SYSTEM SOFTWARE
System software is one of the three major categories of software. Refer to Figure 1.

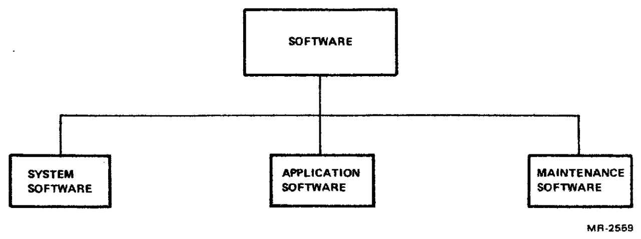


Figure 1   Three Major Categories of Software

System software consists of an operating system or monitor and a library of Commonly Used System Programs (CUSPs). Refer to Figure 2.

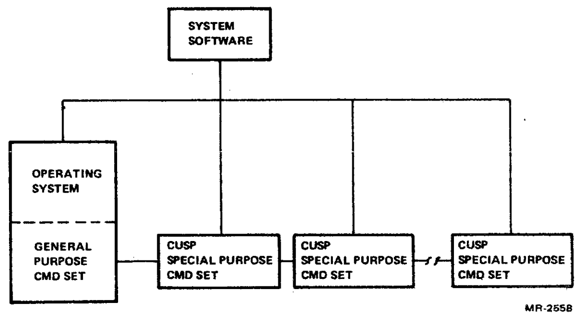

SYSTEM SOFTWARE
OPERATING SYSTEM
GENERAL PURPOSE CMD SET
CUSP SPECIAL PURPOSE CMD SET
CUSP SPECIAL PURPOSE CMD SET
CUSP SPECIAL PURPOSE CMD SET
MR-2558

Figure 2   Component Parts of System Software

The operating system directs and monitors the overall performance of the system and supports a general purpose command set. The CUSPs, in effect, extend the general purpose command set by supporting individual special purpose command sets.

SYSTEM MONITORING - Directing and monitoring the overall performance of the system is the most complex aspect of an operating system. It involves tasks such as scheduling jobs for execution, directing I/O operation, handling interrupts, and managing system resources. Although field maintenance personnel should have an overall understanding of this aspect of operating systems, an in-depth knowledge is not generally required.

COMMONLY USED SYSTEM PROGRAMS (CUSPs) - The number and type of CUSPs associateed with a given system program library depends largely on the intended use of the system. Regardless of the intended use of the system, however, the relationship between the operating system and the CUSPs in the corresponding system program library will remain the same. That is, the operating system will support a set of general purpose commands and each CUSP will support a unique set of special purpose commands. Refer to Figure 3.

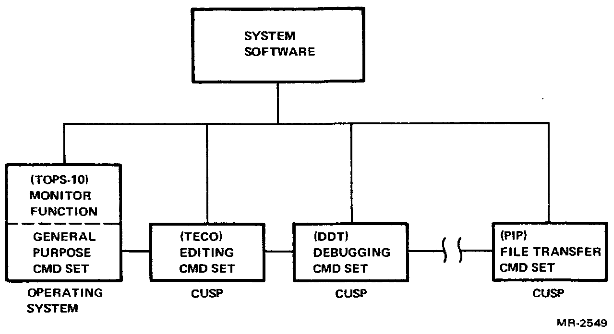


Figure 3 Typical Operating System and CUSP Relationship

Figure 3 uses the TOPS-10 operating system and three CUSPs from the TOPS-10 system program library to illustrate the relationship between operating systems and CUSPs.

The general purpose command set supported by the operating system enables system programmers, operators and users to perform the following functions: gain access to the system, run existing system and application software, communicate with system operators or other users on the system, request system resources and operator services as needed, and gather information concerning job and system performance.

Three of the CUSPs which extend or supplement the TOPS-10 general purpose command set are described below. Note that the CUSP command set is selected for use via one of the general purpose commands, usuallyy GET or RUN (e.g., RUN TECO<CR>).

The Text Editor and COrrector (TECO) supports commands which enable the user to build and edit an ASCII text file. Later, this file may be transformed into a usable program via an assembler or compiler-type CUSP.

The Dynamic Debugging Technique (DDT) supports a command set which allows the user to test and debug his program on-line before putting it into operation.

The Peripheral Interchange Program (PIP) supports commands which enable a user to copy or transfer files between standard peripheral devices.

For field maintenance personnel, command sets are the simplest and most important aspect of system software. Some skill and proficiency in using system software is essential to field maintenance personnel because system software must be used to maintain on-line file storage areas, run on-line (user mode) utility and diagnostic programs, and compile and print system error logs.

System Software Command Format
Operating systems and system library programs use a command format similar to the one illustrated in Figure 4.

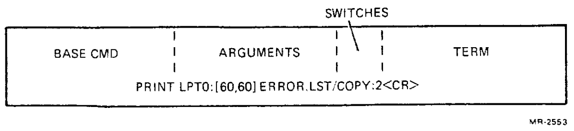


Figure 4 Typical System Software Command Format

BASE CMD - The base command is usually a verb which describes the task that the command will accomplish (e.g., GET, RUN, PRINT,etc.).

ARGUMENTS - The arguments specify the base command parameters. For example, the arguments supplied to the PRINT command illustrated in Figure 4 specify LPT0: as the output device, [60,60] as the project programmer numbers, and ERROR.LST as the file to be printed.

SWITCHES - Switches cause a minor modification to the basic action of the command. For example, the COPY:2 switch illustrated in Figure 4 will cause two copies of the file ERROR.LST to be printed instead of one, which is the default. For example, DIRECT [60,60]/FAST<CR>. The FAST switch associated with the DIRECTORY command will cause an abbreviated form of the directory area to be printed.

TERM - The command terminator, usually a carriage return <CR>, line feed <LF> or altmode <$>, directs the operating system or CUSP to execute the command. As a result of executing the command illustrated by Figure 4, line printer 0 will print two copies of the file ERROR.LST, which is stored in the [60,60] project programmer area of the default input device (in this case the system disk).

Although some system software commands do not require all of the command elements described above, and some will prompt for missing arguments, the basic format (BASE CMD ARGUMENTS SWITCHES TERM) will generally remain the same for all system software. Thus, learning to use system software is a relatively easy task.

Tips on Learning to Use System Software

The following are some tips you may find helpful when learning to use new system software.

1. Study the file structure and organization used by the operating system. This is important because many system software commands are related to file generation, modification and manipulation.

2. Think of system software in terms of command sets. Do not become overly concerned with the monitoring function.

3. Think of each command individually in terms of what task it will accomplish. Do not become overly concerned with how the command achieves the task.

4. Review the general purpose command set supported by the operating system. Become familiar with the type of commands that are available.

5. Review the abstract and command set associated with each CUSP in the system program library. Determine which CUSPs you are most likely to use on a regular basis.

6. Design some exercises which will help you develop skill and proficiency in using the system software. Remember perfect practice makes perfect.

7. Finally, and most important - DON'T be intimidated by system software. It is designed to be easy to use and there are a lot of people using it that know far less about computers than you do.

Table of Contents

GENERAL INFORMATION

The RSX-20F operating system is a real-time system executive for DECsystem-10 and DECSYSTEM-20 console front-end subsystems. RSX-20F supports the operator's console terminal (CTY), the KL10 console operations (start, stop, examine, deposit, etc.) and, in the case of DECSYSTEM-20s, the low-speed I/O devices (i.e., line printers, card readers, etc.).

Due to the limited core size of console front-end subsystems (28K), the general command set supported by RSX-20F resides in a nonresident task file referred to as PARSER. RSX-20F supports only one resident command (↑\). The ↑\ command will cause the PARSER task file to be loaded and the general command set to be made available to the user.

Table 1 RSX-20F Command Summary

Command	Description
↑\	↑\ The control backslash command causes RSX-20F to load and start the PARSER task file. PARSER supports the general command set normally associated with an operating system. Refer to the PARSER module.

ERROR MESSAGES

RSX-20F stop codes are listed and described in Table 2.

Table 2 RSX-20F Stop Codes

Stop Code	Module	Meaning
BF1	QPRDTE	Buffer Failure 1 Free space allocation failed for DTE20 protocol header.
BO2	QPRDTE	Bufffer Overflow 2 Free space allocation failed for data transfer request in a TO11 request.
BO3	SCOMM	Buffer Overflow 3 Free space allocation failed on a TO10 queue request.
CBR	PF	Crobar Error DTE20 power has not returned after a power-fail restart.
DTB	QPRDTE	TO11 DTE20 Transfer Failure The TO11 address after a TO11 transfer is not what was expected.
DTD	LC	DTE20 Is Dead The 11 got a Unibus timeout on the DTE20 with no power fail.
DTF	QPRDTE	TO10 DTE20 Transfer Failure The TO10 address in the DTE20 after a TO10 transfer was not what was expected.
EPE	QPRDTE	EBus Parity Error A DTE20 command has resulted in an KL10 EBus parity error.
ETE	QPRDTE	TO11 Transfer Error After a TO11 transfer, TO11ER (TO11 Error) is on in the DTE20 status register.
FTA	LC	Files Task Aborted A task occupying F11TPD has aborted, TKTN cannot be requested.
IAS	SCH	Unknown Significant Event An unused bit in .SERFG has been set.
ILF	QPRDTE	Illegal Function - Protocol A protocol function within the legal range but currently unimplemented has been received.

Table 2 RSX-20F Stop Codes (Cont)

Stop Code	Module	Meaning
ILQ	QPRDTE	Illegal Queue Count The protocol queue was not expected (i.e., was not incremented by 1).
LRF	SCH	Load Request FAILED An attempt to load a nonresident monitor routine into the F11TPD partition failed.
MPE	LC	Memory Parity ERROR An 11 parity error has occurred.
NPF	DMDTE	Nonprivileged Front End A front end connected to a DTE20 tried to enter boot protocol.
PT1	QPRDTE	Protocol Broken An illegal protocal device number was specified in a TO11 request.
PT2	QPRDTE	Protocol Error 2 An illegal protocol function was specified in a TO11 request.
PT3	QPRDTE	Protocol Error 3 The indirect in progress bit was set in the protocol; however, no TO11 request was in progress.
PT4	QPRDTE	Protocol Error 4 Queue size has exceeded 100(10).
RES	LC	Reserved Instruction Trap This is the PDP-11 trap to location 10. An attempt was made to execute an illegal or reserved instruction.
TO4	LC	Trap at Location 4 Odd address, timeout, stack violation, RP04/06 error.
TBT	LC	T-Bit Trap A BPT instruction was executed or the T-Bit was set by an RTI/RTT.
TET	QPRDTE	TO10 Transfer Error Either TO10ER or MPE11 is up in the DTE20 status register.
UIE	QPRDTE	Unimplemented Protocol Function Either bit 0 or bit 2 was set in the protocol status word.
UNT	LC	Unrecognized Trap Error A trap occurred to an unused vector.

RSX-20F SYSTEM PROGRAM LIBRARY

The RSX-20F System Program Library consists of four kinds of files.

Microcode files. These files are for the KL10 and are listed and described in Table 1.

Boot files. These files are used by the front-end subsystem to boot the KL10. They are listed and described in Table 2.

Automatic task files. These files are used by RSX-20F for various housekeeping tasks and are not normally loaded by the user. They are listed and described (for reference purposes) in Table 3.

User task files. These files are listed and described in Table 4.

Table 1 RSX-20F System Program Library Microcode Files

Task	Description
KLA.MCB	Microcode file for KL10 model PAs.
KLX.MCB	Microcode file for KL10 model PVs.

Table 2 RSX-20F System Program Library Boot Files

Task	Description
BOOT.EXB	Boot Boots KL10 monitor system image into KL's core from RIGID disk; is written in executable binary KL code.
MTBOOT.EXB	Magtape Boot Allows transfer of a program's core image from magtape into KL10's core; is written in executable binary KL code.

