

The command string can be executed once or repeatedly. When the perform command is entered after the CTRL C (↑C) the linked commands are executed once. However, when perform is entered prior to and after the CTRL C (↑C), the first perform command becomes a part of the command list and the linked command list is repeated until a second CTRL C (↑C) is entered.

If a linked command is entered incorrectly, the console outputs an error message when the command containing the syntactic error is executed. A CTRL U (↑U) rejects the current command line.

3.12.2 Link Facility Usage

Syntax - LINK

COMMAND <CR>

COMMAND <CR>

<↑C>

Response - Dependent on command list.

Example
It is desired to repeatedly examine the contents of a device register after a CPU initialize. The sequence of commands is as follows.

>>> LINK	The link mode is entered.
>>> INITIALIZE	An initialize command is entered.
>>> EXAMINE/LONG FFFFABBC	Examine command is entered into string.
>>> PERFORM	Perform is entered prior to linking termination.
>>> C	The linking command is terminated.
>>> PERFORM	Initiates execution.
>>> P 00001234 12A00123	Command string executed continuously.
>>> P 00001234 12A00123	

3.13 CONSOLE I/O MODE ESCAPE

The console I/O mode escape sequence causes the console to move from console I/O mode to program I/O mode. To escape to program I/O enter

>>> SET TERMINAL PROGRAM

Console commands start, continue, and next also enable (escape to) program I/O mode.

The program I/O mode escape sequence causes the console to transition from program I/O mode to console I/O mode. To escape to console I/O mode enter

↑P

The CTRL P (↑P) is not recognized by the console if the console power switch on the console panel is in the REMOTE DISABLE or LOCAL DISABLE position.

3.14 REMOTE CONSOLE ACCESS COMMAND SET

A set of commands is included in the console command language for accessing the console from a remote terminal. The command set for remote access has the following:

a. Talk mode – this provides communication between local and remote terminal operators.

b. Copy control – this permits suppressing or enabling typeout on the local terminal while a remote operator is in control.

c. Transfer control – this permits transferring control of the console between the local and remote operators.

d. Echo control – this provides control of the echo of characters to the remote terminal while in talk mode.

e. Error message suppressor – a method of suppressing lost carrier error messages caused by a loss of carrier on the remote line.

A printout of the remote console command help file is contained in Appendix L.

3.14.1 Enable Talk Mode Command

Syntax – ENABLE <blank> TALK <CR>

Description – The enable talk command places the console into talk mode. While in talk mode, characters typed on the remote keyboard are typed on the local terminal, and vice versa. The console does not echo characters back to the originating keyboard, unless the talk mode echo feature is enabled. Console commands are not recognized while in talk mode.

Talk mode is terminated by typing the escape character, CTRL P (↑P), on either the local or remote keyboard. When talk mode is terminated, console I/O mode is enabled. When the talk mode is entered, the console enables the remote serial interface and sends the data terminal ready signal to the data set. All terminal I/O to the CPU is disabled while in talk mode.

Response

```
>>> ENABLE TALK
.
.
.

↑P

>>>
```

3.14.2 Enable/Disable Echo Command

Syntax – (ENABLE ! DISABLE) <blank> ECHO <CR>

Description – With the enable echo command the console echoes characters input from either the remote or local keyboards while in talk mode. The disable echo command suppresses the echo of characters typed from the local or remote keyboards.

Enable and disable echo commands do not have any effect until talk mode is entered. A disable echo is entered each time the console control panel rotary keyswitch is put in the LOCAL or LOCAL/DISABLE position, and upon power up of the console.

Response

```
>>> ENABLE ECHO
.
.
.

>>> DISABLE ECHO
```

3.14.3 Enable/Disable Local Copy Command

Syntax - (ENABLE ! DISABLE) <blank> LOCAL <blank> COPY <CR>

Description - With the enable local copy command the local terminal prints a copy of all output directed to the remote terminal. The disable local copy command disables the local terminal from getting a copy of output directed to the remote terminal.

Local copy disable is entered each time the console keyswitch is turned to the LOCAL REMOTE, or LOCAL/DISABLE position. Local copy enable is entered each time the console control panel rotary keyswitch is placed in the REMOTE/DISABLE position.

Response

```
>>> ENABLE LOCAL COPY

>>> DISABLE LOCAL COPY
```

3.14.4 Enable Local Control Command

Syntax - ENABLE <blank> LOCAL <blank> CONTROL <CR>

Description - An enable local control command, entered at the remote terminal (when the console keyswitch is in the REMOTE position), transfers control of the console to the local terminal. This permits the local operator to take control of the console and the CPU, while the remote link is maintained. The remote operator can regain control of the console by typing CTRL P (↑P).

An enable local control command entered from the local terminal has no effect. Local control is enabled when the console control panel rotary keyswitch is placed in the LOCAL or LOCAL/DISABLE position.

Response

```
>>> ENABLE LOCAL CONTROL
.
.
.
↑P

>>>
```

3.14.5 Enable/Disable Carrier Error Command

Syntax - (ENABLE ! DISABLE <blank> CARRIER <blank> ERROR <CR>

Description - With the enable carrier error command the message ?CARRIER LOST is printed each time a loss of carrier on the remote line is detected.

Response

```
>>> ENABLE CARRIER ERROR
.
.
.
? CARRIER LOST

>>>
```

The disable carrier error command causes the console to inhibit the carrier lost message. An enable carrier error is automatically done whenever the console keyswitch is placed in the LOCAL or LOCAL/DISABLE position.

3.15 CONSOLE ERROR MESSAGES

All console error messages are prefixed by a question mark to distinguish them from informational messages. A description of the console error messages is contained in Appendix E. To obtain a set from your VAX-11/780 enter

```
>>>@ERROR.HLP
```

CHAPTER 4
DIAGNOSTIC SUPERVISOR

4.1 INTRODUCTION

The diagnostic supervisor is a program that controls the operation of a set of diagnostic programs. The diagnostic supervisor includes a command language for the operator's use in controlling the operation of the diagnostic supervisor program and the diagnostic program being run under the diagnostic supervisor.

The overall features of the diagnostic supervisor and the associated command language are described in the following paragraphs. For detailed information on the diagnostic supervisor and the diagnostic programs refer to the *DS780 Diagnostic System Technical Description.*

4.2 FEATURES

The diagnostic supervisor program is contained on those floppy disks that contain diagnostic programs controlled by the diagnostic supervisor (Appendix M).

To load the diagnostic supervisor program from a floppy disk, the VAX-11/780 must be in the console I/O mode of operation (the console command language is addressable).

To load the diagnostic supervisor program from the system device, the VAX-11/780 must be in the program I/O mode of operation. The program is addressed through a user terminal or through the console terminal. In the program I/O mode the data is handled by the VAX-11/780 operating system.

The diagnostic supervisor program level of operation is indicated by DS> prompt at the operator's terminal. From this level the diagnostic supervisor commands can be entered. The diagnostic program must be loaded (with a console command language load command) prior to reaching the DS> level prompt.

Once the diagnostic supervisor is running (DS> is displayed) a diagnostic program can be started. The diagnostic supervisor reads the header data and the initial operator diagnostic test program dialogue to determine how it is to control the running of the diagnostic test program.

There are two VAX-11/780 operational levels for the diagnostic test programs: the standalone mode and the user mode. In the standalone mode, the VAX-11/780 system is dedicated to the diagnostic test program. In the standalone mode of operation, the diagnostic supervisor simulates the necessary VAX/VMS functions. In the user mode the diagnostic test program shares in the VAX 11/780 operation (VAX/VMS is operational).

4.3 DIAGNOSTIC SUPERVISOR COMMAND LANGUAGE

The diagnostic supervisor commands are English-language entries or their accepted abbreviations entered by the operator at the terminal. The commands can be entered when the system is at the diagnostic supervisor level – a diagnostic supervisor prompt (DS>) is present at the operator's terminal. When a valid command is entered the cited action, as described in the following paragraphs, is initiated.

The diagnostic supervisor command descriptions include, as applicable, the following:

1. Syntax – the structure or elements of the command

2. Command description – a brief description of the command operation, general restrictions, and available option

3. Response – a description of diagnostic supervisor program responses to the command.

The terms and symbols used in the diagnostic supervisor command language are identical to those used in the console command language (Table 3-1).

NOTE

Every diagnostic supervisor command line must be terminated with a carriage return (CR).

4.3.1 Command Abbreviations

The diagnostic supervisor commands can be abbreviated. The minimum abbreviation for each command is specified in parentheses in each command description title. If an improper abbreviation and, therefore, an illegal command is given, the diagnostic supervisor responds by outputting an illegal command.

4.3.2 Load Command (L)

Syntax – LOAD<file spec> [PHYSICAL:<address>] <CR>

where:

<file spec> is the alphabetic code (mnemonic) assigned the diagnostic test program.

[PHYSICAL:<address>] is an optional qualifier that causes the program to be loaded in physically contiguous memory.

Description – The load command initiates a transfer of the diagnostic test program (<file spec>) from the system device to main memory. When the load is completed, the diagnostic test program may be run.

The diagnostic supervisor (DS>) load command may be used only when the VAX-11/780 is in the user mode. In the standalone mode, the diagnostic test program is loaded via the console command language (>>>) load command.

Response – The completion of the loading of the specified diagnostic test program <file spec> is signified by a load done message.

```
>>>LOAD ESSAA.EXE/ST:FE00

    LOAD DONE, 0000D000 BYTES LOADED
>>>LOAD ESCAA.EXE/ST:200

    LOAD DONE, 0000B400 BYTES LOADED
```

After the desired programs have been loaded, an initialized (INIT) and an unjam (UNJAM) command should be given prior to starting the loaded programs.

4.3.3 Start Command (ST)

Syntax - START [/TEST:<first>[<last> ! /SUBTEST:<number>]]
[/PASS:<count>] [SEC: <sect name>] <CR>

where:

[/TEST:<first>[:<last>] is an optional qualifier for selecting the beginning (<first>) and end (<last>) test of the diagnostic test program to be run. The <last> default is the last test of the program.

or

[/TEST:<first>[:/SUBTEST:<number>] is an optional qualifier for selecting a subtest (<number>) within a test.

[/PASS: <count>] is an optional qualifier for selecting the number (<count>) of times test is to be run.

[/SEC: <sect name>] is an optional qualifier for selecting a section within a diagnostic test program.

A description of the tests and sections within each program is contained in *DS780 Diagnostic System Technical Description.*

Description - The start command initiates the execution of the diagnostic test program presently in main memory. If none of the qualifiers are used, the complete program runs one time. Control flags can be used to modify (control) the running of the program.

Response

```
>>>ST 10000

DIAGNOSTIC SUPERVISOR. ZZ-ESSAA-4.02-417 21-JUL-1978 01:29:16.60

DS> ST

PROGRAM: MAINDEC ZZ-ESCAA-5.1 RH780 DIAGNOSTIC, REV 5, DEPO 1, 20 TESTS.

****** SIZER NOT YET IMPLEMENTED ******
```

The ST 10000 command loads the diagnostic supervisor beginning at location 10000. The DS> prompt indicates that the diagnostic supervisor is loaded. The ST command initiates the series of questions. The operator response to the questions is to the right of the colon.

```
SELECT UNITS: MBA:

FOR DEVICE: MBA
  ENTER TR NUMBER [1-16(D)]: 8
  ENTER ADAPTER BR LEVEL [4-7(D)]: 5

RHO BASE ADDRESS IS:20010000
NO DRIVE RESPONDED.
******* NO MBE PRESENT ON MASSBUS *******
END OF RUN, 1 PASS. 0 ERRORS DETECTED, TIME: 21-JUL-1978 01:30:41.93

DS>
```

4.3.4 Restart Command (RE)

Syntax - RESTART [/TEST:<first>[:<last>!/SUBTEST:<number>]]
[/PASS: <count>] [SEC: <sect name>] <CR>

The meanings for the optional qualifiers are exactly the same as those described for the start command.

Description - The restart command initiates the diagnostic test program into execution. The program runs in accordance with the qualifier defined under the start command unless new entries are made under the restart command. The control flags cannot be modified.

Response

DS>RESTART

PROGRAM: ZZ-ESCBA-3.0-DW780 REPAIR DIAGNOSTICS, REV DEP0

The diagnostic test program is identified and, with the exception that there is no program operator interface, the program runs as if it were a start command.

4.3.5 Run Command (RUN)

Syntax - RUN <filespec>[/TEST:<first>[:<last>!/SUBTEST:<number>]]
[/PASS: <count>] [SEC: <sect.name>]<CR>

where <file spec> is the alphabetic code or mnemonic for the diagnostic test program. The meanings for the optional qualifiers are exactly the same as those described for the start command.

Description - The run command initiates a combined load and a start command sequence. The run command can be initiated only when the VAX-11/78[illegible] is in the user mode.

Response

DS> RUN/ESCAA/2/3

4.3.6 Continue Command (CONT)

Syntax - CONTINUE <CR>

Description - The continue command restarts the diagnostic test program at the point at which the program was suspended (interrupted). The interruption may be the result of a breakpoint halt, an error halt, or a CTRL C (↑C).

Response

Interruption initially caused by ↑C, or error halt, or breakpoint halt

DS> CONT

CON - PROGRAM CONTINUING AT PC 00015CFD

XXXXXXXX DW780 REPAIR DIAGNOSTIC CODE ZZ-3.0 - 0.0XXXXXXXX

The response to the continue command is an identity of program counter status (where in the program the restart is being made) and an identification of the diagnostic test program.

4.3.7 Summary Command

Syntax - SUMMARY <CR>

Description – The summary command initiates the summary report (statistical report) of the diagnostic test program. This command is run after a diagnostic test program has been run.

Response
DS> SUMMARY

.

.

057 .

4.3.8 Abort Command (A)

Syntax – ABORT <CR>

Description – The abort command initiates the program's cleanup code and then returns control to the diagnostic supervisor (DS>). The clean up code resets the device under test to a set of initial test condition states.

Response
DS> ABORT

ABO - PROGRAM ABORTED AT PASS 0 CLEANUP SECTION PC 00014EEB.

4.3.9 Control and Special Characters

There are a number of special characters that can be used at the diagnostic supervisor level. These special characters and their descriptions are contained in Table 4-1.

Table 4-1 Control/Special Character Description

Character	Description
CTRL C (↑C)	Returns program control to the diagnostic supervisor command language prompt level (>>>), a command wait state. The operator may then enter any diagnostic supervisor command.
CTRL O (↑O)	Suppresses or enables (on a toggle basis) the console terminal output. The console terminal output is always enabled with a diagnostic supervisor input prompt (DS>). The diagnostic supervisor overrides the CTRL O (maintains and/or reinstates the operator terminal output) whenever system errors, CLI prompts, or forced messages are being serviced.
CTRL U (↑U)	Deletes all characters indexed by the operator since the last line terminator (<CR>). The console terminal echoes U <CR> <LF>
DELETE	Typing DELETE (the DELETE key) or rubout deletes the last character typed by the operator. Only characters entered since the last line terminator (<CR>) can be deleted. Several characters can be deleted by typing successive deletes. The first delete echoes as a backslash (\) followed by the deleted character (....VAN\N). Additional deletes cause only the characters to be echoed (....VAN\NAV). The first character entered that is not a delete causes a backslash (\) to be printed. (....VAN\NA\C).
Carriage Return <CR>	Entering a carriage return (<CR>) terminates a command line.

4.3.10 Directory Command (@DIR)

Syntax - @ DIRECTORY <CR>

Description - The directory command initiates a listing of the contents of the floppy presently installed in the RX01.

Response

```
>>>@DIR

!
!  DIRECTORY     DXO:
!  19-JUL-78
!
!  ESZAE .VCO      0. 19-JUL-78
!  ESCAA           1. 19-JUL-78
!  ESCBA           1. 19-JUL-78
!  CONSOL.SYS     31. 19-JUL-78
!  WCS115.PAT     25. 19-JUL-78
!  ESSAA .EXE    104. 19-JUL-78
!  ESCAA .EXE     90. 19-JUL-78
!  ESCBA .EXE    124. 19-JUL-78
!  DIR             2. 19-JUL-78
!  < UNUSED >    102.
!
!  102. FREE BLOCKS
!
!  TOTAL OF 378. BLOCKS IN 9. FILES
!
!
!  DESCRIPTION OF DIAGNOSTICS ON FLOPPY DISKETTE
!
!  ESSAA.EXE     : VAX DIAGNOSTIC SUPERVISOR  :AUTOMATICALLY LOADED
!  ESCAA.EXE     : RH780 MASSBUS DIAGNOSTIC   : TO RUN TYPE: @ ESCAA
!  ESCBA.EXE     : DW780 UNIBUS DIAGNOSTIC    : TO RUN TYPE: @ ESCBA
!
<@EOF>
<@EXIT>

>>>
```

4.4 EXECUTION CONTROL COMMANDS

Through the execution control commands the operator can alter the operational characteristics (statically or dynamically) of the diagnostic program and/or the diagnostic supervisor. These commands are implemented by flags that reside in both the diagnostic supervisor and the diagnostic program. The diagnostic program event flags are supported by the diagnostic supervisor and the operating system (VAX/VMS).

These commands are used to control the printing of error messages, the ringing of the bell, the halting and looping of the program, etc. Flags are provided to set the diagnostic supervisor for the dialogue characteristic desired by the operator.

The operator has access to a set of event flags that can be used in the diagnostic program.

4.4.1 Set Control Flags Command (SET)

Syntax - SET FLAGS <argument list> <CR>

where <argument list> is one or more of the control flags, as selected by the operator. The control flags and a description of them are contained in Table 4-2.

Table 4-2 Control Flags

Mnemonic/Name	Description
HALTD	Halt on error detection. With this flag set, the diagnostic supervisor enters into the command wait state (DS> is displayed) whenever the diagnostic program detects a failure. All error messages associated with that failure are printed on the console terminal prior to entering the wait state. The operator may then continue, restart, or abort the program. This flag takes precedence over the loop flag.
HALTI	Halt on error isolation. With this flag set, the diagnostic supervisor outputs all error messages associated with the error and then enters into the command wait state (DS> is displayed) when the diagnostic program isolates a failure. The operator may then continue, restart, or abort the program. This program is for use with those programs that perform fault isolation and fault detection at different levels.
LOOP	Loop on error. With this flag set, the diagnostic program enters into a repeat cycle (oscilloscope testing loop) on a test or subtest in which a failure is detected. The repeated cycle continues until the operator enters a CTRL C (↑C) placing the diagnostic supervisor into the command wait state (DS> is displayed). The operator may then continue, clear the flag and continue, restart, or abort the diagnostic program.
BELL	Bell on error. With this flag set, the diagnostic supervisor outputs a bell signal to the console terminal every time the diagnostic program detects a failure.
IE1	Inhibit error messages at level 1. When this flag is set all error messages, except those that are forced by the program, are suppressed.
IE2	Inhibit error messages at level 2. With this flag set, only the header information for each failure is output. The basic and extended information about the failure is suppressed.
IE3	Inhibit error messages at level 3. When this flag is set, the header and basic information messages for each failure are output to the console terminal. The extended information on the failure is suppressed.

Table 4-2 Control Flags (Cont)

Mnemonic/Name	Description
IE/S	Inhibit summary report. When this flag is set, the statistical report messages are suppressed.
QUICK	Quick verify. When this flag is set, the diagnostic supervisor runs only those tests in the diagnostic program that are identified as a part of the quick verify operation. Some tests of long duration are not run. Not all diagnostic programs have a quick verify capability.
SPOOL	List error messages on line printer. When this flag is set, all diagnostic supervisor error messages are sent to the line printer. In the user mode (VAX/VMS operational) the messages are not printed until the diagnostic supervisor logs off.
TRACE	Report the execution of each test. When this flag is set, the diagnostic supervisor outputs the name of each test within the diagnostic supervisor. The test name is sent to the operator terminal as each test is begun.
LOCK	Lock on physical memory. When this flag is set, the self-relocating programs are locked into their current physical memory space (the program relocation function is disabled).
OPERATOR	Operator present. When this flag is set, operator interaction with the diagnostic supervisor is possible. When this flag is not set, the diagnostic supervisor acts to ensure that the test session bypasses any tests that require operator intervention.
PROMPT	Display long dialogue. When this flag is set, the limits and defaults for all questions in the program are printed.
ALL	All flags in this list. When this flag is set, all flags within the above list are set.

Description – The set flags command sets the selected flags (<argument list>) within the diagnostic program. When the diagnostic program is run, the diagnostic supervisor responds to the flags that are set.

Response

```
DS> SET FLAGS BELL,IEI, TRACE

DS>
```

The bell, IE1, and trace flags are set within the diagnostic program. When the program is run, the diagnostic supervisor monitors the program and rings a bell for each error diagnosed, but inhibits all error messages except for those that are forced by the diagnostic program. The tests are listed as they are started.

4.4.2 Control Flags

The control flags and their description are contained in Table 4-2.

4.4.3 Clear Control Flags Command (CLEAR)

Syntax - CLEAR FLAGS <argument list> <CR>

where <argument list> is one or more of the control flags. These are selected by the operator from those flags that have been set. The prompt and operator flags are default set.

Description - The clear flags command clears the selected flags (<argument list>) within the diagnostic program.

Response

DS> CLEAR ALL

DS> SHOW FLAGS

CONTROL FLAGS SET:
CONTROL FLAGS CLEAR: PROMPT, OPER, LOCK, TRACE, SPOOL, QUICK, IES, IE3, IE2, IE1, BELL, LOOP, HALT

All flags are cleared.

4.4.4 Set Control Flags Default Command

Syntax - SET FLAGS DEFAULT <CR>

Description - With this command all flags are returned to their initial or default state. The initial states are for the operator and prompt flags to be set and all other flags cleared.

Response

DS> SET FLAGS DEFAULT

DS> SHOW FLAGS

CONTROL FLAGS SET: PROMPT, OPER
CONTROL FLAGS CLEAR: LOCK, TRACE, SPOOL, QUICK, IES, IE3, IE2, IE1, BELL, LOOP, HALT

4.4.5 Show Control Flags Command

Syntax - SHOW FLAGS <CR>

Description - With this command all flags that set are listed followed by a listing of those flags that are clear.

Response

```
DS> SHOW FLAGS

CONTROL FLAGS SET:      PROMPT, OPER
CONTROL FLAGS CLEAR:    LOCK, TRACE, SPOOL, QUICK, IES, IE3, IE2, IE1, BELL,
                        LOOP, HALT

DS>
```

The two flags set are default flags.

4.4.6 Set Event Flags Command

Syntax - SET EVENT FLAGS <argument list> ! ALL <CR>

where <argument list> is one or more of the event flags, as selected by the operator. The optional ALL calls for all event flags to be set.

```
DS> SET EVENT FLAGS ALL

DS> SHOW EVENT FLAGS

EVENT FLAGS SET:        23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6,
                        5, 4, 3, 2, 1

DS>
```

Description - The set event flags command sets the selected flags (<argument list>) within the diagnostic program. The flags are a string of numbers from 1 through 23. Their use and function are diagnostic program dependent. (They are listed with each diagnostic program.)

4.4.7 Clear Event Flags Command

Syntax -]LEAR EVENT FLAGS <argument list>!ALL<CR>

where <argument list> is one or more of the event flags. These are selected by the operator from those flags that have been set. The ALL option clears all flags.

Description - The clear event flags command clears the selected flags (<argument list>) within the diagnostic program.

Response

```
DS> CLEAR EVENT FLAGS    /23,19,15,11,1

DS> SHOW EVENT FLAGS

EVENT FLAGS SET:         22, 21, 20, 18, 17, 16, 14, 13, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2

DS>
```

The event flags 1, 3, and 4 for the diagnostic program are cleared.

4.4.8 Show Event Flags Command

Syntax - SHOW EVENT FLAGS <CR>

Description - This command causes the list of event flags currently set to be output to the operator terminal.

Response

```
DS> SHOW EVENT FLAGS

EVENT FLAGS SET: 1, 2, 3, 6

DS>
```

4.5 DEBUG AND UTILITY COMMANDS

This group of diagnostic supervisor commands provides the operator with a means to troubleshoot and modify the diagnostic program. The modified diagnostic programs can then be used for testing the associated portion of the VAX-11/780 and also be transferred to a load device for future use.

4.5.1 Set Base Command

Syntax - SET BASE <address> <CR>

Description - This command establishes an <address> bias (offset) for all subsequent examine, deposit, or breakpoint commands. It is commonly set to the base of a program section to allow program listing addresses to be referenced directly.

Response

```
DS > SET BASE 1000

DS >
```

4.5.2 Set Breakpoint Command

Syntax - SET BREAKPOINT <address> <CR>

Description - This command causes control to pass to the supervisor when program execution encounters the <address> previously specified by this command. A maximum of sixteen simultaneous breakpoints can be set within the diagnostic program.

Response

```
DS > SET BREAKPOINT E5000, E6000, E7000

DS >
```

4.5.3 Clear Breakpoint Command

Syntax - CLEAR BREAKPOINT <address> ! ALL <CR>

Description - This command clears the previously set breakpoint at the memory location specified by <address>. If no breakpoint existed at the specified address, no error message is output. An optional argument of ALL clears all previously defined breakpoints.

Response

```
DS> CLEAR BREAKPOINT ALL

DS> SHOW BREAKPOINTS
      NONE CURRENTLY SET

DS>
```

4.5.4 Show Breakpoints Command

Syntax - SHOW BREAKPOINTS <CR>

Description - This command displays all currently defined breakpoints.

Response

```
DS> SHOW BREAKPOINTS

CURRENT BREAKPOINTS:
        00002001(X)
        00003000(X)

DS >
```

4.5.5 Set Default Command

Syntax - SET DEFAULT <argument list> <CR>

Description - This command is used to set the default qualifiers for the examine and deposit commands. The <argument list> argument consists of a data length default and/or radix default qualifiers. They are separated by a comma if both are present. If only one default qualifier is specified, the other one is not affected. Default qualifiers are:

Data Length: BYTE, WORD, LONG

Radix: HEXADECIMAL, DECIMAL, OCTAL

Response

```
DS > SET DEFAULT BYTE, OCTAL

DS >
```

4.5.6 Examine Command

Syntax - EXAMINE [/<qualifiers>] [<address>] <CR>

Description - The examine command displays the contents of memory in the format described by the qualifiers. If no qualifiers are specified, the default qualifiers set by a previous default command are implemented. The applicable qualifiers are described in Table 4-3.

Response

```
DS> EXAMINE/B/O 4263
    00004263 132

DS>
```

When specified, the <address> argument is accepted in hexadecimal format by default. Optionally, <address> may be specified as decimal or octal by immediately preceding the address argument with %D or %O, respectively. If no <address> is specified, the next memory location is displayed.

Table 4-3 Qualifier Descriptions

Qualifier	Description
/B	Address points to a byte
/W	Address points to a word
/L	Address points to a longword
/X	Display in hexadecimal radix
/D	Display in decimal radix
/O	Display in octal radix
/A	Display in ASCII bytes
/P	Address specifies a processor register

4.5.7 Deposit Command

Syntax - DEPOSIT [/<qualifiers>] <address> <data> <CR>

Description - The command accepts data and writes it into the memory location specified by <address> in the format described by the qualifiers. If no qualifiers are specified, the default qualifiers are implemented. The applicable qualifiers are identical to those of the examine command and described in Table 4-3.

The <address> argument is accepted in hexadecimal format by default. Optionally <address> may be specified as decimal or octal by immediately preceding <address> with %D or %O, respectively.

Response

```
DS> DEPOSIT/W/D 5678

DS>
```

CHAPTER 5
PERIPHERAL DEVICES OPERATING INFORMATION

5.1 GENERAL

This chapter contains a brief description of the peripheral equipments and the console equipments (LA36, LSI-11, and DX01). The description includes the equipment specifications, the operating controls and indicators, the operating procedures, and customer care.

5.2 LA36 TERMINAL

The LA36 DECwriter II is a medium-sized, low-cost, interactive data communications terminal (Figure 5-1). It is designed as an I/O device that is used in the VAX-11/780 system console subsystem and may be used as a remote communications terminal or a user terminal.

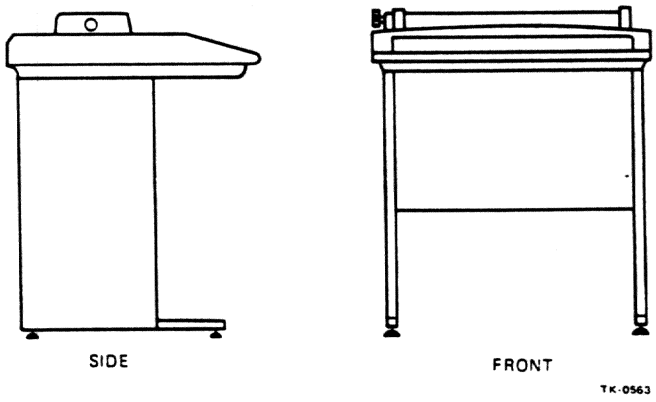


Figure 5-1 LA36 DECwriter II

The LA36 incorporates a low-speed impact printer for hard copy output and a standard ASCII-coded keyboard for transmitting data to the host system or to another data communication terminal(s).

The printer is capable of handling multipart and multiform line printer paper from 7.62 to 37.78 cm (3 to 14-7/8 in) wide.

Preprinted forms can be positioned for the best print-to-line-positioning incidence through the use of the manual vernier paper advance adjustment.

The keyboard is a standard ASCII-coded keyboard, consisting of alphanumeric characters and nonprinting system control codes. The ASCII code set consists of upper- and lower-case characters, and prints at a horizontal spacing of 10 char/in and a vertical spacing of 6 lines/in. The CAPS LOCK switch on the keyboard allows selection of a reduced number of ASCII characters.

5.2.1 Options

The basic LA36 DECwriter II can be expanded with the following options.

Options	Name	Control Switch/Indicator	Description
LAXX-KG	EIA/CCITT Adapter for LA36 terminal	None	The interface includes a 3.45 m (9 ft) cable terminated with a standard EIA connector.

5.2.2 LA36 Specifications

Printing	Switch selectable: 10, 15, or 30 char/s throughput
Line Length	132 characters maximum
Spacing	10 char/in (horizontal) 6 lines/in (vertical)
Characters	96 upper/lower case ASCII 7 × 7 dot matrix 1.77 × 2.54 mm (0.07 × 0.10 in)
Paper	Variable width: 7.62 to 37.78 cm (3 to 14-7/8 in)
Single-Part	6.8 kg (15 lb) paper minimum Card stock thickness of 0.25 mm (0.010 in) maximum
Multipart	2- to 6-part (see Notes) Thickness of 0.50 mm (0.020 in) maximum Tractor-drive, pin-feed

NOTES

1. **Multipart forms may have only one card part. The card must be the last part.**
2. **NCR or 3M paper, up to 6-part, must use ribbon on top copy. First surface impact paper is not recommended.**
3. **Continuous-feed, fan-fold business forms with 3- or 4-prong margin crimps on both margins (multipart) are recommended.**

Stapled forms are not recommended and may damage tractors and other areas of the machine. Dot or line glue margins are acceptable (if line is on one margin only). Do not line glue both margins as air will not be able to escape and poor impressions will result.

Keyboard	Standard ANSI keyboard layout Multikey roll-over
Interface	Integrated 20 mA current loop
Ribbon	DIGITAL-specified nylon fabric (Part No. 36-10558) spool assembly: 1.27 cm (0.5 in.) wide × 36.58 m (40 yd) long
Power	90–132 Vac or 180–264 Vac 50 or 60 Hz ± 1 Hz 300 W maximum (printing) 160 W maximum (idle)
Dimensions	69.85 cm (27.5 in) wide 85.09 cm (33.5 in) high 60.69 cm (24.0 in) deep
Weight	46.36 kg (102 lb)

5.2.3 LA36 Operator Controls and Indicators

5.2.3.1 Keyboard Controls – The following is a list of the keyboard controls (Figure 5-2).

POWER ON/OFF Switch
The POWER ON/OFF push-to-toggle switch connects and disconnects the line voltage to the LA36 DECwriter II. The POWER switch should be in the ON position for normal unit operation. When changing paper, ribbon, or adjusting the print head, the switch should be turned OFF.

LINE/LOC Switch
The LINE/LOC switch is a 2-position pushbutton switch. When in the LINE position (up), the LA36 is enabled to transmit and receive data. When in the LOC position (depressed), the LA36 receive/transmit lines are disabled and only local operation can be performed.

FDX/HDX Switch
The FDX/HDX 2-position pushbutton switch controls the printing of transmitted keyboard characters. When in the FDX position (up), characters typed on the keyboard are transmitted; only received characters are printed. When in the HDX position (down), keyboard characters are both transmitted and printed; received characters are also printed. The operator should not attempt to transmit data when receiving data in the HDX mode.

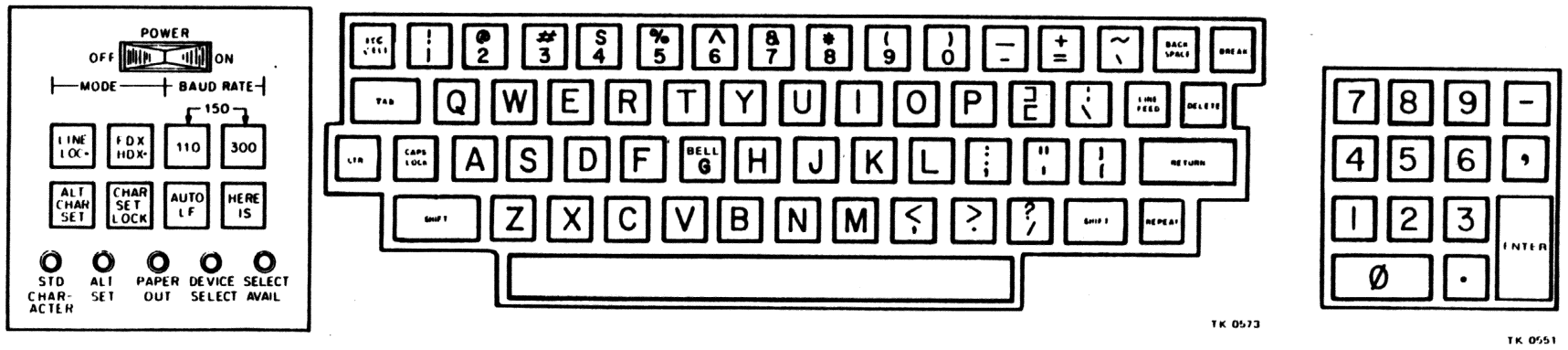


Figure 5-2 LA36 Keyboard

BAUD RATE Switches
The BAUD RATE pushbutton switches select (when pressed) the rate at which characters are transmitted and received over the communication line.

Switch	Character Rate
110	10 (char/s)
300	30 (char/s)
110 and 300	15 (char/s)

The 30 char/s rate is the normal VAX-11/780 operating rate.

ALT CHAR SET Switch
When an alternate character set is installed, the ALT CHAR SET 2-position pushbutton switch allows the operator to select either the standard ASCII character set (switch up) or the alternate character set (switch pressed).

CHAR SET LOCK Switch
The CHAR SET LOCK 2-position pushbutton switch allows the operator to select manual (switch pressed) or automatic (switch up) character set switching. In the automatic mode, the transmitting device can change the character set by issuing the character codes switch in (Control O) or switch out (Control N). The switch in code selects the standard ASCII character set. The switch out code selects the alternate character set. In the manual mode, the character set is selected by the ALT CHAR SET control switch.

STD/ALT CHARACTER SET Indicators
The STD/ALT CHARACTER SET indicators give a visual indication of the selected character set. If no alternate character set option is installed, the STD indicator is always illuminated when power is on.

AUTO LF Switch
Not applicable to the VAX-11/780.

HERE IS Switch
Not applicable to the VAX-11/780.

PAPER OUT Indicator
When illuminated, the PAPER OUT indicator gives a visual indication that the LA36 is out of paper. Printing stops when the out of paper condition is detected.

DEVICE SELECT Indicator
Not applicable.

SELECT AVAIL Indicator
Not applicable.

CTRL Key
The CTRL key provides the LA36 operator with a method of transmitting ASCII control codes (000_8–037_8). Holding the CTRL key down and pressing any alphanumeric key or control key changes the standard alphanumeric ASCII code for that key to a control code (000_8–037_8).

CAPS LOCK Key
The 26 letter keys transmit only upper case when the CAPS LOCK 2-position pushbutton switch is pressed. None of the other keys are affected.

SHIFT Key
This is a momentary switch which, when pressed, allows selection of upper case for all printable characters.

TAB Key
Not applicable.

ESC (SEL) Key
Not applicable.

REPEAT Key
The REPEAT key does not generate an ASCII code. However, when the REPEAT key is held down and any key is pressed, it causes the ASCII code for that character to be transmitted and printed at a repetition rate of approximately 15 char/s (until the key is released).

RETURN Key
The RETURN key generates code 015_8. The printer control logic causes the print head to be repositioned to the left-hand margin each time the carriage return character code is received. If a line feed (LF) character code follows the carriage return code, the line feed operation is executed simultaneously with the carriage return.

LINE FEED Key
The LINE FEED key generates code 012_8. The printer advances the paper one line each time the LF code is received.

NOTE
Rapid paper advance can be obtained by placing the LA36 in the LOCAL mode and pressing the LINE FEED and REPEAT keys.

BACKSPACE Key
The BACKSPACE key generates code 010_8. The printer control logic causes the print head to move one position to the left each time a BACKSPACE code is received, until the print head reaches the left-hand margin.

DELETE Key
The DELETE key generates code 177_8. The printer does not respond to the delete code.

BREAK Key
The BREAK key is provided for users that use the half-duplex mode of transmission. The BREAK key allows the LA36 operator to interrupt incoming data flow by forcing the communication line from a mark mode into the space mode (until the BREAK key is released.

Numeric Keypad
The numeric keypad, which is in an adding machine layout, enables numbers to be entered. Each key in the numeric keypad generates the same ASCII character as the corresponding key in the main keyboard. The ENTER key corresponds to the RETURN key.

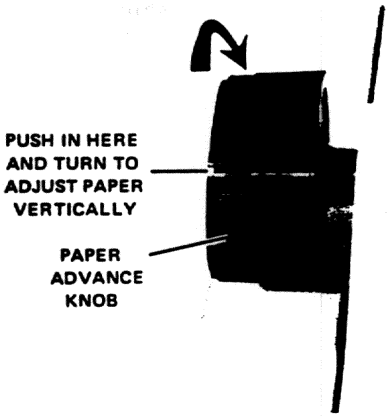
5.2.3.2 Non-Keyboard Controls – The following are non-keyboard controls (Figure 5-3).

Carriage Adjustment Lever
The carriage adjustment lever controls the print head gap for single-part or multipart forms.

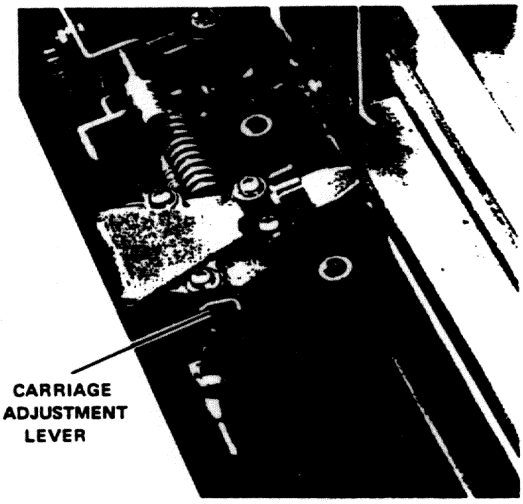
Paper Advance Knob
The paper advance knob pushes in to disengage the line feed gear train, allowing fine vertical adjustment of paper position. Coarse vertical adjustment is achieved by simply turning the knob. Each detent turned represents one line.

Tractor Adjust Knob
The tractor adjust knob allows fine horizontal adjustment of the tractors for holding the forms.

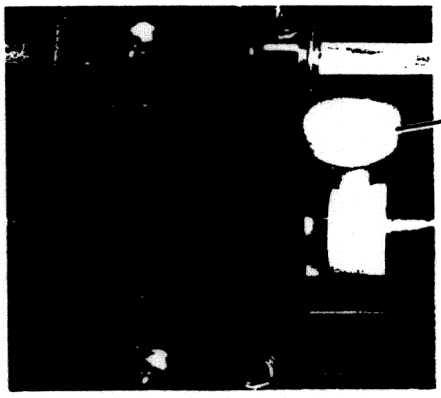
Bell
The bell indicates that a bell code was received, or that the print head has reached column 64.



7595-4-A0199



7666-16-A0218



7595-7-A0198

Figure 5-3 Non-Keyboard Controls

5.2.4 Operating Procedures

5.2.4.1 Turn-On Procedure – These steps are all that may be required to place your LA36 on-line (Figure 5-2):

1. Load paper.
2. Set LINE/LOC switch to LOC.
3. Set POWER ON/OFF switch to ON.
4. Select full-duplex (VAX-11/780) or half-duplex (FDX/HDX switch).
5. Select character set (ALT CHAR SET and CHAR SET LOCK switches), if applicable.
6. Select desired baud rate (BAUD RATE switches).
7. Set the LINE/LOC switch to LINE.

The LA36 is now on-line and ready to receive and transmit data.

5.2.4.2 Loading Paper and Changing the Ribbon – Loading Paper
The LA36 can accept multipart forms, with widths from 3 to 14-7/8 inches. When loading new forms, it is necessary to perform two adjustments:

1. Paper positioning
2. Impression adjustment.

In addition, there is a horizontal positioning and vertical positioning adjustment. The horizontal positioning adjustment allows the paper to be shifted left or right slightly. This procedure is especially useful when typing on preprinted forms with defined horizontal zones. The vertical positioning adjustment enables the paper to be adjusted vertically. Once these adjustments have been performed, reloading paper becomes quick and simple, requiring a minimum of interruption.

5.2.4.3 Loading New Forms – Paper Positioning Procedure

1. Set the POWER switch to OFF.

2. Lift the cover.

3. Place the tractor-feed paper on the floor between the legs of the LA36. (The term tractor-feed refers to the holes on either side of the paper.)