Table 3 RSX-20F System Program Library Auto Tasks

Task	Description
F11ACP.TSK	Files-11 Ancillary Control Processor File handler for front-end disk files (performs file access, management, and control functions).
KLE.TSK	KL Error Error processing of KL10 errors. Uses diagnostic DTE functions. Produces "snapshot" of KL10 error conditions for troubleshooting. Calls KLINIT when done.
KLI.TSK	KL Initialization Initializes the KL10 processor (produces installation dialogue, loads microcode, runs bootstrap, etc.). Called whenever system comes up.
KLR.TSK	KLINIK Request Checks KLINIK time window and KLINIK password when KLINIK line rings. If they are correct, it then enables KLINIK.

Table 3 RSX-20F System Program Library Auto Tasks (Cont)

Task	Description
KLX.TSK	KL Transfer Transfers KLEER.SNP to SYSERR file in KL10. (Not to be confused with KLX.MCB, which is the filename of the KL10-PV microcode.)
MIDNIT.TSK	Midnight Roll over time of day at midnight.
SETSPD.TSK	Set Speed Sets line speed table for -10 after restart and sets the time in the -10. NOTE Do not confuse this with the TOPS-20 program SETSPD.EXE. SETSPD.TSK is a front-end task and it does not access CNFG.CMD.
TKTN.TSK	Task Termination Program Outputs task termination notification and provides orderly termination for front-end tasks. Interfaces between KLINIT and KLERR (lets KLE call KLI).
T20APC.TSK	TOPS-20 Ancillary Control Processor File handler for files to be transferred to and from the KL10's disk area. Interacts with TOPS-20 area in terms compatible with FILES-11 operations.
UFD.TSK	User File Directory Sets up directories in FILES-11 area. Directories are "named" by a UIC (user identification code) and enclosed in brackets: [X, Y].

Table 4 RSX-20F System Program Library User Tasks

Task	Description
COP.TSK	Copy Floppy disk copy utility. Also allows verification of physical state of the disk, as well as verification of successful copying.
DMO.TSK	Dismount Removes a device from the front-end system's knowledge, making its contents inaccessible to the user.
FEDDT.TSK	Front-End DDT Symbolic debuger for RSX-20F. Permits user to read and print selected portions of front-end crashes.
INT.TSK	Initialize Initializes FILES-11 devices to be recognizable FILES-11 "VOLUMES". Sets up master directory space, index and home blocks, etc.
MOU.TSK	Mount Makes a device known to the system so that it can be accessed by a given user.
PARSER.TSK	Command Parser Primary means of access to front-end programs. Provides access to KL10's memory for diagnostic functions, as well as debugging tools. Will interface with KLINIK in future versions.
PIP.TSK	Peripheral Interchange Program Performs general file transfer and some maintenance functions among FILES-11 devices and other peripherals (e.g., floppy-to-disk file transfers, file deletions, typing directories at console, etc.).
RED.TSK	Redirect Changes front-end system's "home" from one FILES-11 device to another, and tells system where it resides presently.
SAV.TSK	Save Saves core image of front-end on RIGID disk in FILES-11 area.
ZAP.TSK	Zap Permits direct examination and modification of files on a FILES-11 volume. Patch task images and data files in an interactive environment.

PARSER

GENERAL INFORMATION

The command PARSER runs as a task under the RSX-20F executive. Its primary function is to receive ASCII command strings, usually from the console terminal, and perform console functions on the KL10 or PDP-11 computer.

	Control Backslash - Command to RSX-20F to load and run PARSER
PAR>	Prompt - Indicates PARSER is ready to accept commands, and the KL10 clock and run flip-flop are on
PAR%	Prompt - Indicates PARSER is ready to accept commands, the KL10 run flip-flop is off, and the KL10 clock is on
PAR#	Prompt - Indicates PARSER is ready to accept commands, and the KL10 clock is off. This may indicate an error condition
QUIT or ^Z or SET CON/USER	Exit PARSER - Return to RSX-20F command mode. The CTY is connected to the program running in the KL10
Note	1. Commands and arguments may be abbreviated to the simplest form that uniquely identifies them; e.g., the EXAMINE command may be typed as E since no other commands begin with E. 2. The maximum number of characters in a command line is 280. 3. Numeric arguments default to decimal unless they are address or data arguments. Then they default to octal.

COMMAND CONVENTIONS

The command conventions and special characters used by PARSER are described in Table 1.

COMMAND SUMMARY

The command PARSER has four modes of operation. The mode is set by the SET CONSOLE command.

Maintenance Mode - Enables the commands described in Table 2.

Operator Mode - Enables the commands listed in Table 3.

Programmer Mode - Enables the commands listed in Table 4.

User Mode - Connects the console to the program running in the KL10. No PARSER commands are in effect.

For a description of the commands listed in Table 3 and Table 4, refer to Table 2.

Table 1 Command PARSER Special Characters

Character	Meaning
?	PAR>?<CR> or PAR>SET?<CR> A question mark typed at PARSER command, subcommand, or argument level will cause a brief help message to be displayed.
;	PAR>E PC;E 20;SH<CR> Used to separate individual commands within a command line.
!	PAR>REP 5;E PC! SEE IF CPU IS IN HALT LOOP<CR> Indicates a comment line.
<CR>	PAR>SH<CR> Command line terminator - causes the command line to be executed.
-<CR>	PAR>ST M0-<CR> Nullifies the <CR> terminator - allows the command line to be continued on the next line. The continuation line will prompt with another dash.

Table 1 Command PARSER Special Characters (Cont)

Character	Meaning
^C	PAR<DE T 100:^C5<CR> Digits preceded by an up arrow and a C are interpreted as 1's complement.
^D	PAR>DE E 200:^D5<CR> Digits preceded by an up arrow and a D are interpreted as decimal.
^B	PAR>DE T 200:^B1010<CR> Digits preceded by an up arrow and a B are interpreted as binary.
^O	PAR>DE T 200:^O5252<CR> Digits preceded by an up arrow and an O are interpreted as octal (default).
^O	A control O can also be used to suppress printouts.
^Z	A control Z causes PARSER to exit. The console is connected to the program running in the KL10.
'	PAR>E E 34'<CR> A single quote adds the current value of the relocation switch to the number. See SET OFFSET.
"	PAR>E E 34"CR> A double quote subtracts the current value of the relocation switch from the number. See SET OFFSET.
-	PAR>DE T 30:-1<CR> A string of digits preceded by a hyphen (minus sign) is interpreted as the 2's complement of the value of the string.
+-*/	Two numeric expressions separated by plus, minus, asterisk, or slash are evaluated by applying the operations of addition, subtraction, multiplication or division, respectively.
_	Two numeric expressions separated by underscore are evaluated by shifting the first left by the second. Example: 1_3 is 10 octal.
(2*8)/4	Parentheses may be used to enclose expressions. Thus parentheses can be used to change the implicit order of arithmetic operations.

Table 2 PARSER Maintenance Mode Command Summary

Command	Description	Cross Ref.
ABORT	PAR>A<CR> Force the KL10 into the HALT loop. See HALT.	1
CLEAR	PAR>CL arg<CR> The CLEAR command accepts the following arguments. See SET commands. CLOCK e.g., PAR>CL CL CON<CR> The CLEAR CLOCK command accepts the following arguments. CONTROL e.g., PAR>CL CL CON<CR> Disable the control logic clock. CRAM e.g., PAR>CL CL CR<CR> Disable the CRAM clock. DATA-PATH e.g., PAR>CL CL D<CR> Disable the data path clock. EXTERNAL e.g., PAR>CL CL E<CR> Select the internal KL10 clock source. Same as SET CLOCK INTERNAL.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	FULL e.g., PAR>CL CL F<CR> Set the KL10 clock rate to full speed. Same as SET CLOCK FULL.	
	HALF e.g., PAR>CL CL H<CR> Set the KL10 clock rate to full speed. Same as SET CLOCK FULL.	
	INTERNAL e.g., PAR>CL CL I<CR> Select the internal KL10 clock source. Same as SET CLOCK INTERNAL.	
	MARGIN e.g., PAR>CL CL M<CR> Select the internal KL10 clock source. Same as SET CLOCK INTERNAL.	
	NORMAL e.g., PAR>CL CL N<CR> Set the KL10 clock parameters to internal source and full rate with the CRAM, DATA-PATH and CONTROL clocks enabled.	
	QUARTER e.g., PAR>CL CL Q<CR> Set the KL10 clock rate to full speed. Same as SET CLOCK FULL.	
	SLOW e.g., PAR>CL CL S<CR> Set the KL10 clock rate to full speed. Same as SET CLOCK FULL.	
	CONSOLE e.g., PAR>CL C<CR> Put the console front end into operator mode. Equivalent to SET CONSOLE OPERATOR.	
	DATE e.g., PAR>CL D<CR> Clear the date validity bit and prompt for a new date and time. This command is invalid if RSX-20F is in primary protocol; i.e., if the public structure (PS) is mounted. See SET DATE.	
	FS-STOP e.g., PAR>CL FS<CR> Disable the field service clock error stop feature. Same as CLEAR PARITY-STOP FS-STOP.	
	INCREMENT e.g., PAR>CL I<CR> Set the KL10 increment factor to 0. See SET INCREMENT.	
	KLINIK e.g., PAR>CL K<CR> Clear KLINIK parameters (only).	16
	MEMORY e.g., PAR>CL M<CR> Make KL10 memory the default for deposits and examines. Not to be confused with zeroing memory. See SET MEMORY and ZERO.	
	NOT e.g., PAR>CL NO REL<CR> Used with CLEAR to negate the clear function. It is equivalent to SET.	
	OFFSET e.g., PAR>CL O<CR> Set the value of the PDP-11 relocation counter to 0. See SET OFFSET.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	PARITY-STOP e.g., PAR>CL P ALL<CR> The CLEAR PARITY-STOP command accepts the following arguments.	
	ALL e.g., PAR>CL P ALL<CR> Disable all parity stop features.	
	AR e.g., PAR>CL P AR<CR> Disable the AR and ARX parity stop features.	
	CRAM e.g., PAR>CL P C<CR> Disable the CRAM parity stop feature.	
	DRAM e.g., PAR>CL P D<CR> Disable the DRAM parity stop feature.	
	ENABLE e.g., PAR>CL P E<CR> Clear all parity stop enables. Same as CLEAR PARITY-STOP ALL<CR>	
	FM e.g., PAR>CL P FM<CR> Disable the fast memory (FM) parity stop feature.	
	FS-STOP e.g., PAR>CL P FS<CR> Disable the field service clock error feature. Same as CLEAR FS-STOP.	
	RELOAD e.g., PAR>CL REL<CR> Disable the automatic reloading of the KL10 following a fatal error condition.	
	REPEAT e.g., PAR>CL REP<CR> Set the repeat counter to 0. All subsequent command lines will be repeated once. See SET REPEAT.	
	RETRY e.g., PAR>CL RET<CR> Clear the PARSER RETRY flag. Every KEEP-ALIVE-CEASED error will cause a KLERR snapshot before reloading the KL10.	
	TRACKS e.g., PAR>CL T<CR> Clear the KL10 tracking function. See SET TRACKS.	10
CONTINUE	PAR>CO<CR> Continue the KL10 running if it is continuable (i.e., the KL10 has not been reset). See START.	2
DEPOSIT	PAR>DE T N:500<CR> The DEPOSIT command accepts the following arguments. Default: see SET MEMORY. The previous contents of the location or argument specified will be displayed.	
	AR e.g., PAR>DE A:777777777777<CR> Load data (777777777777) into the AR.	
	ELEVEN e.g., PAR>DE E 2000:500<CR> Deposit data (500) into PDP-11 location specified (2000).	
	DEPOSIT ELEVEN accepts the following arguments. Default: THIS.	
	DECREMENT e.g., PAR>DE E D:500<CR> Deposit data (500) into the last PDP-11 location referenced minus two (-2).	
	INCREMENT e.g., PAR>DE E I:500<CR> Deposit data (500) into the last PDP-11 location referenced plus two (+2).	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	NEXT e.g., PAR>DE E N:500<CR> Same as DE E I:500<CR> (INCREMENT) PREVIOUS e.g., PAR>DE E P:500<CR> Same as DE E D:500<CR> (DECREMENT) THIS e.g., PAR>DE E T:500<CR> Deposit data (500) into the last PDP-11 location referenced. THIS is the default.	
	TEN e.g., PAR>DE T 30000:500<CR> Deposit data (500) into PDP-10 location specified (30000). All references are to a physical address. Paged (user) deposits are not supported by PARSER. DEPOSIT TEN accepts the following arguments. Default: THIS	
	DECREMENT e.g., PAR>DE T D:500<CR> Deposit data (500) into the last PDP-10 location referenced minus the increment value. See SET INCREMENT. INCREMENT e.g., PAR>DE T I:500<CR> Deposit data (500) into the last PDP-10 location referenced plus the increment value. See SET INCREMENT. NEXT e.g., PAR>DE T N:500<CR> Deposit data (500) into the last PDP-10 location referenced plus one (+1). PREVIOUS e.g., PAR>DE T P:500<CR> Deposit data (500) into the last PDP-10 location referenced minus one (-1) THIS e.g., PAR>DE T T:500<CR> Deposit data (500) into the last PDP-10 location referenced. THIS is the default.	
DISCONNECT	PAR>DI<CR> Disconnect the KLINIK link by running KLDISC.TSK. The existing KLINIK parameters are not affected. See CLEAR KLINIK.	
EXAMINE	PAR>EX T 3000<CR. The EXAMINE command accepts the following arguments. Default: see SET MEMORY. ELEVEN e.g., PAR>EX EL 3000<CR> Display the contents of the PDP-11 location specified (3000). EXAMINE ELEVEN accepts the following arguments. Default: THIS.	
	DECREMENT e.g., PAR>EX EL D<CR> Display the contents of the last PDP-11 location referenced minus two (-2). INCREMENT e.g., PAR>EX EL I<CR> Display the contents of the last PDP-11 location referenced plus two (+2). NEXT e.g., PAR>EX EL N<CR> Same as EX EL I<CR> (INCREMENT) PREVIOUS e.g., PAR>EX EL P<CR> Same as EX E D<CR> (DECREMENT)	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	THIS e.g., PAR>EX EL T<CR> Display the contents of the last PDP-11 location referenced. THIS is the default. TEN e.g., PAR>EX T 30000<CR> Display the contents of the PDP-10 location specified (30000). All references are to a physical address. Paged (user) examines are not supported by PARSER. EXAMINE TEN accepts the following arguments. Default: THIS. DECREMENT e.g., PAR>EX T D<CR> Display the contents of the last PDP-10 location referenced minus the increment value. See SET INCREMENT. INCREMENT e.g., PAR>EX T I<CR> Display the contents of the last PDP-10 location referenced plus the increment value. See SET INCREMENT. NEXT e.g., PAR>EX T N<CR> Display the contents of the last PDP-10 location referenced plus one (+1). PREVIOUS e.g., PAR>EX T P<CR> Display the contents of the last PDP-10 location referenced minus one (-1). THIS e.g., PAR>EX T T<CR> Display the contents of the last PDP-10 location referenced. THIS is the default. AB e.g., PAR>EX AB<CR> Display the contents of the Address Break register. AD e.g., PAR>EX AD<CR> Display the state of the ADder. ADX e.g., PAR>EX ADX<CR> Display the state of the ADder Extended AR e.g., PAR>EX AR<CR> Display the contents of the Arithmetic Register. ARX e.g., PAR>EX ARX<CR> Display the contents of the Arithmetic Register eXtended. BR e.g., PAR>EX BR<CR> Display the contents of the Buffer Register. BRX e.g., PAR>EX BRX<CR> Display the contents of the Buffer Register eXtended. CRADDR e.g., PAR>EX CRADDR<CR> Display the contents of the Cram ADDRess register. CRLOC e.g., PAR>EX CRLOC<CR> Display the contents of the CRAM LOCation register.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	DRADDR e.g., PAR>EX DRADDR<CR> Display the contents of the DRAM ADDRess register.	
	DTE-20 e.g., PAR>EX DTE<CR> Display the contents of the three DIAG registers and the status register in the DTE20.	
	EBUS e.g., PAR>EX EBUS<CR> Display the contents of the EBus.	
	FE e.g., PAR>EX FE<CR> Display the contents of the Floating Exponent register.	
	FLAGS e.g., PAR>EX FLAGS<CR> Display the state of the flag bits (00-12) in the left half of the PC: OVF, CY0, CY1, FOV, BIS, USR, UIO, LIP, AFI, AT1, AT0, FUF and NOV.	
	FM e.g., PAR>EX FM<CR> Display the contents of the Fast Memory register.	
	KL e.g., PAR>EX KL<CR> Perform, in order, an EX PC, EX VMA, EX PI and EX FLAGS.	
	MQ e.g., PAR>EX MQ<CR> Display the contents of the Multiplier Quotient register.	
	PC e.g., PAR>EX PC<CR> Display the contents of Program Counter.	
	PI e.g., PAR>EX PI<CR> Display the state of the Priority Interrupt system.	
	REGISTERS e.g., PAR>EX REG<CR> Display the contents of the following registers: AD, ADX, AR, ARX, BR, BRX, EBUS, FM, MQ, and PC.	
	SBR e.g., PAR>EX SBR<CR> Display the contents of the Subroutine Return register.	
	SC e.g., PAR>EX SC<CR> Display the contents of the Shift Count register.	
	VMA e.g., PAR>EX VMA<CR> Display the contents of the Virtual Memory Address register.	
	VMAH e.g., PAR>EX VMAH<CR> Display the contents of the Virtual Memory Address Held register.	
FREAD	PAR>FR 110<CR> Display the result of a diagnostic function read using the function code specified (110). The function code must be in the range of 100 to 177.	
FWRITE	PAR>FW 77:252525777777<CR> Perform a diagnostic function write using the function code (77) and data (252525777777) specified. The function code must be in the range of 40 to 77.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
FXCT	PAR>FX 0<CR> Perform a diagnostic function execute using the function code specified (0). The function code must be in the range of 00 to 37.	4
HALT	PAR>H<CR> Halt the KL10. See ABORT and SHUTDOWN.	5
INITIALIZE	PAR>I<CR> Check the state of the KL10 clock, run flip-flop and opcode enable.	6
JUMP	PAR>J 30000<CR> Start the KL10 at the address specified (30000) and exit. The address is in the executive space and the processor mode is not affected. See START TEN.	
MCR	PAR>M BOOT<CR> Load and start the specified task file (BOOT.TSK). Same as RUN.	
QUIT	PAR>Q<CR> Exit from PARSER. Same as SET CONSOLE USER<CR> or ^Z.	
REPEAT n	PAR>REP 2;EX T N<CR> Cause the command(s) in the remainder of the line to be repeated n(2) times.	7
RESET	PAR>RES ALL<CR> The RESET command accepts the following arguments. Default: <CR>. <CR> e.g., PAR>RES<CR> Cause a master reset of the KL10. The state of the clock enables and parity stops are not affected. This is the default. ALL e.g., PAR>RES AL<CR> Perform a RES APR, RES DTE-20, RES PAG and RES PI command. The KL10 must be halted. APR e.g., PAR>RES AP<CR> Execute a CONO APR,267760. The KL10 must be halted.	
	DTE-20 e.g., PAR>RES D<CR> Reset the DTE20. ERROR e.g., PAR>RES E<CR> Execute a CONO APR,27760 clearing the error flags in the Arithmetic Process Register (APR). INITIALIZE e.g., PAR>RES IN<CR> Perform a KL10 master reset and return clock enables and parity stops to their default. The KL10 must be halted. IO e.g., PAR>RES IO<CR> Execute a CONO APR,200000 which causes an I/O reset. PAGE e.g., PAR>RES PAG<CR> Execute a CONO PAG,0 followed by a DATAO PAG,X (where the contents of X = 100). This will reset the KL10 paging box. PI e.g., PAR>RES PI<CR> Execute a CONO PI,10000 which resets the Priority Interrupt system.	8