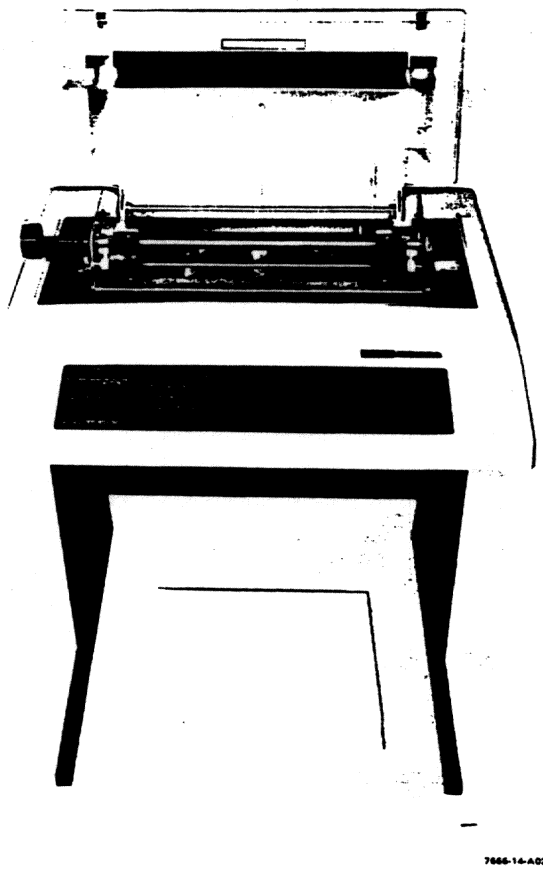
NOTE
Ensure that the leading edge of the forms is directly below and parallel to the feed slot.

4. Open the left tractor cover so that the tractor pins are exposed (Figure 5-4).

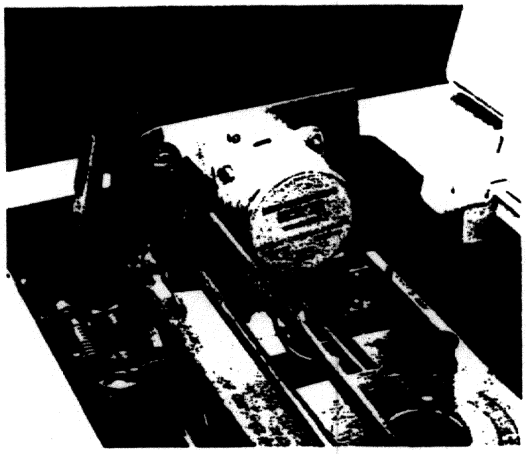
5. Move the carriage adjustment lever to the highest number (toward operator).

6. Feed the paper through the load channel under the terminal and align the left paper margin holes over the left tractor pins (Figure 5-5).

7. Close the left tractor cover.



7666-14-A0220



7666-32-A0215

7666-16-A0218

Figure 5-4 Loading Paper

Figure 5-5 Threading Paper to Tractor Pins

8. Loosen the tractor adjustment knob on the right tractor about 1/2 turn.

9. Open the right tractor cover and slide the tractor to a position where the holes on the right paper margin align directly over the tractor pins.

10. Close the tractor cover.

NOTE

Ensure that the paper does not pull against the tractor pins or bow in the middle.

11. Tighten the tractor adjustment, and proceed to the impression adjustment.

5.2.4.4 Impression Adjustment (Figure 5-6)

NOTE

The carriage adjustment lever is normally set forward (to notch number 1) for single thickness paper. The following procedure is applicable only to multipart forms.

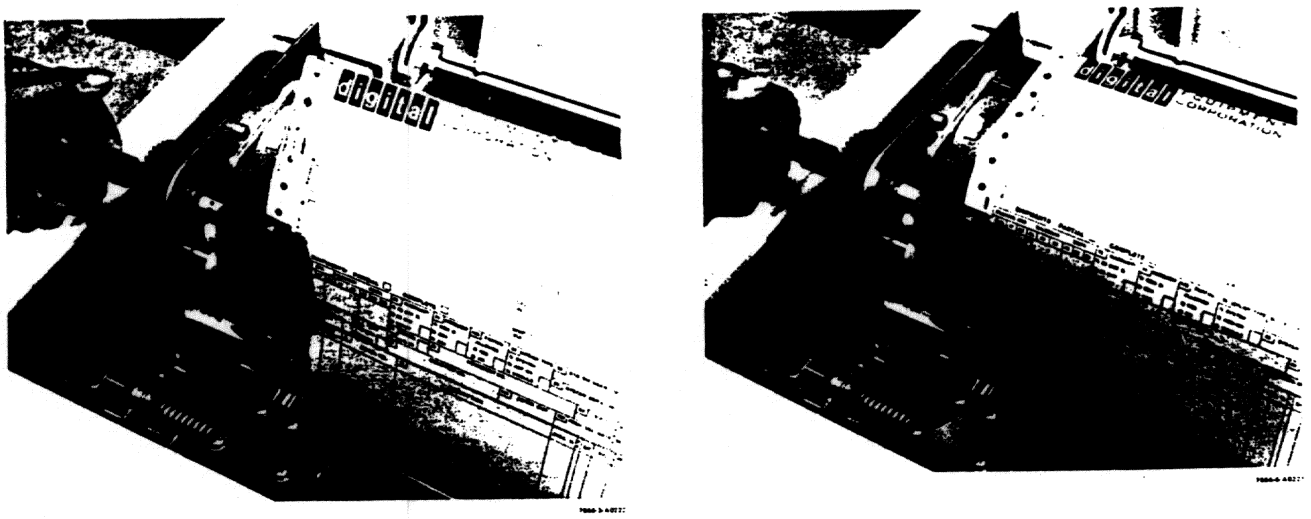


Figure 5-6 Impression Adjustment

1. Set the POWER switch to OFF.
2. Set the carriage adjustment lever to the number corresponding to the number of parts in the form.
3. Turn the paper advance knob counterclockwise while moving the carriage adjustment lever forward one notch at a time until the paper smudges: then move the lever back one notch at a time until the paper no longer smudges.
4. Set the POWER switch to ON and resume operation.

NOTE

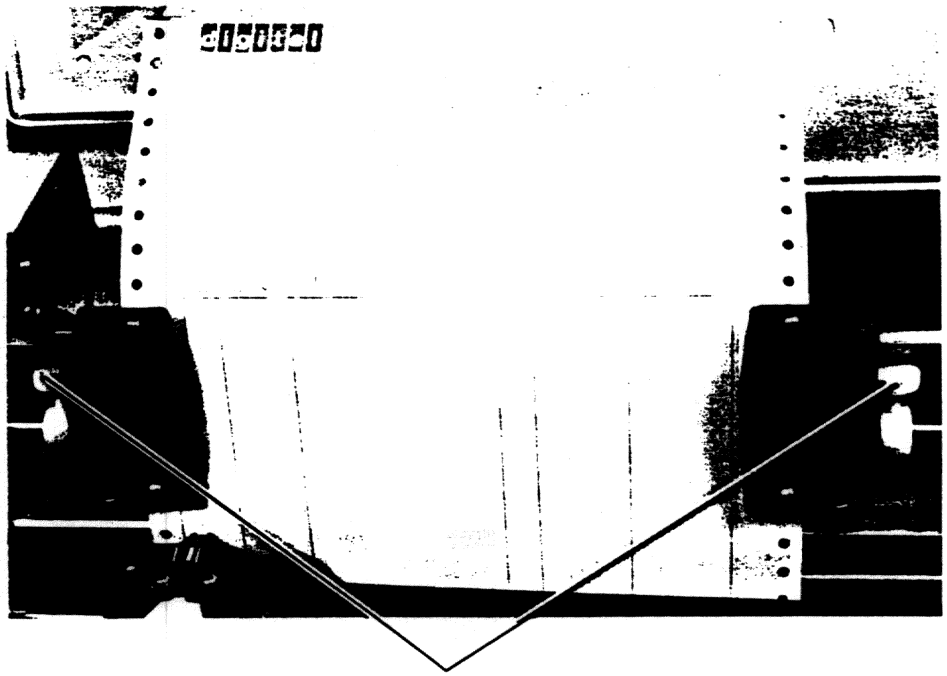
If the impression is unsatisfactory due to a worn ribbon, perform the ribbon installation procedure. An indication of a worn ribbon is that the first copy in a multipart copy is poor but the remaining copies are good.

5.2.4.5 Horizontal Positioning Adjustment (Figure 5-7) – The horizontal positioning adjustment enables the paper to be shifted left or right [1.27 cm (1/2 in.) max]. Shifting the paper provides a simple means of aligning the type within the appropriate columns on the paper.

1. Set the POWER switch to OFF.
2. Lift the cover and loosen both tractor adjustment knobs about 1/2 turn.
3. Move the tractors the desired amount (1/2 in max) to have characters type in the appropriate columns (Figure 5-5).
4. Tighten the tractor adjustment knobs.

NOTE

Ensure that the paper does not pull against the tractor pins or bow in the middle.



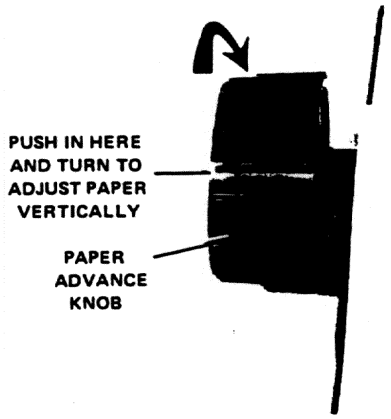


Figure 5-7 Horizontal and Vertical Position Adjustment

5.2.4.6 Fine Vertical Positioning (Figure 5-7) - For fine vertical positioning, press in and turn the paper advance knob.

5.2.4.7 Reloading Paper (Figure 5-4)

1. Set the POWER switch to OFF.
2. Lift the cover.
3. Place the tractor-feed paper on the floor between the legs of the LA36.
4. Open both tractor covers so that the tractor pins are exposed.

NOTE

Ensure that the leading edge of the forms is directly below and parallel to the feed slot.

5. Feed the paper through the load channel under the terminal and align the paper holes over the tractor pins.
6. Close the tractor covers.

5.2.4.8 Ribbon Installation – The printer ribbon should last for 16 to 20 hours of actual printing at 30 char/s (about 2 million characters). After 12 hours, or when the print density becomes too light, remove both ribbon spools from their drive spindles and turn the whole assembly over so that the previous lower edge of the ribbon is now on top. After rethreading the ribbon, another 8 hours (approx) of printing time is possible before the ink is used completely. At that time, the ribbon must be replaced by removing both spools and unthreading the ribbon. Replace with a new spool and ribbon assembly (36-10558) and an empty spool. (One of the old spools may be used if desired.)

NOTE

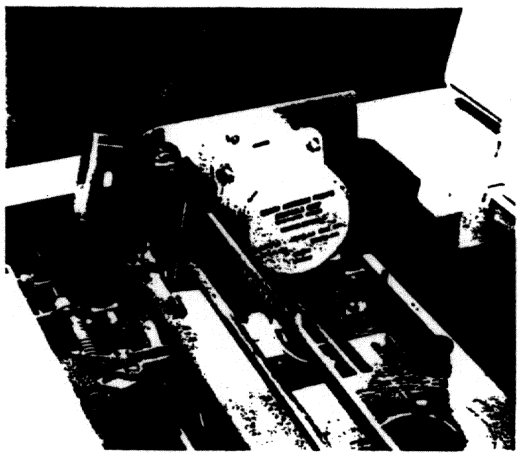
Use only DIGITAL-recommended ribbons (Part No. 36-10558). Use of other ribbons can cause damage and void machine warranty.

1. Set the POWER switch to OFF and lift the cover (Figure 5-8).
2. Record the setting of the carriage adjust lever. Move the carriage adjust lever to the highest number.
3. Remove the ribbon spools and ribbon. Save one spool to be used with the new ribbon.
4. Connect the hook on the end of the ribbon to the empty spool.

NOTE

The rivet located on the ribbon must be on the spool or between the spool and the direction changing guide.

5. Wind 10 turns of ribbon on the empty spool.
6. Place the full spool on the left spindle and turn clockwise until it drops into position.
7. Guide the ribbon around idler spool A, through guide B, and around the outside of idler spools C through E.



7666-32-A0215

7666-17-A0217

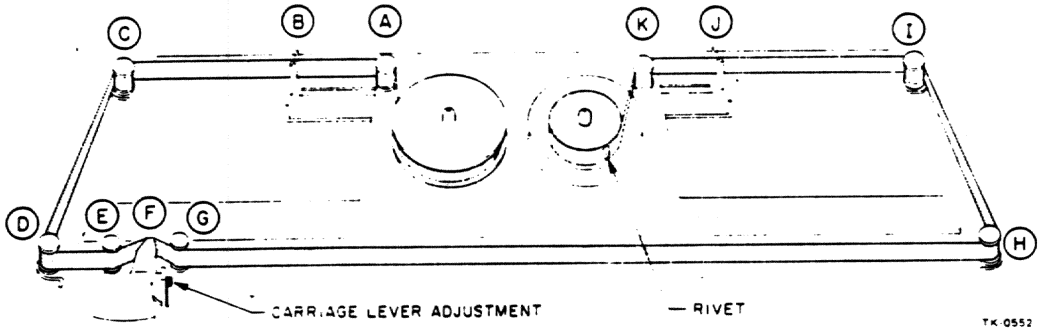


Figure 5-8 Installing a New Ribbon

8. Guide the ribbon around the front of head F and idler spools G through I.

9. Guide the ribbon through slot J (director changing guide) and around idler spool K. Ribbon can be threaded in the opposite direction (from K to A).

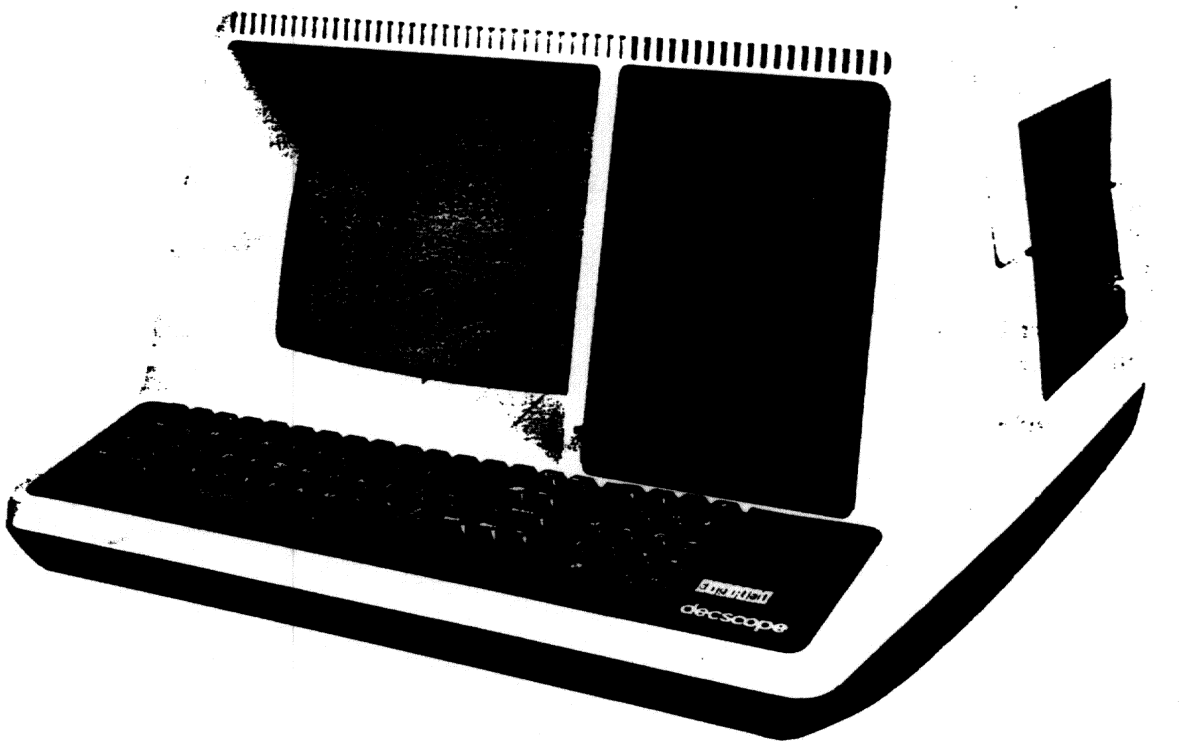
10. Turn the spool clockwise until it drops into position.

11. Take up the slack in the ribbon by turning the free moving spool.

12. Return the carriage adjust lever to its original setting.

5.3 VT52 VIDEO DISPLAY TERMINAL

The VT52 Video Display Terminal (Figure 5-9) is a 24-line, 80-character display terminal that serves as an I/O device to a host processor. It transmits and receives data at speeds up to 9600 baud. The main keyboard uses a standard typewriter layout that supplies upper- and lower-case alphabetic characters as well as numeric, symbol, and control characters to a host. 7 × 7 dot matrix is used to form the displayable characters. All input, output, and display operations are controlled by a ROM resident microprogram.



7966-10-A0208

Figure 5-9 VT52 Video Display Terminal

5.3.1 Options

Design features of the VT52 include a 19-key auxiliary keyboard for applications requiring much numeric input, and a hold-screen mode for operator control of processor files. These and other features are described in the following paragraphs.

5.3.1.1 19-Key Auxiliary Keypad – This feature allows the operator to use the convenient 19-key numeric pad in applications requiring much numeric input. In one mode, numeric keys from the pad transmit the same code as numeric keys from the main keyboard. In the alternate-keypad mode, each key transmits a unique escape sequence that is used to initiate a user-defined function.

The three blank keys on the pad also transmit a unique escape sequence and can be defined by the customer for his particular application.

Four cursor control keys transmit unique escape sequences that move the cursor one position UP, DOWN, LEFT, or RIGHT, in the direction of the arrow, if the code produced by the key is echoed back to the terminal by the host.

5.3.1.2 Hold-Screen Mode – The hold-screen mode allows the operator to regulate the speed of the data received from the host and displayed on the screen. Normally, the host transmits data at a rate that is too fast for the operator to read; as the host adds new information to the bottom line, the information on the top line is scrolled off and lost to the operator. When in hold-screen mode, the terminal will not process information that would cause a line scroll until requested to do so by the operator. The operator can request a new line of characters by typing SCROLL or a new screenful of characters by typing SHIFT, SCROLL.

5.3.1.3 Direct Cursor Addressing – Direct cursor addressing allows the host to move the cursor to any position on the screen by transmitting an escape sequence to the terminal.

5.3.1.4 Identification Feature – This feature allows the host software to poll every terminal on the system, requesting each to identify itself. The VT52 will automatically respond with a 3-character escape sequence that identifies it as a VT52. This allows the VT52 to be mixed with other terminals in a system with the software responding correctly to each different type of terminal.

5.3.1.5 Terminal Interface – The VT52 is available with either the 20 mA current loop interface or, if ordered by the customer, a standard EIA interface.

5.3.1.6 Optional Copier – An electrolytic copier is available to provide hard copy capability to the terminal. The copier prints information one line at a time; each line appears on the paper exactly as it appeared on the screen.

5.3.2 VT52 Specification

Dimensions	Height: 360 mm (14.1 in) Width: 530 mm (20.9 in) Depth: 690 mm (27.2 in) Minimum table depth: 450 mm (17.7 in)
Weight	20 kg (44 lb)
Line Voltage	U.S. model: 100–126 V European model: 191–238 V or 209–260 V
Line Frequency	U.S. model: 60 ± 1 Hz European model: 60 ± 1 Hz or 50 ± 1 Hz

Power Consumption	110 W
Display	
Format	24 lines × 80 characters
Character Matrix	7 × 7
Character Size	2.0 mm × 4.0 mm (0.08 in × 0.16 in)
Screen Size	210 mm × 105 mm (8.3 in × 4.1 in)
Character Set	96-character displayable ASCII subset (upper- and lower-case, numeric, and punctuation)

5.3.3 Controls

Two groups of controls are used in the VT52 Video Display Terminal, the operator controls and the keyboard. The operator controls set the operating parameters of the terminal such as parity, baud rate, etc. The keyboard keys select the codes to be transmitted to the host.

5.3.3.1 Operator Controls – The operator controls and their functions are listed in Table 5-1.

5.3.3.2 Keyboard – Input data to the host is entered by typing keys on the main keyboard or the auxiliary keypad (Figure 5-10). The keyboard transmits upper- and lower-case alphabetic code, numeric code, fixed control code, and user-defined control code.

Alphabetic Keys
All alphabetic keys transmit in upper-case and lower-case code. Upper-case is transmitted when a key is typed while either or both SHIFT keys are down or while the CAPS LOCK key is down. (The CAPS LOCK key does not affect codes transmitted by keys other than the alphabetic keys.)

Numeric/Symbol Keys
Numeric and symbol keys.

Function Keys
Function keys transmit control codes to the host. They cause an action to occur in the terminal such as tab, line feed, etc., if the host echoes these codes back to the terminal.

The CONTROL Key
The CONTROL key is used in conjunction with other keys on the keyboard to produce control codes in the range of 000_8–037_8. When held down, it alters the code normally produced by a typed key by forcing the two high-order bits of the code to zero.

The BREAK key
Typing the BREAK key forces the serial data output line of the terminal to the zero state for as long as the key is held down. The BREAK function is provided for users with software written to operate in half-duplex mode. The VT52 operates in full-duplex mode so there is normally no requirement for the BREAK function.

Auxiliary Keypad
The 19-key auxiliary keypad is provided for applications requiring heavy use of the numeric keys. In addition to the 10 numeric keys, the keypad has a decimal point key, four cursor move keys, three blank keys, and an ENTER key.

Table 5-1 VT52 Operator Controls

Control	Location	Function
ON/OFF	Right side	Applies power to the unit
Baud rate switches: Front, bottom		
S1-1		Off-Line*
S1-2		Full duplex with local copy*
S1-3		Full duplex*
S1-4		300 baud
S1-5		150 baud
S1-6		75 baud
S1-7		4800 baud
S2-A		Match (Bell 103) - Local copy*
S2-B		110 baud
S2-C		Match (Bell 103)*
S2-D		600 baud
S2-E		1200 baud
S2-F		2400 baud
S2-G		9600 baud
S3	Front, bottom	Selects NO (mark) parity or EVEN parity (SPACE or ODD with jumpers)
Brightness	Rear	Sets screen intensity level

*In positions indicated with an asterisk, receiving and transmitting speeds will be the same. In other positions, S1 selects the transmitting speed and S2 selects the receiving speed. S1 and S2 switch positions 1A, 1C, 2A, 2C, 3A, and 3C are illegal.

In normal mode the decimal point key and the numeric keys transmit the same code as the decimal point key and the numeric keys on the main keyboard; the host cannot distinguish between them. The ENTER key transmits the same code as the RETURN key.

In response to a command from the host, the terminal will enter the alternate-keypad mode and the ENTER, decimal point, and numeric keys will each transmit a unique escape sequence. This will allow the host to distinguish between main keyboard entries and auxiliary keypad entries. It also provides the host with 12 user-definable keys to use for his particular application. Appendix J lists the codes transmitted by these keys.

The seven remaining keys on the keypad are the four cursor move keys and three blank keys. The cursor move keys transmit an escape sequence to the host; if the host echoes these codes back to the terminal, the cursor moves one character position up, down, left, or right, depending upon the key typed. The three blank keys transmit user-defined escape sequences; the user can define the meaning of each key to fit his particular application.

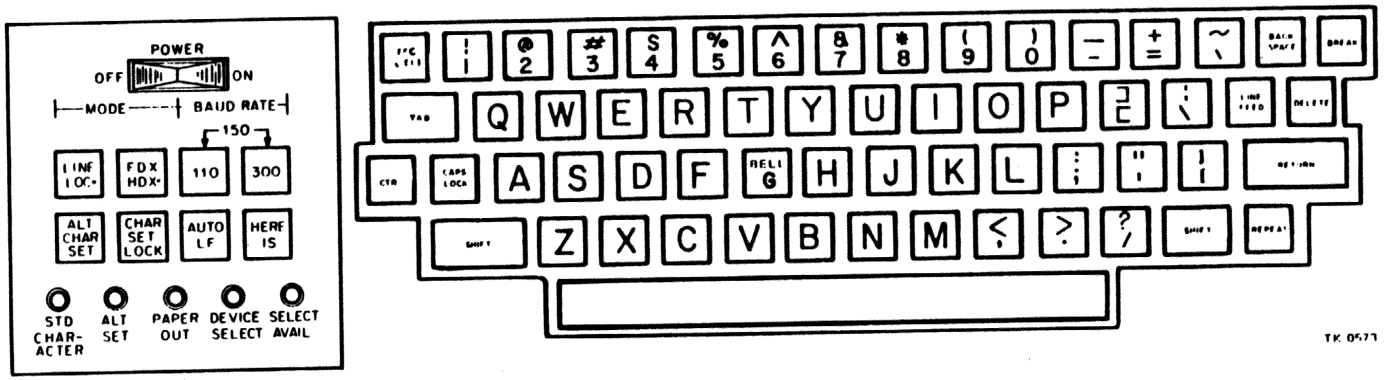


Figure 5-10 VT52 Keyboard

None of the keys on the auxiliary keypad are affected by pressing the SHIFT, CAPS LOCK, or CTRL keys.

The REPEAT key
The REPEAT key is used in conjunction with other keys; it does not transmit a code. Any key that transmits a code to the host will transmit that code repeatedly if pressed while the REPEAT key is down. Keys that transmit more than one code will transmit their sequence repeatedly if pressed while the REPEAT key is down.

The SCROLL Key
The SCROLL key performs a local function; it does not transmit a code to the host. It is used to request more data from the host when the terminal is in hold-screen mode. Typing SCROLL will add one line of characters to the display screen; typing SHIFT SCROLL adds a screenful.

The COPY Key
The COPY key is also a local function key. When pressed, the COPY key will produce a hard copy of the current screenful of characters, at the optional VT50 Series Copier Terminal.

5.3.4 Operating Procedure

5.3.4.1 Off-Line

1. Set the unit for OFF-LINE operation. To do this, set rotary switch S1 to position 1 and rotary switch S2 to position G.

   These switches are located at the front of the unit, under the keyboard. The unit will have to be tipped to gain access to the switches.

2. Apply power to the terminal by setting the ON OFF switch to ON. This switch is located on the right side of the terminal.

3. After warmup (about 1/2 minute) the flashing cursor will appear at the top left of the screen. This is called the home position.

4. Using the brightness control located at the rear of the terminal, adjust the brightness of the cursor to the desired level. For the most comfortable viewing, set the brightness to the lowest possible level while still maintaining good character definition. High screen intensity causes characters to become fuzzy; prolonged viewing of the bright display is very tiresome.

5. Type a character. The character will appear on the screen at the cursor location and the cursor will move one character location to the right.

6. With the REPEAT key held down, type another character. The character will appear in all the remaining locations of the top line and the cursor will be under column 79 (the last column).

7. Type a different character. It will replace the last character in the line because that is the cursor location. The cursor advances automatically as characters are typed, until it reaches the last character position in the line. RETURN and LINE FEED must be typed at the end of each line.

8. Type the RETURN key. The cursor will move to the left end of the line. Now type LINE FEED. The cursor will move down to the next line. Type a few characters to form the second line.

9. Try the rest of the cursor move keys. They will all move the cursor one character position. The BACKSPACE key will move the cursor to the left. The four cursor move keys on the auxiliary keypad will move the cursor one character position in the direction indicated by the arrow.

10. Move the cursor to the left-hand margin on a blank line. Alternately type a character and TAB. The typed characters will appear eight character locations apart. These are the TAB STOP locations. If the cursor is at the end of a line, TAB will not move the cursor.

11. Position the cursor under a displayed character and type SPACE BAR. The character will be replaced with a space and the cursor will be advanced one character position. The space bar may be used for erasing.

12. Type some characters with the SHIFT key and then the CAPS LOCK key held down. Alphabetic characters will appear in upper-case when typed with either SHIFT or CAPS LOCK pressed. Numeric keys are not affected when CAPS LOCK is held down. Symbols associated with the numeric keys are displayed only when the numeric keys are typed with the SHIFT key held down.

13. Move the cursor to the left-hand margin of the bottom line. With REPEAT held down, type a character key and fill the bottom line. Type RETURN and LINE FEED. The terminal will move all lines UP by one position, erasing the bottom line for new entries. The top line is lost in this operation. This is called scrolling and will occur whenever a LINE FEED is typed while the cursor is on the bottom line unless the terminal is in hold-screen mode.

5.3.4.2 On-Line

1. Remove power from the terminal by pressing the ON/OFF push-to-toggle switch to OFF.

2. Tip the unit up and set the baud rate switches to the desired frequency. Transmission speed must be set to the same frequency as the host's receiving speed; receiving speed must be set to the same frequency as the host's transmission speed.

3. Set the ON/OFF switch to ON to restore power to the terminal. After warmup the terminal is ready for communication with the host.

5.3.5 User Maintenance

The keyboard and key-click mechanism are the only moving parts of the terminal and require no preventive maintenance by the owner. The DECscope may be cleaned with soap and water or any mild detergent. Although the terminal's shell is resistant to damage from solvents, cleaners with solvents should not be used.

The DECscope packaging is not weatherproof; there are also several openings in the case through which liquids, coins, paper clips, and other objects can fall. Such objects can disturb the electronic operation of the terminal if they come in contact with the circuitry. For this reason, avoid putting drinks and metal objects on the top of the terminal, or using excessive water to clean the terminal. The keyboard area is an area where the electronics are particularly close to the exterior. Rubbing the keys with a dry or barely moist cloth should suffice to clean them. Do not remove the keycaps to clean them more thoroughly; damage may result to the switch contacts if the caps are replaced incorrectly.

5.4 RM03 DISK SUBSYSTEM

The RM03 Disk Subsystem (Figure 5-11) is a high-performance, direct access, single-head per surface drive designed to enable a data processing system to store and retrieve blocks of data at any location on a rotating disk. The RM03 consists of a disk drive and a Massbus adapter in a free-standing cabinet. The RM03 has a formatted capacity of 67 megabytes.

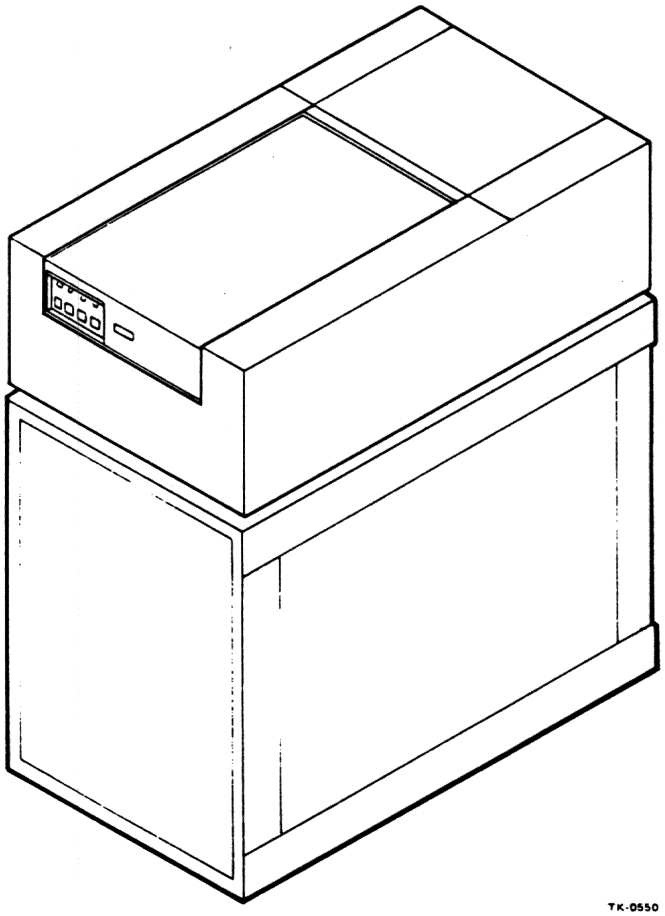


Figure 5-11 RM03 Disk Subsystem

5.4.1 RM03 Specification

The specifications for the RM03 are contained in Table 5-2.

5.4.2 Controls and Indicators

The RM03 Disk Subsystem has operator's controls and indicators located on the drive's front panel, rear panel, and on the Massbus adapter power supply. See Figure 5-12 for their locations. They are illustrated in Figure 5-13.

Table 5-2 RM03 Specifications

Specification	Limit
Electrical	
Voltages available (single-phase)	100 Vac + 10, -10; 60 Hz 120 Vac + 8, -18; 60 Hz
240 Vac + 17, -27; 50 Hz	
Start current for:	
100 Vac, 60 Hz	33 A rms, maximum
120 Vac, 60 Hz	30 A rms, maximum
240 Vac, 50 Hz	22 A rms, maximum
100 Vac, 50 Hz	33 A rms, maximum
Line current	
Disk and carriage in motion	Total
100 Vac, 60 Hz	11 A rms
120 Vac, 60 Hz	11 A rms
240 Vac, 50 Hz	7 A rms
100 Vac, 50 Hz	14 A rms
In standby mode	
100 Vac, 60 Hz	4.5 A rms
120 Vac, 60 Hz	4.5 A rms
240 Vac, 50 Hz	3.5 A rms
100 Vac, 50 Hz	7 A rms
Line Cord Length	213.4 cm (7 ft)
Disk Cartridge Type	RM03P

START Switch
The START switch is a 2-position pushbutton switch. Pressing the START switch when the drive is in the power-off condition (disk pack not spinning), lights the START indicator and initiates the power-up sequence provided the following conditions are met.

- Disk pack is installed.
- Pack access cover is closed.
- All power supply circuits are on.

START Indicator
This indicator lights when the START switch is pressed from the off position provided the three preconditions just mentioned are met. The light indicates that power is on.

READY Indicator
Once the start sequence has been initiated by pressing the START switch, the READY indicator blinks once a second until the disk pack is up to speed.

When the READY indicator stays on permanently, it indicates the following conditions have been met.

- Disk pack is up to speed.
- Heads are loaded.
- No fault conditions exist.

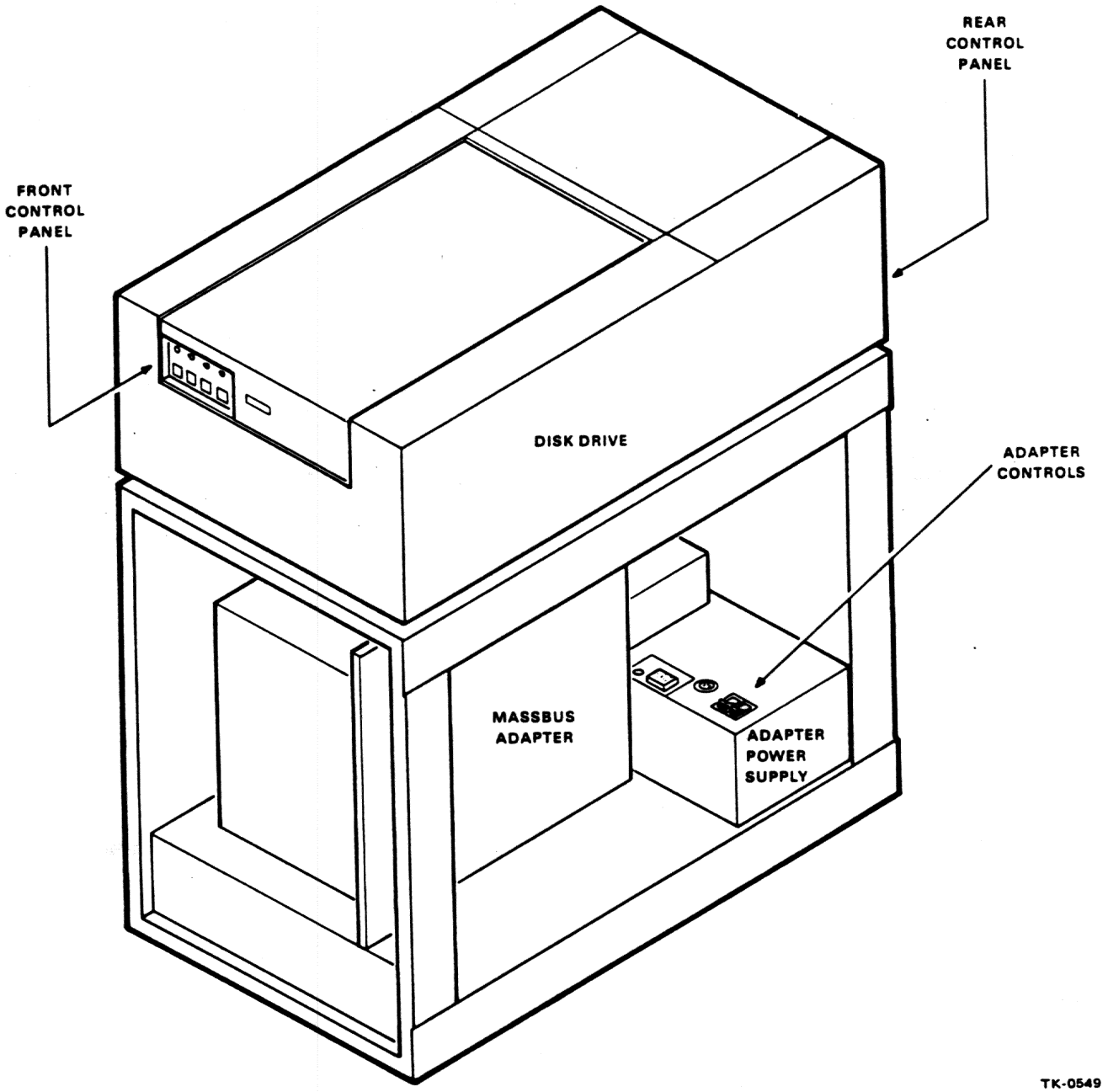


Figure 5-12 Location of RM03 Subsystem Controls

DRIVE SELECT Plug
This plug determines the logical address of the drive. The address can be changed to any number from 0 to 7 by installing the proper plug. If no plug is installed, the plug valid line to the controller is false.

FAULT Indicator
The FAULT indicator lights up when one or more of the following fault conditions exist in the drive.

- Write fault
- Head select fault

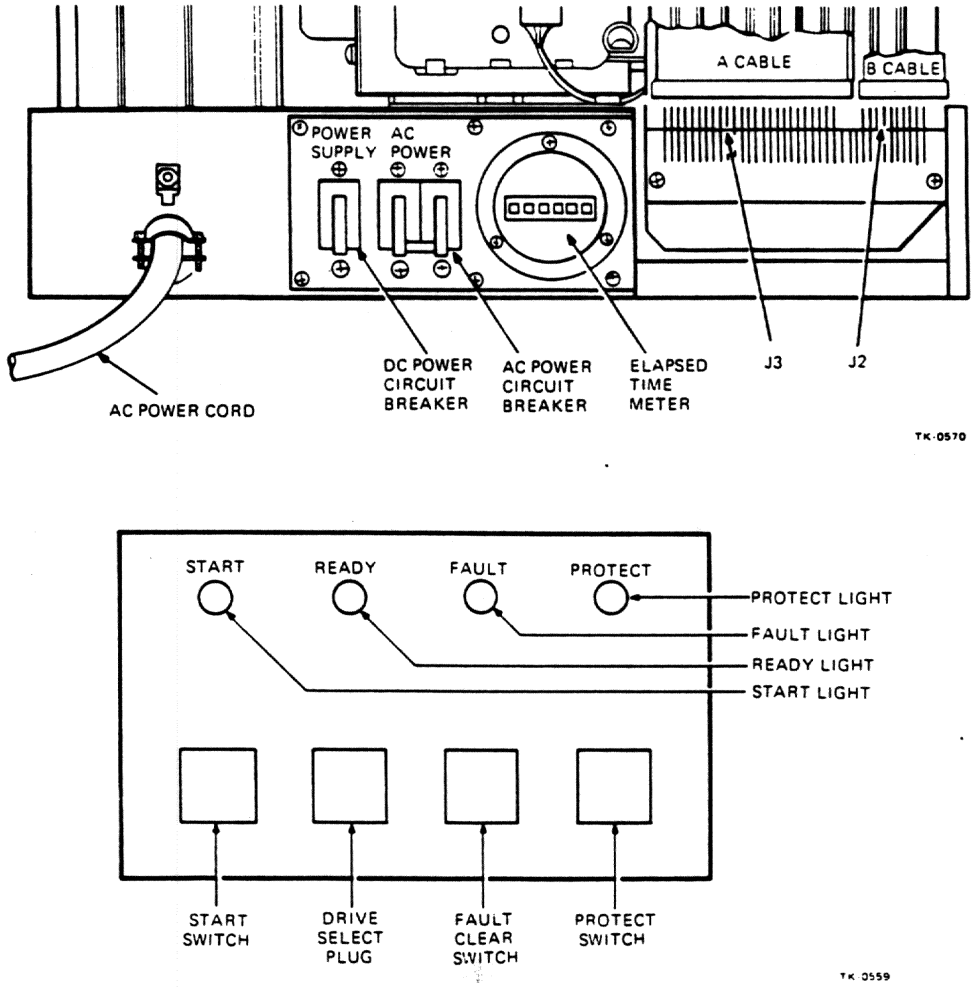


Figure 5-13 RM03 Controls

- Read/write fault
- Read/write while off cylinder fault
- Voltage fault

If the fault condition that caused the FAULT indicator to come on has been removed, then the FAULT indicator can be extinguished in any one of five different ways.

- Pressing the FAULT switch on the operator's control panel
- Pressing the MAINTENANCE FAULT CLEAR switch on the fault card in logic chassis location A04
- Issuing an initialize signal from the controller
- Issuing a fault clear signal from the controller
- Powering down the drive

FAULT CLEAR Switch
This switch is a momentary contact pushbutton used to clear the FAULT indicator. Pressing the FAULT CLEAR switch extinguishes the FAULT indicator only if the fault condition no longer exists.

WRITE PROTECT Indicator
When this indicator is on, it signifies that the drive is in the write protect mode. While in this mode, data cannot be written on the disk pack.

WRITE PROTECT Switch
This switch is a 2-position pushbutton. When pressed to the IN position, it lights the WRITE PROTECT indicator and disables the drive's write circuits. While in this position, data cannot be written onto the pack.

Releasing this switch back to its OUT position removes the disable signal from the write circuits and extinguishes the WRITE PROTECT indicator.

Rear Panel Controls
The rear panel of the disk drive contains the ac power controls for the drive motors and logic circuitry. It also houses an elapsed time meter.