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
RUN	PAR>RU PIP<CR> Load and run the specified task file (PIP.TSK). Same as MCR.	
SET	PAR>SET MEM TEN<CR> The SET command accepts the following arguments.	
	CLOCK e.g., PAR>SET CL N<CR> The SET CLOCK command accepts the following arguments.	
	CONTROL e.g., PAR>SET CL CON<CR> Enable the control logic clock.	
	CRAM e.g., PAR>SET CL CR<CR> Enable the CRAM clock.	
	DATA-PATH e.g., PAR>SET CL D<CR> Enable the data path clock.	
	EXTERNAL e.g., PAR>SET CL E<CR> Set (select) the KL10 external clock source. PARSER will request confirmation.	9
	HALF e.g., PAR>SET CL H<CR> Set the KL10 clock rate to one half of the standard (divide by 2).	
	INTERNAL e.g., PAR>SET CL I<CR> Set (select) the KL10 internal clock source.	
	MARGIN e.g., PAR>SET CL M<CR> Set (select) KL10 clock margins.	
	NORMAL e.g., PAR>SET CL N<CR> Set the KL10 clock rate to the standard (internal source, full rate with CRAM, data-path and control logic clocks enabled).	
	QUARTER e.g., PAR>SET CL Q<CR> Set the KL10 clock rate to one quarter of the standard (divide by 4).	
	SLOW e.g., PAR>SET CL S<CR> Set the KL10 clock rate to one eighth of the standard (divide by 8).	
	CONSOLE e.g., PAR>SET CON M<CR> The SET CONSOLE command accepts the following arguments.	
	MAINTENANCE e.g., PAR>SET CON M<CR> Set the console to maintenance mode. The command set is unrestricted. Refer to Table 2.	
	OPERATOR e.g., PAR>SET CON O<CR> Set the console to operator mode. The command set is restricted to those listed in Table 3.	
	PROGRAMMER e.g., PAR>SET CON P<CR> Set the console to programmer mode. The command set is restricted to those listed in Table 4.	
	USER e.g., PAR>SET CON U<CR> Exit PARSER. Leave the CTY connected to the program running in the KL10.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	DATE e.g., PAR>SET D<CR> Set the date and time to be used by the front-end executive, RSX-20F. This command is illegal if RSX-20F already has a valid date from a previous SET DATE command or a reload of the KL10.	
	FS-STOP e.g., PAR>SET F<CR> Enable the Field Service Clock Error Stop feature in the KL10. This requires backplane jumper wires to be meaningful. Same as SET PARITY-STOP FS-STOP.	
	INCREMENT e.g., PAR>SET I 10<CR> Set the increment and decrement value for KL10 deposit and examine commands to the value specified (10).	
	KLINIK e.g., PAR>SET K<CR> Set the KLINIK link for remote console operation.	15
	MEMORY e.g., PAR>SET M T<CR> The SET MEMORY command accepts the following arguments.	
	ELEVEN e.g., PAR>SET M E<CR> Set the PDP-11 as the default memory for deposits and examines.	
	TEN e.g., PAR>SET M T<CR> Set the KL10 as the default memory for deposits and examines.	
	NOT e.g., PAR>SET NO RELOAD<CR> Used with SET to negate the SET function. It is equivalent to CLEAR.	
	OFFSET e.g., PAR>SET O 101204<CR> Set the PDP-11 relocation counter to the value specified (101204). The relocation counter is initially set to the address of the PARSER root overlay.	
	PARITY-STOP e.g., PAR>SET P ALL<CR> The SET PARITY-STOP command accepts the following arguments.	
	ALL e.g., PAR>SET P ALL<CR> Set the parity stop enable to on and enable the following parity stop features. AR, CRAM, DRAM, FM and FS-STOP.	
	AR e.g., PAR>SET P AR<CR> Add stop on AR and ARX parity error to the parity stop features.	
	CRAM e.g., PAR>SET P C<CR> Add stop on CRAM parity error to the parity stop conditions.	
	DRAM e.g., PAR>SET P D<CR> Add stop on DRAM parity error to the parity stop conditions.	
	ENABLE e.g., PAR>SET P E<CR> Enable (turn on) the selected PARITY-STOP features.	
	FM e.g., PAR>SET P FM<CR> Add stop on a fast memory (FM) parity error to the parity stop conditions.	
	FS-STOP e.g., PAR>SET P FS<CR> Enable the Field Service Clock Error Stop feature in the KL10. This requires backplane jumper wires. Same as SET FS-STOP.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	RELOAD e.g., PAR>SET REL<CR> Enable the automatic reload of the KL10 by the PDP-11 front end. This is the default. See CLEAR RELOAD.	
	REPEAT e.g., PAR>SET REP 5<CR> Set the repeat counter to the decimal value specified. All subsequent command lines will be repeated that number of times. The value will also be used as a multiplier by the REPEAT command.	7
	RETRY e.g., PAR>SET RET<CR> Set the PARSER RETRY flag. See CLEAR RETRY.	17
	TRACKS e.g., PAR>SET T<CR> Display all FR, FW, FX, Examine, Deposit, and DTE-20 operations.	10
SHUTDOWN	PAR>SH<CR> Gracefully shut down the TOPS-10 or TOPS-20 operating system. This is done by depositing a minus 1 in location 30. Timesharing ceases.	11
START	PAR>ST M 0<CR> or PAR>ST T 2000<CR> The START command accepts the following arguments. START with no arguments or an argument of 0 is illegal. If neither TEN nor MICROCODE is specified, TEN is assumed.	
	MICROCODE e.g., PAR>ST M 0<CR> Start the microcode at the address specified (0).	12
	TEN e.g., PAR>ST T 3000<CR> Start the KL10 at the address specified (3000). See CONTINUE and JUMP.	13
WHAT	PAR>W CL<CR> The WHAT command accepts the following arguments.	
	CLOCK e.g., PAR>W CL<CR> Display the current clock state. See SET CLOCK.	
	CONSOLE e.g., PAR>W CON<CR> Display the current console mode. See SET CONSOLE.	
	DATE e.g., PAR>W D<CR> Display the state of the validity flag and the current date and time held by RSX-20F.	
	INCREMENT e.g., PAR>W I<CR> Display the current increment/decrement value. See SET INCREMENT.	
	KLINIK e.g., PAR>W K<CR> Display the current status of the KLINIK link. See SET KLINIK.	15
	MEMORY e.g., PAR>W M<CR> Display the current default memory. See SET MEMORY.	
	OFFSET e.g., PAR>W O<CR> Display the current value of the PDP-11 relocation counter. See SET OFFSET.	

Table 2 PARSER Maintenance Mode Command Summary (Cont)

Command	Description	Cross Ref.
	PARITY-STOP e.g., PAR>W P<CR> Display the current state of the parity stop feature. See SET PARITY-STOP.	
	RELOAD e.g., PAR>W REL Display the current state of the KL10 automatic reload feature (ON or OFF). See SET RELOAD.	
	REPEAT e.g., PAR>W REP<CR> Display the current value of the repeat counter. See SET REPEAT.	
	RETRY e.g., PAR>W RET<CR> Display the state of the PARSER RETRY flag. See SET RETRY.	17
	TRACKS e.g., PAR>W T<CR> Display the current state of the trace enable feature (ON or OFF). See SET TRACKS.	10
	VERSION e.g., PAR>W V<CR> Display the current version of PARSER and RSX-20F.	
XCT	PAR>X 254200000000<CR> Execute the argument (245200000000) as a PDP-10 instruction. The KL10 must be in executive mode.	14
ZERO	PAR>Z 200,277<CR> Zero PDP-10 physical memory from first argument (200) through second argument (277). Note: depending on the amount of memory this may take a while.	

Table 3 PARSER Operator Mode Command Summary

Command	Description
ABORT	PAR>A<CR>
CLEAR	PAR>CL C<CR> or PAR>CL R<CR> etc.
	The CLEAR command accepts the following arguments.
	CONSOLE KLINIK NOT
	INCREMENT MEMORY REPEAT
DISCONNECT	PAR>DI<CR>
EXAMINE	KL e.g., PAR>EX KL<CR>
	PC e.g., PAR> EX PC<CR>
	ELEVEN e.g., PAR>EX EL adr<CR>
	DECREMENT e.g., PAR>EX EL D<CR>
	INCREMENT e.g., PAR>EX EL I<CR>
	NEXT e.g., PAR> EX EL N<CR>
	PREVIOUS e.g., PAR> EX EL P<CR>
	THIS e.g., PAR> EX EL T<CR>
	TEN e.g., PHR>EX T adr<CR>
	DECREMENT e.g., PAR>EX T D<CR>
	INCREMENT e.g., PAR>EX T I<CR>
	NEXT e.g., PAR>EX T N<CR>
	PREVIOUS e.g., PAR>EX T P<CR>
	THIS e.g., PAR>EX T T<CR>

Table 3 PARSER Operator Mode Command Summary (Cont)

Command	Description
JUMP	PAR>J 30000<CR>
MCR	PAR>MCR BOOT<CR>
QUIT	PAR>Q<CR>
REPEAT	PAR>REP 2:EX T N<CR>
RUN	RU PIP<CR>
SET	CONSOLE e.g., PAR>SET CON M<CR>
	The SET console command accepts the following four arguments: USER, OPERATOR, PROGRAMMER and MAINTENANCE.
	INCREMENT e.g., PAR>SET I 10<CR>
	KLINIK e.g., PAR>SET K<CR>
	MEMORY e.g., PAR>SET M E<CR> or PAR>SET M T<CR>

Table 4 PARSER Programmer Mode Command Summary

Command	Description
ABORT	PAR>A<CR>
CLEAR	PAR>CL C<CR> or PAR>CL T<CR> etc.
	The CLEAR command accepts the following arguments.
	CONSOLE MEMORY REPEEAT DATE NOT RETRY INCREMENT OFFSET TRACKS KLINIK RELOAD
CONTINUE	PAR>CO<CR>
DEPOSIT	AR e.g., PAR>DE A:data<CR>
	ELEVEN e.g., PAR>DE E adr: data<CR>
	DECREMENT e.g., PAR>DE E D:data<CR>
	INCREMENT e.g., PAR>DE E I:data<CR>
	NEXT e.g., PAR>DE E N:data<CR>
	PREVIOUS e.g., PAR>DE E P:data<CR>
	THIS e.g., PAR>DE E T:data<CR>
	TEN e.g., PAR>DET adr:data<CR>
	DECREMENT e.g., PAR>DE T D:data<CR>
	INCREMENT e.g., PAR>DE T I:data<CR>
	NEXT e.g., PAR>DE T N:data<CR>
	PREVIOUS e.g., PAR>DE T P:data<CR>
	THIS e.g., PAR>DE T T:data<CR>
DISCONNECT	PAR>DI<CR>

Table 4 PARSER Programmer Mode Command Summary (Cont)

Command	Description
EXAMINE	PAR>EX AB<CR> or PAR>EX PC<CR> etc.
	The EXAMINE command accepts any of the following arguments.
	AB CRLOC MQ
	AD DRADDR PC
	ADX DTE-20 PI
	AR EBUS REGISTERS
	ARX FE SBR
	BR FLAGS SC
	BRX FM VMA
	CRADDR KL VMAH
	ELEVEN e.g., PAR>EX EL adr<CR>
	DECREMENT e.g., PAR>EX EL D<CR>
	INCREMENT e.g., PAR>EX EL I<CR>
	NEXT e.g., PAR>EX EL N<CR>
	PREVIOUS e.g., PAR>EX EL P<CR>
	THIS e.g., PAR>EX EL T<CR>
	TEN e.g., PAR>EX T adr<CR>
	DECREMENT e.g., PAR>EX T D<CR>
	INCREMENT e.g., PAR>EX T I<CR>
	NEXT e.g., PAR>EX T N<CR>
	PREVIOUS e.g., PAR>EX T P<CR>
	THIS e.g., PAR>EX T T<CR>
HALT	PAR>H<CR>
INITIALIZE	PAR>I<CR>
JUMP	PAR>J 30000<CR>
MCR	PAR>MCR BOOT<CR>
QUIT	PAR>Q<CR>
REPEAT	PAR>REP 2;EX T N<CR>
RESET	PAR>RES ALL<CR> or PAR>PAG<CR> etc
	The RESET command accepts the following arguments.
	ALL ERROR PAG
	APR INITIALIZE PI
	DTE-20 I/O
RUN	PAR>RU PIP<CR>
SET	CONSOLE e.g., PAR>SET CON M<CR>
	The SET CONSOLE command accepts four arguments; USER, OPERATOR, PROGRAMMER and MAINTENANCE.
	DATE e.g., PAR>SET D<CR>
	INCREMENT e.g., PAR>SET I 10<CR>
	KLINIK e.g., PAR>SET K<CR>
	MEMORY e.g., PAR>SET M E<CR> or PAR>SET M T<CR>
	The SET MEMORY command accepts two arguments: ELEVEN and TEN.
	NOT e.g., PAR>SET NO arg<CR>
	OFFSET e.g., PAR>SET O 101204<CR>

Table 4 PARSER Programmer Mode Command Summary (Cont)

Command	Description
	RELOAD e.g., PAR>SET REL<CR>
	REPEAT e.g., PAR>SET REP 5<CR>
	RETRY e.g., PAR>SET RET<CR>
	TRACKS e.g., PAR>SET T<CR>
SHUTDOWN	PAR>SH<CR>
START	PAR>ST M<CR> or PAR>ST T 3000<CR> The START command accepts two arguments: MICROCODE and TEN.
WHAT	PAR>W CL<CR> or PAR>W V<CR> etc. The WHAT command accepts the following arguments. CLOCK, CONSOLE, DATE, INCREMENT, KLINIK, MEMORY, OFFSET, PARITY-STOP, RELOAD, REPEAT, RETRY, TRACKS, VERSION
XCT	PAR>X 254200000000<CR>
ZERO	PARZ 200>277<CR>

COMMAND DESCRIPTION
This section describes in detail the commands listed in Table 2.

1 A<CR> - The ABORT command stops the KL10 by trying to force it into the HALT loop. If this fails after a reasonable number of EBox clock ticks, the command tries to START MICROCODE at CRAM address 0, which implies a master reset of the KL10 processor.

NOTE
This is the best way to get the KL10 into a known state when the previous state left it hung.

2 CO<CR> - The CONTINUE command takes the KL10 out of the HALT loop, causing it to execute the instruction pointed to by the PC. If single instruction mode was not set, the KL10 should continue running. If single instruction mode was set via the FXCT 12 function, the instruction is executed, and the KL10 is returned to the HALT loop.