AC Power Circuit Breaker
The ac power circuit breaker applies ac power to the drive's dc power supply circuit breaker and the drive blower motor. It also provides ac power for the drive spindle motor. The spindle motor will not start unless the control panel START switch is pressed to enable the motor start-up sequence.

DC Power Supply Circuit Breaker
This circuit breaker applies ac power to the dc power supply provided that the ac power circuit breaker is on and the power cord is plugged in. The dc power supply provides the dc voltages needed by the drive logic circuits.

Elapsed Time Meter
This meter records the accumulated ac power-on time. The meter starts when the ac power circuit breaker is set to ON.

Adapter Power Supply Controls
The adapter power supply is located under the disk drive inside the subsystem cabinet. Access to it can be gained most easily through the rear door of the cabinet.

On the top surface of the adapter power supply are located some ac power circuit breakers, an ac receptacle, a fuse, and a dc power connector (J9). Figure 5-14 shows these controls and connectors.

AC Receptacle
The ac receptacle on the top of the adapter power supply is used as the source of ac power for the disk drive power cord. When the drive power cord is plugged into this receptacle, the ac power to the drive is controlled by the MAIN AC POWER CIRCUIT BREAKER on top of the adapter power supply.

Main AC Power Circuit Breaker
This circuit breaker controls the ac power to the whole RM03 Disk Subsystem. When in the ON position, it routes ac power to both the adapter dc power supply circuit breaker and to the disk drive ac power cord.

Adapter DC Power Supply Circuit Breaker
This circuit breaker controls the ac power to adapter dc power supply provided the main ac circuit breaker is on. The dc power supply provides the dc voltages needed by the Massbus adapter circuits.

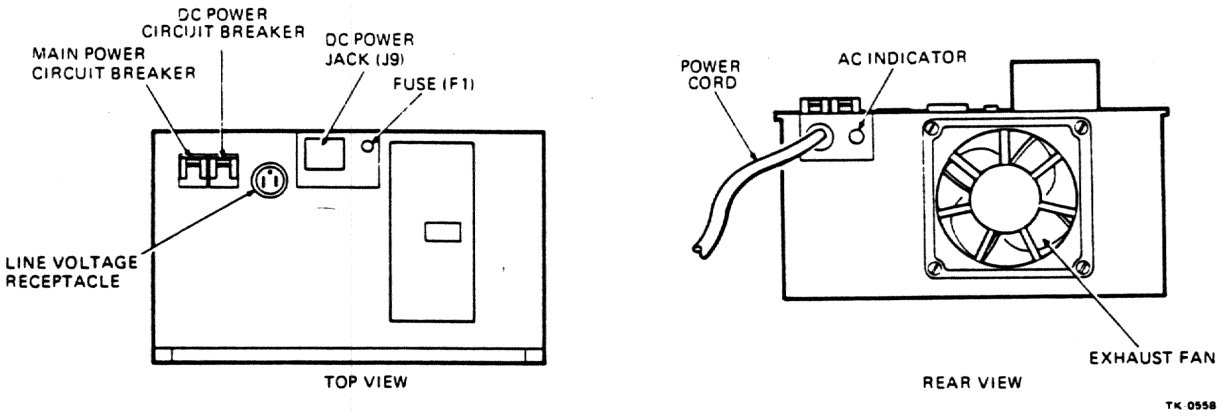


Figure 5-14 Adapter Power Supply Controls

It does this through a power cable that connects to the dc power connector (J9) shown on top of the adapter power supply.

-15 V Supply Fuse (F1)
The fuse (F1) on the top of the adapter power supply is used to protect the transformer from short circuits in the -15 V supply.

AC Indicator
This light is located on the back side of the adapter power supply, in close proximity to the ac power cord. The indicator is on whenever the power cord is plugged into a source of ac power. It is a warning indicator that even though the subsystem is not energized, there is still ac voltage present up to the ac circuit breaker terminal.

5.4.3 Disk Pack Loading and Device Startup

To load a disk pack into the disk drive requires that the drive have power applied. Otherwise, the pack access cover cannot be raised due to a solenoid latch lock. If the drive is already powered on, skip over the following power application procedure and go directly to the disk pack installation.

5.4.3.1 Applying Subsystem Power - The procedure described here for energizing the disk subsystem assumes that it is connected to an ac power source. It also assumes that the LOCAL/REMOTE switch, located on logic card A04, is set to REMOTE. The power application procedure is as follows.

- Verify that the two circuit breakers on top of the adapter power supply and the two circuit breakers on the disk drive rear panel are set to the OFF position.
- Switch on the ac power circuit breaker on the adapter power supply.
- Turn on the dc power supply breaker on the adapter power supply.
- Turn on the ac circuit breaker on the drive rear panel.
- Turn on the dc power supply circuit breaker on the drive rear panel.
- Ensure that the drive blower starts to operate and wait 2 minutes before proceeding.

CAUTION

Failure to allow the blower to operate for 2 minutes before installing the disk pack will not allow sufficient purge time and may cause damage to disk pack or heads.

5.4.3.2 Disk Pack Installation – Assuming the drive is powered on, the disk pack must be installed prior to performing any drive operation. Disk pack installation consists of setting the pack on the drive spindle and locking the pack in place. This procedure is described in detail in the following steps.

NOTE

These drives have a power sequencing feature that is designed to prevent a drive in a daisy-chain configuration from going through the startup cycle unless all the drives previous to it in the string have either gone through the startup cycle, or are powered off. If a drive should fail to start up, check that these conditions have been met.

1. Disengage the bottom dust cover from the pack canister by holding the bottom cover and turning the pack handle counterclockwise (Figure 5-15).

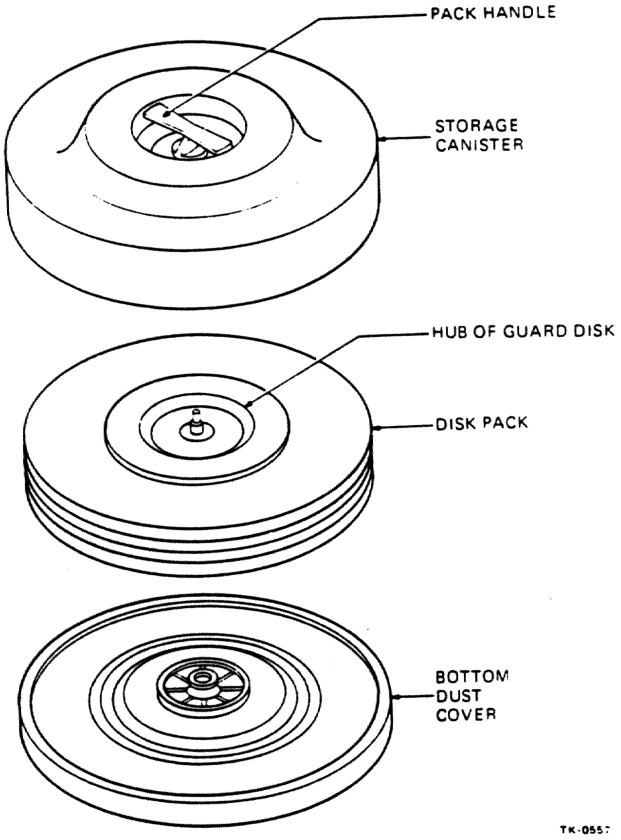


Figure 5-15 RM03 Disk Pack

2. Raise the pack access cover and ensure that the heads are fully retracted (Figure 5-16).

CAUTION
Failure to ensure that heads are fully retracted may damage heads or pack.

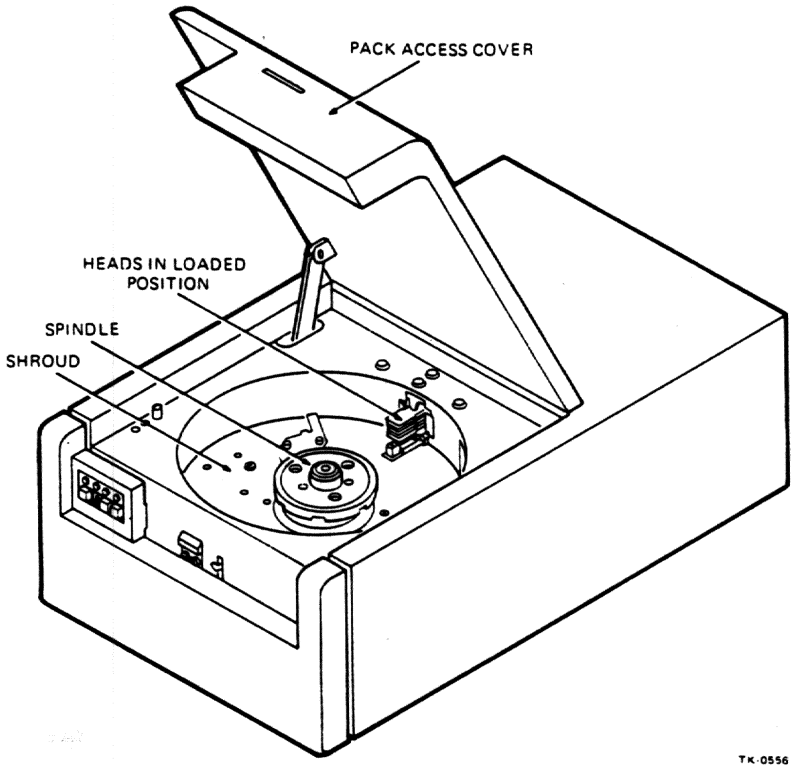


Figure 5-16 Pack Access

3. While holding the disk pack by its handle, carefully set it down on the spindle, avoiding abusive contact between the pack and spindle. Rotate the storage pack handle clockwise until it stops turning. Give one last snugging twist to the handle and lift the storage canister off the pack.

4. Set the storage canister into its bottom dust cover and set aside for later use.

5. Close the pack access cover.

5.4.3.3 Startup Sequence – Once the disk pack is in place and the pack access cover is closed, press the START switch on the control panel to initiate the startup sequence. The START indicator should come on immediately. As the pack comes up to speed, the READY indicator will begin flashing at 1 second intervals. When the READY light finally remains on, the drive is then ready for operation.

5.4.3.4 Disk Pack Removal – Disk pack removal consists of lifting the disk pack off the spindle and storing it in its canister. To accomplish this, follow these simple procedures.

1. Press the START switch to unload the heads and stop the spindle motor. The START light will go off immediately.

2. As the disk pack spins down, the READY light will blink at 1 second intervals. When the READY light is completely off, the pack will be stopped and the pack access cover can be raised. Open the cover and look down between the pack and the shroud to ensure that the heads are fully retracted.

CAUTION

Failure to ensure that heads are fully retracted may damage heads or pack.

3. Place the storage canister over the disk pack so the post protruding from the center of the disk pack is received into the storage canister handle.

4. Rotate the storage canister handle counterclockwise until the pack is free of the spindle as evidenced by a clicking sound.

CAUTION

Avoid abusive contact between disk pack and spindle or one or the other may be damaged.

5. Lift the disk pack out of the drive by the canister handle and close the pack access cover.

6. Install the bottom dust cover on the pack canister by holding the dust cover and turning the pack canister handle clockwise. Exercise caution since excessive force can damage the cover.

5.4.3.5 Removing Subsystem Power – If the subsystem has to be de-energized for some reason, follow these simple instructions.

1. Make sure a disk pack is not left in the drive.
2. Switch off the power supply and the ac power circuit breakers on the drive rear panel.
3. Switch off the power supply and ac power circuit breakers on the adapter power supply.

NOTE

The adapter power supply power cord must be removed from its power source to completely remove all ac voltages. Once this is done, the ac power indicator on the rear of the adapter power supply will go off.

5.4.4 Operator Maintenance

5.4.4.1 Disk Pack Storage – To ensure maximum disk pack life and reliability, observe the following precautions.

- Store disk packs in machine-room atmosphere.

- If the disk pack must be stored in a different environment, allow 2 hours for adjustment to computer environment before use.

- Never store the disk pack in direct sunlight or in a dirty environment.
- Store disk packs flat, not on edge. They may be stacked with similar packs when stored.
- Ensure that both the top and bottom of storage canisters are on disk pack and locked together whenever it is not actually installed in a drive.
- When marking packs, use a pen or felt-tip marker that does not produce loose residue. Never use a lead pencil. Write label before it is applied to disk pack.
- Cleaning of disk surfaces is not recommended. If disks must be cleaned, the cleaning must be done by trained service personnel.
- Do not touch disk surfaces.

5.4.4.2 Disk Pack Handling – Because it is possible for a damaged disk pack to cause damage to the drive heads, or damaged heads to cause damage to a disk pack, it is important that the following be observed.

- Be aware of a sudden increase in error rate related to a specific drive or pack.
- Be aware of any unusual noise, such as screeching or pinging, while heads are loaded on a pack.
- Be aware of a burning odor coming from a drive.
- Be alert to the possibility of contamination such as dust, dirt, grease, oil, or smoke that could accumulate on the pack or heads.

If any of these conditions exist or if there is any doubt about the pack or drive's functional condition, call in trained service personnel.

The operator should keep a copy of the error log printouts on a daily basis.

CAUTION

Do not attempt to operate a suspect disk pack on another drive or operate a new disk pack on a drive with suspect heads. To do so may cause further damage.

5.5 RP05/RP06 DISK DRIVE

The RP05 and RP06 Disk Drives (Figure 5-17) are high-performance, direct-access, single head surface drives designed to enable a data processing system to store and retrieve blocks of data at any location on a rotating disk. The RP05 consists of a 677-51 (RP05) disk drive (manufactured by Memorex) and a device control logic (DCL) unit (manufactured by DIGITAL). The RP06 consists of a 677-01 (RP06) disk drive and a DCL unit (manufactured by DIGITAL). The DCL unit contains the control electronics required to supervise RP06 operations. The disk drive shown in Figure 5-17 is without the DCL.

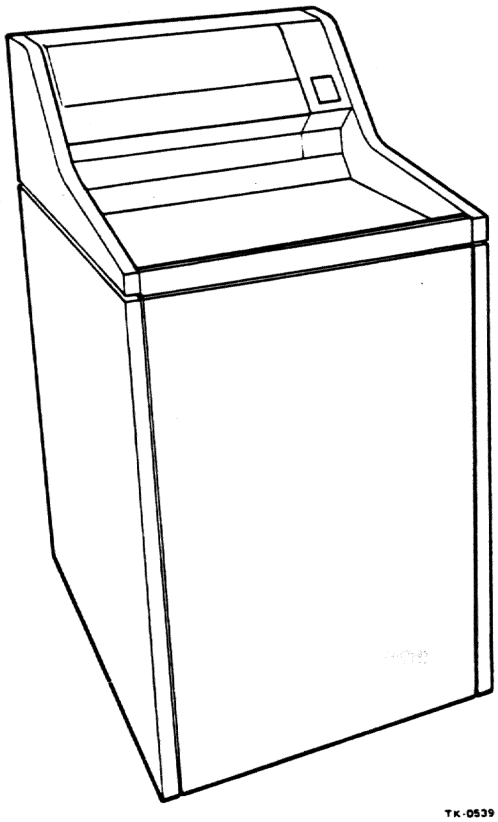


Figure 5-17 RP05/RP06 Disk Drive (without DCL)

The RP05 has a 100 megabyte capacity (using a Mark XI disk pack). The RP06 has a 200 megabyte capacity (using a Mark X disk pack). The RP05 can be upgraded to an RP06 configuration.

5.5.1 RP05/RP06 Specifications

The specifications for the RP05 and RP06 are contained in Table 5-3.

5.5.2 Operator Control and Indicators

5.5.2.1 Operator Controls – All controls and indicators needed for efficient drive operation are located in a single panel designed for use by nontechnical personnel. Names of individual controls (OEM specified) are descriptive of their functions and are printed on the panel. Indicators illuminate white with English words (OEM specified) that identify the condition existing in the drive. One indicator, the word UNSAFE, illuminates when a malfunction is present. The normal response is to position the START/STOP switch to STOP and request maintenance.

Functions of individual controls in the drive's operator panel (Figure 5-18) are described in the following paragraphs.

START/STOP

This is a three-position rocker switch. When pressed to the START position and released, the switch returns to its center position. When pressed to the STOP position, it remains in the STOP position. With the pack access door closed and all interlocks satisfied, pressing the switch to START locks the disk pack access door and initiates a drive power-up sequence. When the disk pack reaches operating speed, the heads are extended and the disk can be loaded. When the switch is pressed to STOP, the carriage moves in a reverse direction to pull the heads out of the pack area. Electrodynamic braking power is applied to stop disk pack rotation, and when stopped the disk pack access door is unlocked. Starting or stopping time is approximately 20 seconds. When pressed to the START position, STANDBY will be reset if it was true. If left in the center position and sequence conditions are satisfied, the drive will initiate a power-up sequence when ac power is applied or restored.

WRITE PROTECT

With this two-position rocker switch in the WRITE PROTECT position, data stored on a disk pack is protected against accidental overwrite. The normal position is off. With the switch in the WRITE PROTECT position, the write function of the drive is inhibited, providing the drive was not writing at the time the switch was engaged. If the switch is engaged while the drive is writing, the inhibit function will operate after the drive stops writing. With the switch in the off position, the reading or writing function of the drive is permitted. If both functions are commanded simultaneously, heads are deselected and both reading and writing are disabled.

Table 5-3 RP05/RP06 Specifications

Data Retrieval Times	
Average latency time	8.33 ms
Maximum access time, track to track	6 ms
Maximum access time, track 000 to track 814	53 ms
Average access time	28.5 ms
Nominal data transfer rate	806,000 bytes/second
Disk Pack Characteristics	
Number of recording disks	10
Number of recording surfaces	19
Track density (677-01) – RP06	370 tracks/in
Track density (677-51) – PP05	192 tracks/in
Tracks/surface (677-01) – RP06	815
Tracks/surface (677-51) – RP05	411
Approximate weight	9.1 kg (20 lb)
Information Storage Capacities	
Track capacity	13,440 bytes
Cylinder capacity (bytes available to a single access)	255,360 bytes
Disk pack capacity (including alternate cylinders)	208,118,400 bytes, (677-01 drive)
	104,952,960 bytes, (677-51 drive)
Dimensions and Weight	
Width (excluding DCL attachment)	55.8 cm (22 in)
Depth	81.3 cm (32 in)
Overall height	120 cm (47 in)
Pack access height (top of sliding door)	89 cm (35 in)
Weight	250 kg (550 lb)
Start/Stop Time	20 seconds
Power Requirements	
Model A:	
Frequency	60 ± 1.0 Hz
Voltage	208/230/240 Vac ± 10%
Phase and configuration	3-phase + ground, Delta
Model B:	
Frequency	50 ± 1.0 Hz
Voltage	220/380/398/416 Vac ± 10%
Phase and configuration, 220 Vac	3-phase + ground, Delta
Phase and configuration, 380/398/416 Vac	3-phase + neutral, Wye

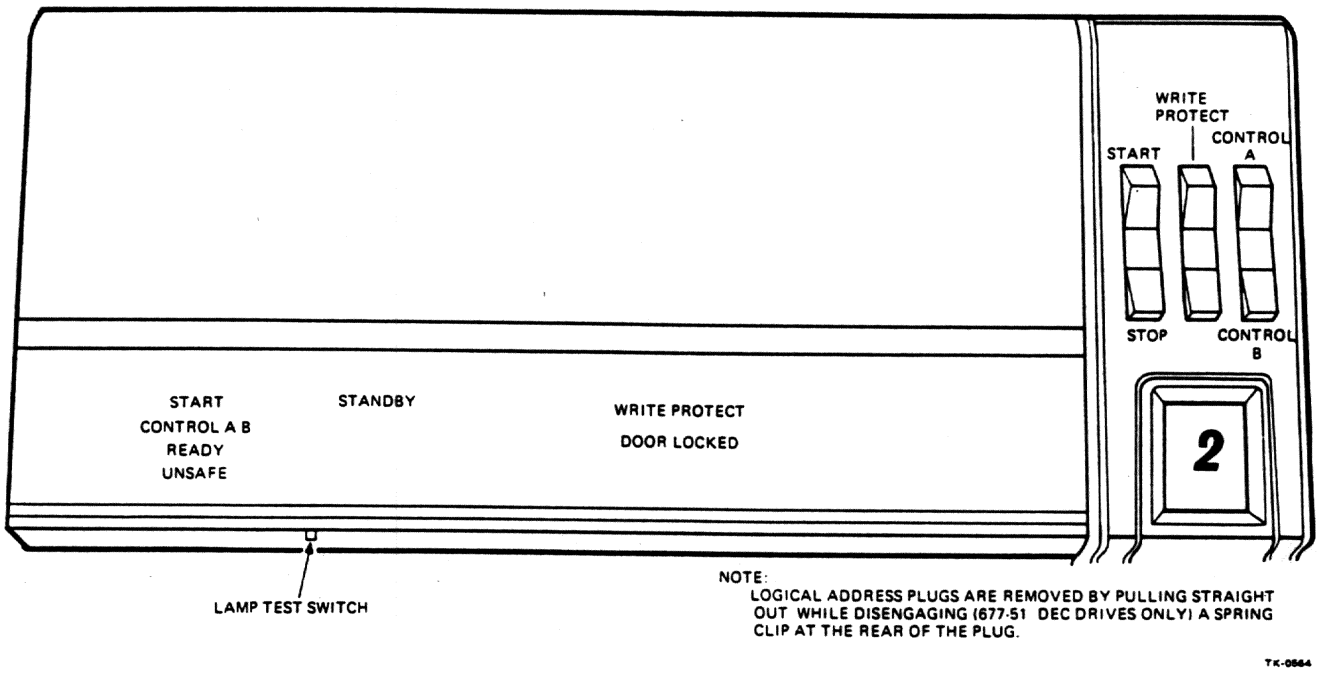


Figure 5-18 RP05/RP06 Operator Control Panel

CONTROL A B
CONTROL A forces the port A locked line on the MDLI to a true level. CONTROL B forces the port B locked line to a true level. In the center position, both lines are false.

Logical Address Plug
This removable plug permits changing the logical address of a drive by simply changing the plug. Recabling is not necessary. Plugs are factory configured to designate the individual drives in the system (maximum of nine plugs including one service plug). Plugs are labeled to indicate drive designations ("0" through "7" and "S"). Two identical plugs are not used on drives attached to the same controller.

Lamp Test
This momentary pushbutton switch provides illumination of all panel indicators for checking purposes.

5.5.2.2 Operator Indicators – Functions of individual indicators in the drive's operator panel (Figure 5-18) are described in the following paragraphs.

START
This indicator illuminates when the START/STOP switch is set to START and all start conditions are satisfied; it remains illuminated until the drive is stopped.

CONTROL A/B
This indicator and letters A and B illuminate to indicate the state of the port A lamp and port B lamp lines.

READY
This indicator illuminates when the drive is on-line.

UNSAFE
This indicator illuminates to indicate a drive malfunction that requires the attention of service personnel. Illumination occurs when either an abnormal-stop or read/write safety error occurs.

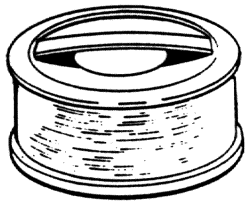
STANDBY
This indicator illuminates to indicate the drive is in the standby mode. Illumination appears when a command to place the drive in standby mode is implemented in the drive circuitry. When in the standby mode, starting the drive is accomplished by setting the START/STOP switch to START, which starts the drive provided: conditions permitting the initiation of a power-up sequence exist while no other drive is in process of powering up, and a disk pack is installed and the access door closed. With these starting conditions satisfied, pressing START causes STANDBY to disappear.

WRITE PROTECT
This indicator illuminates when the WRITE PROTECT switch is in the WRITE PROTECT position.

DOOR LOCKED
This indicator illuminates to indicate that the disk pack access door is seated and locked closed. For personnel safety, pack rotation can begin only when this indicator is illuminated. If the door is forced open during drive operation, pack rotation will commence stopping and the heads will be retracted out of the pack automatically. Normal sequence down will take place under this condition.

5.5.3 Operating Procedures

5.5.3.1 Handling the Disk Pack – The Memorex disk pack (Figure 5-19) is protected during shipping by a special plastic foam container. When the pack is received, examine the container for damage. If the condition appears acceptable, remove the pack and store the container for later use. If the container or pack is damaged, retain the combination in an "as received" condition and notify Memorex.



TK-0547

Figure 5-19 Mark X or XI Disk Pack

A two-piece plastic cover protects the disk pack from dust and moisture when the pack is stored. The top section of the cover contains a handle for carrying the pack. This top cover is never left on a pack that is installed into a drive.

When transporting disk packs:

1. Be sure the pack is securely fastened in its two-piece cover.

2. Use only the specially designed shipping container.

3. Handle the pack only with its top cover on. If the pack is accidentally dropped or receives a sharp impact of any kind, have it inspected by service personnel before using.

For identification purposes, a plastic labeling surface is placed on the center area of the disk pack. When labeling:

1. Use a pen or felt-tip marker that does not produce residue. DO NOT use a lead pencil. Microscopic lead particles can damage disk surfaces and heads.

2. Write on the label before it is applied to the disk pack.

3. Place the label only on the center area.

4. Use a new label if changes are necessary. NEVER ERASE a label that is on a pack. Microscopic eraser particles can damage surfaces and heads.

5. Removal or placement of labels can be done only when the pack is installed. DO NOT attempt to remove the top cover when the pack is not installed on the drive's spindle.

To ensure maximum disk pack life and reliability:

1. Each pack should rest flat on a shelf when storing, not on edge or another pack.

2. Store in a computer room environment. If a pack must be stored in a different environment, allow two hours for temperature adjustment within the computer room before using.

5.5.3.2 Disk Pack Installation and Removal – A disk pack can be installed when main power to the drive is off, or provided the spindle is stopped.

1. If the spindle in the drive is stopped (both START and READY are extinguished), go directly to Step 4; otherwise, perform Steps 2 and 3 before Step 4.

2. Press the START/STOP switch to the STOP position. The START indicator extinguishes.

3. Wait approximately 20 seconds, until the DOOR LOCKED indicator extinguishes. This indicates that the spindle dynamic braking is complete and the disk pack access door is unlocked.

4. Open the glass access door by sliding it all the way to the rear of the drive.

5. Remove the pack (if present) by performing Steps 14 through 16.

6. Remove the bottom cover of the disk pack to be installed by pressing the two handles on the bottom cover together.

7. Place the disk pack, together with its top cover, on the spindle carefully and slowly.

8. Turn the handle on the top cover in a clockwise direction until it comes to a full stop.

9. Carefully lift the top cover straight up from the pack to avoid hitting the edges of the disks.

10. Place the top cover on the bottom cover to create a positive dust seal, and store.

NOTE

Without the disk pack inside, the top and bottom covers are not attachable.

11. Close the glass access door by sliding it all the way to the front of the drive. The access door should always be left closed to prevent contamination.

12. To remove a disk pack, perform Steps 13 through 17.

13. Make sure the spindle is stopped. Open the glass access door by sliding it all the way to the rear of the drive.

14. Carefully lower the top cover straight down over the pack to avoid hitting the edges of the disks. Turn the handle on the top cover in a counterclockwise direction two full turns.

15. Using the handle, remove the pack from the drive.

16. Immediately attach the bottom cover to the pack and store.

17. Close the glass access door by sliding it all the way to the front of the drive. The access door should always be left closed to prevent contamination.

5.5.3.3 Drive Address Assignment

1. The logical address plug determines the drive's assigned address.

2. To change the address, verify that the system is idle before removing the logical address plug.

3. Remove the logical address plug (Figure 5-20) from the drive's operator panel.

4. Insert the appropriately numbered logical address plug into the panel's plug socket by simply pushing straight in.

5.5.3.4 Spindle Motor Start/Stop – To start the drive spindle motor, press the START/STOP switch to START. The drive will sequence to the ready state approximately 20 seconds later, at which time the word READY illuminates. The information that follows covers both the actual starting procedure and the prerequisites to starting.

1. Be sure a logical address plug is installed, a disk pack is installed, and the glass access door is closed.

2. Verify all indicators light by pushing the LAMP TEST switch. This also indicates main power is applied to the drive. If an unsafe condition has been detected (UNSAFE is illuminated), go to Paragraph 2.4.5.1 to clear this condition; if cleared, go to Step 3. If the unsafe condition persists, request maintenance.

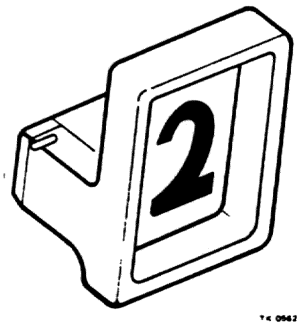


Figure 5-20 Logical Address Plug

3. Press the START/STOP switch to the START position. When pressed, the following events occur: START illuminates, pack begins rotating, and DOOR LOCKED illuminates. If DOOR LOCKED does not illuminate, close the door fully.

4. Wait approximately 20 seconds until the word READY illuminates. It indicates the drive is started and ready to execute commands. START and DOOR LOCKED will remain illuminated.

To stop the drive spindle motor: press the START/STOP switch to STOP. When pressed, the READY and START lamps are extinguished and the drive heads retract. The DOOR LOCKED illumination disappears approximately 20 seconds later when the motor-down sequence is completed.

NOTES

a. **When the START/STOP switch is set to STOP, dynamic braking power is applied to the spindle motor. Approximately 20 seconds later the door is unlocked (and the DOOR LOCKED lamp is extinguished) to allow pack removal.**

b. **The system can stop the drive motor by issuing a command to place the drive in standby mode. Whether stopped by system or operator, the resulting status of indicators is the same with the exception that the system illuminates STANDBY.**

5.5.4 User Responses to Abnormal Conditions

5.5.4.1 Clearing an Unsafe Condition – If a sequence malfunction occurs during a start operation, an unsafe condition will occur (UNSAFE is illuminated) and the drive will automatically perform an abnormal stop sequence. At the end of the stop sequence, the spindle should come to a complete stop. To restart the drive, clear the unsafe condition by pressing START. UNSAFE should disappear; if not, service personnel should be advised that the drive cannot be restarted. If UNSAFE disappears, the drive will then perform its normal starting sequence. If UNSAFE reoccurs, the abnormal stop sequence will perform again automatically; in this event, press STOP and advise service personnel.

NOTE

Anytime the stop sequence malfunctions, as indicated when the spindle does not stop at the end of the sequence, advise service personnel and do not attempt other operations. If it is necessary to remove ac power from the drive, first manually unload the heads.

5.5.4.2 Removing a Pack with No Drive Power – To remove a pack from a drive with no ac power applied perform the following steps.

1. Verify that the spindle is stopped and the heads are retracted.

2. Pull open the drive's front cover. Referring to Figure 5-21, locate the door lock override mechanism: it is an arm protruding from the door lock solenoid. Press the mechanism downward: while holding down, push the door and start sliding it toward the rear. Release the mechanism and push the access door all the way back.

3. Remove the pack using the normal pack removal procedure (Paragraph 5.5.4.2, Steps 13 through 17).

4. Close the drive's front cover.

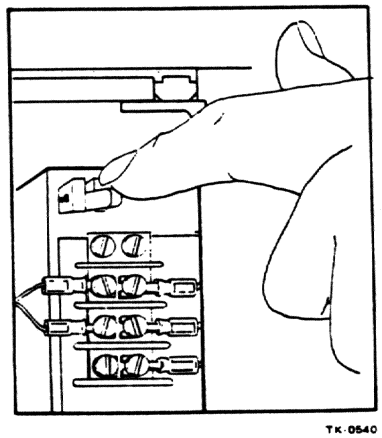


Figure 5-21 Door Lock Override Mechanism

5.5.4.3 Detecting Head-to-Disk Interference (HDI) – HDI results from head contact with a disk surface. Usually a foreign particle in the air stream or a protrusion from the disk surface causes the head to break through the air "bearing" and abrade the disk surface. If the problem is not totally corrected, it will have a propagation effect from pack to pack and, in turn, drive to drive. Try to recognize the following symptoms of HDI:

1. Sudden hard read errors.

2. Black contamination on flying surface of any head.

3. UNSAFE during a write operation.

4. Uncommon noise from the disk, characterized by audible tinkling, zinging, or scratching sounds. If allowed to continue, the noise will progress to a screech.

If any symptom above is exhibited:

1. Stop the drive immediately and contact service personnel.

2. Unless absolutely necessary, do not remove the suspected pack from the drive.

CAUTION

a. **If the suspected pack is replaced with another pack and the drive operated, or the suspected pack is used in another drive, damage to either the second drive or the substituted pack will occur.**

b. **All packs and drives being used when HDI symptoms are exhibited must be checked for HDI by service personnel.**

5.6 RK611/RK06 DISK SUBSYSTEM

The basic RK611/RK06 Disk Subsystem consists of an RK611 controller and from one to eight RK06 moving-head disk drives that are connected to the controller by a daisy-chain bus (Figure 5-22). The RK06 is a moving-head disk drive that functions as a random access mass storage device. A dual-access option for the disk drive permits a configuration whereby each drive on the daisy chain is accessible by one of two controllers. The two controllers can be connected to the same Unibus or two different ones.

Although cleanliness is important in all facets of a computer system, it is particularly crucial in the case of a device such as the RK06 Unibus Disk Subsystem. Disk cartridges are not sealed units, while loading, and are extremely vulnerable to dirt. Even such minute obstructions as smoke particles, fingerprint smudges, or dust specks can cause head crashes and catastrophic destruction of heads and/or disk surfaces.

The RK06 is capable of operating in an ambient atmosphere containing not more than one million particles of 0.5 micron or larger per cubic foot of air.

5.6.1 Specifications

The specifications for the RK06 disk drive are contained in Table 5-4.

5.6.2 Operator Controls/Indicators

The RK06 disk drive contains the following power and control switches and indicators (Figure 5-23).

Rear Panel (Circuit Breaker)

- AC power ON/OFF switch with POWER indicator.

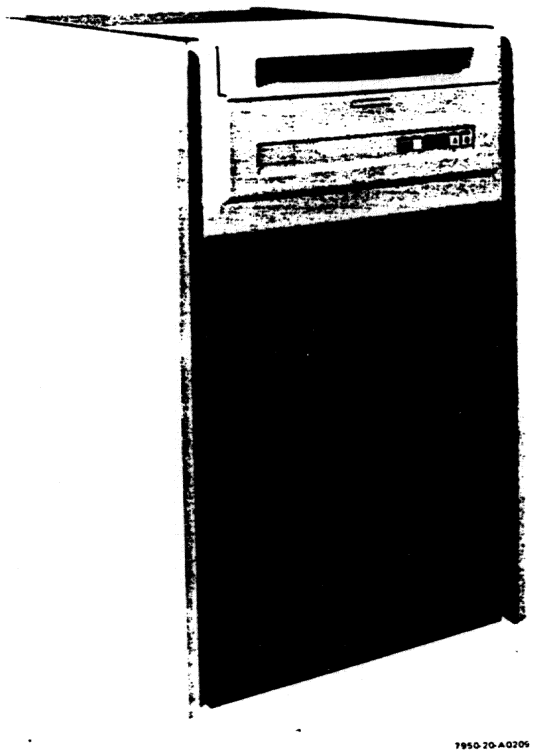


Figure 5-22 RK611/RK06 Disk Drive Console

Table 5-4 RK06 Disk Specifications

Characteristic	Specification	
Seek Times (max)		
Maximum seek (411 cylinders)	75 ms	
One cylinder	8 ms	
Average	38 ms	
Start/stop times	**Maximum**	**Nominal**
Start	60 sec	30 sec
Stop	60 sec	30 sec
Electrical		
Voltages available	90–132 Vac @ 60 ± 0.5 Hz 180–264 Vac @ 60 ± 0.5 Hz 90–132 Vac @ 50 ± 0.5 Hz 180–264 Vac @ 50 ± 0.5 Hz	
Start current		
Low voltage range	10.5 A rms max	
High voltage range	4.5 A rms max	
Power factor	0.80 min	
Input power	450 W nominal, 500 W max @ 60 Hz 500 W nominal, 550 W max @ 50 Hz	

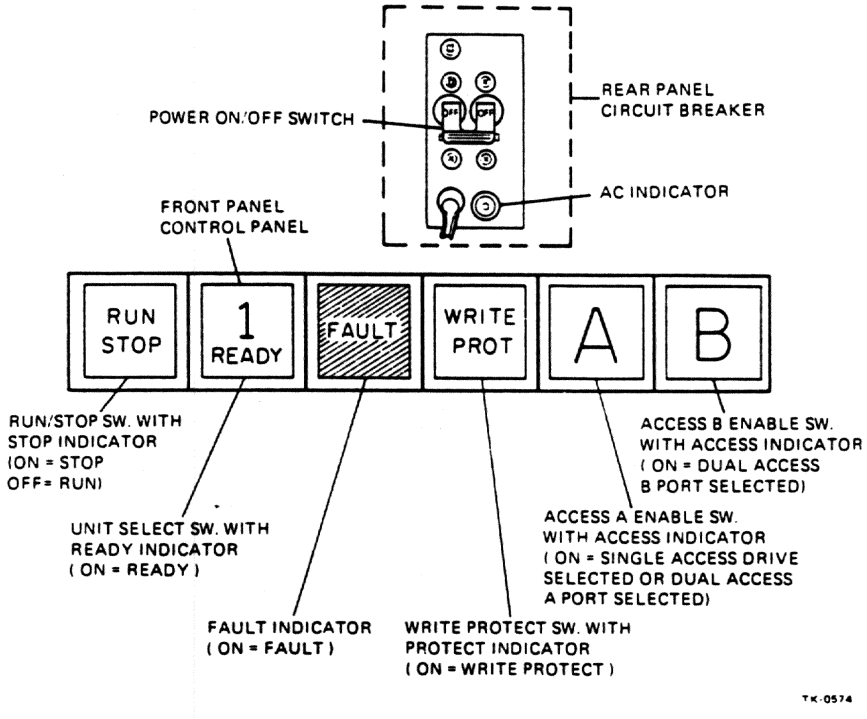


Figure 5-23 RK06 Disk Drive Power and Control Switches

Front Panel (Control Panel):

- Combination RUN/STOP switch with a STOP indicator
- UNIT SELECT switch with a READY indicator
- FAULT indicator
- WRITE PROT switch with a PROTECT indicator
- ACCESS A ENABLE switch with a SELECT A indicator
- ACCESS B ENABLE switch with a SELECT B indicator

Power ON/OFF Circuit Breaker
When the 3-wire plug is inserted into an outlet, ac power is applied to the rear panel circuit breaker on an RK06 disk drive, and the red indicator lamp is illuminated. When the circuit breaker is switched to the ON position, ac power is applied to the drive and the blower motor is energized.

RUN/STOP Switch with STOP Indicator
The STOP indicator is illuminated during STOP conditions when the heads and brushes are home, the spindle is completely stopped, and the spindle motor is not energized. Under these conditions, when the alternating action pushbutton is initially pressed (RUN), the spindle motor is energized, the STOP indicator is extinguished, and the heads are loaded. When these initial operational conditions are met (STOP indicator extinguished and heads loading), a RUN condition is entered. However, if the heads do not load (due to a malfunction), the spindle motor is immediately deenergized and the STOP indicator remains illuminated.

If the pushbutton is pressed (STOP) when the heads are loaded, the heads will be unloaded and the spindle motor deenergized. When these conditions are met (heads home and spindle stopped), the STOP indicator is again illuminated.

If the RUN/STOP switch is in the RUN position and the drive has been cycled down (heads unloaded and spindle completely stopped), due to an unload command, a start spindle command from the controller can be used to restart the spindle and reload the heads. However, if the cycle down condition occurred as the result of an error, a start spindle command is not required, since the heads will immediately load when the error is cleared.

Finally, since the RUN/STOP switch has a mechanical memory, a power interrupt followed by power restoration will cause the drive to automatically cycle up if the switch is currently located in the RUN position.

NOTE

The drive cover (lid) is interlocked. This prevents the lid from being raised when the STOP light is extinguished (RUN) or drive power is off.

UNIT SELECT Switch with READY Indicator
The UNIT SELECT switch is a cam-operated assembly that is actuated by the insertion of a plastic plug. The face side of each actuator plug contains a single number (0–7), while the other side consists of raised plastic cam actuators that are separately configured to encode the plug with the number. With this arrangement, the insertion of a plug into any UNIT SELECT switch assembly will automatically encode the associated plug number when power is applied to the drive. When the heads of a drive are settled (not in motion), the numbered indicator on the UNIT SELECT switch assembly associated with the drive is illuminated (READY).

FAULT Indicator
The FAULT indicator is illuminated whenever any one of the following error conditions is detected in the drive:

- More than one drive selected.
- Positioner, when detented, has moved too far (e.g., the drive has been jarred).
- A parity error occurring in a message transmitted from the controller to the drive.
- A read/write unsafe condition in the drive (e.g., servo track signal errors, write gate signal errors, etc.)
- A write lock error condition (i.e., the receipt of a write gate signal when the drive is in write protect mode).
- Low ac voltage in the drive.
- An incomplete seek operation.
- The receipt of write gate or seek signals while volume valid is reset.

WRITE PROT Switch with PROTECT Indicator
If a write gate signal from the controller is not currently asserted in a selected drive, initial depression of the WRITE PROT pushbutton will immediately set the drive in write protect mode, and the PROTECT indicator will be illuminated. When the alternating action pushbutton is again pressed, write protection is removed and the PROTECT indicator is extinguished. However, if a write gate signal is asserted in the selected drive when the switch is initially pressed, write protection will be inhibited and the PROTECT indicator will remain extinguished until the write gate signal is negated. Once the write gate signal is negated, the write protect mode is entered and the PROTECT indicator is illuminated.

ACCESS A ENABLE Switch with SELECT Indicator
The ACCESS A ENABLE switch is an alternating action pushbutton that is found on both single-access drives (port A or port B) and drives containing the dual-access option (port A and port B).

Single-Access Drive (A)
If port A is configured for single access and the switch is initially pressed, bidirectional communication with the controller is enabled via port A. Under these conditions, when the drive is selected by the controller, the SELECT A indicator is illuminated. When the switch is again pressed (released), access by the controller is disabled and the SELECT A lamp cannot be illuminated.

Dual-Access Drive (A and B)
If the ACCESS A ENABLE switch is independently pressed on a dual-access drive, the operations performed are similar to those previously described. However, if both the ACCESS A and ACCESS B switches are initially pressed, arbitration logic will determine, on a priority basis, which port (A or B) will be accessed by its associated controller (A and B).