3 FLAGS<CR> - The PC flag mnemonics displayed are defined as follows.

AFI	Address Failure Inhibit (bit 08)
AT0	Trap 1 (bit 10)
AT1	Trap 2 (bit 09)
BIS	First Part Done (bit 04)
CY0	Carry 0 (bit 01)
CY1	Carry 1 (bit 02)
FOV	Floating Overflow (bit 03)
FUF	Floating Underflow (bit 11)
LIP	Public (bit 07)
NDV	No Divide (bit 12)
OVF	Overflow/Previous Context Public (bit 00)
UIO	User In-Out/Previous Context User (bit 00)
USR	User (bit 05)

4 FX<CR> - The FXCT command accepts a number as a function write code, performs the function write, and displays the result. Useful values are 0 (stops the KL10 clock), and 1 (starts the KL10 clock). Random use of FXCT can cause false CRAM parity errors. (Use the HALT or ABORT commands first.)

5 H<CR> - The HALT command tries to put the KL10 into the HALT loop by clearing RUN, and waiting. If the KL10 is unable to go into the HALT loop, the HALT command tries to force it in by using BURST mode. If this does not work, an error message is displayed.

6 I<CR> - The INITIALIZE command (re)initializes PARSER, and checks the state of the KL10, sets up the KL10 state flag word with default values and restarts the KL10 based on those values. The following KL10 conditions are checked: clock running, run flip-flop set, and opcode enabled. INITIALIZE also checks to see if this PDP-11 is running on a privileged DTE20.

7 REP 2:EX T P<CR> - The REPEAT n command causes the command(s) in the remainder of the line to be repeated n (2) times if the SET REPEAT value is set to 1. See SET REPEAT. If the SET REPEAT value is greater than 1 then it is multiplied by the REPEAT n value and the commands are repeated that many times.

8 RES D<CR> - The RESET DTE-20 command resets the DTE20 by depositing a 1 in bit 6 of DIAG WORD 2 in the DTE20. Then bit 0 in DIAG WORD 1 of the DTE20 is set to 1 indicating word mode transfers.

9 SET CL E<CR> - The SET CLOCK EXTERNAL command selects the external clock source for the KL10. If no external clock source is connected, the KL10 is stuck and can only be reset by powering the system down and then up again.

10 SET T<CR> - The SET TRACKS command causes changes in the internal state of the KL10 to be displayed after each clock tick. This is done via diagnostic reads and is primarily used for debugging hardware or front-end software. This will result in a lot of wasted paper if you are not careful.

11 SH<CR> - The SHUTDOWN command deposits a -1 (minus one) into KL10 executive virtual location 30 (octal). It is used to gracefully bring down the KL10 timesharing systems. It will cause PARSER to exit if the deposit was successful, which will cause the console terminal to be connected to either EDDT (if loaded), or to the dead KL10. If EDDT is not loaded, the KL10 will execute a HALT instruction (TOPS-20 only) as soon as the clock interrupt is serviced.

12 ST M 0<CR> - The START MICROCODE command performs a KL10 master reset and starts the microcode at the microcode address specified. Starting the MICROCODE at addresses other than 0 is probably not helpful for most users.

13 ST T 30000<CR> - The START TEN command starts the KL10 at the address requested using an algorithm determined by the version of the microcode. It puts the KL10 into the HALT loop, loads the address onto the AR, and does a function CONTINUE, causing the KL10 to start at the address requested in EXEC KERNAL mode. To start the KL10 without losing the old processor mode, use the JUMP command, which will accept an address, EXECUTE a JRST (opcode 254) to that address (in EXEC Virtual Space), and continue in whatever mode the processor was in.

14 X 254200000000<CR> - The XCT command takes a 36-bit octal argument and executes it as a KL10 instruction.

NOTE

Executing an instruction with an opcode of zero may cause random results because the microcode uses op-code zero coming out of the HALT loop for START and CONTINUE.

15 SET KLINIK<CR> - The RSX-20F KLINIK link is enabled by issuing a SET KLINIK command to PARSER from the local console (CTY). PARSER will then request and validate the following parameters.

PARSER will request the KLINIK mode desired with the following prompt.

KLINIK MODE:

The acceptable response to this prompt is either USER or REMOTE.

USER indicates that the KLINIK link is to be used as a timesharing terminal line (only). See SET CONSOLE USER.

REMOTE indicates that the KLINIK link is to be used as a remote console line in either Maintenance, Operator or Programmer mode. See SET CONSOLE.

There is no default response to this prompt. If any other response is supplied, the command will abort and the local operator will receive one of the following error messages:

```
PAR [SET] NSK NO SUCH KEYWORD "XXX"
PAR [SET] ILC ILLEGAL CHARACTER "C"
```

where "XXX" and "C" are the offending keyword and character, respectively.

Next PARSER will request the KLINIK ACCESS WINDOW parameters by printing the following prompts and accepting responses in sequence.

```
ACCESS WINDOW OPEN DATE:
ACCESS WINDOW OPEN TIME:
ACCESS WINDOW CLOSE DATE:
ACCESS WINDOW CLOSE TIME:
```

The possible date formats are as follows.

```
DD-MMM-YY
DD-MMM-YYYY
DD MMM YY
DD MMM YYYY
```

DD is the decimal day, MMM is the alphabetic representation of the month, and YY or YYYY is the decimal year in which the KLINIK WINDOW is to open or close. The default response to a date prompt is a <carriage return>. This will set the Window Open Date to TODAY, and the Window Close Date to TODAY + 1. TODAY is the current date obtained from RSX-20F. See WHAT DATE.

The day specified must be within the range of 1-31. Date for months having less than 31 days will be validated. This includes a special check for February in a leap year. The month MMM is composed of the first three letters of the month to be entered. The year may be specified as either a Gregorian year, 19XX, or as a year relative to 1900, (00 through 99) where the first two digits are assumed to be the first two digits of the current century. Failure to adhere to this syntax will cause the command to abort, and one of the following error messages to be printed.

PAR [SET] DOR DAY OUT OF RANGE - If the day specified does not exist in the month specified.

PAR [SET] NSK NO SUCH KEYWORD "XXX" - If the keyword specified for the month cannot be matched.

PAR [SET] AMB AMBIGUOUS KEYWORD "XXX" - If that keyword is ambiguous. "XXX" is the offending keyword.

PAR [SET] YOR YEAR OUT OF RANGE - If the year has been improperly specified.

PAR [SET] DBT DATE BEFORE TODAY - If the entire window open or close date is prior to TODAY.

The Window Open Time and Window Close Time may be specified in either of the following formats.

```
HHMM
HH:MM
```

HHMM is a representation of the hour and minute. In both formats, HH is the hour and must be within the range of 00 to 23, and MM is the minute and must be within the range of 00 to 60. The default response is a <carriage return>. This will set the Window Open Time and the Window Close Time to NOW. NOW is the current time of day obtained from RSX-20F. See WHAT DATE.

Specifying a time which does not conform to this syntax will cause the command to abort and the following error message to be printed.

PAR [SET] TOR TIME OUT OF RANGE

Finally, when the complete specifications for both the Window Open and Window Close times and dates have been specified, the Window Open time and date will be checked to ensure that it does precede the Window Close time and date. If this is not the case, the command will abort and the following error message will be printed.

PAR [SET] KWE KLINIK WINDOW ERROR

If the KLINIK mode specified was USER, the dialogue will terminate at this point, as all necessary parameters have been input. If the specified KLINIK mode was REMOTE, two more parameters will be solicited from the operator. PARSER will first request a password with the following prompt.

PASSWORD:

The local operator must communicate this password to the remote KLINIK user in order that he be allowed access to the KLINIK link.

The password must be at least one and not more than six numeric or uppercase alphabetic characters, with no imbedded or trailing blanks. There are no default responses. The operator's response to this prompt will be echoed on the local console (CTY).

Failure to provide a password in this form will cause the command to abort and one of the following messages to be printed.

PAR [SET] NPI NULL PASSWORD ILLEGAL - If no password was specified.

PAR [SET] PTL PASSWORD TOO LONG - If more than six characters were typed.

PAR [SET] IPC ILLEGAL PASSWORD CHARACTER "C" - If a nonalphanumeric character was typed as a password character. "C" is the offending character.

PARSER will next request that the operator specify the highest PARSER console mode to be allowed while the KLINIK link is active with the following prompt.

HIGHEST CONSOLE MODE:

The acceptable responses to this prompt areas follows (See SET CONSOLE).

MAINTENANCE
OPERATOR
PROGRAMMER

While the KLINIK link is active, PARSER will not allow the remote or the local console to raise the command PARSER console mode, to a level higher than that specified in response to this prompt. There is no default response to this prompt.

Failure to provide the proper response to this prompt will cause the command to abort and the following error message to be printed:

PAR [SET] NSK NO SUCH KEYWORD "XXX"

where "XXX" is the offending keyword.

If all parameters have been properly input and validated, PARSER will return to command level after displaying the KLINIK enable parameters in the following format.

```
KLINIK [<ACTIVE> <INACTIVE> <DISABLED>]
ACCESS WINDOW OPEN: DD-MMM-YY HH:MM
ACCESS WINDOW CLOSED: DD-MM-YYY HH:MM
KLINIK MODE: [<REMOTE> <USER>]
```

ACTIVE indicates that the KLINIK user is connected to the RSX-20F KLINIK link.

INACTIVE indicates that the KLINIK parameters have been set, but access has not yet been allowed (i.e., the WINDOW is not open yet).

DISABLED indicates that no KLINIK parameters have been set.

If the KLINIK mode is REMOTE, one additional line will be displayed describing the highest PARSER console mode to be allowed.

```
CONSOLE MODE LIMIT: [<MAINTENANCE> <OPERATOR> <PROGRAMMER>]
```

Upon receipt of these parameters RSX-20F will log the SET KLINIK command and the parameters that were accepted. Further, RSX-20F will pass these parameters to the KL10 operating system (TOPS-20 or TOPS-20), to facilitate KLINIK recovery from a PDP-11 reboot.

16 CLEAR KLINIK<CR> - The RSX-20F KLINIK link is disabled via the CLEAR KLINIK command. This command does not accept arguments, it simply clears the KLINIK WINDOW. If the KLINIK link is active, the CLEAR KLINIK command will cause the following message to be printed on both the local and the remote consoles.

```
KLD KLINIK ACCESS TERMINATED BY OPERATOR
```

The current KLINIK enable parameters will be reset and passed to the KL10 operating system (TOPS-10 or TOPS-20). The KLINIK ACCESS WINDOW will close and RSX-20F will log the KLINIK mode termination on the CTY. The modem will not be hung up; however, all input from and output to the remote console will be ignored and all subsequent calls made to the KLINIK LINK will be acknowledged and rejected until such time as a new KLINIK WINDOW is set by the local operator. The rejection message will be in the following format.

```
KLR--KLINIK RING KLINIK-WINDOW CLOSED
```

This rejection message will appear on both the local and remote consoles.

17 CL RET<CR> - When the RETRY flag is set, the occurrence of a KEEP-ALIVE-CEASED error will result in the execution of the instruction in location 71. The instruction typically branches to a routine that will cause the KL10 operating system (TOPS-10 or TOPS-20) to dump memory and request a reload. If this can not be accomplished before the end of the keep-alive period (5 seconds), then RSX-20F assumes that the KL10 is incapacitated. In this case KLERR is called to take a KL10 hardware snapshot and then reload the KL10.

If the RETRY flag is clear (CLEAR RETRY command) every occurrence of a KEEP-ALIVE-CEASED error will result in a KLERR snapshot and reload of the KL10.

PARSER Error Message Summary

APE	KL APR ERROR - The PARSER encountered a CPU error (nonexistent memory, parity error, etc.
BAE	BURST ARGUMENT ERROR - This is an internal programming failure. It may require a software specialist.
CAE	KLCRAM ADDRESS ERROR - This is an internal programming failure. It may require a software specialist.
CBO	COMMAND BUFFER OVERFLOW - You typed a command line that was more than 280 characters in length. Reenter as two or more command lines.
CDI	CLEAR DATE ILLEGAL - You tried to clear the internal date while the KL10 was in primary protocol.

CES	CLOCK ERROR STOP - code ERROR STOP - The variable, code, is either CRAM, DRAM, FM, or FS-STOP. This message is displayed when the CPU encounters a fatal internal hardware error.
CFH	CAN'T FIND KL HALT LOOP - The PARSER tried to halt the KL10 but failed.
CLE	CONSOLE LIMIT EXCEEDED - You tried to set a console mode that was higher than the console mode specified in the SET KLINIK command dialogue. This is not allowed while the KLINIK link is active in remote mode.
CNR	COMMAND IS NOT REPEATABLE - You tried to repeat a command that cannot be repeated. However, the command has been executed once.
DAV	DATE ALREADY VALID - You tried to set a new internal date and the date validity flag was on.
DBT	DATE BEFORE TODAY - While in the SET KLINIK command dialogue, you tried to specify an open or close date that was prior to the current date.
DCK	DIVIDE CHECK - This is an internal programming error. It may require a software specialist.
DMF	DEPOSIT KL MEMORY FAILED - This is an internal programming failure. RSX-20F did not accept a deposit directive. It may require a software specialist.
DNP	DTE-20 IS NOT PRIVILEGED - This is a fatal error. The DTE20 mode switch is in the wrong position.
DOR	DAY OUT OF RANGE - You specified a day that does not exist in the month you entered.
DSF	DTE-20 STATUS FAILURE - A read or write to one of the DTE20 status registers failed.
DTC	DTE-20 CONFUSED - RUN AND HALT LOOP - This is a fatal error. The run and halt loop flags were set simultaneously, an impossible situation.
ECT	EBOX CLOCK TIMEOUT - While the PARSER was doing an execute function, the KL10 failed to reenter the halt loop within the allotted time.
EMF	EXAMINE KL MEMORY FAILED - This is an internal programming failure. RSX-20F did not accept an examine directive. It may require a software specialist.
EOC	END OF COMMAND REQUIRED - The command was ended with a ? and no additional arguments are required. Retype the command and press the RETURN key.
EPE	EBUS PARITY ERROR - This is a fatal error. The PARSER encountered an EBus parity error.
ESD	EBOS STOPPED - DEPOSIT - The PARSER executed a deposit directive and found that the KL10 clock was stopped.
ESE	EBOX STOPPED - EXAMINE - The PARSER executed an examine directive and found that the KL10 clock was stopped.
FRF	FUNCTION READ nnn FAILED - A diagnostic function read with function code nnn has failed. This is a fatal error.
FWF	FUNCTION WRITE nnn FAILED - A diagnostic function write with function code nn has failed. This is a fatal error.
FXF	FUNCTION XCT nn FAILED - A diagnostic function execute with function code nn has failed. This is a fatal error.

IDF	ILLEGAL DATE FORMAT - You entered a date in the wrong format. The correct format is: dd-mmm-yy where the hyphens may be replaced by spaces and the year may be entered as four digits. The day and year must be numeric and the month must be alphabetic. The month may be truncated to the point where it retains its uniqueness.
IFC	ILLEGAL FUNCTION CODE - This is either an internal programming error or the result of entering a diagnostic command with an invalid function code. The valid function codes are as follows. FREAD command takes codes 100-177; FWRITE command takes codes 40-77; FXCT command takes codes 0-37. If the message was not a result of entering a diagnostic command, contact a software specialist.
ILC	ILLEGAL CHARACTER "c" - The PARSER found an illegal character in the command line and c is the character's printing equivalent. Nonprinting characters are preceded by an up-arrow (^) and converted to their printing equivalent for output.
ILS	ILLEGAL SEPARATOR CHARACTER "s" - The PARSER found an illegal separator character in the command line and s is the illegal character. Nonprinting characters are preceded by an up-arrow and converted to their printing equivalent for output. Note that a tab is converted to one space.
IOC	ILLEGAL KL OPCODE - Either you or the PARSER tried to execute a KL instruction with an illegal op-code. If this was not the result of an XCT command, you may need a software specialist.
IPC	ILLEGAL PASSWORD CHARACTER "c" - During the SET KLINIK dialogue, you typed a password containing "c," an illegal character. You must use only numeric or uppercase alphabetic characters in the password.
IRC	ILLEGAL REPEAT COUNT - You typed a 0 or negative argument to either the REPEAT or SET REPEAT command.
ITF	ILLEGAL TIME FORMAT - You entered a time of day that was not in the proper format. The PARSER expects a numeric value of the form hh:mm or hhmm.
ITN	ILLEGAL TASK NAME - The RUN or MCR command was entered with no task name.
KCN	KL CLOCK IS OFF - The KL10 clock is off and you tried to execute a command that requires the clock to be on.
KLA	KL ADDRESS ERROR - You specified a KL10 address that was out of range (over 22 bits), negative, or not in octal radix.
KLR	ILLEGAL WHILE KL RUNNING - You tried to execute a command that is illegal while the KL10 is running.
KNC	KL IS NOT CONTINUABLE - You tried to resume processing with the CONTINUE command but the KL10 was not in a continuable state. For example, you cannot CONTINUE after a RESET command.
KWE	KLINIK WINDOW ERROR - During the SET KLINIK dialogue, you specified a window close date and time that is prior to the window open date and time.
MRA	MISSING REQUIRED ARGUMENT - You did not specify all of the necessary arguments for the command.
NDI	NULL DATE ILLEGAL - During the SET DATE dialogue, you answered the DATE: prompt with a carriage return. You must supply a date.

NER	NUMERIC EXPRESSION REQUIRED - You entered a command that expects a numeric expression as an argument and something else was entered.
NOR	INPUT NUMBER OUT OF RANGE - You specified a number that was out of range or in the wrong radix.
NPI	NULL PASSWORD ILLEGAL - During the SET KLINIK dialogue, you answered the PASSWORD: prompt with a carriage return. You must supply a password if one is requested.
NSK	NO SUCH KEYWORD "xxx" - You entered a command containing the invalid keyword xxx.
NST	NO SUCH TASK - You specified a nonexistent task in an MCR or RUN command.
NTI	NULL TIME ILLEGAL - During the SET DATE dialogue, you answered the TIME: prompt with a carriage return. You must specify a time.
OAI	ODD ADDRESS ILLEGAL - You tried to examine an odd-numbered PDP-11 address.
OFC	ODD FUNCTION CODE - This is an internal programming error. It may require a software specialist.
PTL	PASSWORD TOO LONG - During the SET KLINIK dialogue you specified a password that was more than six characters in length.
RPM	RIGHT PARENTHESIS MISSING - You omitted a right parenthesis in a numeric expression.
SCF	SET CLOCK FAILED - The PARSER cannot validate the clock enable parameters it has just set. This is a hardware error.
SKI	SET KLINIK ILLEGAL WHILE KLINIK ACTIVE - You tried to set new KLINIK parameters while the KLINIK link was active. If you want to change the parameters, you must first disconnect the KLINIK link by typing DISCONNECT or CLEAR KLINIK.
SPF	SET PARITY FAILED - The PARSER cannot validate the parity stop parameters it has just set. This is a hardware error.
SZI	START AT ZERO ILLEGAL - You tried to start the KL10 at location 0; this is illegal.
TAA	TASK ALREADY ACTIVE - You issued a RUN or MCR command for a task that was already active.
TOR	TIME OUT OF RANGE - You specified a time in which the hours were greater than 23 or the minutes were greater than 59.
UNL	KL MICROCODE NOT LOADED - The system tried to start the KL10 microcode and found that it was not loaded or was not functioning. Use disk, DECtape, floppy, or the switch register to reload the microcode and the system.
VFY	VERIFY FAILED - The PARSER cannot verify the correct execution of a DEPOSIT command. This may require a software specialist.
WRM	COMMAND NOT AVAILABLE IN THIS CONSOLE MODE - You entered a command that is not available in the current console mode. Use the SET CONSOLE command to change mode.
XTO	KL EXECUTE TIMED OUT - The KL10 failed to reenter the halt loop within the allotted time while performing a fast internal execute function.
YOR	YEAR OUT OF RANGE - You specified the year incorrectly.

Table of Contents

TOPS-10 SYSTEM PROGRAM LIBRARY
The programs in the TOPS-10 System Program Library are listed and described in Table 1.

Table 1 TOPS-10 System Program Library

Program	Description
AID	Algebraic Interpretive Dialogue. Each command occupies one line and can be executed immediately or stored as part of a routine for later execution. This interpreter requires no previous programming experience.
ALCFIL	A program used for allocating space for a new file or reallocating space for an existing file in one contiguous region on the disk.
ALGOL	ALGOrithmic Language. A scientifically oriented language that contains a complete syntax for describing computational algorithms.
BACKUP	A program used to save disk files on magnetic tape, and later to restore any or all of these files to disk. Magnetic tape is the medium used for backup storage of disk files and for transporting files between sites.
BASIC	Beginner's All-purpose Symbolic Instruction Code. A time-sharing computer programming language that is used for direct communication between terminal units and computer centers. The language was developed at Dartmouth College.
BATCON	The Batch controller. This program reads a job's control file, starts the job, and controls the job by passing commands and data to it.
BLISS	A programming language that enables users to write programs consisting only of declarations, which establish structure, and expressions, which compute values. It is specifically designed for implementing system software.
BOOTS	A bootstrap program whose main functions are to load a program into core from a SAVE file on a disk unit and/or to dump core as a SAVE file for later analysis.
CHKPNT	A program used to gather the information on the utilization of the DECsystem-10 for accounting and billing purposes.
COBDDT	The COBOL Dynamic Debugging Technique. With COBDDT the user can: 1. Change data-name contents, 2. Set breakpoints, 3. Continue the program, 4. Display the contents of a data-name, and 5. Trace paragraphs and sections.
COBOL	COmmon Business Oriented Language. A programming language used in programming data processing applications.
COMPIL	A utility program that allows the user to type a short, concise command string in order to cause a series of operations to be performed. COMPIL deciphers the command and constructs new command strings for the system program that actually processes the command. Several of the commands that invoke COMPIL are EDIT, COMPILE, CREF, and EXECUTE.
CREF	A program which produces a sequence-numbered assembly listing followed by tables showing cross references for all operand-type symbols, all user-defined operators, and/or all operation codes and pseudo-op codes.
DAEMON	A program for writing all or parts of a job's core area and associated monitor tables onto disk.
DATDMP	A program for dumping the core data base.

Table 1 TOPS-10 System Program Library (Cont)

Program	Description
DDT	The Dynamic Debugging Technique program used for on-line checkout, testing, examination, modification, and program composition of object programs.
DIRECT	A program for producing directory listings of disks and DECtapes.
DSKLST	A program which gives status and statistics of all user disk files at a given time.
DSKRAT	A damage assessment program that scans a file structure and reports any inconsistencies detected.
DTBOOT	A bootstrap program used to save and restore core images on DECtape or magnetic tape. It operates only in executive mode.
DUMP	A program that outputs selected portions of a file in one of the various formats that can be specified by the user.
EDDT	Executive DDT (Dynamic Debugging Technique). A version of DDT used for debugging programs, such as the monitor, in executive mode.
EDIT	A program used to build and edit ASCII text files.
FAILSAFE	A program used to save the contents of the disk on magnetic tape and later restore the saved contents back onto disk.
FILDDT	File DDT (Dynamic Debugging Technique). A version of DDT used for examining and changing a file on disk instead of in core memory. This program is used to examine a monitor for debugging purposes.
FILEX	A general file transfer program used to convert between various core image formats and to read and write various DECtape directory formats and standard disk files.
FORTRAN	FORmula TRANslator. A procedure-oriented programming language designed for solving scientific-type problems by expressing the procedure for their solution as arithmetic formulas. The language is widely used in many areas of engineering, mathematics, physics, chemistry, biology, psychology, industry, military, and business.
FUDGE 2	A program used to update libraries containing one or more relocatable binary modules and to manipulate modules within these libraries.
GLOB	A program used to read collections of relocatable binary modules which have been loaded together (from both library files and separate files) in order to generate an alphabetical cross-referenced list of all the global symbols encountered. When a program is composed of many modules which communicate via global symbols, it is useful to have an alphabetical list of all global symbols with the names and modules in which they are defined and referenced.
GRIPE	A program that accepts text from the user and records it in a disk file for later examination by the operations staff.
INITIA	A program for performing standard system initialization for a particular terminal. It is used to initiate specific programs, such as the spooling programs, on the designated terminal.
LINK	A program that provides automatic loading and relocation of binary programs, producing an optional storage map, and performs loading and library searching. Also, the program loads and links relocatable binary programs and generates a symbol table in core for execution under DDT.