ACCESS B ENABLE Switch with SELECT Indicator
The ACCESS B ENABLE switch is an alternating action pushbutton that is found on both single-access drives (port A or B) and drives containing the dual-access option (port A and B).

Single-Access Drive (B)
If the B port is configured for single access and the switch is initially pressed, bidirectional communication with the controller is enabled via port B. Under these conditions, when the drive is selected by the controller, the SELECT B indicator is illuminated. When the switch is again pressed (released), access by the controller is disabled and the SELECT B lamp cannot be illuminated.

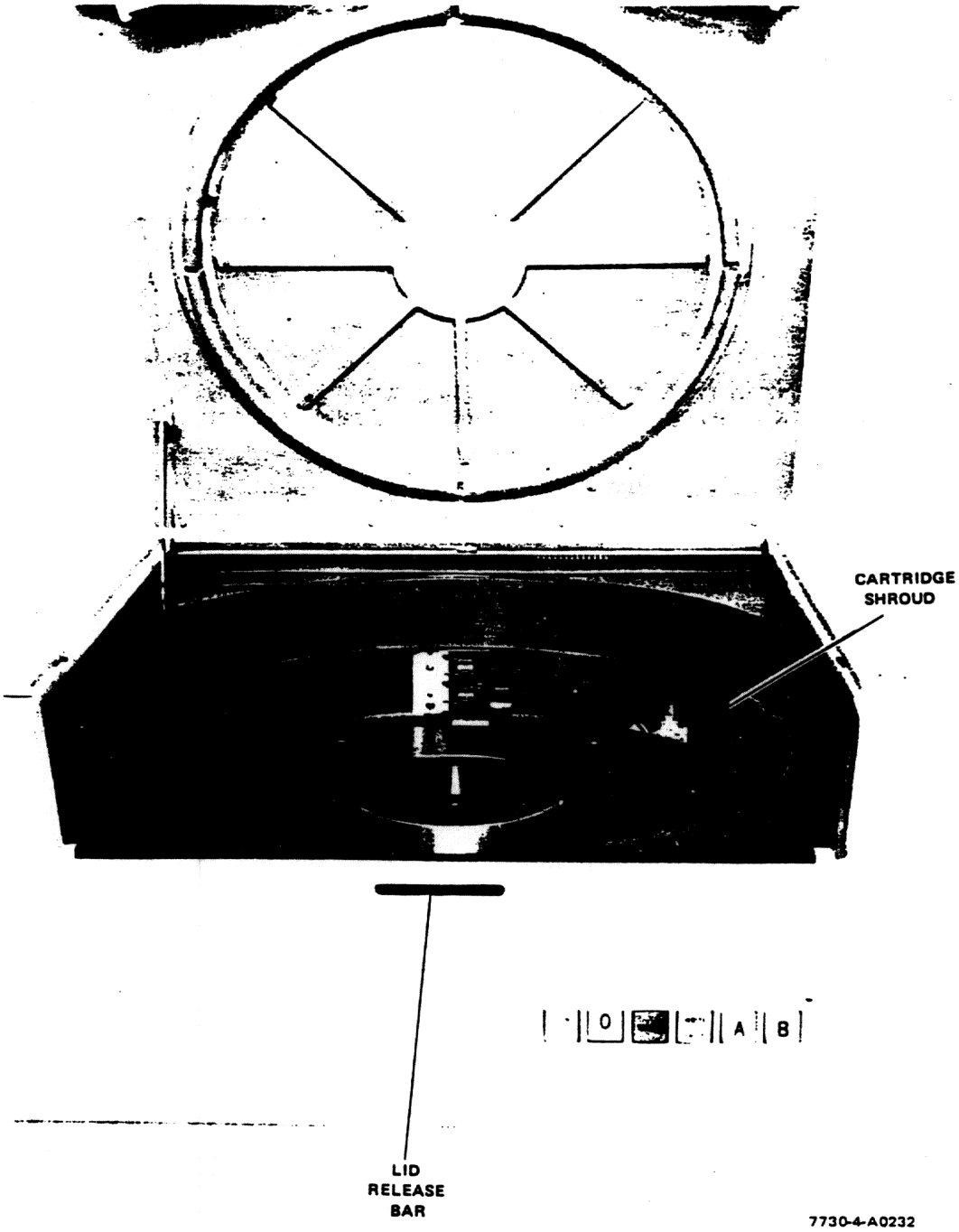
Dual-Access Drive (A and B)
If the ACCESS B ENABLE switch is independently pressed on a dual-access drive, the operations performed are similar to those previously described. However, if both the ACCESS B and ACCESS A switches are initially pressed, arbitration logic will determine, on a priority basis, which port (B or A) will be accessed by its associated controller (B or A).

5.6.3 Operating Procedures

This material describes RK06K cartridge loading and cycle-up procedures that are required to place an RK06 disk drive on-line. The procedure assumes that ac power is available (red indicator lamp on the rear panel circuit breaker is on), the ac circuit breaker is on (blower motor is energized), and the STOP indicator lamp is illuminated on the control panel. If these conditions are met, initiate the following.

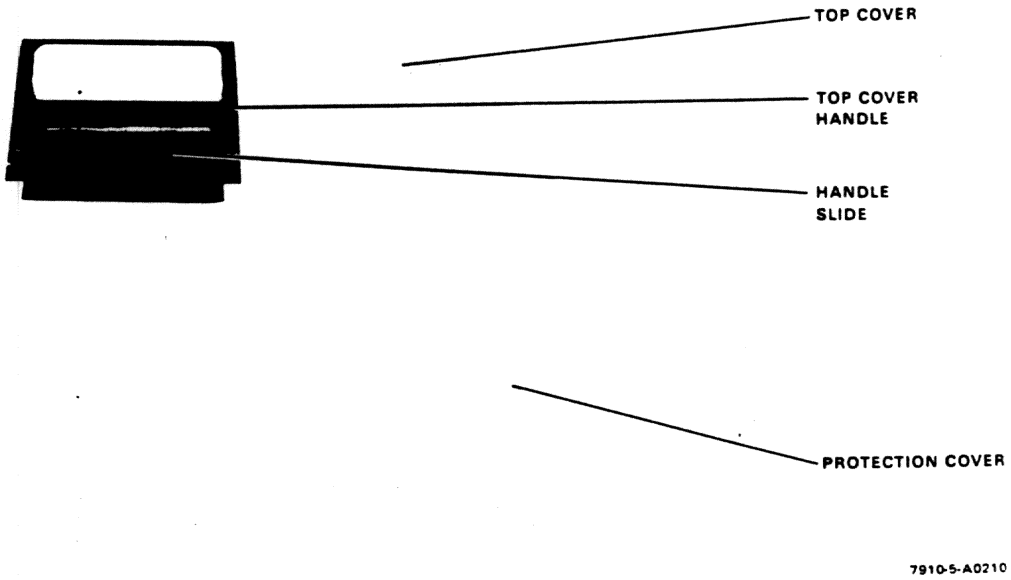
5.6.3.1 RK06K Cartridge Loading

1. Press the drive lid release bar and raise the lid (Figure 5-24).
2. Prepare an RK06K cartridge for loading as follows:
   a. Lift the cartridge by grasping the top cover handle with the right hand.
   b. Support the cartridge with the left hand by holding the protection cover handle.
   c. Lower the top cover handle and push the handle slide to the left with the thumb of the right hand. Again raise the handle to its full upright position to release the protection cover.
3. Lift the cartridge from the protection cover and place it into the drive shroud with the top cover handle recess facing the rear of the machine.
4. Rotate the top cover handle a few degrees clockwise and counterclockwise to ensure that the shroud locating studs are properly seated within the cartridge housing detent slots.



a. RK06 Disk Drive with Lid Released

Figure 5-24 Loading a Disk Cartridge (Sheet 1 of 2)



b. RK06-K Disk Cartridge

Figure 5-24 Loading a Disk Cartridge (Sheet 2 of 2)

5. Gently lower the top cover handle to a horizontal position to engage the drive spindle, and place the protection cover over the top cover.

6. Close the lid.

7. Press one or both of the ACCESS A and B ENABLE switches, depending on the port capability.

8. If write protection is required, press the WRITE PROT switch.

9. Press the RUN/STOP switch (RUN). If the lid is properly closed and no drive errors exist (FAULT extinguished), the spindle will turn and the STOP indicator will be extinguished in approximately 1 second.

When the drive has completed the start spindle sequence and the heads are detented on cylinder 0, the READY indicator on the numbered UNIT SELECT switch will be illuminated.

5.6.3.2 RK06K Cartridge Unloading

1. Press the RUN/STOP switch (STOP).

2. Wait for the STOP indicator to illuminate.

3. Press the lid release bar and raise the lid.

4. Remove the RK06K cartridge as follows:

a. Remove the protection cover and hold the cover in the left hand.

b. Push the handle slide to the left before raising the top cover handle.

c. Raise the top cover handle to a full upright position to release the cartridge from the drive spindle.

d. Lift the cartridge up and out of the shroud, and place it in the protection cover.

e. Lower the top cover handle to the horizontal position to lock the protection cover in place.

5.7 TE16 MAGNETIC TAPE SUBSYSTEM

The TE16 DECmagtape Transport (Figure 5-25) is an industry-compatible 9- or 7-track tape transport capable of reading and writing on 1.27 cm (0.5 in) magnetic tape at 1600 char/in (PE) and 800 char/in (NRZI). Tape density and tape character format are program selectable. Forward/reverse tape speed is 1.14 m/s (45 in/s), while rewind is performed at 3.8 m/s (150 in/s). The tape transport consists of a tape deck with associated reel and capstan motors: H607 motor driver module, transformer and capacitor assembly; a logic box that integrates the read/write electronics, the control electronics, and the power supply regulator; and a vacuum assembly. An off-line test function generator (TFG) module is included in the control package, which allows some maintenance procedures to be completed off-line. The TE16 is outfitted with cable connector modules that interface directly to the TM02. Up to eight (8) TE16 transports can be interfaced to a TM02.

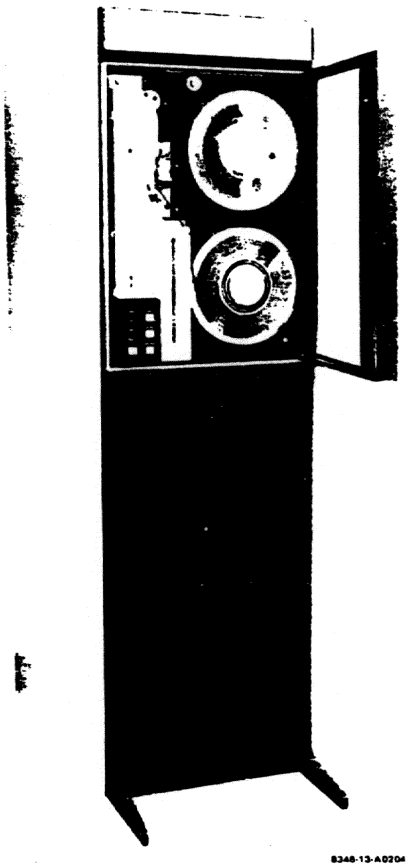


Figure 5-25 TE16 DECmagtape System

5.7.1 Transport Specifications

Table 5-5 contains the TE16 DECmagtape transport specifications.

Table 5-5 TE16 Specifications

Category	Parameter	Specification
Main Specifications	Storage Medium	1.27 cm (1/2 in) wide magnetic tape (industry compatible)
	Capacity/tape reel	23 million characters (NRZI); 46 million characters (PE)
	Data transfer rate	36,000 char/s (NRZI); 72,000 char/s (PE)
	Transports/formatter, max	8
Mechanical	Tape transport mounting,	Mounts on slides in a standard 48.3 cm (19 in) cabinet
	TE16 transport	Without cabinet [with H950 cabinet]
	Depth	0.64 m (25 in) [0.76 m (30 in)]
	Width	0.48 m (25 in) [0.53 m (21 in)]
	Height	0.66 m (72 in) [0.83 m (72 in)]
	Weight	70 kg (150 lb) [204 kg (450 lb)]
	861 Power Controller	
	Depth	0.20 m (8 in)
	Width	0.48 m (19 in)
	Height	0.13 m (5 in)
	Weight	4.54 kg (10 lb)
Power	Input current	8 A @ 115 V; 6 A @ 230 V
	Input power	920 VA; 1380 VA
	Voltage	115/230 ±10%
	Frequency	47–63 Hz; single phase

5.7.2 Controls and Indicators

The operator control panel (Figure 5-26) is located at the lower left-hand corner. The functions of the control box switches and indicators are listed in Table 5-6. All operator controls are momentary-contact switches. Indicators are yellow and red.

5.7.2.1 Address Selection Plug Receptacle – This receptacle accepts plugs labeled 0 through 7, which define the logical address of the device. If no plug is in the receptacle, the transport cannot be selected. This is not an indicating plug.

5.7.2.2 Maintenance Aids – Two switches are located underneath the logic assembly to assist in TE16 maintenance. Both switches are deactivated when the transport is on-line.

1. FWD/REV – The FWD/REV switch determines the direction of tape motion. It does not, however, initiate tape motion.

NOTE

Off-line tape motion may be initiated by the START/STOP switch, or by the TFG module.

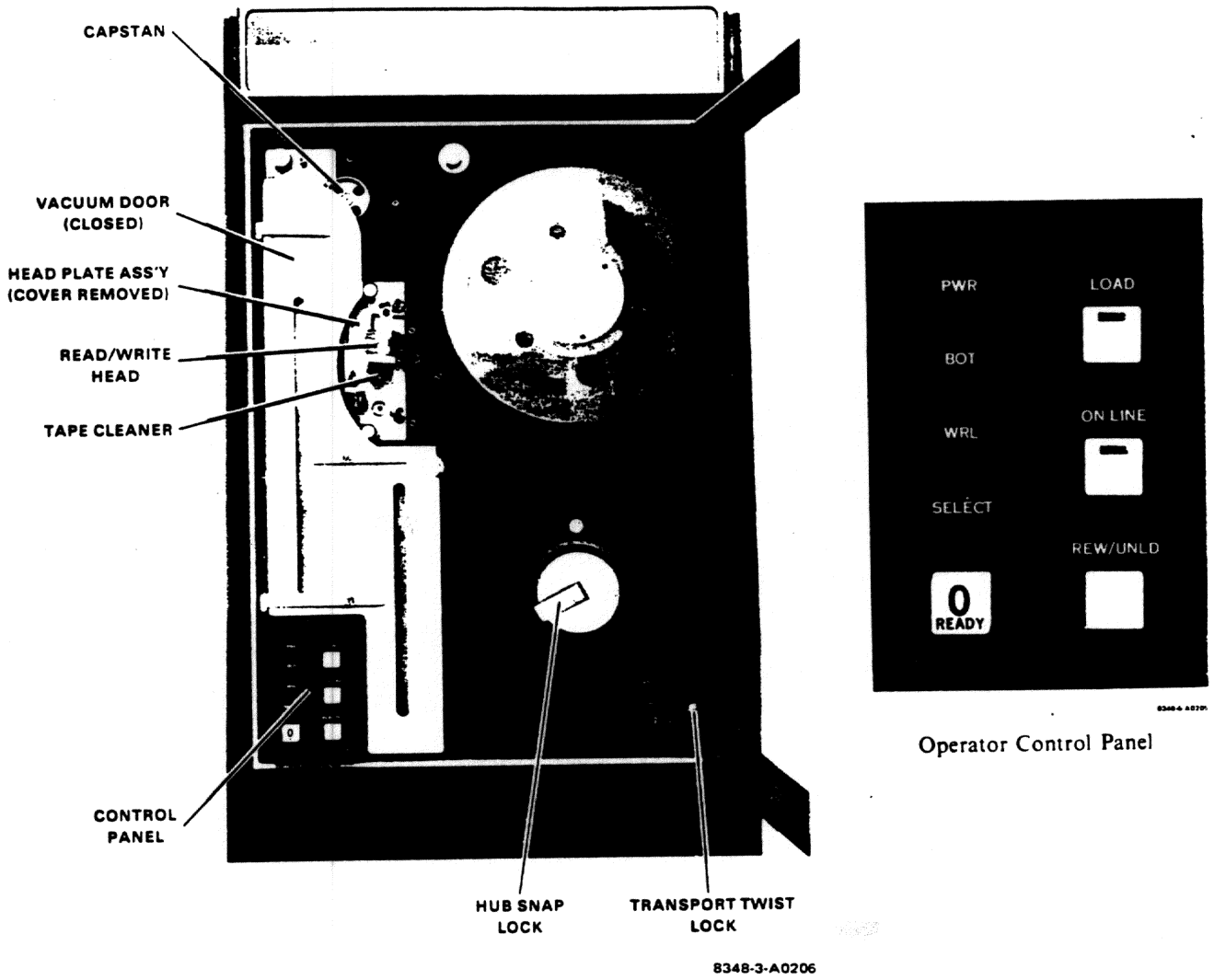


Figure 5-26 Operator Control, TE16

2. START/STOP – This switch initiates tape motion when moved from the STOP to the START position. It halts tape motion when switched from the START to the STOP position.

5.7.3 Operating Procedures

TE16 operating procedures are described in the following paragraphs.

5.7.3.1 Application of Power

1. If the 861 power controller REMOTE ON/OFF/LOCAL ON switch is in the REMOTE ON position, TE16 power is controlled by the processor POWER key switch. This method is used in normal operation.

2. If the processor POWER key switch is not activated, TE16 power may be turned on locally by setting the 861 power controller REMOTE ON/OFF/LOCAL ON switch to LOCAL ON. This method may be used during maintenance.

Table 5-6 TE16 Switch Functions/Indicator

Switch/Function	Function
LOAD	Pushing this switch (when tape is correctly threaded) causes tape to be pulled into the vacuum columns and the transport to seek the BOT marker and go on-line when it is at rest at BOT. The indicator on this switch glows when tape is in the vacuum columns and proper vacuum has been established. **NOTE** **A seek-for-BOT sequence consists of 5 seconds of forward motion, followed by a rewind to BOT.**
ON LINE	Pushing this switch causes the transport to make a transition between the on-line and off-line states. For example, pushing the switch when the drive is off-line causes it to go on-line. This switch is inoperative when the LOAD indicator is off and no LOAD sequence is happening. The indicator on this switch glows when the transport is on-line.
REWIND/UNLOAD	This switch is inoperative when the transport is on-line. Pushing this switch when the transport is off-line and not at BOT causes the drive to execute a high-speed rewind to BOT. Pushing this switch when the transport is off-line and at BOT causes the transport to execute an UNLOAD sequence, gently winding all tape onto the lower reel. This is not an indicating switch. **NOTE** **Pushing this switch while the transport is executing a LOAD sequence (LOAD indicator on) and seeking forward for the BOT marker, aborts the LOAD sequence and causes a normal off-line rewind procedure. Never push this switch before the BOT marker is sensed.**
POWER	This indicator glows when ac power has been applied to the device. It is powered by the +5 V regulated supply.
BOT	This indicator glows when the BOT marker is positioned over its sensor.
WRL	This RED indicator glows when either no tape reel is mounted, or a reel is mounted without a write-enable ring.
SELECT	This indicator glows when the drive is both on-line and selected.

5.7.3.2 Loading and Threading Tape – Use the following procedure to mount and thread magnetic tape.

1. Apply power to the transport. Ensure that the transport is off-line (the ON LINE switch does not glow).

2. Place a write-enable ring in the tape reel groove if data is to be written on the tape. Ensure that there is no ring in the groove if data on the tape is not to be erased or written over.

3. Mount the file reel onto the lower hub in the following manner: release the snap-lock lever on the lower hub by pulling it firmly outward on the rim end. Then, with the reel groove facing away from the operator, mount the file reel onto the lower hub. When the
4. Manually unwind tape from the file reel and thread the tape by the tape guides and head assembly as shown in Figure 5-27.

5. Wind about four turns of tape onto the take-up reel. Ensure that the tape is in the guides.

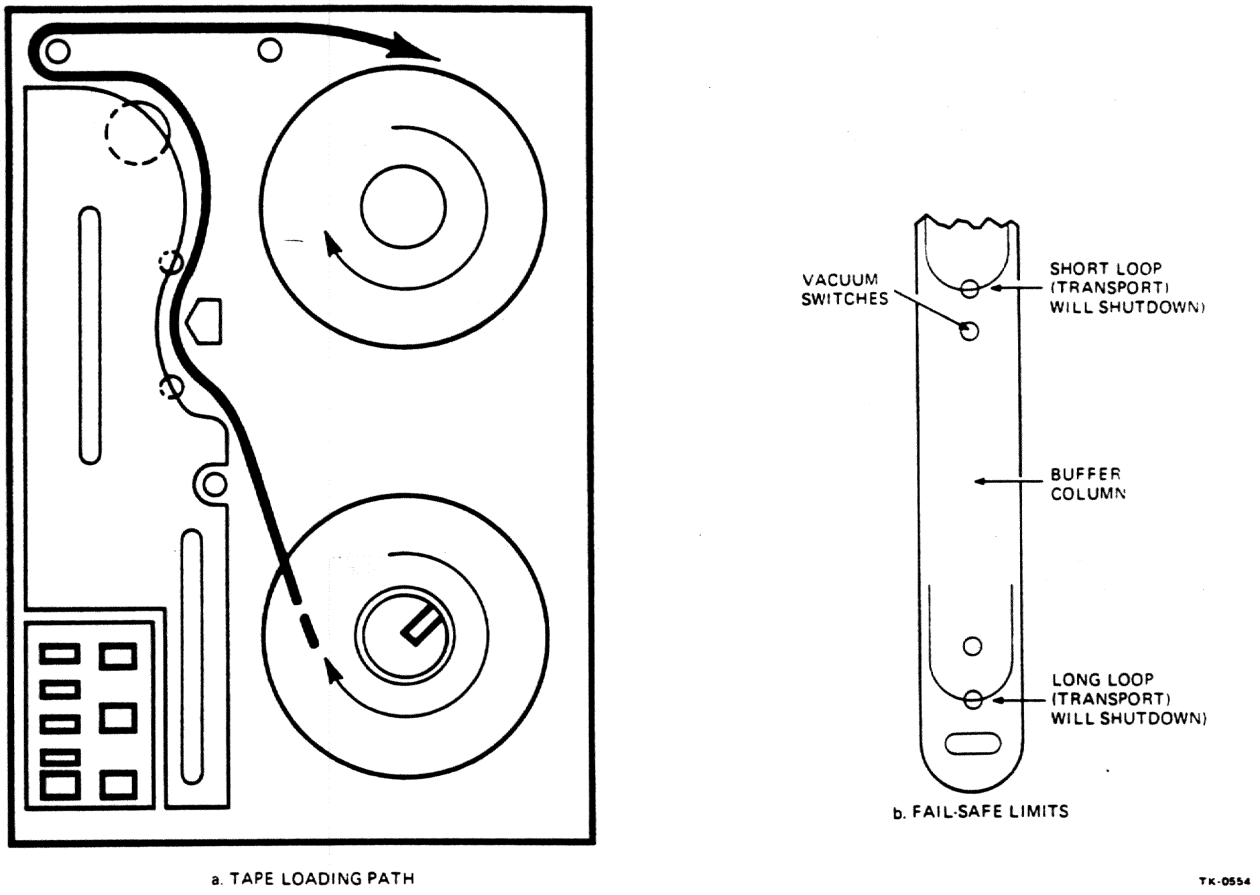


Figure 5-27 Tape Loading Path/Fail-Safe Limits

CAUTION

Wind tape flat onto the take-up reel. Do not bend tape back or place tape end outside of reel (out the window). While winding tape on take-up reel, simultaneously unwind the file reel to relieve tension on the tape. Rotate the reels gently. Do not jerk the tape, as this could cause the tape to stretch.

6. When the tape is mounted and threaded, press the LOAD button. After a 2-second delay, to allow vacuum to be established, tape will be pulled into the vacuum columns and the seek-for-BOT sequence will begin. This sequence consists of five seconds of forward tape motion followed by a rewind to BOT. When the BOT marker is sensed, tape motion stops and the transport goes on-line (both the ON LINE switch and BOT indicators glow). The tape is now positioned at the load point (at the BOT marker) and the transport is ready for use.

5.7.3.3 Unloading Tape – Depending on whether or not tape is at the BOT marker, use one of the following unloading procedures.

Tape Not at BOT

1. Ensure the transport is off-line.

2. With the transport off-line and the tape not at BOT, press the REWIND/UNLOAD button.

3. The transport executes a high-speed rewind operation. When the BOT marker is sensed, tape motion stops.

4. Press the REWIND/UNLOAD button again. An unload sequence begins with the tape gently winding onto the lower file reel. When all the tape is on the file reel, tape motion stops.

5. If you wish to remove the file reel, unlock the snap-lock lever on the hub and gently pull the reel off.

Tape at BOT

1. Ensure the transport is off-line.

2. With the transport off-line and the tape at the BOT marker, press the REWIND/UNLOAD button.

3. The unload sequence begins, with the tape gently winding onto the lower file reel. When all the tape is on the file reel, tape motion stops.

4. If you wish to remove the file reel, unlock the snap-lock on the hub and gently pull the reel off.

5.7.3.4 Restart After Power Failure – In the event of a power failure, the TE16 automatically shuts down and tape motion stops, without physical damage to the tape. However, if the transport was on-line and was either reading or writing at the time of the power failure, the last record should be assumed lost. Refer to the system recovery procedures documentation if this happens. To restart the transport, proceed as follows:

NOTE
Return of power is indicated when the POWER LED glows.

1. Ensure that the TE16 is off-line.

2. Using the top tape reel, manually wind up any loose tape so that it is just snug in the tape loading path (Figure 5-27).

3. Press the LOAD button. This causes the tape to be drawn into the vacuum columns, go forward for five seconds, and rewind to BOT.

5.7.3.5 Restart After Fail-Safe – If the tape loop in either buffer column exceeds the fail-safe limit shown in Figure 5-27, the vacuum system automatically shuts down and tape motion stops without damage to the tape. When this fail-safe condition occurs, the TE16 does not respond to either on-line or off-line commands. To restart the transport, follow the procedure listed in Paragraph 5.7.3.4.

Before restarting, this failure should be noted in the system log, together with an indication of which fail-safe switch was exceeded and the distance by which it was exceeded.

This information will aid Field Service in preventing a reoccurrence, should the fail-safe condition be due to a tape transport problem.

5.7.4 Operator Troubleshooting

Before any maintenance personnel are called to correct a problem, the operator can make several checks with minimal effort. These precautions may isolate an easily correctable error:

1. Ensure that the vacuum door is closed and sealed properly.
2. If the tape does not stop at BOT, be certain the tape has a BOT marker.
3. Ensure that the write-enable ring is inserted in the tape reel if a write operation is to be performed.
4. Clean the tape path according to the daily (8 hour) preventive maintenance procedures.
5. Check POWER indicator. If it is not glowing, ensure that the 861 power control circuit breaker is on. Also ensure that the REMOTE/OFF/LOCAL 861 power control switch is in the REMOTE position.

5.7.5 Care of Magnetic Tape

1. Do not expose magnetic tape to excessive heat or dust. Most tape read errors are caused by dust or dirt on the read head; keeping tape clean is imperative.
2. Always store tape reels inside containers when the tape is not in use; keep the empty containers tightly closed to keep out dust and dirt.
3. Never touch the portion of tape between the BOT and EOT markers: oil from fingers attracts dust and dirt.
4. Never use a contaminated reel of tape; this spreads dirt to clean tape reels and could have an adverse effect on tape transport reliability.
5. Always handle tape reels by the hub hole; squeezing the reel flanges could lead to tape edge damage in winding or unwinding tapes.
6. Do not smoke near the tape transport or storage area; tobacco smoke and ashes are especially damaging to tapes.
7. Do not place magnetic tape near any line printer or other device that produces paper dust.
8. Do not place magnetic tape on top of the tape transport, or in any other location where it might be affected by hot air.
9. Do not store magnetic tape in the vicinity of electric motors.

5.8 RX11 FLOPPY DISK

The RX11 Floppy Disk consists of an RX01 subsystem and an interface for the VAX-11/780.

The RX01 is a low cost, random access, mass memory device that stores data in fixed length blocks on a preformatted, IBM-compatible, flexible diskette. Each drive can store and retrieve up to 256K 8-bit bytes of data. The RX01 consists of a flexible disk, a read/write electronics module, a microprogrammed controller module, and a power supply, enclosed in a rack mounted, 10-1/2 inch, self-cooled chassis. A cable is included for connection to a PDP-11 interface for use on the PDP-11 Unibus. The RX01 is shown in Figure 2-1.

The RX01 connects to the RXV-11 controller, which converts the RX01 I/O bus to the console Q bus structure. It controls interrupts to the CPU initiated by the RX01, decodes Unibus addresses for register selection, and handles data interchange between the RX01 and the host CPU.

The interface modules are dc powered by the host processor.

All components except the interface are housed in a 10-1/2 inch rack mountable box. The power supply and the logic modules are mounted above the drives. Interconnection from the RX01 to the interface is with a 40-conductor BC05L-15 cable.

5.8.1 Specifications

The specifications for the RX11 are given below.

System Reliability

Drive Performance

Capacity	
Per diskette	256,256 bytes
Per track	3,328 bytes
Per sector	128 bytes
	Data transfer rate
Diskette to controller buffer	4 μs/data bit (250K bytes/s)
Buffer to CPU interface	2 μs/bit (500K bytes/s)
CPU interface to I/O bus	18 μs/8-bit byte (50K bytes/s)

Environmental Characteristics

Temperature	15° to 32° C (60° to 90° F)

NOTE

Media temperature must be within operating temperature range before use.

Electrical

Power consumption	
RX01	3 A @ 24 V (dual), 75 W 5 A @ 5 V, 25 W
M7846 interface	Not more than 1.5 A @ 5 Vdc
AC power input	4 A @ 115 Vac 2 A @ 230 Vac

5.8.2 Operation

5.8.2.1 Operator Control – The simplicity of the RX01 precludes the necessity of operator controls and indicators. A convenient method of opening the unit for diskette insertion and removal is provided. On each drive is a simple pushbutton, which is compressed to allow the spring-loaded front cover (top) to open. The diskette may be inserted or removed (with the label up). The front cover automatically locks when the bar is pushed down.

CAUTION

The drive(s) should not be opened while they are being accessed because data may be recorded incorrectly, resulting in a CRC error when the sector is read.

5.8.2.2 Diskette Handling Practices and Precautions – To prolong the diskette life and prevent errors when recording or reading, reasonable care should be taken when handling the media. The following handling recommendations should be followed to prevent unnecessary loss of data or interruption of system operation.

1. Do not write on the envelope containing the diskette. Write any information on a label prior to affixing it to the diskette.

2. Paper clips should not be used on the diskette.

3. Do not use writing instruments that leave flakes, such as lead or grease pencils, on the jacket of the media.

4. Do not touch the disk surface exposed in the diskette slot or index hole.

5. Do not clean the disk in any manner.

6. Keep the diskette away from magnets or tools that may have become magnetized. Any disk exposed to a magnetic field may lose information.

7. Do not expose the diskette to a heat source or sunlight.

8. Always return the diskette to the envelope supplied with it to protect the disk from dust and dirt. Diskettes not being used should be stored in the file box if possible.

9. When the diskette is in use, protect the empty envelope from liquids, dust, and metallic materials.

10. Do not place heavy items on the diskette.

11. Do not store diskettes on top of computer cabinets or in places where dirt can be blown by fans into the diskette interior.

12. If a diskette has been exposed to temperatures outside of the operating range, allow 5 minutes for thermal stabilization before use. The diskette should be removed from its packaging during this time.

5.8.2.3 Diskette Storage - Short Term (Available for Immediate Use)

1. Store diskettes in their envelopes.

2. Store horizontally, in piles of ten or less. If vertical storage is necessary, the diskettes should be supported so that they do not lean or sag, but should not be subjected to compressive forces. Permanent deformation may result from improper storage.

3. Store in an environment similar to that of the operating system; at a minimum, store within the operating environment range.

Long Term - When diskettes do not need to be available for immediate use, they should be stored in their original shipping containers within the nonoperating range of the media.

5.9 LSI-11 MICROCOMPUTER

The PDP-11/03 is a packaged version of the LSI-11 microcomputer. It includes a rack mountable enclosure containing the LSI-11 processor, memory, an LSI-11 bus-structured backplane, a power supply for the processor and options contained in the box, and a control panel (part of the power supply assembly) containing three indicators and three switches. The LSI-11 control panel is shown in Figure 2-1.

5.9.1 Operation

The operator inputs commands, data, and instructions via the console terminal, which also serves as an output device. All console functions previously provided by a control panel containing switches and lights are provided by this terminal, including starting programs, depositing and examining memory and register locations, and, with most terminals, halting program execution.

Bootstrap, loader, and diagnostic programs are included in PROMs. The bootstrap program allows bootstrapping the RX01 disk system by entering a command on the console device.

To apply power, proceed as follows:

1. Ensure that the system is configured properly.
2. Place the DC ON/OFF switch in the down position (DC OFF).
3. Place the AC ON/OFF switch on the rear of H780 power supply in the AC ON position.
4. Place the HALT/ENABLE switch in the desired power-up position.

NOTE

DC power can be applied with the HALT/ENABLE switch in either position. However, processor power-up response is affected by this switch.

5. Place the LTC ON/OFF switch in the OFF position.
6. Place the DC ON/OFF switch in the up (DC ON) position. The console terminal should respond with a printout.

5.10 LA11 LINE PRINTER

The LA11 Line Printer consists of an M7258 Interface Unit (LA11 Controller) and an LA180 Printer.

The LA180 is a small, low-cost, high-speed printer that prints at speeds of up to 180 char/s. Data can be received in standard ASCII codes over a parallel or serial interface.

The printer produces a hard copy original plus up to five duplicate copies on tractor-driven continuous forms varying in width from 3 to 14-7/8 inches. Preprinted forms can be positioned in exact vertical alignment by operating a manual clutch on the tractor drive. The standard set of 95 upper- and lowercase ASCII characters is printed at a horizontal spacing of 10 char/in and a vertical spacing of 6 lines/in.

5.10.1 Technical Characteristics

The technical characteristics of the LA180 DECprinter I are listed in Table 5-7.

Table 5-7 LA11 Technical Characteristics

Line Length
132 characters maximum

Spacing
10 char/in (horizontal)

Characters
96 upper/lower case ASCII
7 × dot matrix (0.07 × 0.10 in)

Paper
Variable Width: 3 to 14-7/8 in
Single-Part Paper:
15 lb paper minimum
Card stock thickness of 0.010 in max

Multipart Paper:
2- to 6-part (see notes)
Thickness of 0.020 in max
Tractor-drive, pin-feed

Ribbon
DIGITAL-specified nylon fabric
(Part No. 36-12153)
Spool assembly: 0.5 in wide × 60 yd long

Power
90–132 Vac or 180–264 Vac
50 or 60 Hz ± 1 Hz
400 W max (printing)
200 W max (idle)

Dimensions
70 cm (27.5 in) wide
85 cm (33.5 in) high
55 cm (21.7 in) deep

Weight
46 kg (102 lb)

NOTES

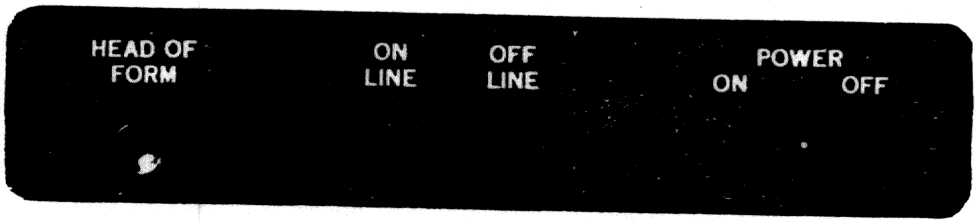
1. **Multipart forms may have only one card part. The card must be the last part.**
2. **NCR or 3M paper, up to 6-part, must use ribbon on top copy. First surface impact paper is not recommended.**
3. **Continuous-feed, fan-fold business forms with 3- or 4-prong margin crimps on both margins (multipart) are recommended. Stapled forms are not recommended and may damage tractors and other areas of the machine. Dot or line glue margins are acceptable (if line is on one margin only). Do not line glue both margins as air will not be able to escape and poor impressions will result.**

5.10.2 Operating Controls and Indicators

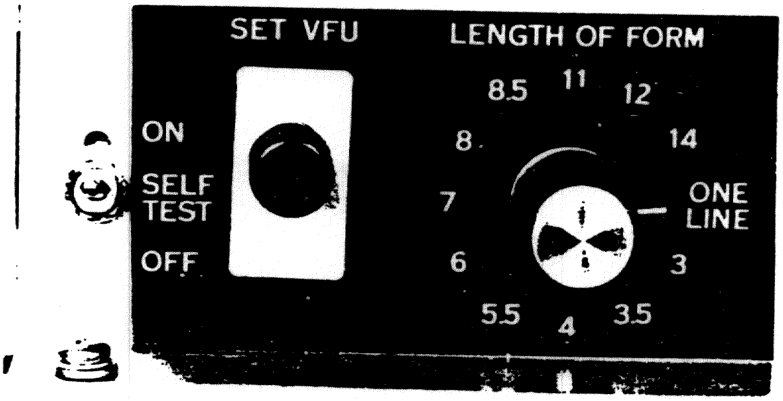
5.10.2.1 LA180 Alarm Signals – The LA180 produces three different audible alarm signals. The operator should become familiar with these alarms in order to determine the correct response.

- Continuous Tone – Indicates a carriage jam or failure. To turn the alarm off, set the POWER switch to OFF. Turing the POWER switch OFF, then ON, resets the alarm.
- Repetitive Beeping – Indicates an out-of-paper condition or bell code. To turn the alarm off when out of paper, set the ON LINE/OFF LINE switch to OFF LINE.
- Single Beep – Indicates a bell code.

5.10.2.2 LA180 Operator Controls (Figures 5-28 and 5-29)



7595-9-A0200



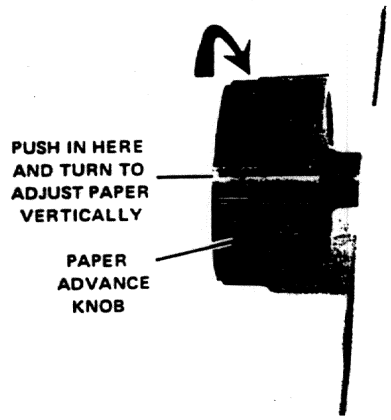
7595-10-A0202

Figure 5-28 LA180 Operator Control Panels

HEAD OF FORM Pushbutton
The HEAD OF FORM pushbutton, when pressed, causes the line printer paper to advance to the top of the form, as determined by the SET VFU and LENGTH OF FORM switches. The HEAD OF FORM pushbutton is only active OFF LINE.

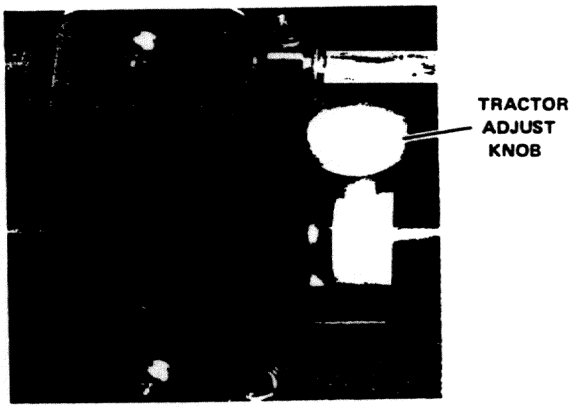
ON LINE/OFF LINE Switch
The ON LINE/OFF LINE switch enables the operator to place the LA180 off-line or on-line with an operating system. When off-line, the HEAD OF FORM switch is activated.

POWER ON/OFF Switch
The POWER switch controls power application to the LA180.



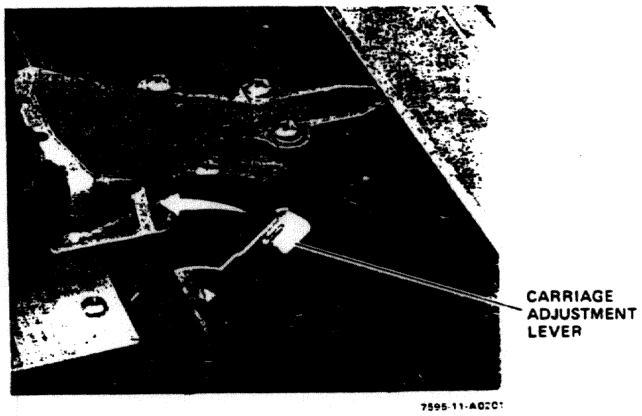
7595-4-A0199

a. Vertical Adjustment



7595-7-A0198

b. Tractor Knob Adjustment



c. Carriage Adjustment Lever

Figure 5-29 Operator Controls

SELF TEST Switch
In the on-line mode, setting the SELF TEST switch to ON causes the LA180 to print out all the characters.

NOTE
When performing a self-test, 14-7/8 inch (132 column) paper must be used.

SET VFU Pushbutton
The LENGTH OF FORM switch enables the LA180 to determine the length of the forms; however, the LA180 does not know where the form begins. Pressing SET VFU establishes the starting point on the form.

LENGTH OF FORM Switch
The LENGTH OF FORM switch is set to the number corresponding to the form length in inches. To set the form length to 11 inches, set the switch to 11. (The new switch setting is delayed until one form passes through the LA180 at the old switch setting. Pressing SET VFU eliminates the one-form delay.)

Paper Advance Knob (Figure 5-29)
The paper advance knob, when turned, advances the paper one step at a time. Pressing in and turning the paper advance knob enables the paper to be rolled freely in either direction, and allows precise vertical forms positioning.

Tractor Adjust Knobs
The tractor adjust knobs aliow fine horizontal adjustment of forms.

Carriage Adjustment Lever
The carriage adjustment lever controls the print head gap for single or multipart forms.

NOTE
Refer to the impression adjustment procedure in Paragraph 5.10.3.1 when setting the carriage adjustment level.

Cover Interlock Switch
The cover interlock switch (when installed) prevents the LA180 from printing when the cover is open. Opening the cover when the ON LINE/OFF LINE switch is ON LINE causes a repetitive beeping. To turn the alarm off, set the ON LINE/OFF LINE switch to OFF LINE.