Table 1 TOPS-10 System Program Library (Cont)

Program	Description
LINKER	A program that combines many input modules into a single module for loading purposes. Thus, it allows for independent compilations of modules. Typically, it satisfies global references and may combine control sections.
LINKING LOADER	A program that provides automatic loading, relocation, and linking of compiler- and assembler-generated object modules.
LOGIN	The system program by which the system users gain access to the computing system.
LOOKFL	A program for typing the characteristics of a single disk file, such as creation date and number of words written, on the terminal.
MONEY	A program for reading the system's time accounting file and assigning a monetary charge for each user according to the time and resources that he has used on the system.
MONGEN	The monitor generator dialogue program that enables the system programmer to define the hardware configuration of his individual installation and the set of software options that he wishes to select for his system.
OMOUNT	A program that interfaces with the operator in order to handle requests concerning removable media.
OPSER	The OPerator SERvice program that facilitates multiple job control from a single terminal by allowing the operator or user to initiate several jobs from his terminal.
PIP	The Peripheral Interchange Program which transfers data files from one standard I/O device to another and performs simple editing functions, such as sequencing, trailing blank suppression, and compressing blanks into tabs, and magnetic tape control functions.
PLEASE	A program that provides the user with two-way communication with the operator via an operator's terminal that is reserved for PLEASE commands and the user's terminal.
QMANGER	The Batch queue manager. QMANGR is called by BATCON to schedule jobs by computing and dynamically revising job priorities.
QUEUE	The system program that allows users to add, delete, list, or modify queue entries in the various system queues.
QUOLST	A program that prints the user's quotas for each file structure in his search list and the number of free blocks available in each file structure.
REACT	A program for maintaining administrative control files. It can be used to create, modify, delete or list entries in a file.
RUNOFF	A program that facilitates the preparation of typed or printed manuscripts by performing formatting, case shifting, line justification, page numbering, titling, and indexing.
SCRIPT	A program that sends predetermined sequences of characters over multiple pseudoterminals in order to simulate a load on the system for testing, measurement, and analysis.
SETSRC	A program that allows the user to list or change his search list.
SOUP	The SOftware Updating Package that consists of a set of programs for facilitating the updating of system or user source files.

Table 1 TOPS-10 System Program Library (Cont)

Program	Description
SPRINT	The Batch input stacker. SPRINT reads any sequential input stream, sets up the job's control file and data files, and enters the job into the Batch input queue.
SYSDPY	A variation of the SYSTAT program which runs on a keyboard display terminal (at up to 2400 baud). SYSDPY maintains a dynamic display of system status by periodically altering lines of the display to replace old information with the latest information.
SYSERR	SYSERR is the report generating portion of the DECsystem-10 and DECSYSTEM-20 error detection, recovery, and reporting system. As an error is detected by the monitor, various pieces of information describing pertinent hardware and software status are gathered and appended to a disk file. SYSERR is a user-mode program which lists the contents of this file at the direction of the command string.
SYSTAT	A program that outputs to the user's terminal status information on the system as a whole, on selected aspects of the system, or on a selected job or set of jobs.
TECO	A sophisticated Text Editor and Corrector program that allows simple editing requests, character string searches, complex program editing, command repetition, and text block movement. TECO editing is performed on files consisting of ASCII characters.
UMOUNT	A program for user interfacing for the handling of requests concerning removable media.

TOPS-10 COMMAND LANGUAGE
The TOPS-10 Operating System supports approximately 96 commands. The conventions used to illustrate these commands are described in Table 1. The individual commands are arranged in alphabetical order in Table 2.

Note that the complete command format has been shown for the commands. Depending on the circumstances, only part of this format may be required. Refer to the DECsystem-10 Operating System Commands manual to determine the arguments required for a particular task. In addition, the commands can be abbreviated as long as the abbreviation does not conflict with any other command abbreviation.

Many command strings allow wild-card characters to be used in place of alphanumeric characters. These characters permit more than one file or directory to be referenced by a single specification. Two such wild-card characters are available:

1. * - The asterisk is a wild card for an entire field. When positioned in the appropriate context, it means:

		Examples	
a.	any filename or extension	*.EXT	FILNAM.*
b.	any project number or programmer number (also, any subfile directory)	[*,1164]	[27,*]

   Note that *.* and [*,*] are also possible.

2. ? - The question mark is a wild card for a single character. It can be used in any field mentioned above, provided the * does not share the field. It means: any character.

   Examples:

   .EX? FI???.EX? ?ILNAM. [27,116?] [*,11??]

In addition, the directory name can be specified with the project number, the programmer number, or both numbers missing.

ERROR MESSAGES
TOPS-10 operating systems use four types of stop codes.

DEBUG - If a priority interrupt is in progress, the condition is not immediately harmful to the system or any job. The monitor types out a message on the console terminal and continues. If no priority interrupt is in progress, a DEBUG stopcode acts the same as a JOB stopcode.

JOB - If no priority interrupt is in progress, the condition jeopardizes the integrity of the current job. The monitor sends a message to both the console terminal and the user's terminal and aborts the job. If a priority interrupt is in progress, then a JOB stopcode acts like a STOP stopcode.

STOP - This condition jeopardizes the integrity of the entire system. The monitor sends a message to the console terminal, aborts all jobs, and reloads the system.

HALT - This condition is so serious that the monitor is not going to do anything that might affect stored data. The system executes a HALT instruction and waits for the operator to initiate a reload.

Table 3 lists and describes the STOP CODES associated with a TOPS-10 operating system (6.03 release).

Table 1 TOPS-10 Command Conventions

Convention	Description
adr	An octal address.
arg	A letter or word specifying the desired function of the command.
control file	The name of the control file for the Batch System.
core	Decimal number of blocks (n or nK) or pages (nP) of core.
dev:	Any physical (or logical, normally) device name (e.g., MTA:). The colon must be included.
devn:	Any phsical device name of three characters followed by a unit number of one to three numerals (e.g., DTA3:). The colon must be included.
devSnn:	Any physical device name of three characters followed by the letter S and a station number (e.g., LPTS2:). The colon must be included.
[directory]	A designation identifying a particular disk area. This designation can be in the form [proj,prog] which identifies a UFD or [proj,prog,sfd,sfd, ...] which identifies a sub-file directory path branching from a UFD. The square brackets are required.
drives	The physical drives on which a unit is to be mounted.
file.ext	Any legal filename from one to six characters followed by a dot and an extension of zero to three characters.
file structure	The name of a particular disk. This name is usually in the form DSKA, DSKB, etc.
input specifications	File specifications for the disk files to be processed.
jobn	A user's job number assigned by the system.
jobname	A name of up to six characters of the job being entered into one of the system queues.
lh	Left half of a 36-bit word.
logdev:	Any logical device name from one to six alphanumeric characters. The colon should be included.
log file	The name to be given to the log file created by the Batch system.
n or m	A number.
x	A letter.
<nnn>	A three-digit octal code indicating the protection of a file. This code can appear only on the output side of the command string and must be enclosed in angle brackets.
prog	A program name of six or fewer characters.
rh	Right half of a 36-bit word.
/S	One or more switches used to modify the command string.
↑tape id↑	A one to six character identifying name recorded on a DECtape.
text	A message to be sent to the designated user or terminal.

Table 1 TOPS-10 Command Conventions (Cont)

Convention	Description
[user number]	A numeric identification assigned to the user for the purpose of gaining access to the system. It is usually two numbers separated by a comma.
=	An equal sign used in command strings to separate the output specification (left of the equal sign) from the input specification (right of the equal sign).