5.10.3 Operating Procedure

5.10.3.1 Loading Paper – The LA180 is a highly flexible printer that can accept multipart forms, with widths from 3 to 14-7/8 inches. When loading new forms, it is necessary to perform two adjustments:

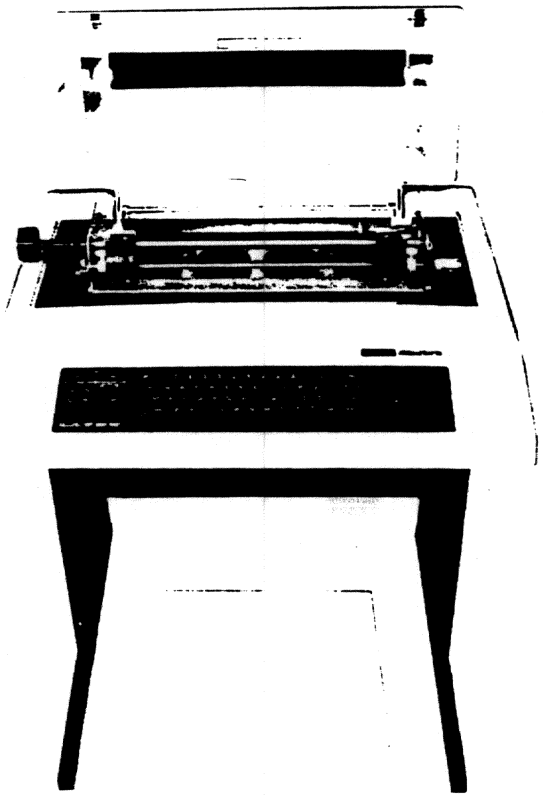
1. Paper positioning.
2. Impression adjustment.

In addition, there is a horizontal positioning and vertical positioning adjustment. The horizontal positioning adjustment allows the paper to be slightly shifted left or right. This procedure is especially useful when typing on preprinted forms with defined horizontal zones. The vertical positioning adjustment enables the paper to be adjusted vertically. Once these adjustments have been performed, reloading paper becomes quick and simple, requiring a minimum of interruption.

5.10.3.2 Loading New Forms – Paper Postioning Procedure (Figure 5-30)

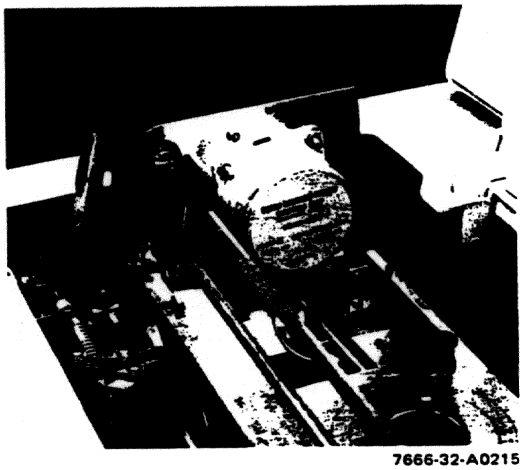
1. Set the POWER switch to OFF.
2. Lift the cover.
3. Place the tractor-feed paper on the floor between the legs of the LA180. (The term tractor-feed refers to the holes on either side of the paper.)

NOTE
Ensure that the leading edge of the forms is directly below and parallel to the feed slot.



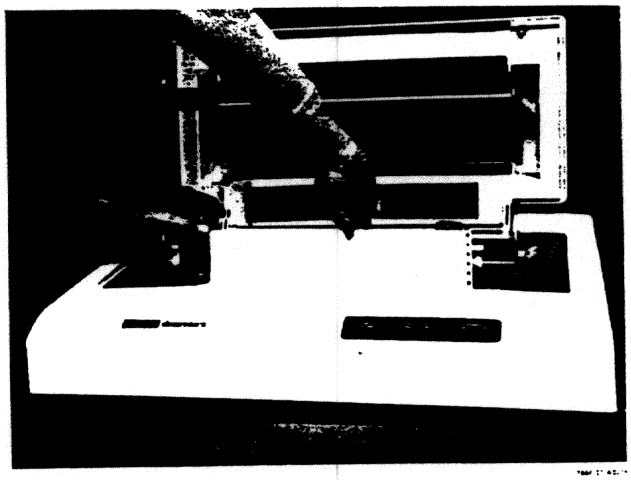
7666-14-A0220

a. Cover Open

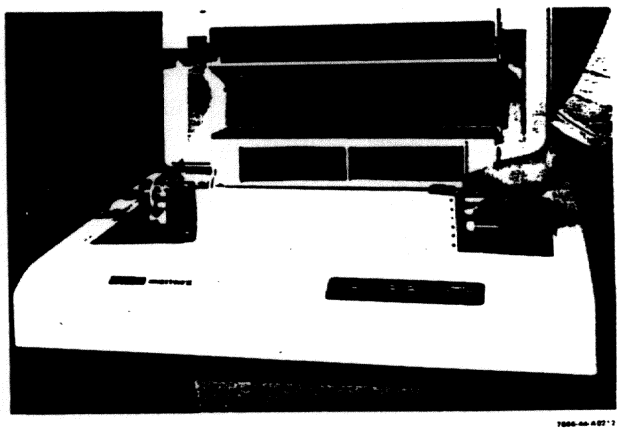


7666-32-A0215

b. Tractor Cover Open



c. Paper Fed to Tractor Pins



d. Tractor Adjustment

Figure 5-30 Loading New Forms

4. Open the left tractor cover so that the tractor pins are exposed.

5. Move the carriage adjustment lever to the highest number (toward operator).

6. Feed the paper through the load channel under the terminal and align the left paper margin holes over the left tractor pins.

7. Close the left tractor cover.

8. Loosen the tractor adjustment knob on the right tractor about 1/2 turn.

9. Open the right tractor cover and slide the tractor to a position where the holes on the right paper margin align directly over the tractor pins.

10. Close the tractor cover.

NOTE

Ensure that the paper does not pull against the tractor pins or bow in the middle.

11. Tighten the tractor adjustment.

12. Set the LENGTH OF FORM switch (Figure 5-28) to the number corresponding to the form length and proceed to the impression adjustment.

Impression Adjustment

NOTE

The carriage adjustment lever (Figure 5-29) is normally set forward (to notch number 1) for single thickness paper. The following procedure is applicable only to multipart forms.

1. Set the POWER switch to OFF.

2. Set the carriage adjustment lever to the number corresponding to the number of parts in the forms.

3. Turn the paper advance knob counterclockwise while moving the carriage adjustment lever forward one notch at a time until the paper smudges. Then move the lever back one notch at a time until the paper no longer smudges.

4. Set the POWER switch to ON (Figure 5-28) and resume operation.

NOTE

If the impression is unsatisfactory due to a worn ribbon, perform the ribbon installation procedure. A worn ribbon is indicated when the first copy in a multipart copy is poor but the remaining copies are good.

Horizontal Positioning Adjustment
The horizontal positioning adjustment enables the paper to be shifted left or right (1/2 in maximum). Shifting the paper provides a simple means of aligning the type within the appropriate columns on the paper.

1. Set the POWER switch to OFF (Figure 5-28).
2. Lift the cover and loosen both tractor adjustment knobs about 1/2 turn (Figure 5-29).
3. Move the tractors the desired amount (1/2 in maximum) to have characters type in the appropriate columns.
4. Tighten the tractor adjustment knobs.

NOTE
Ensure that the paper does not pull against the tractor pins or bow in the middle.

Fine Vertical Positioning
For fine vertical positioning, press in and turn the paper advance knob (Figure 5-28).

NOTE
When rolling paper backwards, use care to ensure that paper perforations do not snag on the print head.

5.10.3.3 Reloading Paper

1. Set the POWER switch to OFF.
2. Lift the cover.
3. Place the tractor-feed paper on the floor between the legs of the LA180.
4. Open both tractor covers so that the tractor pins are exposed.

NOTE
Ensure that the leading edge of the forms is directly below and parallel to the feed slot.

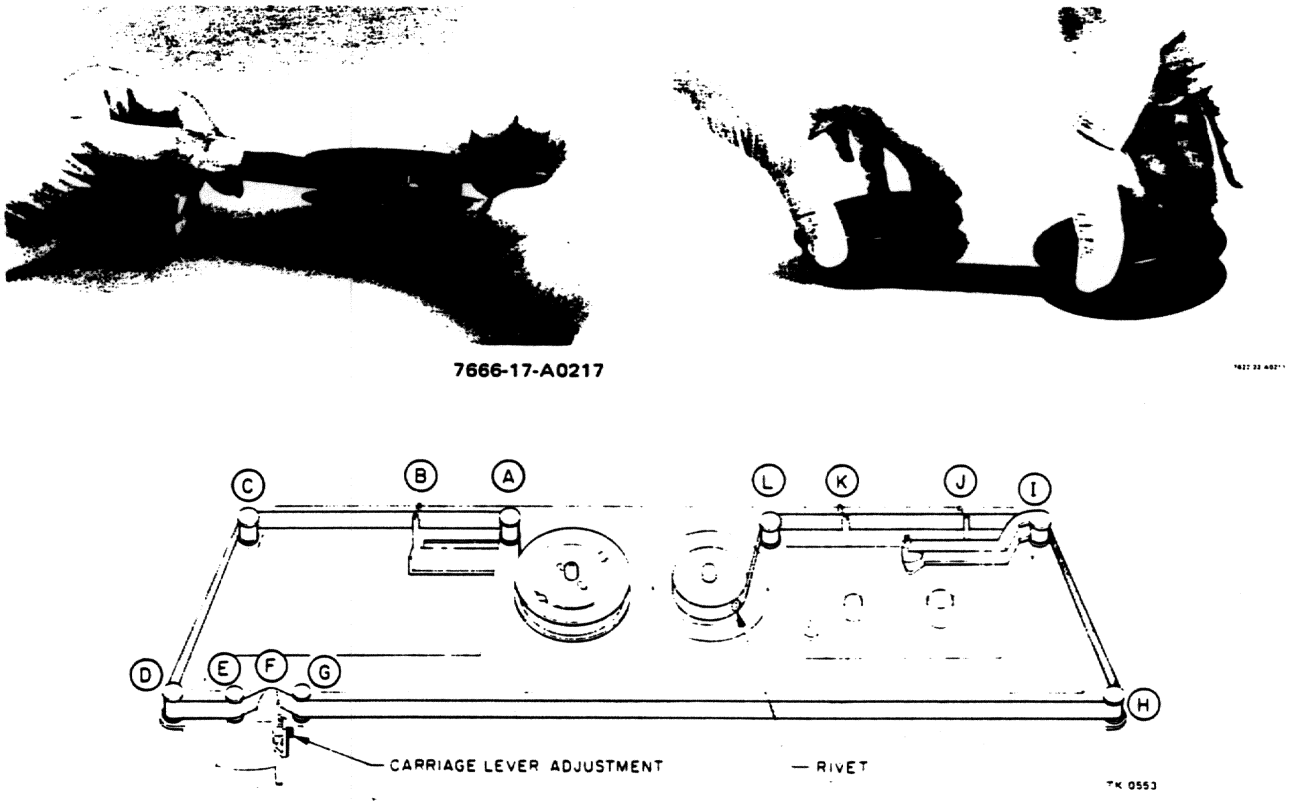
5. **Feed the paper through the load channel under the terminal and align the paper holes over the tractor pins.**
6. **Close the tractor covers.**

5.10.3.4 Ribbon Installation – The printer ribbon should last for 8 to 12 hours of actual printing at 180 char/s (about 4 million characters). After 8 hours, or when the print density becomes too light, remove both ribbon spools from their drive spindles and turn the whole assembly over so that the previous lower edge of the ribbon is now on top. After rethreading the ribbon, another 4 hours (approximately) of printing time is possible before the ink is completely used. At that time, the ribbon must be replaced by removing both spools and unthreading the ribbon. Replace with a new spool and ribbon assembly (Part No. 36-12153) and an empty spool. (One of the old spools may be used if desired.)

NOTE

Use only DIGITAL-recommended ribbons (Part No. 36-12153). Use of other ribbons can void machine warranty. Ribbons must be made of nylon fabric with nonabrasive inks. Carbon-based inks are extremely abrasive and should not be used.

1. Set the POWER switch to OFF and lift the cover.
2. Record the setting of the carriage adjustment lever. Move the carriage adjustment lever to the highest number.
3. Remove the ribbon spools and ribbon. Save one spool to be used with the new ribbon.
4. Connect the hook on the end of the ribbon to the empty spool (Figure 5-31).



Ribbon Installation Diagram

Figure 5-31 Reloading Ribbon

5. Wind 10 turns of ribbon on the empty spool.
6. Place the full spool on the left spindle (Figure 5-31) and turn clockwise until is drops into position.

7. Guide the ribbon around idler spool A through guide B, and around the outside of idler spools C through E.

8. Guide around the front of head F and idler spools G and H.

9. Guide the ribbon around tension arm I, through slots J and K (dire tion changing guide), and around idler spool L.

10. Turn the spool clockwise until it drops into position.

11. Take up the slack in the ribbon by turning the free moving spool.

12. Return the carriage adjustment lever to its original setting.

NOTE

The rivet located on the ribbon must be on the spool or between the spool and the direction changing guide.

Ribbon can be threaded in the opposite direction (from L to A).

5.10.3.5 Using the LA180 – Using the LA180 requires almost no operator intervention. These few steps are all that may be required to place your LA180 on-line.

1. Load the paper.
2. Set the ON LINE/OFF LINE switch to OFF LINE.
3. Set the SELF TEST switch to OFF.
4. Set the POWER switch to ON.
5. Set the top of form, if applicable.
6. Set the ON LINE/OFF LINE switch to ON LINE.

The LA180 is now on-line and fully operational.

5.10.3.6 Self-Testing the LA180 – The self-test procedure provides a simple and accurate method of testing the LA180.

1. Set the POWER switch to ON.

2. Set the ON LINE/OFF LINE switch to OFF LINE.

3. Set the LENGTH OF FORM switch to the number corresponding to the form length in inches.

4. Turn the paper advance knob until the bottom of the first line to be typed is aligned with the top of the Top of Form bar.

5. Press the SET VFU pushbutton.

6. Press the HEAD OF FORM pushbutton several times to ensure that the forms advance to the correct position.

NOTE

If the LA180 is turned off and on, the top of form starting point must be reestablished using the SET VFU pushbutton.

5.11 LP11 LINE PRINTER

The LP11 Line Printer (Figure 5-32) is a high-speed printer designed to interface with the VAX-11/780 family of processors to provide impact line printing at speeds up to 1250 lines/minute (speed is dependent on line length).

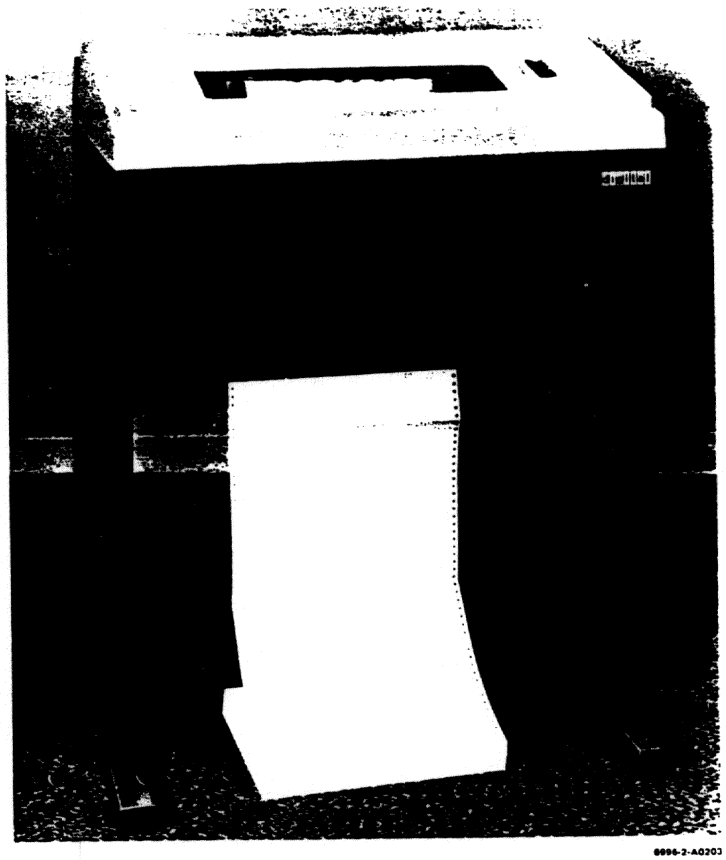


Figure 5-32 LP05 Line Printer

The LP11 system consists of two major components: a Data Products Corporation line printer (Model 2230, 2310, 2410, or 2470); and an M7930 or M7258 interface unit, referred to as the LP11 controller. The Data Products line printer models (designated by DIGITAL as LP05). These DIGITAL designations are used throughout this guide. Figure 5-32 shows the LP05 line printer.

5.11.1 General Information

The LP11 Line Printer system is designed to operate on-line with the VAX-11/780 system and associated peripherals such as paper tape readers, magnetic tape units, card readers, or communications terminals. The line printer is mounted on a free-standing cabinet. The controller, which interfaces the line printer to the Unibus, is a single quad module.

The line printer is a high-speed printer that produces a hard copy output at rates up to 1250 lines/min. The printer employs an impact type mechanism with a revolving character drum and one hammer per column. Forms making up to six copies can be used when multiple copy printing is desired. The printer is available in three versions: 80 columns, with a speed of up to 356 lines/min for a full line; 132 columns, with a speed of up to 245 lines/min for a full line; 132 columns, with a speed of up to 1250 lines/min for a full line; or 132 columns, with a speed of up to 300 lines/min for a full line. All of the preceding print rates are based on the 64-character drum.

The four line printer models can be ordered with either a 64- or 96-character drum. The DIGITAL model designations are listed below.

		Model Designations Number of	
Line Printer	**Designation**	**Characters**	**Power**
LP05 132 Column (DP 2230)	LP11-VA	64	115 V, 60 Hz
	LP11-VD	64	230 V, 50 Hz
	LP11-WA	96	115 V, 60 Hz
	LP11-WD	96	230 V, 50 Hz

A brief description of the four line printer models and both character sets is contained in subsequent paragraphs.

The LP11-V, -W 132-column line printer has a maximum line length of 132 columns and prints at a rate of 300 lines/min when using the 64-character set, and 240 lines/min when using the 96-character set. A 132-column memory is contained within the line printer. Printing is accomplished by dividing the 132 columns into odd and even positions and sharing a hammer and associated drive circuit between two positions. The LP11-V, -W will print a line only after one of three control characters is sent to it.

The LP11 controller interfaces the line printer to the Unibus and is under program control. The controller does not store information but synchronizes the data transfer between the bus and the printer. The prime functions of the controller are: indicate to the VAX-11/780 the operational status of the line printer; control transfer of data from the VAX-11/780 into the printer; and enable the line printer to gain control of the bus and perform an interrupt routine.

5.11.2 Specifications

Electrical, operating, physical, environmental, and printing characteristics and performance specifications are presented in Table 5-8.

Table 5-8 LP11-V, -W Line Printer Specifications

Electrical	
Printer	115 Vac ± 10%, 50/60 Hz ± 3 Hz 230 Vac ± 10%, 50/60 Hz ± 3 Hz
Controller	1.5 A (derived from power supply in mounting box where controller is installed)
Physical Characteristics	
Printer	
Height	1.14 m (45 in)
Width	0.81 m (32 in)
Depth	0.56 m (22 in)
Weight	150 kg (330 lb)
Controller	The LP11 controller is a small peripheral controller consisting of a single quad module and occupying 1/4 of a DD11 or one of the controller slots in a processor system unit such as the KA11 or PDP-11/05.
Ribbon Characteristics	
Type	Inked roll
Width	15 in
Length	20 yd
Thickness	0.004 in
Paper Characteristics	
Type	Standard fanfold, edge punched, 11 in between folds
Width	4 in to 16-3/4 in
Weight	15 lb bond minimum (single copy) 12 lb bond with single-sheet carbon for up to six parts (multiple copy)

5.11.3 Loading Paper

Use the following procedure when loading paper into the LP05 line printer. More detailed instructions including illustrations are given in Section 4 of Data Products Model 2230 Line Printer Technical Manual.

1. Lift printer cover.
2. Pull drum gate latch forward.
3. Swing drum gate fully open.
4. Set the power circuit breaker to on and press and release the TOP OF FORM switch on the control panel. The tractors should automatically advance to the top-of-form position.
5. Open the spring-loaded pressure plates.
6. Place the paper in the tractors and close the pressure plates.
7. Loosen both of the paper width adjustment guides and move both of the tractors laterally to adjust the correct paper width. Tighten the paper width adjustment guides.

8. Align the perforation in the paper with the top-of-form index line by setting the FORMS RESET switch to the down position and rotating the tractor shaft by using the coarse paper advance adjustment knob. Then set the FORMS RESET switch to the up position.

9. Adjust the horizontal position of the paper by using the horizontal paper positioning knob. Use the horizontal indentation index marks as a guide.

10. Close and latch the drum gate.

11. Close the printer cover.

12. Use the fine paper advance adjustment knob to correct any small misalignment in the print-out during operation.

5.12 CR11/CR11A CARD READER

The CR11/CM11 Card Reader System is a card handling system designed to read marked or punched Hollerith data cards at rates up to 600 cards/min. The CR11/CM11 consists of two distinct components: a card reader and an interface unit, which is referred to as the CR11 Controller.

a. Card Reader	A motorized card handling device that reads information from EIA standard (Hollerith code) cards. The unit reads 11-row, 80-column cards at a nominal rate of 200 to 600 cards/min, depending on the specific card reader used. The CR11 system is fully compatible with Documation™ punched card and mark/sense readers and GDI punched card and mark/sense readers.
b. CR11 Controller	An interface between the card reader and the PDP-11 Unibus. Controls data transfers between the card reader and other devices. Provides a 12-bit output character for standard data transfers and an 8-bit output character (one byte) when it is desired to use the proposed compressed Hollerith code. Also monitors reader operations and issues appropriate control commands. The controller is completely plug-compatible with the above-mentioned card readers: no modifications are required. Also, referred to as "control unit," "interface," and "reader control."

5.12.1 Specifications

Specifications for all Documation type card readers are given in Table 5-9, specifications for GDI (General Design, Inc.) card readers are given in Table 5-10, and Table 5-11 lists the model variations of optical card readers.

5.12.2 Operator Controls and Indicators

The CR11/CR11-A card reader operating controls and indicators are contained in Table 5-12.

™Documation is a trademark of Documation, Incorporated, Melbourne, Florida.

Table 5-9 Documation Card Reader Specifications

All Punched Card Models		OM200		
Card Type:	Standard 80-column EIA	Standard 80-column EIA card		
Light Source:	Infrared light emitting diode	Fiber Optics, 13 channel		
Variations		**Model M200 & OM200**	**Model M300**	**Model M600**
Card Rate (cards/min)		285	300	600
Hopper/Stacker Card Capacity		550	1000	1000
Power (in VA):				
Starting Load		1600	1600	1600
Running Load		600	600	600
Dimensions:				
Height	11 in	13-9/16 in		19-9/16 in
Width	19-1/4 in	23-1/16 in		23-1/16
Depth	14 in	18 in		18 in
Weight	60 lb	75 lb		75 lb

Table 5-10 GDI Card Reader Specifications

Both Models (100 and 100M)	
Card Type	Standard 80-column EIA card
Card Rate	200 cards/min (nominal)
Hopper/Stacker Card Capacity	450 cards
Input Power	115 ± 10 Vac, 60 Hz, single phase
Power Consumption	Less than 300 W
Size	14 in wide, 18 in deep, 18 in high
Variations	
Weight	Model 100M = 47 lb Model 100M = 52 lb
Card Type	Model 100 - Reads only 80-column punched cards Model 100M - Reads punched or marked 40-column cards with clock track

Table 5-11 Optical Card Reader Models

Option No.	Manufacturer & Model No.	Voltage	Frequency (in Hz)
CM11	GDI, 100M (40 column)	115	60
CM11-A	GDI, 100M, (40 column)	230	50
CM11-FA	Documation, OM200	115	60
CM11-FB	Documation, OM200	230	50

Table 5-12 Rear Panel Controls

Control	Type	Function
LAMP TEST	pushbutton	When pressed, illuminates all indicators on the front control panel to determine if any of the indicator lamps are faulty.
SHUTDOWN	2-position toggle	Controls automatic operations of the input hopper blower. MAN position - blower operates continuously whether or not cards are in the input hopper. AUTO position - causes the blower to shut down automatically whenever the input hopper is emptied. Blower automatically restarts when cards are loaded into the hopper and the RESET switch is pressed. Blower activates approximately 3 seconds after RESET is pressed.
MODE switch	2-position toggle	Permits selection of either on-line or off-line operation. LOCAL position - removes the READ COMMAND input from the controller to allow the operator to run the reader off-line by using the RESET and STOP switches on the front control panel. REMOTE position - enables the READ COMMAND input from the controller to allow normal on-line operation under program control once RESET is pressed.

5.12.3 Card Handling Procedures

The following paragraphs present the recommended procedures for loading the input hopper, unloading the output stacker, and correcting error conditions.

5.12.3.1 Loading Cards - The following procedure is used to load the input hopper with punched cards to be read.

1. Pull the hopper follower back with one hand and begin loading card decks into the hopper. Ensure that the first card to be read is placed at the front with the "9" edge down, column 1 to the left.

2. Continue placing cards into the input hopper until it is loosely filled (this is the approximate amount of cards listed as the capacity for a particular card reader model).

CAUTION

Do not pack the input hopper so full that the air from the blower cannot riffle the cards properly. If the cards are packed too tightly, it impairs proper operation of the vacuum picker.

3. Cards may continue to be loaded while the reader is operating provided tension is maintained on the front portion of the deck as cards are added to the rear. Additional cards should not be loaded, however, until the hopper is approximately 1/2 to 1/3 full.

CAUTION

When maintaining tension on the card deck, use just enough pressure to maintain the riffle action to prevent card damage and jamming of the reader.

4. Normally, all cards are moved through the reader into the stacker. However, if it is necessary to remove cards from the input hopper, simply pull back the follower and remove the card deck.

5.12.3.2 Unloading Cards – To unload cards from the output stacker, pull the stacker follower back with one hand and remove the card deck from the stacker, being careful to maintain the order of the deck. The stacker may be unloaded while the cards are being read.

5.12.3.3 Correcting Error Conditions – The four error alarm indicators on the front control panel of the card reader normally indicate a condition that can be corrected by operator intervention.

5.12.4 Operating Procedures

The CR11 Card Reader System can be used in either a local or remote operating mode. The local (off-line) mode is controlled by switches on the front and rear panels of the card reader. The remote (on-line) mode is controlled by programmed commands from the VAX-11/780.

The following paragraphs present procedures for both off-line and on-line operation of the card reader. Although procedures are given for setting up on-line operaton, it is beyond the scope of this manual to give any programming details.

5.12.4.1 Off-Line Operation – Off-line operation of the card reader is used primarily for setting up and checking reader operation prior to switching to on-line use; for correcting error conditions; and for maintenance tests. When placed off-line, the reader can be operated locally from the control panels. The following procedure is used to energize the reader and check off-line operation prior to switching to on-line operation.

1. Ensure that the ac power cord is plugged in and that the circuit breaker on the rear base panel of the reader is in the ON position.

2. Set MODE switch to LOCAL position.

3. Set SHUTDOWN switch to AUTO postion.

4. Press POWER switch to energize reader. Note that POWER indicator lights but blower does not come on.

5. Press LAMP TEST switch and observe that all front panel indicators are lit.

6. Load a card deck into the input hopper.

7. Press RESET switch and observe that the associated green indicator comes on. After approximately 3 seconds, cards should start being picked and moved through the read station into the output stacker.

8. When the input hoppwr is empty, observe that the HOPPER CHECK indicator lights, the green RESET indicator goes out, and the read STOP indicator lights.

9. The card reader may now be operated locally or switched to on-line operation.

5.12.4.2 On-Line Operation – The following procedure is used to place the CR11 Card Reader System on-line. When placed on-line, the system is controlled by programmed commands from the VAX-11/780.

1. Ensure that the card reader is operational by performing the procedure given in Paragraph 5.10.4.1.

2. Ensure that the output stacker is empty. Load the input hopper with the cards to be read.

3. Set the MODE switch to REMOTE.

4. Press the RESET switch and observe that the associated green indicator lights. The system is now on-line.

5. If the system goes off-line because of an error alarm, it can be placed back on-line by correcting the error and then pressing the RESET switch.

   Any time it is desired to go off-line, press the STOP switch.

APPENDIX A
VIRTUAL MEMORY CONCEPTS

Virtual memory provides major advantages over traditional computer system designs. Virtual memory benefits are:

1. Allows the user to run programs larger than the size of real memory
2. Eliminates overlays
3. Is transparent to the user
4. Provides direct improvement in performance as real memory is added.

The programmer is freed from restrictive programming practices and memory constraints. However, the programmer may fine-tune program performance and efficiency.

The following paragraphs develop some basic concepts (of virtual memory systems) that are prerequisite to an understanding of the VAX-11/780 system design.

A.1 VIRTUAL ADDRESS

A source program is composed of data and instructions that are processed by a compiler or assembler to produce executable code. During compiling or assembly, the data and instructions of the program are allocated virtual addresses (addresses used to identify or represent memory locations). For example, in Figure A-1, source Program A is processed and allocated virtual addresses that start at address 0 and continue through address 1000.

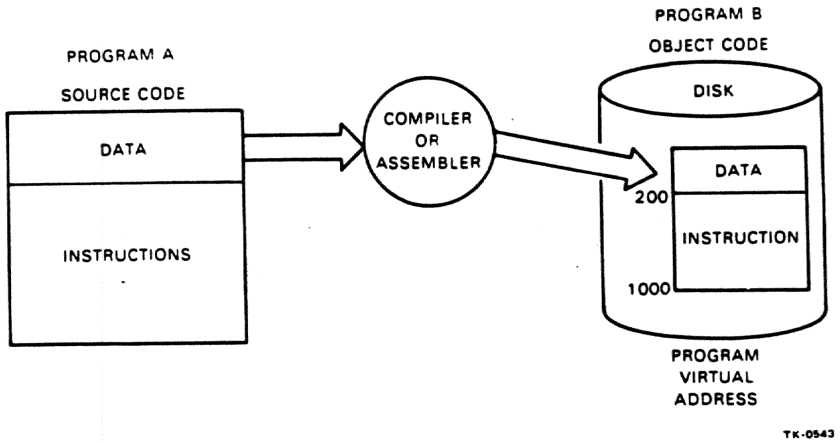


Figure A-1 Program Virtual Address

For the object program to be executed it must be loaded into physical memory. The simplest kind of program loading is absolute program loading. A software program called an absolute loader loads the object program into the same physical addresses as the program's virtual addresses. In this case, the virtual address is the same as the physical address. Additional mechanisms in the hardware and software are required to make the virtual address anything different from the physical address.

A system that uses absolute loading has limited flexibility because a program is always loaded into the same physical addresses as its virtual addresses. Separately compiled or assembled routines all start with virtual address 0. To combine several routines into one program, the addresses of each routine must be changed so they do not conflict.

A.2 PROGRAM LINKING

Linking is the process of combining object programs (e.g., a main routine and several subroutines). The linker program relocates (changes) the virtual addresses of each routine to prepare the combined programs for loading into memory. The programmer can instruct the linker to assign specific virtual addresses to programs, so that two or more programs will not use the same virtual addresses.

This kind of relocation is static address relocation (done at link time), and is the linking process available in systems that have no memory management or relocation hardware. Static address relocation can be used also to resolve addressing conflicts between separate programs when they reside concurrently in unmapped physical memory. This technique is used by multiprogramming operating systems.

A.3 MAPPING

Dynamic address relocation (sometimes called mapping) is done after program loading. Mapping continually relocates addresses in the program during program execution. Special mapping hardware is used to change a program's virtual addresses into physical addresses used by the hardware.

Figure A-2 shows three separate programs, A, B, and C, each of which has different virtual memory requirements starting at address 0.

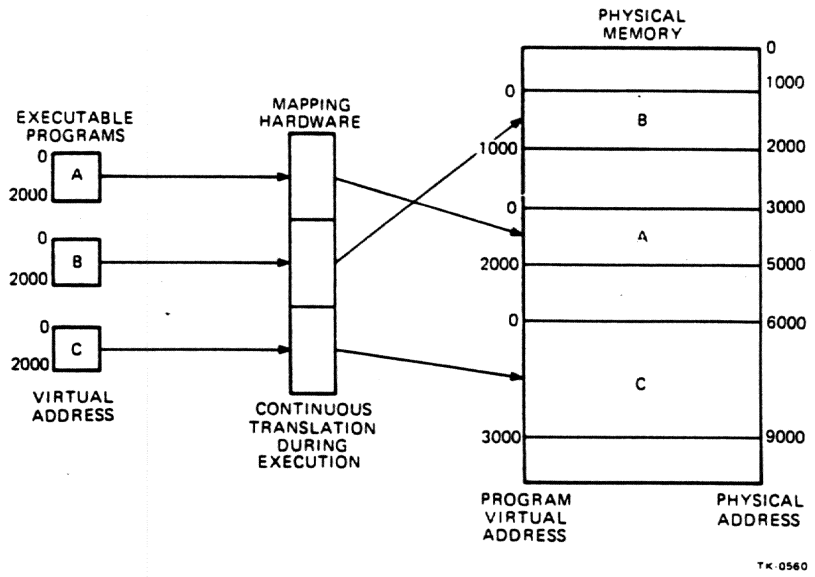


Figure A-2 Mapping

A benefit of mapping hardware is that a program can be run in different areas of physical memory without relinking. The relocation is done at run-time by special mapping hardware. The hardware is very fast and does not significantly degrade the performance of the program.

A.4 MULTIPROGRAMMING

In order to make full use of computer hardware, it is often desirable to have several programs in memory concurrently. This technique is called multiprogramming. Mapping is often used to resolve address conflicts between programs in a multiprogramming environment.

A.5 MEMORY FRAGMENTATION

Memory fragmentation is a problem that occurs when the available memory is broken into segments that are too small to accommodate each program in its entirety.

Figure A-3 shows three executable programs, A, B, and C. Program A is 4000 bytes long, B is 4000 bytes long, and C is 2000 bytes long. The available physical memory in this example is 11000 bytes, so there is clearly sufficient physical memory to hold all three programs. However, if program A has been loaded at address 1000 and B at address 6000 as shown here, there is no single contiguous 2000 byte area for C.

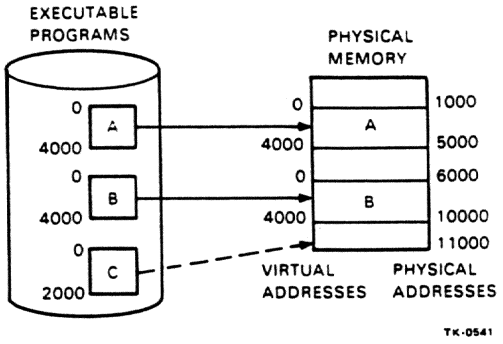


Figure A-3 Physical Memory Fragmentation

One solution employed, when holes develop in memory is to shuffle programs toward one end of memory, leaving a contiguous area of available memory at the other end.

Another solution is to map programs in segments that are smaller than the entire program.

A.6 NONCONTIGUOUS MAPPING

Although programs have been shown as being mapped as a single unit, it is not a requirement that the program be mapped as one unit.

Breaking the program into units and mapping the units separately is called noncontiguous mapping. Because the program is mapped in several pieces, the program need not be loaded in contiguous memory. Mapping executable programs in small increments reduces the problem of memory fragmentation because there is a higher probability of finding areas of memory that are large enough to hold the smaller units of a program.

A.7 PAGED MAPPING

A logical extension of the concepts of noncontiguous mapping is the concept of paged mapping. Paged mapping refers to the mapping of programs and data in units that have a fixed length. These units are called pages.

Mapping a program in pages further reduces memory fragmentation. Since memory is mapped in equal length pages, all free memory is divided into pages and is the correct size for pages to be loaded.

Figure A-4 illustrates a program composed of three sections, main routine A, and subroutines B and C. These routines are subdivided into fixed length pieces (pages); A is shown as being two pages long, B four pages long, and C two pages long.

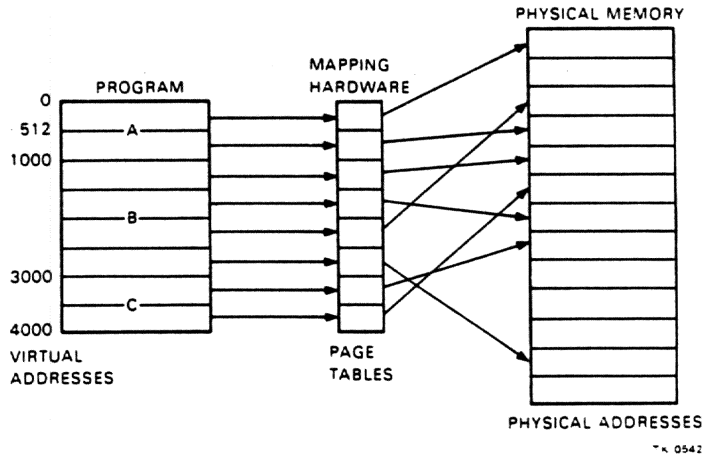


Figure A-4 Paged Mapping

Mapping hardware can now locate each page separately.

One of the benefits of mapping, and paged mapping in particular, is the ability for programs to easily share data or code. The specification of data or code to be shared is done explicitly by the programmer at link time. When the programs are physically loaded into memory, the operating system recognizes those pages to be shared between two or more programs and maps the common data or code to the same area of physical memory.

So far, discussion has been limited to situations in which the virtual memory requirements of the program(s) involved do not exceed the physical memory capabilities of the system. What if a program's virtual memory requirement is greater than the available physical memory of the system?

A.8 OVERLAYING

Overlay structures have been used to overcome the limitation of not enough memory. Overlays are logical segments of a program and are specified at link-time. The overlay structure defines a root section (which is always resident in memory), and two or more overlay sections (which are loaded into the same memory during program execution).

In Figure A-5, the root section is linked for addresses 0–1000, each overlay is linked for addresses 1000–2000. In this scheme, overlays 1A and 1B share the same physical memory, but at different times during the execution of the program. The overlay (either 1A or 1B) is loaded from disk as needed.

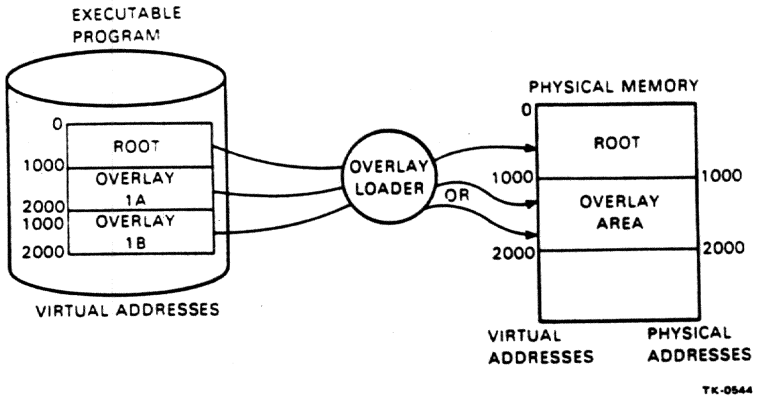


Figure A-5 Overlays

A different type of overlay scheme can be used when the available physical memory is larger than the virtual address space (Figure A-6).

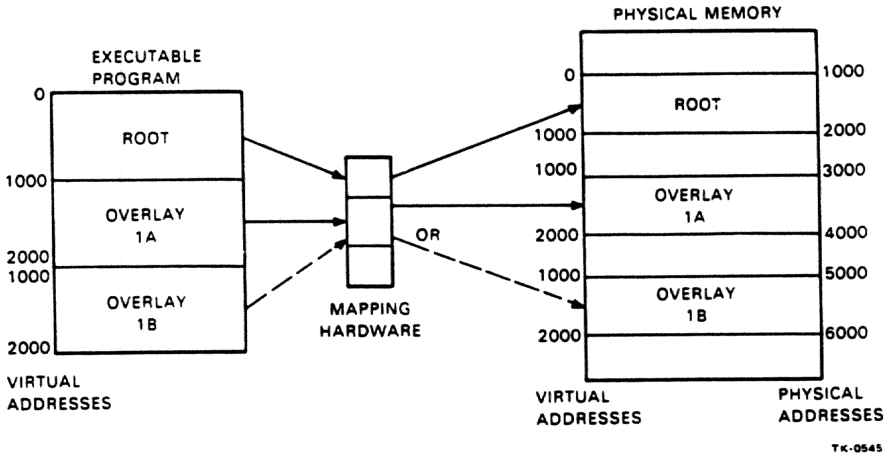


Figure A-6 Mapping Overlays

This example shows the same root and overlay sections; but in this case, mapping hardware does continual virtual to physical address translation at run time. Overlays 1A and 1B may be in physical memory at the same time. The programmer must still define the overlay structure at link time. This process is known as the in-core overlay technique, which is implemented by program logical address space (PLAS).

Thus, defining overlay structures and using overlay mapping are two ways of circumventing the problem of small virtual memory. However, in each case, it is the programmer's responsibility to plan and specify an overlay structure and to ensure that it works properly.

A.9 PROCESS

Programs on the VAX-11/780 system are run as part of a process. A process is like a task on other systems. When a program is running, it is known to VAX/VMS as a process.

A.10 PAGING

As the concept of overlaying has implied, it is not necessary to have all of a program resident in memory at one time. In fact, only those segments that are actually required for execution at any given time need be resident.

In the VAX-11/780 system, the set of segments, called pages, that reside in memory for any given process is called the process working set. Usually the working set is a subset of the total number of pages of the entire process, and will change over time as the program runs.

As a process executes, pages are brought into memory as required. Eventually, as execution proceeds, the process references pages that are not in memory. Whenever a process references a page not in memory, a page fault occurs. In order for execution to continue, the required page must be brought into memory. This is done by the operating system, which is transparent to the user. This is called paging.

As shown in Figure A-7, pages move back and forth between a file on disk and physical memory under the control of a page fault handler, which is part of the operating system.

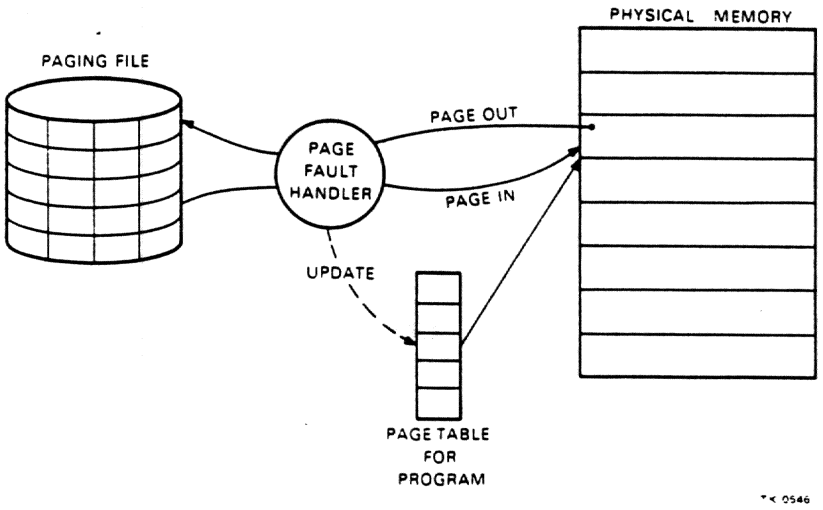


Figure A-7 Paging

On the VAX-11/780 operating system, all processes have a maximum working set size established for them. The maximum working set size is the total number of pages that a particular process may have resident in physical memory at any given time. The maximum working set for each process is decided by the system manager, based on both total number of users and the size of physical memory.

In order to avoid exceeding the maximum working set size for a process, a page from the current working set of the process may have to be removed.

The VAX-11/780 uses process-oriented paging. It discards least-used pages from the process in operation in order to make room for new pages. This is called paging against the process. Because a process in VAX/VMS pages only against itself, performance of individual processes can be controlled by the user. The performance of each process is more predictable than with system-wide paging.

A.11 SWAPPING

In a multiprogramming system, where two or more processes may be in physical memory at the same time, the term balance set is used to define the number of processes that are in memory. The balance set may be less than or equal to the total number of active processes in the system.

The number of processes in the balance set is controlled by the system and is determined by the amount of available physical memory. The system tries to balance the number of active processes in memory against the resources that are available to serve them.

It is the function of the scheduler, a system service, to select one process for execution from those processes currently in the balance set.

Figure A-8 shows five processes A, B, C, D, and E, all currently active. However, only the working sets of processes A, B, and C are in physical memory (i.e., in the balance set). The working sets of processes D and E are not in physical memory; consequently, processes D and E are not in the balance set.

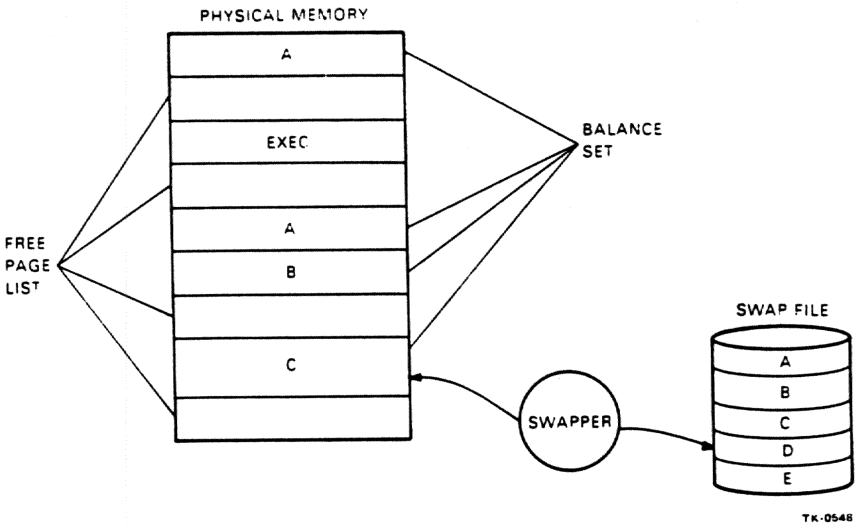


Figure A-8 Balance Set

In order for nonresident processes D or E to be scheduled to execute, they must first be brought into the balance set. To avoid exceeding the number of programs allowed in the balance set processes D and E must be substituted (swapped) for active processes already in the balance set.

Swapping, then, is the moving of entire working sets between physical memory and disk so that a nonresident process can be scheduled to execute.

Note the contrast between swapping and paging. Swapping is the moving of the working set (i.e., all the pages that make up the working set) of a process to allow a new process to be scheduled. Paging is the replacement of just a single page to allow a single process to continue to execute.

APPENDIX B
BOOTSTRAPPING SEQUENCES

The VAX-11/780 system can be bootstrapped from a number of sets of conditions. Operator participation in the bootstrapping operation is limited. All bootstrap sequences are performed with the control panel key selector switch in the LOCAL position.

Console and CPU power is applied and the AUTO RESTART switch is OFF. This is an automatic bootstrap sequence that is performed when power is applied. When completed, the console subsystem is booted and operating in the console I/O mode. The CPU subsystem is not booted. The sequence for this bootstrap is as follows.

1. Apply power to the console and CPU subsystems. Set the five position rotary key selector switch to LOCAL. (Disk ZZ-ESZAB-7-0 must be installed in the RX01 disk drive.)

2. The console boot program (resident in the LSI-11 ROM) is initiated.

3. The console boot begins loading the console program from the disk to the LSI-11 RAM. When loading is completed, the console program is executed.

4. The console boot displays the VAX-11/780 status. During this operation, the following data is printed on the terminal

   CPU HALTED, SOMM CLR

5. The console is placed in the console I/O mode of operation. The CPU is initialized to the architecture defined state. When the initializing of the CPU is completed, the following data is printed on the terminal

   INIT SEQ DONE (CARRIAGE RETURN) HALTED AT 000000

6. The WCS/ECO microcode is loaded from the disk to the writable control store (WCS) area in the CPU subsystem. When the WCS is loaded, the following data is printed on the terminal

   (RELOADING WCS) CARRIAGE RETURN LOAD DONE,

   NNN BYTES LOADED

7. The console program tests the status of the AUTO RESTART switch. If the switch is ON, skip to the sequence described in the next paragraph (Step 8). If the switch is OFF, continue to Step 8 of this paragraph.

8. The console remains in the console I/O mode of operation and the terminal prints

   >>>

   This is an input prompt signifying that the console subsystem is booted and operator console command language commands can be entered.

Console and CPU power is applied and the AUTO RESTART switch is ON. This is an automatic boot sequence that is performed when the power is applied. When completed, the VAX-11/780 is booted and operated in the program I/O mode. The sequence for this bootstrap is as follows.

1-7. This is the same sequence as described in Steps 1 through 7 of the preceding paragraph.

8. The console program initiates the restart referee program (contained in a CPU ROM) and the following is printed by the terminal

   (AUTO RESTART)

9. The restart referee tests to determine if this is a cold start or a warm start.

10. If this boot is part of a warm start, the restart referee goes to VAX/VMS/SEP restart vector and the following is printed on the terminal

    VAX/UMS/SEP prompt

    The VAX-11/780 is ready for user input.

11. If this boot is part of a cold start, the restart referee returns to the console program to obtain the default reading for the system load device.

12. The VAX-11/780 primary boot (on the RX01 disk) is loaded to VAX-11 memory and started. The console enters program I/O mode.

13. The VAX-11/780 primary boot initiated the memory sizer. The memory sizer searches memory for 128K bytes of contiguous main memory.

APPENDIX C
CONSOLE HELP FILE

The console help file contains a list of the console command language commands and a brief description of the action resulting from using the command. A printout of the console help file is obtained by entering HELP at the console terminal with the console in the console I/O mode (enter CTRL P if a console prompt is not present). A sample printout of the console help file is given below.

```
>>>HELP
!VAX-11/780 CONSOLE HELP FILE REV. 5 27-JAN-1978
!TO STOP PRINTING, TYPE ↑C
!FOR ABBREVIATION RULES, TYPE '@ABBREV.HLP'
!FOR ERROR MESSAGE HELP, '@ERROR.HLP'
!FOR REMOTE ACCESS HELP, TYPE '@REMOTE.HLP'
!GENERAL: <ADDRESS> IS A <NUMBER>, OR ONE OF THE FOLLOWING
!    SYMBOLIC <ADDRESSES>(ONLY FOR EXAMINE & DEPOSIT
!    COMMANDS)
!    'R0,R1,R2.....R11,AP,FP,SP,PC' (GENERAL REGISTERS)
!    'PSL'              (PROCESSOR STATUS WORD)
!    '*'                (LAST ADDRESS)
!    '+'                (ADDRESS FOLLOWING 'LAST' (*) ADDRESS)
!    '-'                (ADDRESS PRECEDING 'LAST' (*) ADDRESS)
!    '@'                (USES LAST EXAMINE/DEPOSIT DATA FOR ADDRESS)
!
!    <NUMBER> = STRING OF DIGITS IN CURRENT DEFAULT RADIX,OR
!    STRING OF  DIGITS PREFIXED WITH A DEFAULT RADIX
!    OVERRIDE.(%O FOR OCTAL, %X FOR HEX)
!
!    ALL COMMANDS ARE TERMINATED BY CARRIAGE RETURN
!'EXAMINE <ADDRESS>'              -DISPLAYS CONTENTS OF <ADDRESS>
!'DEPOSIT <ADDRESS> <DATA>'       -DEPOSITS <DATA> TO <ADDRESS>
!    USE A QUALIFIER AFTER THE COMMAND NAME TO SPECIFY
!    THE PROPER ADDRESS SPACE TO USE:
!    '/P'               FOR PHYSICAL MEMORY (THE DEFAULT)
!    '/V'               FOR VIRTUAL MEMORY
!    '/I'               FOR INTERNAL (PROCESSOR) REGISTERS
!    '/G'               FOR GENERAL REGISTERS 0 THRU F (R0 THRU PC)
!    '/VB'              FOR VBUS REGISTERS
!    '/ID'              FOR IDBUS REGISTERS
!    EXAMPLE: TO EXAMINE VIRTUAL ADDRESS 10245, THE SHORTEST
!    UNIQUE COMMAND STRING IS: 'E/V 10245' (SEE ABBREV.HLP)
!
```

```
! 'EXAMINE IR'                    -EXAMINES INSTRUCTION REG. (IR). DISPLAYS
!                                 OP CODE. SPECIFIER. & EXECUTION POINT
!                                 COUNTER
! 'START <ADDRESS>'               -INITIALIZES THE CPU. DEPOSITS <ADDRESS> TO
!                                 THE PC. ISSUES A CONTINUE TO THE ISP.
! 'CONTINUE'                      -ISSUES A CONTINUE TO THE ISP.
! 'HALT'                          -HALTS THE ISP
! 'BOOT'                          -BOOTS THE CPU FROM DEFAULT DEVICE
! 'INITIALIZE'                    -INITIALIZES THE CPU
! 'SHOW'                          -SHOWS CONSOLE AND CPU STATE
! 'SHOW VERSION'                  -SHOWS VERSIONS OF MICROCODE AND CONSOLE
! 'TEST'                          -RUNS MICRODIAGNOSTICS
! 'TEST/COM'                      -LOADS MICRODIAGNOSTICS. AWAITS COMMANDS
! 'UNJAM'                         -UNJAMS THE SBI
! 'SET STEP BUS'                  -ENABLE SINGLE BUS CYCLE CLOCK MODE
! 'SET STEP STATE'                -ENABLE SINGLE TIME STATE CLOCK MODE
! 'SET STEP INSTUCTION'           -ENABLES SINGLE INSTRUCTION MODE
! 'CLEAR STEP'                    -ENABLE NORMAL (NO STEP) MODE
! 'NEXT <NUMBER>' -<NUMBER> STEP CYCLES ARE DONE. TYPE OF
!                                 STEP DEPENDS ON LAST 'SET STEP' COMMAND
! 'QCLEAR <ADDRESS>'              -DOES A QUAD CLEAR TO <ADDRESS>. WHICH IS
!                                 FORCED TO A QUADWORD BOUNDARY (CLEARS ECC
!                                 ERRORS)
! 'SET SOMM'                      -SET 'STOP ON MICROMATCH' ENABLE
! 'CLEAR SOMM'                    -CLEAR 'STOP ON MICROMATCH' ENABLE
!                       NOTE: ID REGISTER 21 IS THE MICROMATCH REGISTER.
! 'SET CLOCK SLOW'  -SET CPU CLOCK FREQ TO SLOW
! 'SET CLOCK FAST'  -SET CPU CLOCK FREQ TO FAST
! 'SET CLOCK NORMAL'              -SET CPU CLOCK FREQ TO NORMAL
! 'SET RELOCATION:<NUMBER>'             -PUT <NUMBER> INTO CONSOLE'S
!                                 RELOCATION REGISTER. RELOCATION REGISTER
!                                 IS ADDED TO EFFECTIVE ADDRESS OF PHYSICAL
!                                 AND VIRTUAL EXAMINES AND DEPOSITS.
! 'SET DEFAULT <OPTION>,...,<OPTION>'
-SET CONSOLE DEFAULTS
!                       NOTE:     <OPTIONS> ARE: OCTAL,HEX,PHYSICAL,VIRTUAL,
!                                 INTERNAL,GENERAL,VBUS,IDBUS,BYTE,
!                                 WORD,LONG,QUAD
! 'SET TERMINAL FILL: <NUMBER>'         -SET FILL COUNT FOR # OF BLANKS
!                                 WRITTEN TO THE TERMINAL AFTER
!                                 <CR> OR <LF>
! 'SET TERMINAL PROGRAM'          -PUT CONSOLE TTY INTO 'PROGRAM I/O' !
                        MODE
! '↑P' (CONTROL-P)      -PUT CONSOLE TTY INTO 'CONSOLE I/O' MODE
!                                 (UNLESS MODE SWITCH IN 'DISABLE')
! 'HELP'                          -PRINTS THIS FILE
! '@<FILENAME>'                   -PROCESS AN INDIRECT COMMAND FILE
! 'LOAD <FILENAME>'               -LOAD FILE TO MAIN MEMORY.
! 'LOAD/WCS <FILENAME>'           -LOAD FILE SPECIFIED TO WCS
```

```
!                    NOTE:   THE '/START:<ADDRESS>' QUALIFIER MAY ALSO !
!                            BE USED TO SPECIFY THE STARTING ADDRESS FOR
!                            A LOAD. OTHERWISE LOAD WILL BEGIN WITH
!                            LOCATION 0.
! 'LINK'                     -CAUSES CONSOLE TO BEGIN COMMAND LINKING.
!                            CONSOLE PRINTS REVERSED PROMPT TO INDICATE
!                            LINKING. ALL COMMANDS TYPED BY USER WHILE
!                            LINKING ARE STORED IN AN INDIRECT COMMAND
!                            FILE FOR LATER EXECUTION. CONTROL-C
!                            TERMINATES LINKING. (SEE PERFORM)
! 'PERFORM'                  -EXECUTE A FILE OF LINKED COMMANDS
!                            PREVIOUSLY GENERATED VIA A 'LINK' COMMAND.
! 'REPEAT <ANY-CONSOLE-COMMAND>'    -CAUSES THE CONSOLE TO
!                            REPEATEDLY EXECUTE THE <CONSOLE-COMMAND>.
!                            UNTIL STOPPED BY A ↑C
! 'WCS'                      -CALLS MICRO-DEBUGGER. (FOR DEBUGGER HELP.
!                            TYPE '@WCSMON.HLP')
! 'ENABLE DX1:'              -ENABLES CONSOLE SOFTWARE TO ACCESS FLOPPY
!                            DRIVE 1 ON THOSE SYSTEMS WITH DUAL
!                            FLOPPIES.
! 'REBOOT'                   -CAUSES A CONSOLE SOFTWARE RELOAD
! 'WAIT DONE'                -WHEN EXECUTED FROM AN INDIRECT COMMAND
!                            FILE, THIS COMMAND WILL CAUSE COMMAND FILE
!                            EXECUTION TO STOP UNTIL: A) A 'DONE' SIGNAL
!                            IS RECEIVED FROM THE PROGRAM RUNNING IN THE
!                            VAX-11/780 (COMMAND FILE EXECUTION WILL
!                            CONTINUE), OR B) THE VAX-11/780 HALTS, OR
!                            OPERATOR TYPES A ↑C (COMMAND FILE EXECUTION
!                            WILL TERMINATE).
! <END-OF-CONSOL.HLP>
<@EOP>
<@EXIT>

>>>
```

APPENDIX D
CONSOLE ABBREVIATIONS HELP FILE

The console abbreviations help file contains a list of the console command language commands and the shortest abbreviation recognized as that command. A printout of the console abbreviation help file is obtained by entering @ABBREV.HLP at the console terminal with the console in the console I/O mode (enter CTRL P if a console prompt is not present). A sample of the console abbreviation help file is given below.

```
>>>@ABBREV>HLP
    ?FILE NAME ERR
    ?'ABBREV>HLP' IS INCORRECT
                        >>>@ABBREV>HLP.....
! VAX-11/780 CONSOLE ABBREVIATION RULES REV-5 8-NOV-77
! THIS FILE SHOWS THE SHORTEST UNIQUE COMMAND STRINGS THAT WILL BE .
! RECOGNIZED AS THE CONSOLE COMMAND LISTED.
!
!    COMMAND                          SHORTEST ABBREVIATION RECOGNIZED
!'EXAMINE <ADDRESS>'                   'E <ADDRESS>'
!'DEPOSIT <ADDRESS> <DATA>'            'D <ADDRESS> <DATA>'
!'START <ADDRESS>'                     'S <ADDRESS>'
!'CONTINUE'                            'C'
!'HALT'                                'H'
!'HELP'                                'HE'
!'BOOT'                                'B'
!'INITIALIZE'                          'I'
!'SHOW'                                'SH'
!'SHOW VERSION'                        'SH V'
!'TEST'                                'T'
!'UNJAM'                               'U'
!'SET RELOCATION:<NUMBER>'             'SE R:<NUMBER>'
!'SET STEP BUS'                        'SE S B'
!'SET STEP STATE'                      'SE S S'
!'SET STEP INSTRUCTION'                'SE S I'
!'CLEAR STEP'                          'CL S'
!'NEXT <NUMBER>'                       'N <NUMBER>'
!'QCLEAR <ADDRESS>'                    'Q <ADDRESS>'
!'SET SOMM'                            'SE SO'
!'CLEAR SOMM'                          'CL SO'
!'SET CLOCK FAST'                      'SE C F'
!'SET CLOCK SLOW'                      'SE C S'
!'SET CLOCK NORMAL'                    'SE C N'
```

```
! '@<FILENAME>'                          '@<FILENAME>'
! 'LOAD <FILENAME>'                      'L <FILENAME>'
! 'LINK'                                 'LI'
! 'PERFORM'                              'P'
! 'REPEAT <CONSOLE-COMMAND>'             'R <CONSOLE-COMMAND>'
! 'WCS'                                  'W'
! 'ENABLE DX1:'                          'EN DX1:'
! 'REBOOT'                               'REB'
! 'SET TERMINAL FILL:<NUMBER>'           'SE T F:<NUMBER>'
! 'SET TERMINAL PROGRAM'                 'SE T PR'
! 'SET DEFAULT <OPTION-LIST>'            'SE D <OPTION-LIST>'
! 'WAIT DONE'                            'WA D'
!
! QUALIFIERS                             SHORTEST ABBREVIATION RECOGNIZED
! /BYTE                                  /B
! /WORD                                  /W
! /LONG                                  /L
! /QUAD                                  /Q
! /OCTAL                                 /O
! /HEX                                   /H
! /PHYSICAL                              /P
! /VIRTUAL                               /V
! /INTERNAL                              /I
! /GENERAL                               /G
! /VBUS                                  /VB
! /IDBUS                                 /ID
! /WCS                                   /WC
! /NEXT:<NUMBER>                         /N:<NUMBER>
! /COMMAND                               /C
! /START:<ADDRESS>                       /S:<ADDRESS>
! <END-OF-ABBREV.HLP>
<@EOF>
<@EXIT>

>>>
```

APPENDIX E
CONSOLE ERROR MESSAGE HELP FILE

The console error message help file contains a list of the console error messages and a brief description of what the error message means. A printout of the console error message help file is obtained by entering @ERROR.HLP at the console terminal with the console in the console I/O mode (enter CTRL P if a console prompt is not present). A sample printout of the console errors message help file is given below.

```
>>>@ERROR.HLP
!VAX-11/780 ERROR MESSAGE HELP FILE   REV-2   8-NOV-77
!
! ?'<TEXT-STRING>' IS INCOMPLETE
!     THE <TEXT-STRING> IS NOT A COMPLETE CONSOLE COMMAND
! ?'<TEXT-STRING>' IS INCORRECT
!     THE <TEXT-STRING> IS NOT RECOGNIZED AS PART OF A COMMAND
! ?FILE NAME ERR
!     A <FILE-NAME> GIVEN WITH A COMMAND CAN NOT BE TRANSLATED
!     TO RAD50
! ?FILE NOT FOUND
!     A <FILE-NAME> GIVEN WITH A 'LOAD' OR '@' COMMAND DOES NOT
!     MATCH ANY FILE ON THE CURRENTLY LOADED FLOPPY DISK. CAN
!     ALSO BE GENERATED BY 'HELP', 'BOOT', OR AN ATTEMPTED WCS
!     LOAD IF HELP FILE, BOOT FILE, OR WCS FILE IS MISSING FROM
!     FLOPPY.
! ?NO CPU RESPONSE
!     CONSOLE TIMED-OUT WAITING FOR A RESPONSE FROM CPU
! ?CPU NOT IN COSOLE WAIT LOOP, COMMAND ABORTED
!     A CONSOLE COMMAND REQUIRING ASSISTANCE FROM CPU WAS
!     ISSUED WHILE THE CPU WAS NOT IN THE CONSOLE SERVICE LOOP
! ?CPU CLK STOP, COMMAND ABORTED
!     A CONSOLE COMMAND THAT REQUIRES THE CPU CLOCK TO BE
!     RUNNING WAS ISSUED WHILE THE CLOCK WAS STOPPED
! ?FLOPPY ERR, CODE=X
!     THE CONSOLE FLOPPY DRIVER DETECTED AN ERROR. CODES ARE AS
!     FOLLOWS: (CODES ARE ALWAYS PRINTED IN HEX RADIX)
!     CODE=1          FLOPPY HARDWARE ERROR (CRC,PARITY,ETC)
!     CODE=2          FILE NOT FOUND
!     CODE=3          FLOPPY DRIVER QUEUE OVERFLOW
!     CODE=4          CONSOLE SOFTWARE REQUESTED IN ILLEGAL SECTOR
!                     NUMBER
! ?FLOPPY NOT READY
```

```
!     THE CONSOLE FLOPPY DRIVE FAILED TO BECOME READY WHEN
!     BOOTING.
! ?NO BOOT ON FLOPPY
!     CONSOLE ATTEMPTED TO BOOT FROM A FLOPPY THAT DOES NOT
!     CONTAIN A VALID BOOT BLOCK.
! ?FLOPPY ERROR ON BOOT
!     A FLOPPY ERROR WAS DETECTED WHILE ATTEMPTING A CONSOLE
!     BOOT.
! ?MIC-ERR ON FUNCTION
!     A MICROERROR OCCURRED IN THE CPU WHILE SERVICING A
!     CONSOLE REQUEST. SBI ERROR REGISTERS ARE DUMPED AFTER
!     THIS MESSAGE IS PRINTED.
! ?INT-REG ERR
!     A MICROERROR OCCURRED WHILE ATTEMPTING TO REFERENCE A
!     CPU INTERNAL (PROCESSOR) REGISTER. AN ILLEGAL ADDRESS
!     WILL CAUSE THIS ERROR.
! ?MICROERR, CODE=X
!     AN UNRECOGNIZED MICROERROR OCCURRED. THE CODE RETURNED
!     BY THE CPU IS NOT IN THE RANGE OF RECOGNIZED ERROR CODES.
!     'X' IS THE CODE THAT WAS RETURNED BY THE CPU.
! ?INT-STK INVLD
!     THE CPU HALTED BECAUSE THE INTERRUPT STACK WAS MARKED
!     INVALID
! ?ILL I/E VEC
!     THE CPU DETECTED AN ILLEGAL INTERRUPT/EXCEPTION VECTOR
! ?NO USR WCS
!     CPU DETECTED AN INTERRUPT/EXCEPTION VECTOR TO USER WCS
!     AND NO USER WCS EXISTS
! ?CHM ERR
!     A CHANGE MODE INSTRUCTION WAS ATTEMPTED FROM THE
!     INTERRUPT STACK
! ?MEM-MAN FAULT, CODE=XX
!     A VIRTUAL EXAMINE OR DEPOSIT CAUSED AN ERROR IN THE
!     MEMORY MANAGEMENT MICROROUTINE. 'XX' IS A ONE BYTE
!     ERROR CODE RETURNED BY THE ROUTINE, WITH THE FOLLOWING
!     BIT ASSIGNMENTS:
!     BIT 0 = LENGTH VIOLATIONS (BITS NUMBERED FROM RIGHT)
!     BIT 1 = FAULT WAS ON A PTE REFERENCE
!     BIT 2 = WRITE OR MODIFY INTENT
!     BIT 3 = ACCESS VIOLATION
!     BIT 4 THRU 7 SHOULD BE IGNORED
! ?IND-COM ERR
!     THE CONSOLE DETECTED AN ERROR IN THE FORMAT OF AN
!     INDIRECT COMMAND FILE. POSSIBLE ERRORS ARE: 1) MORE THAN
!     80 CHARACTERS IN AN INDIRECT COMMAND LINE, 2) A COMMAND
!     LINE DID NOT END WITH A CARRIAGE RETURN AND LINE FEED.
! INT PENDING
!     THIS IS NOT ACTUALLY AN ERROR, BUT INDICATES THAT AN
!     ERROR WAS PENDING AT THE TIME THAT A CONSOLE-REQUESTED
!     HALT WAS PERFORMED. CONTINUE CPU TO CLEAR INTERRUPT.
```

```
! ?WARNING-WCS & FPLA VER MISMATCH
!     THE MICROCODE IN WCS IS NOT COMPATIBLE WITH FPLA. THIS
!     MESSAGE IS PRINTED ON EACH ISP START OR CONTINUE, BUT
!     NO OTHER ACTION TAKEN BY CONSOLE.
! ?FATAL-WCS & PCS VER MISMATCH
!     THE MICROCODE IN PCS IS NOT COMPATIBLE WITH THAT IN WCS.
!     ISP START AND CONTINUE ARE DISABLED BY CONSOLE.
! ?MICROMACHINE TIME OUT
!     INDICATES THAT THE VAX-11/780 MICROMACHINE HAS FAILED TO
!     STROBE INTERRUPTS WITHIN THE MAX TIME PERIOD ALLOWED.
! ?REMOTE ACCESS NOT SUPPORTED
!     PRINTED WHEN CONSOLE MODE SWITCH ENTERS A 'REMOTE'
!     POSITION, AND THE REMOTE SUPPORT SOFTWARE IS NOT INCLUDED
!     IN THE CONSOLE.
! ?TRAP-4, RESTARTING CONSOLE
!     THE CONSOLE TOOK A TIME-OUT TRAP. CONSOLE WILL RESTART.
! ?UNEXPECTED TRAP
! MOUNT CONSOLE FLOPPY, THEN TYPE ↑C
!     CONSOLE TRAPPED TO AN UNUSED VECTOR. CONSOLE REBOOTS WHEN
!     ↑C TYPED
! '?Q0BLKD'
!     CONSOLE'S TERMINAL OUTPUT QUEUE IS BLOCKED. CONSOLE WILL
!     REBOOT.
! <END-OF-ERROR.HLP>

<@EOF>
<@EXIT>

>>>@REMOTE.HLP
!VAX-11/780 CONSOLE - REMOTE ACCESS HELP FILE REV-01 28-DEC-77
!
! /ENABLE TALK/                    -ESTABLISH TERMINAL TO TERMINAL
!                                  COMMUNICATION BETWEEN LOCAL AND
!                                  REMOTE TERMINAL. KEYS STRUCK ON ONE
!                                  TERMINAL ARE PRINTED ON THE OTHER.
!                                  CONTROL-P TERMINATES TALK.
! 'ENABLE ECHO'                    -CAUSES CHARACTERS TYPED IN TALK MODE
!                                  TO BE ECHOED BACK TO THE ORIGINAL
!                                  TERMINAL.
! 'ENABLE LOCAL COPY'              -CAUSES THE LOCAL TERMINAL TO GET A
!                                  COPY OF OUTPUT BEING SENT TO REMOTE
!                                  TERMINAL.
! 'ENABLE CARRIER ERROR'           -CAUSE CONSOLE TO PRINT '?CARRIER'
!                                  LOST WHEN A LOSS OF CARRIER IS
!                                  DETECTED AT REMOTE INTERFACE.
! 'DISABLE ECHO/'                  -INHIBITS ECHO OF CHARACTERS TYPED IN
!                                  TALK MODE.
! 'DISABLE LOCAL COPY'             -DISABLE LOCAL TERMINAL FROM
!                                  RECEIVING COPY OF OUTPUT TO REMOTE
```

```
!                                       TERMINAL.
! 'DISABLE CARRIER ERROR'               -CAUSES CONSOLE TO INHIBIT PRINTING
!                                       CARRIER DETECTED.
! 'ENABLE LOCAL FLOPPY'                 -(AFFECTS PROTOCOL OPERATION ONLY) ON
!                                       ATTEMPT TO OPEN A FILE, THE DIRECTORY
!                                       OF LOCAL FLOPPY WILL BE SEARCHED
!                                       FIRST. IF FILE IS NOT FOUND, 'REMOTE'
!                                       FLOPPY'S DIRECTORY IS SEARCHED FOR
!                                       FILE.
! 'DISABLE LOCAL FLOPPY'                -(AFFECTS PROTOCOL OPERATION ONLY) ON
!                                       AN ATTEMPT TO OPEN A FILE, THE FILE !
IS SEARCHED FOR ON THE 'REMOTE' !       FLOPPY ONLY.
! END-OF-REMOTE.HLP

<@EOF>
<@EXIT>

>>>
```

APPENDIX F
MICRODEBUGGER HELP FILE

The microdebugger help file contains a list of microdiagnostic program command language and a brief description of the action resulting from using the command. A printout of the microdebugger help file is obtained by entering '@WCSMON.HLP.' A sample printout of the microdebugger is given below.

```
    @WCSMON.HLP
!MICRODEBUGGER HELP FILE REV-0 MAY 1977
!TO STOP PRINTING, TYPE ≥C
!
! DEBUGGER COMMANDS (ALL TERMINATED BY CARRIAGE RETURN)
!
! `E/P <ADDRESS>`-EXAMINE PHYSICAL MEMORY
! `E/ID <ADDRESS>`                        -EXAMINE ID BUS REGISTER
!
! `E <ADDRESS>`   -EXAMINE WCS LOCATION, DISPLAY
!                                         ALL FIELDS
! `E <ADDRESS> <FIELDNAME-1>, <FIELDNAME-2>,,,, <FIELDNAME-N>`
!     EXAMINE WCS LOCATION, DISPLAY ONLY FIELDS THE
!     FIELDS SPECIFIED.
!     NOTE: <FIELDNAMES> =                ACR,ACM,ADS,ALU,BEN,BMX,CCK,
!                                         CID,DK,DT,EAL,EBM,FEK,FS,IBS,
!                                         IEK,UJM,KMX,MCT,MSC,PCK,QK,RMX,
!                                         SCK,SGN,SHF,SI,SMX,SPO,USU,VAK
! `E RA <ADDRESS>`                        -EXAMINE AN RA REGISTER
! `E RC <ADDRESS>`                        -EXAMINE AN RC REGISTER
!
! `E <SYMBOLIC-NAME>`                     -EXAMINE ONE OF THE
!                                         SYMBOLICALLY NAMED REGISTERS
!     NOTE: <SYMBOLIC-NAMES> =            DR,FER,IBA,LA,LB,LC,Q,RL,SC,
!                                         SR,UPC
!
! `D/P <ADDRESS> <DATA>`                  -DEPOSIT <DATA> TO PHYSICAL
!                                         MEMORY
! `D/ID <ADDRESS> <DATA>`                 -DEPOSIT <DATA> TO ID BUS
!                                         REGISTER
!
! `D <ADDRESS> <FIELDNAME-1> <DATA-1>, <FIELDNAME-2> <DATA-2>`,....
!                                         -DEPOSIT TO WCS LOCATION,
!                                         PUTTING <DATA-1> INTO
!                                         <FIELDNAME-1>, ETC.
```

```
!                                       UNSPECIFIED FIELDS ARE
!                                       UNCHANGED.
!    NOTE:          THE '/S' QUALIFIER MAY BE USED TO CAUSE ALL
!                   UNSPECIFIED FIELDS TO BE CLEARED.
!
! 'D RA <ADDRESS> <DATA>'               -DEPOSIT <DATA> TO AN RA
!                                       REGISTER
! 'D RC <ADDRESS> <DATA>'               -DEPOSIT <DATA> TO AN RC
!                                       REGISTER
!
! 'D <SYMBOLIC-NAME> <DATA>'            -DEPOSIT <DATA> TO ONE OF THE
!                                       SYMBOLICALLY NAMED REGISTERS
!                                       (SEE LIST ABOVE).
!    NOTE: DEPOSITS TO THE LOG STACK (RL) ARE NOT SUPPORTED.
!
! 'CONTINUE'                            -RESUME MICROINSTRUCTION
!                                       EXECUTION AS SPECIFIED BY
!                                       CONTENTS OF MICRO-PC (UPC)
!
! 'START <ADDRESS>'                     -START MICROSEQUENCER AT
!                                       <ADDRESS>.
!
! 'HALT'                                -HALT THE MICROSEQUENCER
!
! 'SET SOMM'                            -SET THE 'STOP ON MICROMATCH'
!                                       ENABLE
! 'CLEAR SOMM'    -CLEAR THE 'STOP ON MICROMATCH'
!                                       ENABLE
!
! 'SET STEP'                            -ENABLE SINGLE MICROINSTRUCTION
!                                       STEP MODE. START OR CONTINUE
!                                       WILL ALLOW ONE MICROINSTRUCTION
!                                       TO EXECUTE, THEN HALT THE
!                                       MICROSEQUENCER.
!
! 'CLEAR STEP'      -DISABLE SINGLE MICROINSTRUCTION
!                                       STEP MODE.
!
! 'RETURN'                              -RETURN TO THE CONSOLE PROGRAM
!
! 'OPEN <FILENAME>'                     -OPEN SPECIFIED FILE ON FLOPPY
!                                       DRIVE 0
! 'OPEN DX1:<FILENAME>'                 -OPEN SPECIFIED FILE ON FLOPPY
!                                       DRIVE 1
!    NOTE:          'OPEN' IS USED TO SPECIFY A FILE CONTAINING THE
!                   MICROCODE CURRENTLY LOADED IN THE WCS PORTION OF
!                   THE CONTROL STORE. (ADDRESSES 1000(16) & UP IN THE
!                   CONTROL STORE)
!                   THIS FILE WILL BE USED FOR ALL EXAMINES OF THE WCS,
!                   SINCE THE WCS IS NOT DIRECTLY READABLE.
```

```
! <END-OF-WCSMON.HLP>

<@EOF>
<@EXIT>

>>>
```

APPENDIX G
GLOSSARY

abort
An exception that occurs in the middle of an instruction that could leave the registers and memory in a state, such that the instruction cannot be restarted.

absolute indexed mode
An indexed addressing mode in which the base operand is addressed in absolute mode.

absolute mode
In absolute mode addressing, the program counter (PC) is used as the register in autoincrement deferred mode. The PC contains the address of the actual operand.

absolute time
Time values expressing a specific date (month, day, and year) and time of day. Absolute time values are always expressed in the system as positive numbers.

access mode
1. Any of the four processor access modes in which software executes. Processor access modes are, in order from most to least privileged and protected: kernel (mode 0), executive (mode 1), supervisor (mode 2), and user (mode 3). When the processor is in kernel mode, the executing software has complete control of, and responsibility for, the system. When the processor is in any other mode, the processor is inhibited from executing privileged instructions. The processor status longword contains the current access mode field. The operating system uses access modes to define protection levels for software executing in the context of a process. For example, the executive runs in kernel and executive mode and is most protected. The command interpreter is less protected and runs in supervisor mode. The debugger runs in user mode and is not more protected than normal user programs.

2. See record access mode.

access type
1. The way in which the processor accesses instruction operands. Access types are: read, write, modify, address, and branch.

2. The way in which a procedure accesses its arguments.

access violation
An attempt to reference an address that is not mapped into virtual memory or an attempt to reference an address that is not accessible by the current access mode.

account name
A string that identifies a particular account used to accumulate data on a job's resource use. This name is the user's accounting charge number, not the user's user identification code (UIC).

address
A number used by the operating system and user software to identify a storage location. See also virtual address and physical address.

address access type
The specified operand of an instruction is not directly accessed by the instruction. The address of the specified operand is the actual instruction operand. The context of the address calculation is given by the data type of operand.

addressing mode
The way in which an operand is specified; for example, the way in which the effective address of an instruction operand is calculated using the general registers. The basic general register addressing modes are called: register, register deferred, autoincrement, autoincrement deferred, autodecrement, displacement, and displacement deferred. In addition, there are six indexed addressing modes using two general registers and literal mode addressing. The PC addressing modes are called: immediate (for register deferred mode using the PC), absolute (for autoincrement deferred mode using the PC), and branch.

address space
The set of all possible addresses available to a process. Virtual address space refers to the set of all possible virtual addresses. Physical address space refers to the set of all possible physical addresses sent out on the SBI.

allocate device
To reserve a particular device unit for exclusive use. A user process can allocate a device only when that device is not allocated by any other process.

alphanumeric character
An upper or lower case letter (A–Z, a–z), a dollar sign ($), an underscore (_), or a decimal digit (0.9).

American standard code for information interchange (ASCII)
A set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol.

ancillary control process (ACP)
A process that acts as an interface between user software and an I/O driver. An ACP provides functions supplemental to those performed in the driver, such as file and directory management. Three examples of ACPs are: the Files-11 ACP (F11ACP), the magnetic tape ACP (MTACP), and the networks ACP (NETACP).

argument pointer
General register 12 (R12). By convention, argument pointer contains the address of the base of the argument list for procedures initiated using the CALL instructions.

assign a channel
To establish the necessary software linkage between a user process and a device unit before a user process can transfer any data to or from that device. A user process requests the system to assign a channel and the system returns a channel number.

asynchronous
A mode of activity that operates without respect to a clock. For example, asynchronous hardware uses ready and done signals to schedule operations rather than time intervals. If two activities are asynchronous, the second can begin at a random interval with respect to the first.

asynchronous record operation
A mode of record processing in which a user program can continue to execute after issuing a record retrieval or storage request without having to wait for the request to be fulfilled.

asynchronous system trap (AST)
A software-simulated interrupt to a user-defined service routine. ASTs enable a user process to be notified asynchronously with respect to its execution of the occurrence of a specific event. If a user process has defined an AST routine for an event, the system interrupts the process and executes the AST routine when that event occurs. When the AST routine exits, the system resumes the process at the point where it was interrupted.

asynchronous system trap level (ASTLVL)
A value kept in an internal processor register that is the highest access mode for which an AST is pending. The AST does not occur until the current access mode drops in priority (raises in numeric value) to a value greater than or equal to ASTLVL. Thus, an AST for an access mode will not be serviced while the processor is executing in a higher priority access mode.

authorization file
See user authorization file.

autodecrement indexed mode
An indexed addressing mode in which the base operand specifier uses autodecrement mode addressing.

autodecrement mode
In autodecrement mode addressing, the contents of the selected register are decremented and the result is used as the address of the actual operand for the instruction. The contents of the register are decremented according to the data type context of the register: 1 for byte, 2 for word, 4 for longword and floating, 8 for quadword and double floating.

autoincrement deferred indexed mode
An indexed addressing mode in which the base operand specifier uses autoincrement deferred mode addressing.

autoincrement deferred mode
In autoincrement deferred mode addressing, the specified register contains the address of a longword that contains the address of the actual operand. The contents of the register are incremented by 4 (the number of bytes in a longword). If the PC is used as the register, this mode is called absolute mode.

autoincrement indexed mode
An indexed addressing mode in which the base operand specifier uses autoincrement mode addressing.

autoincrement mode
In autoincrement mode addressing, the contents of the specified register are used as the address of the operand, then the contents of the register are incremented by the size of the operand.

balance set
The set of all process working sets currently resident in physical memory. The processes whose working sets are in the balance set have memory requirements that balance with available memory. The balance set is maintained by the system swapper process.

base operand address
The address of the base of a table or array referenced by index mode addressing.

base operand specifier
The register used to calculate the base operand address of a table or array referenced by index mode addressing.

base priority
The process priority that the system assigns a process when it is created. The scheduler never schedules a process below its base priority. The base priority can be modified only by the system manager or the process itself.

base register
A general register used to contain the address of the first entry in a list, table, array, or other data structure.

binding
See linking.

bit string
See variable-length bit field.

block
1. The smallest addressable unit of data that the specified device can transfer in an I/O operation (512 contiguous bytes for most disk devices).

2. An arbitrary number of contiguous bytes used to store logically related status, control, or other processing information.

block I/O
A data accessing technique in which the program manipulates the blocks (physical records) that make up a file, instead of its logical records.

bootstrap block
A block in the index file on a system disk that contains a program that can load the operating system into memory and start its execution.

branch access type
An instruction attribute which indicates that the processor does not reference an operand address, but that the operand is a branch displacement. The size of the branch displacement is given by the data type of the operand.

branch mode
In branch addressing mode, the instruction operand specifier is a signed byte or word displacement. The displacement is added to the contents of the updated PC (which is the address of the first byte beyond the displacement), and the result is the branch address.

bucket
A storage structure of 1 to 32 blocks that is used to store and transfer blocks of data in files with a relative file organization.

bucket locking
A facility that prevents access to any record in a bucket by more than one user until that user releases the bucket.

buffered I/O
See system buffered I/O.

bug check
The operating system's internal diagnostic check. When so indicated by the check the system logs the failure and crashes the system.

byte
A byte is eight contiguous bits starting on an addressable byte boundary. Bits are numbered from the right, 0 through 7, with bit 0 the low-order bit. When interpreted arithmetically, a byte is a two's complement integer with significance increasing from bits 0 through 6. Bit 7 is the sign bit. The value of the signed integer is in the range -128 to 127 decimal. When interpreted as an unsigned integer, significance increases from bits 0 through 7 and the value of the unsigned integer is in the range 0 to 255 decimal. A byte can be used to store one ASCII character.

cache memory
A small, high-speed memory placed between slower main memory and the processor. A cache increases effective memory transfer rates and processor speed. It contains copies of data recently used by the processor, and fetches several bytes of data from memory in anticipation that the processor will access the next sequential series of bytes.

call frame
See stack frame.

call instructions
The processor instructions CALLG (call procedure with general argument list) and CALLS (call procedure with stack argument list).

call stack
The stack, and conventional stack structure, used during a procedure call. Each access mode of each process context has one call stack, and interrupt service context has one call stack.

channel
A logical path connecting a user process to a physical device unit. A user process requests the operating system to assign a channel to a device so the process can transfer data to or from that device.

character
A symbol represented by an ASCII code. See also alphanumeric character.

character string
A contiguous set of bytes. A character string is identified by two attributes: an address and a length. Its address is the address of the byte containing the first character of the string. Subsequent characters are stored in bytes of increasing addresses. The length is the number of characters in the string.

character string descriptor
A quadword data structure used for passing character data (strings). The first word of the quadword contains the length of the character string. The second word can contain type information. The remaining longword contains the address of the string.

cluster
1. A set of contiguous blocks that is the basic unit of space allocation on a Files-11 disk volume.
2. A set of pages brought into memory in one paging operation.
3. An event flag cluster.

command
An instruction, generally an English word, typed by the user at a terminal or included in a command file that requests the software monitoring a terminal or reading a command file to perform some well-defined activity. For example, typing the COPY command requests the system to copy the contents of one file into another file.

command file
A file containing command strings. See also command procedure.

command interpreter
Procedure-based system code that executes in supervisor mode in the context of a process to receive, syntax check, and parse commands typed by the user at a terminal or submitted in a command file.

command parameter
The positional operand of a command delimited by spaces, such as a file specification, option, or constant.

command procedure
A file containing commands and data that the command interpreter can accept in lieu of the user typing the commands individually on a terminal.

command string
A line (or set of continued lines), normally terminated by typing the carriage return key, containing a command and, optionally, information modifying the command. A complete command string consists of a command, its qualifiers, if any, and its parameters (file specifications, for example), if any, and their qualifiers, if any.

common
A FORTRAN term for a program section that contains only data.

common event flag cluster
A set of 32 event flags that enables cooperating processes to post event notification to each other. Common event flag clusters are created as they are needed. A process can associate with up to two common event flag clusters.

compatibility mode
A mode of execution that enables the central processor to execute nonprivileged PDP-11 instructions. The operating system supports compatibility mode execution by providing an RSX-11M programming environment for an RSX-11M task image. The operating system compatibility mode procedures reside in the control region of the process executing a compatibility mode image. The procedures intercept calls to the RSX-11M executive and convert them to the appropriate operating system functions.

compiler
A program that translates a program written in a high-level language (such as FORTRAN or BASIC) into an object program.

condition
An exception condition detected and declared by software. For example, see failure exception mode.

condition codes
Four bits in the processor status word that indicate the results of previously executed instructions.

condition handler
A procedure that is called upon by the system when an exception condition occurs. When an exception condition occurs, the operating system searches for a condition handler and, if found, initiates the condition handler immediately. The condition handler may perform some action to change the situation that caused the exception condition and continue execution for the process that incurred the exception condition. Condition handlers execute in the context of the process at the access mode of the code that incurred the exception condition.

condition value
A 32-bit quantity that uniquely identifies an exception condition.

context
The environment of an activity. See also process context, hardware context, and software context.

context indexing
The ability to index through a data structure automatically because the size of the data type is known and used to determine the offset factor.

context switching
Interrupting the activity in progress and switching to another activity. Context switching occurs as one process after another is scheduled for execution. The operating system saves the hardware context of the interrupted process in its hardware process control block (PCB) using the save process context instruction. It also loads hardware PCB into the hardware context of another process using the load process context instruction, scheduling that process for execution.

continuation character
A hyphen at the end of a command line signifying that the command string continues on to the next command line.

console
The manual control unit integrated into the central processor. The console includes an LSI-11 microprocessor, a DX01 floppy disk, and a serial line interface connected to an LA36 terminal. It enables the operator to start and stop the system, monitor system operation, and run diagnostics.

console terminal
The hard copy terminal (LA36) connected to the central processor console.

control region
The highest addressed half of virtual memory per-process space (the P1 region). Control region virtual addresses refer to the process-related information used by the system to control the process, such as: the kernel, executive, and supervisor stacks, the permanent I/O channels, exception vectors, and dynamically used system procedures (such as the command interpreter and RSX-11M programming environment compatibility mode procedures). The user stack is also normally found in the control region, although it can be located elsewhere.

control region base register (P1BR)
The processor register, or its equivalent in a hardware process control block, that contains the base virtual address of a process control region page table.

control region length register (P1LR)
The processor register, or its equivalent in a hardware process control block, that contains the number of nonexistent page table entries for virtual pages in a process control region.

copy-on-reference
A method used in memory management for sharing data until a process accesses it, in which case it is copied before the access. Copy-on-reference allows sharing of the initial values of a global section whose pages have read/write access but contain pre-initialized data available to many processes.

counted string
A data structure consisting of a byte-sized length followed by the string. Although a counted string is not used as a procedure argument, it is a convenient representation in memory.

current access mode
The processor access mode of the currently executing software. The current mode field of the processor status longword indicates the access mode of the currently executing software.

cylinder
The tracks at the same radius on all recording surfaces of a disk.

data base
1. All the occurrences of data described by a data base management system.
2. A collection of related data structures.

data structure
Any table, list, array, queue, or tree whose format and access conventions are well-defined for reference by one or more images.

data type
In general, the way in which bits are grouped and interpreted. In reference to the processor instructions, the data type of an operand identifies the size of the operand and the significance of the bits in the operand. Operand data types include: byte, word longword, and quadword integer, floating and double floating, character string, packed decimal string, and variable-length bit field.

deferred echo
Refers to the fact that terminal echoing does not occur until a process is ready to accept input entered by type ahead.

delta time
A time value expressing an offset from the current date and time. Delta times are always expressed in the system as negative numbers whose absolute value is used as an offset from the current time.

demand zero page
A page, typically of an image stack or buffer area, that is initialized to contain all zeros when dynamically created in memory as a result of a page fault. This feature eliminates the waste of disk space that would otherwise be required to store blocks (pages) that contain only zeros.

descriptor
A data structure used in calling sequences for passing argument types, addresses, and other optional information. See character string descriptor.

detached process
A process that has no owner. The parent process of a tree of subprocesses. Detached processes are created by the job controller when a user logs on the system or when a batch job is initiated. The job controller does not own the user processes it creates; these processes therefore are detached.

device
The general name for any physical terminus or link connected to the processor that is capable of receiving, storing, or transmitting data. Card readers, line printers, and terminals are examples of record-oriented devices. Magnetic tape devices and disk devices are examples of mass storage devices. Terminal line interfaces and interprocessor links are examples of communications devices.

device interrupt
An interrupt received on interrupt priority level 16 through 23. Device interrupts can be requested only by devices, controllers, and memories.

device name
The field in a file specification that identifies the device unit on which a file is stored. Device names also include the mnemonics that identify an I/O peripheral device in a data transfer request. A device name consists of a mnemonic followed by a controller identification letter (if applicable), followed by a unit number (if applicable). A colon (:) separates it from following fields.

device queue
See spool queue.

device register
A location in device controller log used to request device functions (such as I/O transfers) and/or report status.

device unit
One drive, and its controlling logic, of a mass storage device system. A mass storage system can have several drives connected to it.

diagnostic
A program that tests logic and reports any faults it detects.

direct I/O
An I/O operation in which the system locks the pages containing the associated buffer in memory for the duration of the I/O operation. The I/O transfer takes place directly from the process buffer. Contrasts with system buffered I/O.

direct mapping cache
A cache organization in which only one address comparison is needed to locate any data in the cache because any block of main memory data can be placed in only one possible position in the cache. Contrasts with fully associative cache.

directory
A file used to locate files on a volume that contains a list of file names (including type and version number) and their unique internal identifications.

directory name
The field in a file specification that identifies the directory file in which a file is listed. The directory name begins with a left bracket ([or <) and ends with a right bracket (] or >).

displacement deferred indexed mode
An indexed addressing mode in which the base operand specifier uses displacement deferred mode addressing.

displacement deferred mode
In displacement deferred mode addressing, the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of a longword that contains the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The base address is the address of the first byte beyond the specifier extension.

displacement indexed mode
An indexed addressing mode in which the base operand specifier uses displacement mode addressing.

displacement mode
In displacement mode addressing, the specifier extension is a byte, word, or longword displacement. The displacement is sign extended to 32 bits and added to a base address obtained from the specified register. The result is the address of the actual operand. If the PC is used as the register, the updated contents of the PC are used as the base address. The base address is the address of the first byte beyond the specifier extension.

double floating datum
Eight contiguous bytes (64 bits), starting on an addressable byte boundary, which are interpreted as containing a floating point number. The bits are labeled from right to left, 0 to 63. A four-word floating-point number is identified by the address of the byte containing bit 0. Bit 15 contains the sign of the number. Bits 14 through 7 contain the excess 128 binary exponent. Bits 63 through 16 and 6 through 0 contain a normalized 56-bit fraction with the redundant most significant fraction bit not represented. Within the fraction, bits of decreasing significance go from 6 through 0, 31 through 16, 47 through 32, then 63 through 48. Exponent values of 1 through 255 in the 8-bit exponent field represent true binary exponents of –128 to 127. An exponent value of 0, together with a sign bit of 0, represent a floating value of 0. An exponent value of 0 with a sign bit of 1 is a reserved representation; floating-point instructions processing this value return a reserved operand fault. The value of a floating datum is in the approximate range (+ or –) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 2^{55} or sixteen decimal digits.

drive
The electromechanical unit of a mass storage device system on which a recording medium (disk cartridge, disk pack, or magnetic tape reel) is mounted.

driver
The set of code that handles physical I/O to a device.

dynamic access
A technique in which a program switches from one record access mode to another while processing a file.

echo
A terminal handling characteristic in which the characters typed by the terminal user on the keyboard are also displayed on the screen or printer.

effective address
The address obtained after indirect or indexing modifications are calculated.

entry mask
A word whose bits represent the registers to be saved or restored on a subroutine or procedure call using the call and return instructions.

entry point
A location that can be specified as the object of a call. It contains an entry mask and exception enables known as the entry point mask.

equivalence name
The string associated with a logical name in a logical name table. An equivalence name can be, for example, a device name, another logical name, or a logical name concatenated with a portion of a file specification.

error logger
A system process that empties the error log buffers and writes the error messages into the error file. Errors logged by the system include memory system errors, device errors and timeouts, and interrupts with invalid vector addresses.

escape sequence
An escape is a transition from the normal mode of operation to a mode outside the normal mode. An escape character is the code that indicates the transition from normal to escape mode. An escape sequence refers to the set of character combinations starting with an escape character that the terminal transmits without interpretation to the software set up to handle escape sequences.

event
A change in process status or an indication of the occurrence of some activity that concerns an individual process or cooperating processes. An incident reported to the scheduler that affects the ability of a process to execute. Events can be synchronous with the process's execution (a wait request), or they can be asynchronous (I/O completion). Some other events include: swapping, wake request, page fault.

event flag
A bit in an event flag cluster that can be set or cleared to indicate the occurrence of the event associated with that flag. Event flags are used to synchronize activities in a process or among many processes.

event flag cluster
A set of 32 event flags that are used for event posting. Four clusters are defined for each process: two process-local clusters, and two common event flag clusters. Of the process-local flags, eight are re served for system use.

exception
An event detected by the hardware (other than an interrupt or jump, branch, case, or call instruction) that changes the normal flow of instruction execution. An exception is always caused by the execution of an instruction or set of instructions (whereas an interrupt is caused by an activity in the system independent of the current instruction). There are three types of hardware exceptions: traps, faults, and aborts. Examples are: attempts to execute a privileged or reserved instruction, trace traps, compatibility mode faults, breakpoint instruction execution, and arithmetic traps such as overflow, underflow, and divide by zero.

exception condition
A hardware- or software-detected event other than an interrupt or jump, branch, case, or call instruction that changes the normal flow of instruction execution.

exception dispatcher
An operating system procedure that searches for a condition handler when an exception condition occurs. If no exception handler is found for the exception or condition, the image that incurred the exception is terminated.

exception enables
See trap enables.

exception vector
See vector.

executable image
An image that is capable of being run in a process. When run, an executable image is read from a file for execution in a process.

executive
The generic name for the collection of procedures included in the operating system software that provides the basic control and monitoring functions of the operating system.

executive mode
The second most privileged processor access mode (mode 1). The record management services (RMS) and many of the operating system's programmed service procedures execute in executive mode.

exit
An image exit is a rundown activity that occurs when image execution terminates either normally or abnormally. Image rundown activities include deassigning I/O channels and disassociation of common event flag clusters. Any user-or-system-specified exit handlers are called.

exit handler
A procedure executed when an image exits. An exit handler enables a procedure that is not on the call stack to gain control and clean up procedure-own data bases before the actual image exit occurs.

extended attribute block (XAB)
A record management service user data structure that contains additional file attributes beyond those expressed in the file access block (FAB), such as boundary types (aligned on cylinder, logical block number, virtual block number) and file protection information.

extension
The amount of space to allocate at the end of a file each time a sequential write exceeds the allocated length of the file.

extent
The contiguous area on a disk containing a file or a portion of a file. Consists of one or more clusters.

failure exception mode
A mode of execution selected by a process indicating that it wants an exception condition declared if an error occurs as the result of a system service call. The normal mode is for the system service to return an error status code for which the process must test.

fault
A hardware exception condition that occurs in the middle of an instruction and leaves the registers and memory in a consistent state, such that elimination of the fault and restarting the instruction will give correct results.

field
1. See variable-length bit field.
2. A set of contiguous bytes in a logical record.

file access block (FAB)
A record management service user data structure that represents a request for data operations related to the file as a whole, such as OPEN, CLOSE, or CREATE.

file header
A block in the index file describing a file on a Files-11 disk structure. The file header identifies the locations of the file's extents. There is a file header for every file on the disk.

file name
The field preceding a file type in a file specification that contains a 1- to 9-character logical name for a file.

file name extension
See file type.

file organization
The particular file structure used to record the data comprising a file on a mass storage medium. Record management services (RMS) file organizations are: sequential, relative, direct, and indexed.

Files-11
The name of the on-disk structure used by the RSX-11, IAS and VAX operating systems. Volumes created under this structure are transportable between these operating systems.

file specification
A unique name for a file on a mass storage medium. It identifies the node, the device, the directory name, the file name, the file type, and the version number under which a file is stored.

file structure
The way in which the blocks forming a file are distributed on a disk or magnetic tape to provide a physical accessing technique suitable for the way in which the data in the file is processed.

file system
A method of recording, cataloging, and accessing files on a volume.

file type
The field in a file specification that is preceded by a period or a dot (.) and consists of a zero- to three-character type identification. By convention, the type identifies a generic class of files that have the same use or characteristics, such as ASCII text files, binary object files, etc.

fixed control area
An area associated with a variable length record available for controlling or assisting record access operations. Typical uses include line numbers and printer format control information.

fixed length record format
A file format in which all records have the same length.

floating (point) datum
Four contiguous bytes (32 bits) starting on an addressable byte boundary. The bits are labeled from right to left from 0 to 31. A two-word floating-point number is identified by the address of the byte containing bit 0. Bit 15 contains the sign of the number. Bits 14 through 7 contain the excess 128 binary exponent. Bits 31 through 16 and 6 through 0 contain a normalized 24-bit fraction with the redundant most significant fraction bit not represented. Within the fraction, bits of decreasing significance go from bit 6 through 0, then 31 through 16. Exponent values of 1 through 255 in the 8-bit exponent field

represent true binary exponents of −128 to 127. An exponent value of 0, together with a sign bit of 0, represent a floating value of 0. An exponent value of 0, with a sign bit of 1, is a reserved representation; floating-point instructions processing this value return a reserved operand fault. The value of a floating datum is in the approximate range (+ or −) 0.29×10^{-38} to 1.7×10^{38}. The precision is approximately one part in 2^{23} or seven decimal digits.

foreign volume
Any volume other than a Files-11 formatted volume that may or may not be file structured.

fork process
A dynamically created system process such as a process that executes device driver code or the timer process. Fork processes have minimal context. Fork processes are scheduled by the hardware rather than by the software. The timer process is dispatched directly by software interrupt. I/O driver processes are dispatched by a fork dispatcher. Fork processes execute at software interrupt levels and are dispatched for execution immediately. Fork processes remain resident until they terminate.

frame pointer
General register 13 (R13). By convention, the frame pointer (FP) contains the base address of the most recent call frame on the stack.

fully associative cache
A cache organization in which any block of data from main memory can be placed anywhere in the cache. Address comparison must take place against each block in the cache to find any particular block. Contrasts with direct mapping cache.

general register
Any of the sixteen 32-bit registers used as the primary operands of the native mode instructions. The general registers include 12 general purpose registers that can be used as accumulators, as counters, and as pointers to locations in main memory, and the frame pointer (FP), argument pointer (AP), stack pointer (SP), and program counter (PC) registers.

generic device name
A device name that identifies the type of device but not a particular unit; a device name in which the specific controller and/or unit number is omitted.

giga
Metric term used to represent the number 1 followed by nine zeros. Equivalent to billions.

global page table
The page table containing the master page table entries for global sections.

global section
A data structure (e.g., FORTRAN global common) or sharable image section potentially available to all processes in the system. Access is protected by privilege and/or group number of the user identification code (UIC).

global symbol
A symbol defined in a module that is potentially available for reference by another module. The linker resolves (matches references with definitions) global symbols. Contrasts with local symbol.

global symbol table (GST)
In a library, an index of defined global symbols used to access the modules defining the global symbols. The linker will also put global symbol tables into an image. For example, the linker appends a global symbol table to executable images that are intended to run under the symbolic debugger, and it appends a global symbol table to all sharable images.

group

1. A set of users who have special access privileges to each other's directories and files within those directories (unless protected otherwise), as in the context "system, owner, group, world," where group refers to all members of a particular owner's group.

2. A set of jobs (processes and their subprocesses) that have access privileges to a group's common event flags and logical name tables, and may have mutual process controlling privileges, such as scheduling, hibernation, etc.

group number
The first number in a user identification code (UIC).

hardware context
The values contained in the following registers while a process is executing: the program counter (PC); the processor status longword (PSL); the 14 general registers (R0 through R13); the four processor registers (P0BR, P0LR, P1BR, and P1LR) that describe the process virtual address space; the stack pointer (SP) for the current access mode in which the processor is executing; plus the contents to be loaded in the stack pointer for every access mode other than the current access mode. While a process is executing, its hardware context is continually being updated by the processor. While a process is not executing, its hardware context is stored in its hardware process control block (PCB).

hardware process control block (PCB)
A data structure known to the processor that contains the hardware context when a process is not executing. A process's hardware PCB resides in its process header.

hibernation
A state in which a process is inactive, but known to the system with all of its current status. A hibernating process becomes active again when a wake request is issued. It can schedule a wake request before hibernating, or another process can issue its wake request. A hibernating process also becomes active for the time sufficient to service any asynchronous system trap (AST) it may receive while it is hibernating. Contrasts with suspension.

home block
A block in the index file that contains the volume identification, such as volume label and protection.

image
An image consists of procedures and data that have been bound together by the linker. There are three types of images: executable, sharable, and system.

image activator
A set of system procedures that prepares an image for execution. The image activator establishes the memory management data structures required both to map the image's virtual pages to physical pages and to perform paging.

image exit
See exit.

image I/O segment
That portion of the control region that contains the record management services (RMS) internal file access blocks (IFAB) and I/O buffers for the image currently being executed by a process.

image name
The file name of the file in which an image is stored.

image privileges
The privileges assigned to an image when it is linked. See process privileges.

image section (isect)
A group of program sections (psects) with the same attributes (such as read-only access, read/write access, absolute, relocatable, etc.) that is the unit of virtual memory allocation for an image.

immediate mode
In immediate mode addressing, the program counter (PC) is used as the register in autoincrement mode addressing.

indexed addressing mode
In indexed mode addressing, two registers are used to determine the actual instruction operand: an index register and a base operand specifier. The contents of the index register are used as an index (offset) into a table or array. The base operand specifier supplies the base address of the array (the base operand address or BOA). The address of the actual operand is calculated by multiplying the contents of the index register by the size (in bytes) of the actual operand and adding the result to the base operand address. The addressing modes resulting from index mode addressing are formed by adding the suffix "indexed" to the addressing mode of the base operand specifier: register deferred indexed, autoincrement indexed, autoincrement deferred indexed (or absolute indexed), autodecrement indexed, displacement indexed, and displacement deferred indexed.

indexed file organization
A file organization in which a file contains records and a primary key index (and optionally one or more alternate key indices) used to process the records sequentially by index or randomly by index.

index file
The file on a Files-11 volume that contains the access information for all files on the volume and enables the operating system to identify and access the volume.

index file bit map
A table in the index file of a Files-11 volume that indicates which file headers are in use.

index register
A register used to contain an address offset.

indirect command file
See command procedure.

input stream
The source of commands and data, i.e., the user's terminal, the batch stream, or an indirect command file.

instruction buffer
An 8-byte buffer in the processor used to contain bytes of the instruction currently being decoded and to pre-fetch instructions in the instruction stream. The control logic continuously fetches data from memory to keep the 8-byte buffer full.

interleaving
Assigning consecutive physical memory addresses alternately between two memory controllers.

interprocess communication facility
A common event flag, mailbox, or global section used to pass information between two or more processes.

interrecord gap
A blank space deliberately placed between data records on the recording surface of a magnetic tape.

interrupt
An event other than an exception or branch, jump, case, or call instruction that changes the normal flow of instruction execution. Interrupts are generally external to the process executing when the interrupt occurs. See also device interrupt, software interrupt, and urgent interrupt.

interrupt priority level (IPL)
The interrupt level at which the processor executes when an interrupt is generated. There are 31 possible interrupt priority levels. IPL 1 is lowest, 31 highest. The levels arbitrate contention for processor service. For example, a device cannot interrupt the processor if the processor is currently executing at an interrupt priority level greater than the interrupt priority level of the device's interrupt service routine.

interrupt service routine
The routine executed when a device interrupt occurs.

interrupt stack
The system-wide stack used when executing in interrupt service context. At any time, the processor is either in a process context executing in user, supervisor, executive, or kernel mode; or in system-wide interrupt service context operating with kernel privileges, as indicated by the interrupt stack and current mode bits in the PSL. The interrupt stack is not context switched.

interrupt stack pointer
The stack pointer for the interrupt stack. Unlike the stack pointers for process context stacks, which are stored in the hardware PCB, the interrupt stack pointer is stored in an internal register.

interrupt vector
See vector.

I/O driver
See driver.

I/O function
An I/O operation that is interpreted by the operating system and typically results in one or more physical I/O operations.

I/O function code
A 6-bit value specified in a queue I/O request system service that describes the particular I/O operation to be performed (e.g., read, write, rewind).

I/O function modifier
A 10-bit value specified in a queue I/O request system service that modifies an I/O function code (e.g., read terminal input no echo).

I/O lockdown
The state of a page such that it cannot be paged or swapped out of memory until any I/O in progress to that page is completed.

I/O rundown
An operating system function in which the system cleans up any I/O in progress when an image exits.

I/O space
The region of physical address space that contains the configuration registers, and device control/status and data registers.

I/O status block
A data structure associated with the queue I/O request system service. This service optionally returns a status code, number of bytes transferred, and device- and function-dependent information in an I/O status block. It is not returned from the service call, but filled in when the I/O request completes.

job
1. A job is the accounting unit equivalent to a process and the collection of all the subprocesses, if any, that it and its subprocesses create. Jobs are classified as batch and interactive. For example, the job controller creates an interactive job to handle a user's requests when the user logs onto the system and it creates a batch job when the symbiont manager passes a command input file to it.

2. A print job.

job controller
The system process that establishes a job's process context, starts a process running the LOGIN image for the job, maintains the accounting record for the job, manages symbionts, and terminates a process and its subprocesses.

job queue
A list of files that a process has supplied for processing by a specific device, for example, a line printer.

kernel mode
The most privileged processor access mode (mode 0). The operating system's most privileged services, such as I/O drivers and the pager, run in kernel mode.

librarian
A program that allows the user to create, update, modify, list, and maintain object library, image library, and assembler macro library files.

library file
A direct access file containing one or more modules of the same module type.

limit
The size or number of given items requiring system resources (such as mailboxes, locked pages, I/O requests, open files, etc.) that a job is allowed to have at any one time during execution, as specified by the system manager in the user authorization file. See also quota.

line number
A number used to identify a line of text in a file processed by a text editor.

linker
A program that reads one or more object files created by language processors and produces an executable image file, a sharable image file, or a system image file.

linking
The resolution of external references between object modules used to create an image, the acquisition of referenced library routines, service entry points, and data for the image, and the assignment of virtual addresses to components of an image.

literal mode
In literal mode addressing, the instruction operand is a constant whose value is expressed in a 6-bit field of the instruction. If the operand data type is byte, word, longword, or quadword, the operand is zero-extended and can express values in the range 0 through 63 (decimal). If the operand data type is floating or double floating, the 6-bit field is composed of two 3-bit fields, one for the exponent and the other for the fraction. The operand is extended to floating or double floating format.

locality
See program locality.

local symbol
A symbol meaningful only to the module that defines it. Symbols not identified to a language processor as global symbols are considered to be local symbols. A language processor resolves (matches references with definitions) local symbols. They are not known to the linker and cannot be made available to another object module. They can, however, be passed through the linker to the symbolic debuffer. Contrasts with global symbol.

locate mode
A record access technique in which a program reads records in a record management service (RMS) block buffer working storage area to reduce overhead. See also move mode.

locking a page in memory
Making a page in an image ineligible for either paging or swapping. A page stays locked in memory until it is specifically unlocked.

locking a page in the working set
Making a page in an image ineligible for paging out of the working set for the image. The page can be swapped when the process is swapped. A page stays locked in a working set until it is specifically unlocked.

logical block number
A number used to identify a byte on a mass storage device. The number is a volume-relative address rather than a physical (device-oriented) address or a virtual (file-relative) address. The blocks that comprise the volume are labeled sequentially with logical block number 0.

logical I/O function
A set of I/O operations (e.g., read and write logical block) that allow restricted direct access to device level I/O operations using logical block addresses.

logical name
A user-specified name for any portion or all of a file specification. For example, the logical name INPUT can be assigned to a terminal device from which a program reads data entered by a user. Logical name assignments are maintained in logical name tables for each process, each group, and the system. A logical name can be created and assigned a value permanently or dynamically.

logical name table
A table that contains a set of logical names and their equivalent names for a particular process, a particular group, or the system.

longword
Four contiguous bytes (32 bits) starting on an addressable byte boundary. Bits are numbered from right to left with 0 through 31. The address of the longword is the address of the byte containing bit 0. When interpreted arithmetically, a longword is a two's complement integer with significance increas-

ing from bit 0 to bit 30. When interpreted as a signed integer, bit 31 is the sign bit. The value of the signed integer is in the range -2,147,483,648 to 2,147,483,647. When interpreted as an unsigned integer, significance increases from bit 0 to bit 31. The value of the unsigned integer is in the range 0 through 4,294,967,295.

macro
A statement that requests a language processor to generate a predefined set of instructions.

mailbox
A software data structure that is treated as a record-oriented device for general interprocess communication. Communication using a mailbox is similar to other forms of device-independent I/O. Senders perform a write to a mailbox, the receiver performs a read from that mailbox. Some system-wide mailboxes are defined: the error logger and OPCOM read from system-wide mailboxes.

main memory
See physical memory.

mapping window
A subset of the retrieval information for a file that is used to translate virtual block numbers to logical block numbers.

mass storage device
A device capable of reading and writing data on mass storage such as a disk pack or a magnetic tape reel.

member number
The second number in a user identification code that uniquely identifies that code.

memory management
The system functions that include the hardware's page mapping and protection and the operating system's image activator and pager.

memory mapping enable (MME)
A bit in a processor register that governs address translation.

modify access type
The specified operand of an instruction or procedure is read, and is potentially modified and written, during that instruction's or procedure's execution.

module
1. A portion of a program or program library, as in a *source module*, *object module*, or *image module*.

2. A board, usually made of plastic covered with an electrical conductor, on which logic devices (such as transistors, resistors, and memory chips) are mounted, and circuits connecting these devices are etched, as in a *logic module*.

monitor console routine (MCR)
The command interpreter in an RSX-11 system.

mount a volume
1. To logically associate a volume with the physical unit on which it is loaded (an activity accomplished by system software at the request of an operator).

2. To load or place a magnetic tape or disk pack on a drive and place the drive on-line (an activity accomplished by a system operator).

move mode
A record I/E access technique in which a program accesses records in its own working storage area. See also locate mode.

mutex
A flag that is used to control exclusive access to a region of code that can share a data structure or other resource. The mutex (mutual exclusion) flag ensures that only one process at a time has access to the region of code.

name block (NAM)
An RMS user data structure that contains supplementary information used in parsing file specifications.

native image
An image whose instructions are executed in native mode.

native mode
The processor's primary execution mode in which the programmed instructions are interpreted as byte-aligned, variable-length instructions that operate on byte, word, longword, and quadword integer, floating and double floating, character string, packed decimal, and variable-length bit field data. The instruction execution mode other than compatibility mode.

network
A collection of interconnected individual computer systems.

nibble
The low-order or high-order four bits of a byte.

node
An individual computer system in a network.

null process
A small system process that is the lowest priority process in the system and takes one entire priority class. One function of the null process is to accumulate idle processor time.

numeric string
A contiguous sequence of bytes representing up to 31 decimal digits (one per byte) and possibly a sign. The numeric string is specified by its lowest addressed location, its length, and its sign representation.

object module
The binary output of a language processor such as the assembler or a compiler, which is used as input to the linker.

object time system (OTS)
See run time procedure library.

offset
A fixed displacement from the beginning of a data structure. System offsets for items within a data structure normally have an associated symbolic name used instead of the numeric displacement. Where symbols are defined, programmers always reference the symbolic names for items in a data structure instead of using the numeric displacement.

op code
The pattern of bits within an instruction that specifies the operation to be performed.

operand specifier
The pattern of bits in an instruction that indicates the addressing mode, a register and/or displacement, which, taken together, identify an instruction operand.

operand specifier type
The access type and data type of an instruction's operand(s). For example, the test instructions are of read access type, since they only read the value of the operand. The operand can be of byte, word, or longword data type, depending on whether the op code is for the TSTB (test byte), TSTW (test word), or TSTL (test longword) instruction.

operator communication manager (OPCOM)
A system process that is always active. OPCOM receives input from a process that wants to inform an operator of a particular status or condition, passes a message to the operator, and tracks the message.

operator's console
Any terminal identified as a terminal attended by a system operator.

owner
In the context "system, owner, group, world," an owner is the particular member (or a group) to which a file, global section, mailbox, or event flag cluster belongs.

owner process
The process (with the exception of the job controller) or subprocess that created a subprocess.

packed decimal
A method of representing a decimal number by storing a pair of decimal digits in one byte, taking advantage of the fact that only four bits are required to represent the numbers zero through nine.

packed decimal string
A contiguous sequence of up to 16 bytes interpreted as a string of nibbles. Each nibble represents a digit except the low-order nibble of the highest addressed byte, which represents the sign. The packed decimal string is specified by its lowest addressed location and the number of digits.

page
1. A set of 512 contiguous byte locations used as the unit of memory mapping and protection.

2. The data between the beginning of file and a page marker, between two markers, or between a marker and the end of a file.

page fault
An exception generated by a reference to a page which is not mapped into a working set.

page fault cluster size
The number of pages read in on a page fault.

page frame number (PFN)
The address of the first byte of a page in physical memory. The high-order 21 bits of the physical address of the base of a page.

page marker
A character or characters (generally a form feed) that separates pages in a file that is processed by a text editor.

pager
A set of kernel mode procedures that executes as the result of a page fault. The pager makes the page for which the fault occurred available in physical memory so that the image can continue execution. The pager and the image activator provide the operating system's memory management functions.

page table entry (PTE)
The data structure that identifies the location and status of a page of virtual address space. When a virtual page is in memory, the PTE contains the page frame number needed to map the virtual page to a physical page. When it is not in memory, the PTE contains the information needed to locate the page on secondary storage (disk).

paging
The action of bringing pages of an executing process into physical memory when referenced. When the process executes, all of its pages are said to reside in virtual memory. Only the actively used pages, however, need to reside in physical memory. The remaining pages can reside on disk until they are needed in physical memory. In this system, a process is paged only when it references more pages than it is allowed to have in its working set. When the process refers to a page not in its working set, a page fault occurs. This causes the operating system's pager to read in the referenced page if it is on disk (and, optionally, other related pages depending on a cluster factor), replacing the least recently faulted pages as needed. A process pages only against itself.

parameter
See command parameter.

per-process address space
See process address space.

physical address
The address used by hardware to identify a location in physical memory on directly addressable secondary storage devices such as a disk. A physical memory address consists of a page frame number and the number of a byte within the page. A physical disk block address consists of a cylinder or track and sector number.

physical address space
The set of all possible 30-bit physical addresses that can be used to refer to locations in memory (memory space) or device registers (I/O space).

physical block
A block on a mass storage device referred to by its physical (device-oriented) address rather than a logical (volume-relative) or virtual (file-relative) address.

physical I/O functions
A set of I/O functions that allow access to all device level I/O operations except maintenance mode.

physical memory
The memory modules connected to the SBI that are used to store:

1. instructions that the processor can directly fetch and execute, and
2. any other data that a processor is instructed to manipulate.

Also called main memory.

position dependent code
Code that can execute properly only in the locations in virtual address space that are assigned to it by the linker.

position independent code
Code that can execute properly without modification wherever it is located in virtual address space, even if its location is changed after it has been linked. Generally, this code uses addressing modes that form an effective address relative to the program counter (PC).

primary vector
A location that contains the starting address of a condition handler to be executed when an exception condition occurs. If a primary vector is declared, that condition handler is the first handler to be executed.

private section
An image section of a process that is not sharable among processes. See also global section.

privilege
See process privilege, user privilege, and image privilege.

privileged instructions
In general, any instructions intended for use by the operating system or privileged system programs. In particular, instructions that the processor will not execute unless the current access mode is kernel mode (e.g., HALT, SVPCTX, LDPCTX, MTPR, and MFPR).

procedure
1. A routine entered via a call instruction.
2. See command procedure.

process
The basic entity scheduled by the system software that provides the context in which an image executes. A process consists of an address space and both hardware and software context.

process address space
See process space.

process context
The hardware and software contexts of a process.

process control block (PCB)
A data structure used to contain process context. The hardware PCB contains the hardware context. The software PCB contains the software context, which includes a pointer to the hardware PCB.

process header
A data structure that contains the hardware process control block (PCB), accounting and quota information, process section table, working set list, and the page tables defining the virtual layout of the process.

process header slots
That portion of the system address space in which the system stores the process headers for the processes in the balance set. The number of process header slots in the system determines the number of processes that can be in the balance set at any one time.

process identification (PID)
The operating system's unique 32-bit binary value assigned to a process.

process I/O segment
That portion of a process control region that contains the process permanent record management service (RMS) internal file access block for each open file, and the I/O buffers, including the command interpreter's command buffer and command descriptors.

process name
A 1- to 15-character ASCII string that can be used to identify processes executing under the same group number.

processor register
A part of the processor used by the operating system software to control the execution states of the computer system. They include the system base and length registers, the program and control region base and length registers, the system control block base register, the software interrupt request register, and many more.

processor status longword (PSL)
A system-programmed processor register consisting of a word of privileged processor status and the PSW. The privileged processor status information includes: the current interrupt priority level (IPL), the previous access mode, the current access mode, the interrupt stack bit, the trace trap pending bit, and the compatibility mode bit.

processor status word (PSW)
The low-order word of the processor status longword. Processor status information includes: the condition codes (carry, overflow, zero, negative), the arithmetic trap enable bits (integer overflow, decimal overflow, floating underflow), and the trace enable bit.

process page tables
The page tables used to describe process virtual memory.

process priority
The priority assigned to a process for scheduling purposes. The operating system recognizes 32 levels of process priority, where 0 is low and 31 high. Levels 16 through 31 are used for time-critical processes. The system does not modify the priority of a time-critical process (although the system manager or process itself may). Levels 0 through 15 are used for normal processes. The system may temporarily increase the priority of a normal process based on the activity of the process.

process privileges
The privileges granted to a process by the system, which are a combination of user privileges and image privileges. They include, for example, the privilege to: affect other processes associated with the same group as the user's group, affect any process in the system regardless of user identification code (UIC), set process swap mode, create permanent event flag clusters, create another process, create a mailbox, and perform direct I/O to a file-structured device.

process section
See private section.

process space
The lowest-addressed half of virtual address space, where per process instructions and data reside. Process space is divided into a program region and a control region.

program counter (PC)
General register 15 (R15). At the beginning of an instruction's execution, the PC normally contains the address of a location in memory from which the processor will fetch the next instruction it will execute.

program locality
A characteristic of a program that indicates how close or far apart the references to locations in virtual memory are over time. A program with a high degree of locality does not refer to many widely scattered virtual addresses in a short period of time.

programmer number
See member number.

program region
The lowest-addressed half of process address virtual memory space (P0 space). The program region contains the image currently being executed by the process and other user code called by the image.

program region base register (P0BR)
The processor register, or its equivalent in a hardware control block, that contains the base virtual address of the page table entry for virtual page number 0 in a process program region.

program region length register (P0LR)
The processor register, or its equivalent in a hardware process control block, that contains the number of entries in the page table for a process program region.

program section (psect)
A portion of a program with a given protection and set of storage management attributes. Program sections that have the same attributes are gathered together by the linker to form an image section.

project number
See group number or account number.

pure code
See reentrant code.

quadword
Eight contiguous bytes (64 bits) starting on an addressable byte boundary. Bits are numbered from right to left, 0 to 63. A quadword is identified by the address of the byte containing the low-order bit (bit 0). When interpreted arithmetically, a quadword is a two's complement integer with significance increasing from bit 0 to bit 62. Bit 63 is used as the sign bit. The value of the integer is in the range -2^{63} to $2^{63}-1$.

qualifier
A portion of a command string that modifies a command verb or command parameter by selecting one of several options. A qualifier, if present, follows the command verb or parameter to which it applies and is in the format: "/qualifier:option." For example, in the command string "PRINT filename/COPIES:3," the COPIES qualifier indicates that the user wants three copies of a given file printed.

queue
1. noun. A circular, doubly linked list. See system queues. verb. To make an entry in a list or table, perhaps using the INSQUE instruction.

2. See job queue.

queue priority
The priority assigned to a job placed in a spooler queue or a batch queue.

quota
The total amount of a system resource, such as CPU time, that a job is allowed to use in an accounting period, as specified by the system manager in the user authorization file. See also limit.

random access by record's file address
The retrieval of a record by its unique address, which is provided to the program by record management services (RMS). The method of access can be used to randomly access a sequentially organized file containing variable length records.

random access by relative record number
The retrieval or storage of a record by specifying its position relative to the beginning of the file.

read access type
An instruction or procedure operand attribute indicating that the specified operand is only read during instruction or procedure execution.

record access block (RAB)
A record management services (RMS) user data structure that represents a request for a record access stream. A RAB relates to operations on the records within a file, such as UPDATE, DELETE, or GET.

record access mode
The method used in record management services (RMS) for retrieving and storing a record in a file. One of three methods: sequential, random, and record's file address.

record management services (RMS)
A set of operating system procedures that are called by programs to process files and records within files. RMS allows programs to issue READ and WRITE requests at the record level (record I/O), as well as read and write blocks (block I/O). RMS is an integral part of the system software. RMS procedures run in executive mode.

record-oriented device
A device such as a terminal, line printer, or card reader, on which the largest unit of data a program can access in one I/O operation is the device's physical record.

record's file address
The unique address of a record in a file that allows records to be accessed randomly regardless of file organization.

record slot
A fixed length area in a relatively organized file that is used to contain one record.

reentrant code
Code that is never modified during execution. It is possible to let many users share the same copy of a procedure or program written as reentrant code.

register
A storage location in hardware logic other than main memory. See also general register, processor register, and device register.

register deferred indexed mode
An indexed addressing mode in which the base operand specifier uses register deferred mode addressing.

register deferred mode
In register deferred mode addressing, the contents of the specified register are used as the address of the actual instruction operand.

register mode
In register mode addressing, the contents of the specified register are used as the actual instruction operand.

relative file organization
A file organization in which the file contains fixed length record cells. Each cell is assigned a consecutive number that represents its position relative to the beginning of a file. Records within each cell can be as big as or smaller than the cell. Relative file organization permits sequential record access, random record access by record number, and random record access by the record's file address.

resource
A physical part of the computer system such as a device or memory, or an interlocked data structure such as a mutex. Quotas and limits control the use of physical resources.

resource wait mode
An execution state in which a process indicates that it will wait until a system resource becomes available when it issues a service request requiring a resource. If a process wants notification when a resource is not available, it can disable resource wait mode during program execution.

return status code
See status code.

run time procedure library
The collection of procedures available to native mode images at run time. These library procedures (such as trigonometric functions, etc.) are common to all native mode images, regardless of the language processor used to compile or assemble the program.

scatter/gather
The ability to transfer in one I/O operation data from discontiguous pages in memory to contiguous blocks on disk, or data from contiguous blocks on disk to discontiguous pages in memory.

secondary storage
Random access mass storage.

secondary vector
A location that identifies the starting address of a condition handler to be executed when a condition occurs and the primary vector contains zero or the handler to which the primary vector points chooses not to handle the condition.

section
A portion of process virtual memory that has common memory management attributes (protection, access, cluster factor, etc.). It is created from an image section, a disk file, or as the result of a create virtual address space system service. See global section, private section, image section, and program section.

sequential access mode
The retrieval or storage of records in which a program successively reads or writes records one after the other in the order in which they appear, starting and ending at any arbitrary point in the file.

sequential file organization
A file organization in which records appear in the order in which they were originally written. The records can be of fixed length or variable length. Although one does not speak of record slots with sequentially organized files, for purposes of comparison with relatively organized files one can say that the record itself is the same as its record slot, and its record number is the same as its relative slot number. Sequential file organization permits sequential record access and random access by a record's file address. Sequential file organization with fixed length records also permits random access by relative record number.

sharable image
An image that has all of its internal references resolved, but which must be linked with an object module(s) to produce an executable image. A sharable image cannot be executed. A sharable image file can be used to contain a library of routines. A sharable image can be used to create a global section by the system manager.

shell process
A predefined process that the job initiator copies to create the minimum context necessary to establish a process.

signal
1. An electrical impulse conveying information.
2. The software mechanism used to indicate that an exception condition was detected.

slave terminal
A terminal from which it is not possible to issue commands to the command interpreter. A terminal assigned to application software.

small process
A system process that has no control region in its virtual address space and has an abbreviated context. Examples are the working set swapper and the null process. A small process is scheduled in the same manner as user processes, but must remain resident during its execution.

software context
The context maintained by the operating system that describes a process. See software process control block (PCB).

software interrupt
An interrupt generated on interrupt priority level 1 through 15, which can be requested only by software.

software process control block (PCB)
The data structure used to contain a process's software context. The operating system defines a software PCB for every process when the process is created. The software PCB includes the following kinds of information about the process: current state; storage address if it is swapped out of memory; unique identification of the process, and address of the process header (which contains the hardware PCB). The software PCB resides in system region virtual address space. It is not swapped with a process.

software priority
See process priority and queue priority.

spooling
Output spooling: The method by which output to a low-speed peripheral device (such as a line printer) is placed into queues maintained on a high-speed device (such as disk) to await transmission to the low-speed device.

Input spooling: The method by which input from a low-speed peripheral (such as the card reader) is placed into queues maintained on a high-speed device (such as disk) to await transmission to a job processing that input.

spool queue
The list of files supplied by processes that are to be processed by a symbiont. For example, a line printer queue is a list of files to be printed on the line printer.

stack
An area of memory set aside for temporary data storage, or for procedure and interrupt service linkages. A stack uses the last-in, first-out concept. As items are added to ("pushed on") the stack, the stack pointer decrements. As items are retrieved from ("popped off") the stack, the stack pointer increments.

stack frame
A standard data structure built on the stack during a procedure call, starting from the location addressed by the frame pointer (FP) to lower addresses, and popped off during a return from procedure. Also called call frame.

stack pointer (SP)
General register 14 (R14). SP contains the address of the top (lowest address) of the processor-defined stack. Reference to SP will access one of the five possible stack pointers, kernel, executive, supervisor, user, or interrupt, depending on the value in the current mode and interrupt stack bits in the processor status longword (PSL).

state queue
A list of processes in a particular processing state. The scheduler uses state queues to keep track of processes' eligibility to execute. They include: processes waiting for a common event flag, suspended processes, and executable processes.

status code
A longword value that indicates the success or failure of a specific function. For example, system services always return a status code in R0 upon completion.

store through
See write through.

strong definition
Definition of a global symbol that is explicitly available for reference by modules linked with the module in which the definition occurs. The linker always lists a global symbol with a strong definition in the symbol portion of the map. The librarian always includes a global symbol with a strong definition in the global symbol table of a library.

strong reference
A reference to a global symbol in an object module that requests the linker to report an error if it does not find a definition for the symbol during linking. If a library contains the definition, the linker incorporates the library module defining the global symbol into the image containing the strong reference.

subprocess
A subsidiary process created by another process. The process that creates a subprocess is its owner. A subprocess receives resource quotas and limits from its owner. When an owner process is removed from the system, all its subprocesses (and their subprocesses) are also removed.

supervisor mode
The third most privileged processor access mode (mode 2). The operating system's command interpreter runs in supervisor mode.

suspension
A state in which a process is inactive, but known to the system. A suspended process becomes active again only when another process requests the operating system to resume it. Contrasts with hibernation.

swap mode
A process execution state that determines the eligibility of a process to be swapped out of the balance set. If process swap mode is disabled, the process working set is locked in the balance set.

swapping
The method for sharing memory resources among several processes by writing an entire working set to secondary storage (swap out) and reading another working set into memory (swap in). For example, a process's working set can be written to secondary storage while the process is waiting for I/O completion on a slow device. It is brought back into the balance set when I/O completes. Contrasts with paging.

switch
See (command) qualifier.

symbiont
A full process that transfers record-oriented data to or from a mass storage device. For example, an input symbiont transfers data from card readers to disks. An output symbiont transfers data from disks to line printers.

symbiont manager
The function (in the system process called the job controller) that maintains spool queues, and dynamically creates symbiont processes to perform the necessary I/O operations.

symbol
See local symbol, global symbol, and universal global symbol.

synchronous backplane interconnect (SBI)
The part of the hardware that interconnects the CPU, memory controllers, Massbus adapters, the Unibus adapter.

synchronous record operation
A mode of record processing in which a user program issues a record read or write request and then waits until that request is fulfilled before continuing to execute.

system
In the context "system, owner, group, world," the system refers to the group numbers that are used by the operating system and its controlling users, the system operators and system manager.

system address space
See system space and system region.

system base register (SBR)
A processor register containing the physical address of the base of the system page table (SPT).

system buffered I/O
An I/O operation, such as terminal or mailbox I/O, in which an intermediate buffer from the system pool is used instead of a processor-specified buffer. Contrasts with direct I/O.

system control block (SCB)
The data structure in system space that contains all the interrupt and exception vectors known to the system.

system control block base register (SCBB)
A processor register containing the base address of the system control block.

system device
The random access mass storage device unit on which the volume containing the operating system software resides.

system dynamic memory
Memory reserved for the operating system to allocate as needed for temporary storage. For example, when an image issues an I/O request, system dynamic memory is used to contain the I/O request packet. Each process has a limit on the amount of system dynamic memory that can be allocated for its use at one time.

system identification register
A processor register that contains the processor type and serial number.

system image
The image that is read into memory from secondary storage when the system is started up.

system length register (SLR)
A processor register containing the length of the system page table (SPT) in longwords, that is, the number of page table entries (PTEs) in the system region page table.

system page table (SPT)
The data structure that maps the system region virtual addresses, including the addresses used to refer to the process page tables. The SPT contains one page table entry (PTE) for each page of system region virtual memory. The physical base address of the SPT is contained in a register called the system base register (SBR).

system process
A process that provides system-level functions. Any process that is part of the operating system. See also small process, fork process.

system programmer
A person who designs and/or writes operating systems, or who designs and writes procedures or programs that provide general purpose services for an application system.

system queue
A queue used and maintained by operating system procedures. See also state queues.

system region
The third quarter of virtual address space. The lowest-addressed half of system space. Virtual addresses in the system region are sharable between processes. Some of the data structures mapped by the system region virtual addresses are: system entry vectors, the system control block (SCB), the system page table (SPT), and process page tables.

system services
Procedures provided by the operating system that can be called by user processes.

system space
The highest-addressed half of virtual address space. See also system region.

system virtual address
A virtual address identifying a location mapped by an address in system space.

system virtual space
See system space.

task
An RSX-11/IAS term for a process and image bound together.

terminal
The general name for those peripheral devices that have keyboards and video screens or printers. Under program control, a terminal enables people to type commands and data on the keyboard and receive messages on the video screen or printer. Examples of terminals are the LA36 DECwriter hardcopy terminal and VT52 video display terminal.

time-critical process
A process assigned to a software priority level between 16 and 31, inclusive. The scheduling priority assigned to a time-critical process is never modified by the scheduler, although it can be modified by the system manager or process itself.

timer
A system fork process that maintains the time of day and the date. It also scans for device timeouts and performs time-dependent scheduling upon request.

track
A collection of blocks at a single radius on one recording surface of a disk.

transfer address
The address of the location containing a program entry point (the first instruction to execute).

translation buffer
An internal processor cache containing translations for recently used virtual addresses.

trap
An exception condition that occurs at the end of the instruction that caused the exception. The program counter (PC) saved on the stack is the address of the next instruction that would normally have been executed. All software can enable and disable some of the trap condition with a single instruction.

trap enables
Three bits in the processor status word that control the processor's action on certain arithmetic exceptions.

two's complement
A binary representation for integers in which a negative number is one greater than the bit complement of the positive number.

two-way associative cache
A cache organization that has two groups of directly mapped blocks. Each group contains several blocks for each index position in the cache. A block of data from main memory can go into any group at its proper index position. A two-way associative cache is a compromise between the extremes of fully associative and direct mapping cache organizations that takes advantage of the features of both.

type ahead
A terminal handling technique in which the user can enter commands and data while the software is processing a previously entered command. The commands typed ahead are not echoed on the terminal until the command processor is ready to process them. They are held in a type ahead buffer.

unit record device
A device such as a card reader or line printer.

universal global symbol
A global symbol in a sharable image that can be used by modules linked with that sharable image. Universal global symbols are typically a subset of all the global symbols in a sharable image. When creating a sharable image, the linker ensures that universal global symbols remain available for reference after symbols have been resolved.

unwind the call stack
To remove call frames from the stack by tracing back through nested procedure calls using the current contents of the FP register and the frame pointer (FP) register contents stored on the stack for each call frame.

urgent interrupt
An interrupt received on interrupt priority levels 24 through 31. These can be generated only by the processor for the interval clock, serious errors, and power-fail.

user authorization file
A file containing an entry for every user that the system manager authorizes to gain access to the system. Each entry identifies the user name, password, default account, user identification code (UIC), quotas, limits, and privileges assigned to individuals who use the system.

user environment test package (UETP)
A collection of routines that verify that the hardware and software systems are complete, properly installed, and ready to be used.

user file directory (UFD)
See directory.

user identification code (UIC)
The pair of numbers assigned to users and to files, global sections, common event flag clusters, and mailboxes that specify the type of access (read and/or write access, and in the case of files execute and/or delete access) available to the owners, group, world, and system. It consists of a group number and a member number separated by a comma.

user mode
The least privileged processor access mode (mode 3). User processes and the run time library procedures run in user mode.

user name
The name that a person types on a terminal to log on to the system.

user number
See member number.

user privileges
The privileges granted a user by the system manager. See process privileges.

utility
A program that provides a set of related general purpose functions, such as a program development utility (an editor, a linker, etc.), a file management utility (file copy or file format translation program), or operations management utility (disk backup/restore, diagnostic program, etc.).

value return registers
The general registers R0 and R1 used by convention to return function values. These registers are not preserved by any called procedures. They are available as temporary registers to any called procedure. All other registers (R2, R3, . . ., R11, AP, FP, SP, PC) are preserved across procedure calls.

variable-length bit field
A set of zero to 32 contiguous bits located arbitrarily with respect to byte boundaries. A variable bit field is specified by four attributes:

1. the address A of a byte,

2. the bit position P of the starting location of the bit field with respect to bit 0 of the byte at address A,

3. the size, in bits, of the bit field, and

4. whether the field is signed or unsigned.

variable-length record format
A file format in which records are not necessarily the same length.

variable with fixed-length control (VFC) record format
A file format in which records of variable length contain an additional fixed-length control area. The control area may be used to contain file line numbers and/or print format controls.

vector
1. An interrupt or exception vector is a storage location known to the system that contains the starting address of a procedure to be executed when a given interrupt or exception occurs. The system defines separate vectors for each interrupting device controller and for classes of exceptions. Each system vector is a longword.

2. For the purposes of exception handling, users can declare up to two software exception vectors (primary and secondary) for each of the four access modes. Each vector contains the address of a condition handler.

3. A one-dimensional array.

version number

1. The field following the file type in a file specification. It is separated from file type by a period (.) or semicolon (;) and consists of a number that generally identifies it as one version among all the files having the identical file specification but for version number.

2. The number used to identify the revision level of program.

virtual address
A 32-bit integer identifying a byte "location" in virtual address space. The memory management hardware translates a virtual address to a physical address. The term virtual address may also refer to the address used to identify a virtual block on a mass storage device.

virtual address space
The set of all possible virtual addresses that an image executing in the context of a process can use to identify the location of an instruction or data. The virtual address space seen by the programmer is a linear array of 4,294,967,296 (2^{32}) byte address.

virtual block number
A number used to identify a block on a mass storage device. The number is a file-relative address rather than a logical (volume-oriented) or physical (device-oriented) address. The first block in a file is always virtual block number 1.

virtual I/O functions
A set of I/O functions that must be interpreted by an ancillary control process.

virtual memory
The set of storage locations in physical memory and on disk that are referred to by virtual addresses. From the programmer's viewpoint, the secondary storage locations appear to be locations in physical memory. The size of virtual memory in any system depends on the amount of physical memory available and the amount of disk storage used for nonresident virtual memory.

virtual page number
The virtual address of a page of virtual memory.

volume
A mass storage medium such as a disk pack or reel of magnetic tape.

volume set
The file-structured collection of data residing on one or more mass storage media.

wait
To become inactive. A process enters a process wait state when the process suspends itself, hibernates, or declares that it needs to wait for an event, resource, mutex, etc.

wake
To activate a hibernating process. A hibernating process can be awakened by another process or by the timer process, if the hibernating process or another process scheduled a wake-up call.

weak definition
Definition of a global symbol that is not explicitly available for reference by modules linked with the module in which the definition occurs. The librarian does not include a global symbol with a weak definition in the global symbol table of a library. Weak definitions are often used when creating libraries to identify those global symbols that are needed only if the module containing them is otherwise linked with a program.

weak reference
A reference to a global symbol that requests the linker not to report an error or to search the default library's global symbol table to resolve the reference if the definition is not in the modules explicitly supplied to the linker. Weak references are often used when creating object modules to identify those global symbols that may not be needed at run time.

wild card
A symbol, such as an asterisk, that is used in place of a file name, file type, directory name, or version number in a file specification to indicate "all" for the given field.

window
See mapping window.

word
Two contiguous bytes (16 bits) starting on an addressable byte boundary. Bits are numbered from the right, 0 through 15. A word is identified by the address of the byte containing bit 0. When interpreted arithmetically, a word is a two's complement integer with significance increasing from bit 0 to bit 14. If interpreted as a signed integer, bit 15 is the sign bit. The value of the integer is in the range -32768 to 32767. When interpreted as an unsigned integer, significance increases from bit 0 through bit 15 and the value of the unsigned integer is in the range 0 through 65535.

working set
The set of pages in process space to which an executing process can refer without incurring a page fault. The working set must be resident in memory for the process to execute. The remaining pages of that process, if any, are either in memory and not in the process working set or they are on secondary storage.

working set swapper
A system process that brings process working sets into the balance set and removes them from the balance set.

world
In the context "system, owner, group, world," world refers to all users, including the system operators, the system manager, and users both in an owner's group and in any other group.

write access type
The specified operand of an instruction or procedure is only written during that instruction's or procedure's execution.

write allocate
A cache management technique in which cache is allocated on a write miss as well as on the usual read miss.

write back
A cache management technique in which data from a write operation to cache is copied into main memory only when the data in cache must be overwritten. This results in temporary inconsistencies between cache and main memory. Contrasts with write through.

write through
A cache management technique in which data from a write operation is copied in both cache and main memory. Cache and main memory data are always consistent. Contrasts with write back.

COMMONLY USED MNEMONICS

ACP	Ancillary Control Process
ANS	American National Standard
ASCII	American Standard Code for Information Interchange
AST	Asynchronous System Trap
ASTLVL	Asynchronous System Trap LeVeL
CCB	Channel Control Block
CM	Compatibility Mode bit in the hardware PSL
CRB	Channel Request Block
CRC	Cyclic Redundancy Check
DAP	Data Access Protocol
DDB	Device Data Block
DDCMP	DIGITAL Data Communications Message Protocol
DDT	Driver Data Table
DV	Decimal oVerflow trap enable bit in the PSW
ECB	Exit Control Block
ECC	Error Correction Code
ESP	Executive mode Stack Pointer
ESR	Exception Service Routine
F11ACP	Files-11 Ancillary Control Process
FAB	File Access Block
FCA	Fixed Control Area
FCB	File Control Block
FCS	File Control Services
FDT	Function Decision Table
FP	Frame Pointer
FPD	First Part (of an instruction) Done
FU	Floating Underflow trap enable bit in the PSW
GSD	Global Section Descriptor
GST	Global Symbol Table
IDB	Interrupt Dispatch Block
IPL	Interrupt Priority Level
IRP	I/O Request Packet
ISECT	Image SECTion
ISD	Image Section Descriptor
ISP	Interrupt Stack Pointer
IS	Interrupt Stack bit in PSL
ISR	Interrupt Service Routine
IV	Integer oVerflow trap enable bit in the PSW
KSP	Kernel mode Stack Pointer
MBA	Massbus Adapter
MBZ	Must Be Zero
MCR	Monitor Console Routine
MFD	Master File Directory
MFPR	Move From Process Register instruction
MME	Memory Mapping Enable
MTPR	Move To Process Register instruction
MUTEX	MUTual EXclusion semaphore
NSP	Network Services Protocol
OPCOM	OPeration COMmunication manager

P0BR	Program region Base Register
P0LR	Program region Length Register
P0PT	Program region Page Table
P1BR	control region Base Register
P1LR	control region Limit Register
P1PT	control region Page Table
PC	Program Counter
PCB	Process Control Block
PCBB	Process Control Block Base register
PFN	Page Frame Number
PID	Process Identification Number
PME	Performance Monitor Enable bit in PCB
PSECT	Program SECTion
PSL	Processor Status Longword
PSW	Processor Status Word
PTE	Page Table Entry
QIO	Queue Input/Output request system service
RAB	Record Access Block
RFA	Record's File Address
RMS	Record Management Services
RWED	Read, Write, Execute, Delete
SBI	Synchronous Backplane Interconnect
SBR	System Base Register
SCB	System Control Block
SCBB	System Control Block Base register
SLR	System Length Register
SP	Stack Pointer
SPT	System Page Table
SSP	Supervisor mode Stack Pointer
SVA	System Virtual Address
TP	Trace trap Pending bit in PSL
UBA	Unibus Adapter
UCB	Unit Control Block
UETP	User Environment Test Package
UFD	User File Directory
UIC	User Identification Code
USP	User mode Stack Pointer
VCB	Volume Control Block
VPN	Virtual Page Number
WCB	Window Control Block
WCS	Writable Control Store
WDCS	Writable Diagnostic Control Store

APPENDIX H
NATIVE MODE INSTRUCTION SET

The instruction set that the processor executes is selected under operating system control to either native mode or compatibility mode. The native mode instruction set is based on over 200 different op codes. The op codes can be grouped into classes based on their function and use. Instructions used to manipulate the general data types include:

- integer and floating-point instructions
- packed decimal instructions
- character string instructions
- bit field instructions

Instructions that are used to manipulate special kinds of data include:

- queue manipulation instructions
- address manipulation instructions
- user-programmed general register control instructions

Instructions that provide basic program flow control and enable you to call procedures are:

- branch, jump, and case instructions
- subroutine call instructions
- procedure call instructions

Table H-1 lists the basic instruction operations in order by these classifications. Instructions that enable operating system procedures to provide user mode processes with services requiring privilege are listed in the table, but discussed in the system programming environment section. Instructions that are singular in the functions they provide are listed last.

Table H-1 Native Mode Instruction Set Summary

Integer and Floating-Point Logical Instructions

MOV$	Move (B,W,L,F,D,Q)*
MNEG$	Move Negated (B,W,L,F,D)
MCOM$	Move Complemented (B,W,L)
MOVZ$	Move Zero-Extended (BW,BL,WL)
CLR$	Clear (B,W,L=Q,F=D)
CVT$	Convert (B,W,L,F,D) (B,W,L,F,D)
CVTR$	Convert Rounded (F,D) to Longword
CMP$	Compare (B,W,L,F,D)
TST$	Test (B,W,L,F,D)
BIS$2	Bit Set (B,W,L) 2-Operand
BIS$3	Bit Set (B,W,L) 3-Operand
BIC$2	Bit Clear (B,W,L) 2-Operand
BIC$3	Bit Clear (B,W,L) 3-Operand
BIT$	Bit Test (B,W,L)
XOR$2	Exclusive OR (B,W,L) 2-Operand
XOR$3	Exclusive OR (B,W,L) 3-Operand
ROTL	Rotate Longword

Integer and Floating-Point Arithmetic Instructions

INC$	Increment (B,W,L)
DEC$	Decrement (B,W,L)
ASH$	Arithmetic Shift (L,Q)
ADD$2	Add (B,W,L,F,D) 2-Operand
ADD$3	Add (B,W,L,F,D) 3-Operand
ADWC	Add with Carry
ADAWI	Add Aligned Word Interlocked
SUB$2	Subtract (B,W,L,F,D) 2-Operand
SUB$3	Subtract (B,W,L,F,D) 3-Operand
SBWC	Subtract with Carry
MUL$2	Multiply (B,W,L,F,D) 2-Operand
MUL$3	Multiply (B,W,L,F,D) 3-Operand
EMUL	Extended Multiply
DIV$2	Divide (B,W,L,F,D) 2-Operand
DIV$3	Divide (B,W,L,F,D) 3-Operand
EDIV	Extended Divide
EMOD$	Extended Modulus (F,D)
POLY$	Polynomial Evaluation (F,D)

Packed Decimal Instructions

MOVP	Move Packed
CMPP3	Compare Packed 3-Operand
CMPP4	Compare Packed 4-Operand
ASHP	Arithmetic Shift Packed and Round
ADDP4	Add Packed 4-Operand
ADDP6	Add Packed 6-Operand
SUBP4	Subtract Packed 4-Operand
SUBP6	Subtract Packed 6-Operand
MULP	Multiply Packed
DIVP	Divide Packed
CVTLP	Convert Long to Packed
CVTPL	Convert Packed to Long
CVTPT	Convert Packed to Trailing
CVTTP	Convert Trailing to Packed
CVTPS	Convert Packed to Separate
CVTSP	Convert Separate to Packed
EDITPC	Edit Packed to Character String

Table H-1 Native Mode Instruction Set Summary (Cont)

Character String Instructions

MOVC3	Move Character 3-Operand
MOVC5	Move Character 5-Operand
MOVTC	Move Translated Characters
MOVTUC	Move Translated Unit Character
CMPC3	Compare Characters 3-Operand
CMPC5	Compare Characters 5-Operand
LOCC	Locate Character
SKPC	Skip Character
SCANC	Scan Characters
SPANC	Span Characters
MATCHC	Match Characters

Variable-Length Bit Field Instructions

EXTV	Extract Field
EXTZV	Extract Zero-Extended Field
INSV	Insert Field
CMPV	Compare Field
CMPZV	Compare Zero-Extended Field
FFS	Find First Set
FFC	Find First Clear

Index Instruction

INDEX	Compute Index

Queue Instructions

INSQUE	Insert Entry in Queue
REMQUE	Remove Entry from Queue

Address Manipulation Instructions

MOVA$	Move Address (B,W,L=F,Q=D)
PUSHA$	Push Address (B,W,L=F,Q=D) on Stack

General Register Manipulation Instructions

PUSHL	Push Longword on Stack
PUSHR	Push Registers on Stack
POPR	Pop Registers from Stack
MOVPSL	Move from Processor Status Longword
BISPSW	Bit Set Processor Status Word
BICPSW	Bit Clear Processor Status Word

Unconditional Branch and Jump Instructions

BR	Branch with (Byte, Word) Displacement
JMP	Jump

Table H-1 Native Mode Instruction Set Summary (Cont)

Branch on Condition Code

BLSS	Less Than
BLSSU	Less than Unsigned
BLEQ	Less than or Equal
BLEQU	Less than or Equal Unsigned
BEQL	Equal
(BEQLU)	(Equal Unsigned)
BNEQ	Not Equal
(BNEQU)	(Not Equal Unsigned)
BGTR	Greater than
BGTRU	Greater than Unsigned
BGEQ	Greater than or Equal
BGEQU	Greater than or Equal Unsigned
(BCC)	(Carry Cleared)
BVS	Overflow Set
BVC	Overflow Clear

Branch on Bit

BLB$	Branch on Low Bit (Set, Clear)
BB$	Branch on Bit (Set, Clear)
BBS$	Branch on Bit Set and (Set, Clear) bit
BBC$	Branch on Bit Clear and (Set, Clear) bit
BBSSI	Branch on Bit Set and Set bit interlocked
BBCCI	Branch on Bit Clear and Clear bit interlocked

Loop and Case Branch

ACB$	Add, Compare and Branch (B,W,L,F,D)
AOBLEQ	Add One and Branch Less Than or Equal
AOBLSS	Add One and Branch Less Than
SOBGEQ	Subtract One and Branch Greater Than or Equal
SOBGTR	Subtract One and Branch Greater Than
CASE$	Case on (B,W,L)

Subroutine Call and Return Instructions

BSB$	Branch to Subroutine with (B,W) Displacement
JSB	Jump to Subroutine
RSB	Return from Subroutine

Procedure Call and Return Instructions

CALLG	Call Procedure with General Argument List
CALLS	Call Procedure with Stack Argument List
RET	Return from Procedure

Table H-1 Native Mode Instruction Set Summary (Cont)

Protected Procedure Call and Return Instructions

CHM$	Change Mode to (Kernel, Executive, Supervisor, User)
REI	Return from Exception or Interrupt
PROBER	Probe Read
PROBEW	Probe Write

Privileged Processor Register Control Instructions

SVPCTX	Save Process Context
LDPCTX	Load Process Context
MTPR	Move to Process Register
MFPR	Move from Processor Register

Special Function Instructions

CRC	Cyclic Redundancy Check
BPT	Breakpoint Fault
XFC	Extended Function Call
NOP	No Operation
HALT	Halt

APPENDIX I
COMPATIBILITY MODE INSTRUCTION SET (PDP-11)

The compatibility mode instruction set is the PDP-11 instruction set. Programs previously constructed with this instruction set can be run directly on the VAX-11/780. New programs can also be developed on the VAX-11/780 using this instruction. Programs using the native mode instruction set can be run concurrent with programs using the compatibility mode. Table I-1 lists the basic instruction operations.

Table I-1 Compatibility Mode Instruction Set Summary

Single Operand Instructions

CLR(B)	Clear Destination
DEC(B)	Decrement Destination
INC(B)	Increment Destination
NEG(B)	Negate Destination
TST(B)	Test Destination
COM(B)	Complement Destination
ASR(B)	Arithmetic Shift Right Destination
ASL(B)	Arithmetic Shift Left Destination
ASH	Shift Arithmetically
ASHC	Arithmetic Shift Combined
ADC(B)	Add Carry Destination
SBC(B)	Subtract Carry Destination
SXT	Shift Extend Destination
ROL(B)	Rotate Left Destination
ROR(B)	Rotate Right Destination
SWAB	Swap Bytes Destination

Double Operand Instructions

MOV(B)	Move Source to Destination
ADD	Add Source to Destination
MUL	Multiply
DIV	Divide
XOR	Exclusive OR
BIS(B)	Bit Set
BIT(B)	Bit Test
BIC(B)	Bit Clear

Table I-1 Compatibility Mode Instruction Set Summary (Cont)

Program Control Instructions

BR	Branch (Unconditional)
BEQ	Branch on Equal (Zero)
BNE	Branch Not Equal (Zero)
BMI	Branch on Minus
BPL	Branch on Plus
BCS	Branch on Carry Set
BCC	Branch on Carry Clear
BVS	Branch on Overflow Set
BVC	Branch on Overflow Clear
BLT	Branch on Less than (Zero)
BGE	Branch on Greater than or Equal (Zero)
BLE	Branch on Less than or Equal (Zero)
BGT	Branch on Greater than (Zero)
BHI	Branch on Higher
BLOS	Branch on Lower or Same
BHIS	Branch on Higher or Same
BLO	Branch on Lower
JSR	Jump to Subroutine
MARK	Mark
RTS	Return from Subroutine
SPL	Set Priority Level
JMP	Jump
SOB	Subtract One and Branch
EMT	Emulator Tapes
TRAP	Trap
BPT	Breakpoint Trap
IOT	I/O Trap
RTI	Return From Interrupt
RTT	Return From Trap

Miscellaneous

HALT	Halt
WAIT	Wait for Interrupt
RESET	Reset External Bus
MTPI	Move to Previous Instruction Space
MTPD	Move to Previous Data Space
MFPI	Move From Previous Instruction Space
MFPD	Move From Previous Data Space

Condition Code Operators

CLC	Clear C
CLV	Clear V
CLZ	Clear Z
CLN	Clear N
SEC	Set C
SEV	Set V
SEZ	Set Z
SEN	Set N
SCC	Set all CC
CCC	Clear all CCs

APPENDIX J
7-BIT ASCII CODE

OCTAL CODE	CHAR	OCTAL CODE	CHAR	OCTAL CODE	CHAR	OCTAL CODE	CHAR
000	NUL	040	SP	100	@	140	\
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	] or ↑	175	}
036	RS	076	>	136	^	176	~
037	US	077	?	137	_ or ←	177	DEL

APPENDIX K
REMOTE CONSOLE COMMAND HELP FILE

The remote console command help file contains a list of the remote console access command set. A printout of the remote console command help file is obtained by entering '@REMOTE HELP' at the console terminal with the console in the console I/O mode (enter CTRL P if a console prompt is not present). A sample printout is given below.

```
! VAX11/780 CONSOLE - REMOTE ACCESS HELP FILE   REV-01   28-DEC-77
!
! 'ENABLE TALK'                  -ESTABLISH TERMINAL TO TERMINAL
!                                COMMUNICATION BETWEEN LOCAL AND REMOTE
!                                TERMINAL. KEYS STRUCK ON ONE TERMINAL
!                                ARE PRINTED ON THE OTHER. CONTROL-P
!                                TERMINATES TALK.
! 'ENABLE ECHO'                  -CAUSES CHARACTERS TYPED IN TALK MODE
!                                TO BE ECHOED BACK TO THE ORIGINATING
!                                TERMINAL.
! 'ENABLE LOCAL COPY'            -CAUSES THE LOCAL TERMINAL TO GET A
!                                COPY OF OUTPUT BEING SENT TO REMOTE
!                                TERMINAL.
! 'ENABLE LOCAL CONTROL'         -ALLOWS LOCAL TERMINAL TO CONTROL
!                                SYSTEM WHEN CONSOLE SWITCH IS IN REMOTE
!                                POSITION(S). DISABLED BY A CONTROL-P
!                                FROM THE REMOTE TERMINAL.
! 'ENABLE CARRIER ERROR'         -CAUSES CONSOLE TO PRINT '?CARRIER
!                                LOST' WHEN A LOSS OF CARRIER IS
!                                DETECTED AT REMOTE INTERFACE.
! 'DISABLE ECHO'                 -INHIBITS ECHO OF CHARACTERS TYPED IN
!                                TALK MODE.
! 'DISABLE LOCAL COPY'           -DISABLE LOCAL TERMINAL FROM RECEIVING
!                                COPY OF OUTPUT TO REMOTE TERMINAL.
! 'DISABLE CARRIER ERROR'        -CAUSES CONSOLE TO INHIBIT PRINTING OF
!                                CARRIER LOST MESSAGE WHEN LOSS OF
!                                CARRIER DETECTED.
! 'ENABLE LOCAL FLOPPY'          -(AFFECTS PROTOCOL OPERATION ONLY) ON
!                                AN ATTEMPT TO OPEN A FILE, THE
!                                DIRECTORY OF LOCAL FLOPPY WILL BE
!                                SEARCHED FIRST. IF FILE IS NOT FOUND,
!                                'REMOTE'. FLOPPY'S DIRECTORY IS
!                                SEARCHED FOR FILE.
```

```
! 'DISABLE LOCAL FLOPPY'
!
!
!
! END-OF-REMOTE.HLP
```

-(AFFECTS PROTOCOL OPERATION ONLY) ON AN ATTEMPT TO OPEN A FILE, THE FILE IS SEARCHED FOR ON THE 'REMOTE' FLOPPY ONLY.

APPENDIX L
DIAGNOSTIC PROGRAMS

A series of diagnostic programs are available for use in troubleshooting the VAX-11/780 and its peripheral equipment. These diagnostics are available on floppy disks and on the operating system disk (VAX PAX). The floppy disks are listed in Table L-1.

Table L-1 Diagnostic Programs Floppy Disks

The following color code has been adopted for floppy diskette media distribution.

MAINDEC Code	Title	Color
ZZ-ESZAB	VAX-11/780 Console Package	Green
ZZ-ESZAC	VAX-11/780 Micro Diagnostics #1	Red
ZZ-ESZAD	VAX-11/780 Micro Diagnostics #2	Red
ZZ-ESZAE	VAX-11/780 Bus Adapter Diagnostics	Blue
ZZ-ESZAF	VAX-11/780 CPU Cluster Exerciser	Blue
ZZ-ESZ1A	RP0X Load Device (RP0X Repair)	Yellow
ZZ-ESZ2A	Magtape Load Device (TM03/TEE16 DFT)	Yellow
ZZ-ESZ3A	COM Option Repair Package	Yellow
ZZ-ESZ4A	RK611 Load Device #1	Yellow
ZZ-ESZ5A	RK611 Load Device #2	Yellow
ZZ-ESZ6A	RK611 Load Device #3	Yellow
ZZ-ESZ1B	User Mode Diagnostics #1 (Distribution Only, Non-Executable)	Brown
ZZ-ESZ2B	User Mode Diagnostics #2 (Distribution Only, Non-Executable)	Brown

The diagnostic programs are listed on Table L-2. All of the diagnostic programs are contained on the VAX PAX. Some of the programs are contained on more than one floppy disk. Table L-3 is a list of the CPU cluster diagnostic programs and the floppy disks on which they are present.

Table L-2 Diagnostic Programs

MAINDEC Code	Title
ZZ-ESCAA	VAX Line Printer Diagnostic
ZZ-ESCAA	VAX Massbus Adapter (RH780) Program
ZZ-ESCBA	VAX Unibus Adapter (DW780) Program
ZZ-ESDBB	VAX DMC11 Exerciser Program
ZZ-ESKAA	VAX-11/780 Console (KC780) Program
ZZ-ESKAB	VAX-11/780 (KA780) Microdiagnostic Monitor
ZZ-ESKAC	VAX-11/780 (KA780) Hardcore Monitor
ZZ-ESKAD	VAX-11/780 (KA780) Hardcore Test Stream
ZZ-ESKAE	VAX-11/780 (KA780) Micro Test Monitor
ZZ-ESKAF	VAX-11/780 (KA780) Microdiagnostic Command Parser
ZZ-ESKAG	VAX-11/780 (KA780) Microdiagnostic Directory Search Routine
ZZ-ESKAH	VAX-11/780 (KA780) Microdiagnostic Test #1
ZZ-ESKAJ	VAX-11/780 (KA780) Microdiagnostic Test #2
ZZ-ESKAK	VAX-11/780 (KA780) Microdiagnostic Fail Chain Monitor
ZZ-ESKAL	VAX-11/780 (KA780) Fail Chain Test One
ZZ-ESKAX	VAX-11/780 (KA780) CPU Cluster Exerciser Diagnostic
ZZ-ESMAA	VAX TM03/TEE16 Tape Reliability
ZZ-ESMAB	VAX TM03/TEE16 Drive Function Timer
ZZ-ESRAA	VAX RP0X/RK06 Reliability
ZZ-ESRAB	VAX RP0X/RK Formatter
ZZ-ESRBA	VAX RP0X Functional Diagnostic
ZZ-ESRCA	VAX DCL/RP04,5,6 Repair
ZZ-ESREA	VAX RK611 Diagnostic Part A
ZZ-ESREB	VAX RK611 Diagnostic Part B
ZZ-ESREC	VAX RK611 Diagnostic Part C
ZZ-ESRED	VAX RK611 Diagnostic Part D
ZZ-ESREE	VAX RK611 Diagnostic Part E
ZZ-ESREF	VAX RK611 Manual Interaction Test
ZZ-ESSAA	VAX Diagnostic Supervisor
ZZ-ESTAA	VAX Terminal Diagnostic
ZZ-ESTBA	VAX Terminal Exerciser
ZZ-ESXBA	VAX Bus Interaction Program
ZZ-ES0AA	WCS Microcode
ZZ-ES0AB	PCS Program
ZZ-ES0AE	Memory Management Script File
ZZ-ES0AF	Memory Management Program

Table L-3 Diagnostic Programs Contained on Floppy Disk

MAINDEC Code/Title		DECO/DEPO	Resident Media Code
ZZ-ESKAA	VAX-11/780 Console Program	3.08	ZZ-ESZAB/ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAB	VAX-11/780 Microdiag Monitor	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAC	VAX-11/780 Hardcore Monitor	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAD	VAX-11/780 Hardcore Test Stream	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESLAE	VAX-11/780 Micro Test Monitor	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAF	VAX-11/780 Microdiag CMD Parser	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAG	VAX-11/780 Microdiag DIR Routine	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAH	VAX-11/780 Microdiag Test 1	8.0	ZZ-ESZAC
ZZ-ESKAJ	VAX-11/780 Microdiag Test 2	8.0	ZZ-ESZAD
ZZ-ESKAK	VAX-11/780 Fail Chain Monitor	8.0	ZZ-ESZAC/ZZ-ESZAD
ZZ-ESKAK	VAX-11/780 Fail Chaintest #1	8.0	ZZ-ESZAC
ZZ-ESKAX	VAX-11/780 CPU Cluster Exerciser	8.0	ZZ-ESZAF
ZZ-ESCBA	VAX-11/780 Unibus Adapter Diagnostic	4.0	ZZ-ESZAE/ZZ-ESZAA
ZZ-ES0AA	VAX-11/780 WCS Microcode	1.11	ZZ-ESZAB/ZZ-ESZAC/ZZ-ESZAD/ZZ-ESZAF
ZZ-ESSAA	VAX Diagnostic Supervisor	3.06	ZZ-ESZAE/ZZ-ESZAF/ZZ-ESZ1A/ZZ-ESZ2A ZZ-ESZ3A/ZZ-ESZ4A/ZZ-ESZ5A/ZZ-ESZ6A ZZ-ESZ1B/ZZ-ESZ2B/ZZ-ESZA

The coding convention used on the diagnostic programs is contained in Table L-4.

Table L-5 defines the diagnostic program features. For additional details on the diagnostic programs, see "Diagnostic System Operation Procedure."

Table L-4 VAX MAINDEC Coding Convention

ZZ-SPXYD-R.P

ZZ = Diagnostic Package Identifier Designator

S = System
- E = VAX

P = Processor
- S = Star
- V = All VAX Processors

X = Option
- A = Application (CR, LP, A-D, Misc.)
- C = Channel Adapters (MBA, UBA)
- D = Communication Devices (DMC, DZ11)
- K = CPU
- M = Magnetic Tape (TE, TU, DT)
- R = Rotating Memory (RP, RK, RM, RS, RX)
- S = System Monitor (Supervisor, Tape Monitor)
- T = Terminal (LA36, VT52)
- U = Utilities (Script File, Copy, Update)
- X = System Exerciser (EXR/BUSINIT)
- Z = Media Packages
- 0-9 = Engineering File

Y = Hardware Level or Device Class*

Y = Hardware Microcode Base Level
- A = Base Level One (Grandfather)
- B = Base Level Two (Father)
- C = Base Level Three (Son)

Y = Device Class

(A)	A = LP
(A)	B = CR
(C)	A = MBA
(C)	B = UBA
(D)	A = DZ11
(D)	B = DMC
(M)	A = TE16/77
(M)	B = TM03/TUX
(R)	A = RP0X/RK06
(R)	B = RP0X
(R)	C = DCL/RP0X
(R)	D = RK06
(R)	E = RK611
(S)	A = SUPERVISOR
(S)	B = MAGTAPE MONITOR
(T)	A = LA36/VT52
(T)	B = TERMX
(U)	A = SCRIPT FILE
(U)	B = COPY
(X)	A = EXR/BUSINIT
(0–9)	= ENGINEERING MANUFACTURING OR ARCHIVAL FILE

D = DIAGNOSTIC. A THROUGH Z AND 1 THROUGH 9

R = REVISION (DECO)
P = PATCH (DEPO)

*The "Y" field will have a unique application when identifying CPU, device type or media packages. In the case of CPU diagnostics or hardware microcode that is hardware dependent, the "K" option will be followed by a hardware base level. These levels will be sequentially alphabetized following a "son/father/grandfather" rationale. The VAX CDSP (media) will follow these rules also for those packages containing CPU diagnostics that are hardware microcode dependent. In the case of peripheral or device type packages the "Y" field will be numeric, allowing for media expansion (i.e.: 1A through 9Z).

Examples: ZZ-ESZAA-A.0

This would be a VAX/STAR CPU CDSP that supports the first hardware base level.

ZZ-ESZ1A-A.0
This would indicate a VAX/STAR CDSP containing diagnostics for a device type or peripheral option.

Table L-5 Diagnostic Program Features

	Run Environment and Load Device			I/O Type		Resolution	
Program	**Standalone, Load from Floppy**	**Standalone, Load from System Device**	**User Mode Load from System Device**	**Direct I/O**	**Q I/O**	**Failing Module Callout**	**Failing Function Callout**
Microdiagnostics	X					X	
CPU Cluster Exerciser	X	X		X			X
Massbus Adapter Diagnostic	X	X		X		X	
Unibus Adapter Diagnostic	X	X		X		X	
RP0X DCL Test	X	X		X		X	
RK611 Diagnostic Parts A-E	X	X		X		X	
RK611 Manual Invention Tests	X	X		X			X
DMC Exerciser	X	X		X			X
RP0X Functional Diagnostic	X	X		X			X
Tape Drive Function Timer	X	X		X			X
RK/RP/RM Disk Reliability	X	X	X		X		X
Tape Reliability	X	X	X		X		X
RK/RP/RM Formatter	X	X	X		X		X
Multiterminal Exerciser	X	X	X		X		X
Load Terminal Diagnostic	X	X	X		X		X
Line Printer Diagnostic	X	X	X		X		X
Bus Interaction	X	X	X		X		X
EXER			X		X		X
UETP			X		X		X