Table 2 TOPS-10 Command Summary

Command	Description
ALCFIL	R ALCFIL<CR> Allocates space for a new file or reallocates space for an existing file in one contiguous region on the disk.
ASSIGN	ASSIGN dev:logdev:<CR> ASSIGN devSnn:logdev:<CR> ASSIGN devn:logdev:<CR> Allocates an I/O device to the user's job without operator intervention.
ATTACH	ATTACH jobn [user number]<CR> Detaches the current job and connects the terminal to the specified detached job.
BACKSPACE	BACKSPACE MTAn:m FILES<CR> BACKSPACE MTAn:m RECORDS<CR> Spaces a magnetic tape backward the specified number of files or records.
CCONTINUE	CCONTINUE<CR> Continues the program from the point at which it was interrupted, but leaves the terminal in monitor mode.
CLOSE	CLOSE dev:<CR> Terminates I/O currently in progress on the specified device, performs the CLOSE UUO, but does not release the device.
COMPILE	COMPILE dev:file.ext [directory]/S,...<CR> Produces relocatable binary files (.REL files) for the specified source files.
CONTINUE	CONTINUE<CR> Continues the program from the point at which it was interrupted.
COPY	COPY dev: ↑tape id↑ file.ext [directory] <nnn> = dev:file.ext [directory], file.ext [directory], ...<CR> Transfers files from one I/O device to another.
CORE	CORE core<CR> Types or modifies the amount of core assigned to the user's job.
CPUNCH	CPUNCH jobname = dev:file.ext [directory]/s, ...<CR> Places entries into the card punch output spooling queue.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
CREATE	CREATE file.ext<CR> Opens a new file on disk for creation with LINED.
CREF	CREF<CR> Lists on LPT: any cross-referenced listing files generated by a previous COMPILE, LOAD, EXECUTE, or DEBUG command.
CSTART	CSTART adr<CR> Begins execution of a program that was either loaded with a GET command or interrupted, but leaves the terminal in monitor mode.
D(eposit)	D lh rh adr<CR> Deposits information in the user's core area.
DAYTIME	DAYTIME<CR> Types the current date followed by the time of day.
DCORE	DCORE dev:file.ext [directory]<CR> Writes a core image file of the user's core area.
DDT	DDT<CR> Copies the saved program counter and starts the program at the beginning address of DDT if DDT was loaded with the program (automatic in 6.01).
DEASSIGN	DEASSIGN dev:<CR> Returns devices assigned to the user's job to the monitor's pool of available devices and clears logical names.
DEBUG	DEBUG dev:file.ext [directory]/s, ...<CR> Produces relocatable binary files (.REL files) for the specified source files, loads the .REL files along with an appropriate system debugging program, and prepares for debugging.
DELETE	DELETE dev:file.ext [directory], ...<CR> Deletes files from DECtape or disk.
DETACH	DETACH<CR> Disconnects the terminal from the current job without affecting the status of the job.
DIRECT	DIRECT dev:file.ext [directory] = dev:file.ext [directory]/s, ...<CR> Lists the directory entries for the specified arguments.
DISMOUNT	DISMOUNT dev:/s, ...<CR> Returns, via the operator, devices assigned to the user's job to the monitor's pool of available devices.
DSK	DSK jobn<CR> Types disk usage for the combined structures of the specified job.
DTCOPY	R DTCOPY<CR> Copies contents of one DECtape to another, clears the blocks on a DECtape and clears the directory, compares two DECtapes, and/or loads and writes a bootstrap loader.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
DUMP	DUMP/S ...<CR> Writes a core image file, analyzes the file written, and provides printed output.
DUMP	R DUMP<CR> Provides printable output of data files in specified forms and modes.
E(xamine)	E adr<CR> Examines the specified core location in the user's area.
EDIT	EDIT file.ext<CR> Opens the specified file already existing on disk for editing with LINED.
EOF	EOF MTAn:<CR> Writes an end-of-file mark on the specified magnetic tape.
EXECUTE	EXECUTE dev:file.ext [directory]/s, ...<CR> Produces relocatable binary files (.REL files) for the specified source files, loads the .REL files, and begins execution.
FAILSAFE	R FAILSAFE<CR> Saves and restores disk files.
FILCOM	R FILCOM Compares two versions of a file and outputs any differences.
FILE	FILE arg, ↑tape id↑, file.ext, file.ext, ...<CR> Provides remote control, via the operator, of DECtape-to-disk and disk-to-DECtape transfers.
FILEX	R FILEX<CR> Converts between various core image formats, and reads and writes various directory formats.
FINISH	FINISH dev:<CR> Terminates I/O in progress on the specified device and performs the RELEASE UUO and DEASSIGN command.
FUDGE	FUDGE<CR> Creates a library REL file by reading a temporary file generated by a previous COMPILE, LOAD, EXECUTE, or DEBUG command containing the /FUDGE switch.
FUDGE2	R FUDGE2<CR> Updates files containing relocatable binary programs, and manipulates the programs within these files.
GET	GET dev:file.ext [directory] core<CR> Loads a core image from the specified device, but does not begin execution.
GLOB	R GLOB<CR> Reads multiple binary files to produce an alphabetical cross-referenced listing of all global symbols encountered.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
GRIPE	R GRIPE<CR> Accepts text from a user and records it in a disk file for the operations staff.
HALT	HALT<CR> or ↑C Stops the job and stores the program counter in the job data area. Control C can be used at user level as well as at monitor level.
HELP	HELP dev:prog<CR> or HELP dev:*<CR> Outputs useful documentation on various system features.
INITIA	INITIA<CR> Performs standard system initialization for the terminal issuing the command.
JCONT	JCONT jobn<CR> Continues the specified job if it was in a ↑C state because of a call to the device error message routine (HNGSTP).
KJOB	KJOB logfile = file structures/s<CR> Gives up access to the system.
LABEL	LABEL DEV: ↑tape id↑<CR> Writes an identifier onto a DECtape.
LIST	LIST dev:file.ext [directory]/s, ...<CR> Lists the specified files on the line printer.
LOAD	LOAD dev:file.ext [directory]/s, ...<CR> Produces relocatable binary files (.REL files) for the specified files and loads the .REL files generated.
LOCATE	LOCATE nn<CR> Establishes, logically, the user's job at a specified station.
LOGIN	LOGIN user number/s ...<CR> Provides access to the system.
MAKE	MAKE dev:file.ext [directory]<CR> Opens a new file on disk for creation with TECO.
MOUNT	MOUNT dev:logdev:/s drives<CR> Allocates an I/O device to the user's job via the operator.
OPSER	R OPSER<CR> Provides multiple job control from a single terminal.
PJOB	PJOB<CR> Outputs the job number to which the terminal is currently attached.
PLEASE	PLEASE dev:prog! text<CR> Provides two-way communication between the user and the operator.
PLOT	PLOT jobname = dev:file.ext [directory]/s, ...<CR> Places entries into the plotter output spooling queue.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
PRESERVE	PRESERVE file.ext, file.ext, ...<CR> Renames the specified files with the standard protection inclusively ORed with 100.
PRINT	PRINT jobname = dev:file.ext [directory]/s, ...<CR> Pleaces entries into the line printer output spooling queue.
PROTECT	PROTECT file.ext<nnn>, file.ext<nnn>, ...<CR> Sets the specified files to the requested protections.
PUNCH	PUNCH jobname = dev:file.ext [directory]/s, ...<CR> Places entries into the paper tape punch output spooling queue.
QUEUE	QUEUE queue name:jobname = input specifications<CR> Enters items into the specified system queue.
QUOLST	R QUOLST<CR> Types the used, loggin-in quota, and logged-out quota for each file structure to which the user has access, followed by the number of free blocks left on that structure.
R	R file.ext core<CR> Loads a core image from the system device (SYS:) and starts it at the location specified within the file.
REASSIGN	REASSIGN dev:jobn<CR> Gives the specified device to the designated job.
REATTA	R REATTA<CR> Transfers the job from the current terminal to the specified terminal.
REENTER	REENTER<CR> Starts the program at an alternate entry point specified by the program.
RENAME	RENAME new = old, new = old, ...<CR> Changes the name and protection of one or more files on DECtape or disk.
RESOURCES	RESOURCES<CR> Outputs the names of all available devices (except for terminals and PTYs), all file structures, and all physical units not in file structures.
REWIND	REWIND dev:<CR> Rewinds a magnetic tape or DECtape.
RUN	RUN dev:file.ext [directory] core<CR> Loads a core image from the specified device and starts it at the location specified within the file.
SAVE	SAVE dev:file.ext [directory] core<CR> Writes a core image of the user's core area on the specified device.
SCHED	SCHED<CR> Outputs the schedule bits set by the last SET SCHED command.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
SEND	SEND dev:text<CR> SEND jobn text<CR> Provides a one-way interconsole line of communication.
SET BLOCKSIZE	SET BLOCKSIZE dev:nnnn<CR> Sets the default blocksize for the specified magnetic tape.

SET BREAK	SET BREAK AT adr ON arg, ...<CR>
	SET BREAK NO arg, ...<CR>
	SET BREAK NONE<CR>
	
	Sets address break in program according to specified conditions used with KI10 processors only.

SET CDR	SET CDR file<CR> Sets the filename for the next card-reader spooling intercept.

SET CPU	SET CPU CPxn<CR>
	SET CPU NO CPxn<CR>
	SET CPU ALL<CR>
	SET CPU ONLY CPxn<CR>
	
	Sets the CPU specification for the job. This command is only available on multiprocessor systems (1055, 1077) and requires certain bits be set in the privilege word.

SET DENSITY	SET DENSITY dev:nnn<CR> Sets the default density for the specified magnetic tape.

SET DSKFUL	SET DSKFUL ERROR<CR>
	SET DSKFUL PAUSE<CR>
	
	Controls the job when the user has exhausted his disk space.

SET DSKPRI	SET DSKPRI n<CR> Sets the priority for the job's disk operations (data transfers and head positionings). Requires certain bits to be set in the privilege word.
SET HPQ	SET HPQ n<CR> Sets the high priority scheduler run queue for the job. Requires certain bits to be set in the privilege word.

SET PHYSICAL	SET PHYSICAL LIMIT core<CR>
	SET PHYSICAL GUIDELINE CORE<CR>
	
	Specifies when the job will go virtual and specifies a guideline for the page fault handler if GUIDELINE is designated. Used with KI10 processors only.

SET SPOOL	SET SPOOL dev:, dev:, ...<CR>
	SET SPOOL ALL<CR>
	SET SPOOL NONE<CR>
	SET SPOOL NO dev:, dev:, ...<CR>
	
	Adds devices to or deletes devices from the list of spooled devices for this job.

SETSRC	R SETSRC<CR> Manipulates the job's search list or system's search list.
SET TIME	SET TIME n<CR> Sets the central processor time limit for the job.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description

SET TTY	SET TTY NO arg<CR>
	SET TTY arg
	
	Sets properties to be associated with the terminal.

SET VIRTUAL LIMIT	SET VIRTUAL LIMIT core<CR> Specifies the limit on the virtual memory for a job. Used with KI10 processors only.

SET WATCH	SET WATCH arg, arg, ...<CR>
	SET WATCH ALL<CR>
	SET WATCH NONE<CR>
	SET WATCH NO arg, arg, ...<CR>
	
	Sets the output of incremental job statistics.

SKIP	SKIP MTAn:m FILES<CR> SKIP MTAn:m RECORDS<CR> SKIP MTAn:EOT<CR> Moves the specified magnetic tape forward the designated number of files or records or to the logical end of tape.
SSAVE	SSAVE dev:file.ext [directory] core<CR> Writes a core image of the user's core area on the specified device. When it is loaded with a GET (or RUN) command, the high segment will be sharable.
START	START adr<CR> Begins execution of a program either previously loaded with the GET command or interrupted while running.
SUBMIT	SUBMIT jobname = control file, log file/s<CR> Places entries into the Batch input queue.
SYSTAT	SYSTAT/S<CR> Prints information about the current status of the system.
TECO	TECO dev:file.ext [directory]<CR> Opens the specified file for editing with TECO.
TIME	TIME jobn<CR> Outputs the running time for the specified job.
TPUNCH	TPUNCH jobname = dev:file.ext [directory]/s, ...<CR> Places entries into the paper tape punch output spooling queue.
TTY	TTY NO arg<CR> TTY arg<CR> Sets properties to be associated with the terminal.
TYPE	TYPE dev:file.ext [directory]/s, ...<CR> Types the specified files on the user's terminal.
UNLOAD	UNLOAD dev:<CR> Rewinds and unloads the specified magnetic tape or DECtape.
USESTAT	USESTAT<CR> or ↑T Prints information on the terminal concerning the user's job. Control T can be used at user level also.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
VERSION	VERSION<CR> Outputs the version number of a program on the terminal.
WHERE	WHERE dev:<CR> Outputs the station number of the specified device.
ZERO	ZERO dev: [directory]<CR> Clears the directory of the specified device.

Table 3 TOPS-10 STOP CODE Summary

Monitor Module	STOPCD Name	STOPCD Type	Comment
XTCSER	28B	DEBUG	DA28 is broken
FSXKON	4IF	DEBUG	RS04 is not fancy
D85INT	5WE	DEBUG	DC75 wrong PDP-11 code
D6SINT	6DD	DEBUG	11 gave too much direct data
D6SINT	6DI	DEBUG	Unexpected TO10 DONE interrupt
D6SINT	6ID	DEBUG	11 gave too much indirect data
D76INT	6MS	DEBUG	DC76 message is short
D76INT	6QF	DEBUG	DC76 queue full
D78INT	8BI	JOB	????????????
D78INT	8IN	JOB	Input chararacter count is not 0
D78INT	8NC	JOB	Not enough monitor free core
D78INT	8ON	JOB	Output chararacter count is not 0
D78INT	8PI	JOB	Positive IOWD
D60INT	8VI	DEBUG	DN60 wrong PDP-11 code
D78INT	8VI	DEBUG	Version incorrect
FILFND	AAD	DEBUG	A. T. already dormant
KISER	AAO	JOB	Access allowed off
KLSER	AAO	JOB	Access allowed off
COMMON	AD#	STOP	CPU n address parity error
FILFND	AES	JOB	Abnormal end of search list
FILIO	AHB	DEBUG	Already have buffer
ONCMOD	AHS	HALT	Already have structure
FILFND	AOC	DEBUG	Already own CB
VMSER	APF	DEBUG	Allocated page free
ONCMOD	ARl	DEBUG	ASKDEC returned CPOPJ1
DTESER	ARD	STOP	Runaway driver
KISER	ARF	STOP	Attempt to return free page
KLSER	ARF	STOP	Attempt to return free page
FILFND	ARM	DEBUG	Access rings all messed up
DTESER	BAA	STOP	Buffer already there
CORE1	BAC	DEBUG	Bit already clear
FILFND	BAD	JOB	Block already dormant
FILIO	BAO	DEBUG	Bit already one
FILIO	BAZ	DEBUG	Bit already zero
DTESER	BDN	STOP	Bad device number
TAPUUO	BFO	DEBUG	Better find one
NETSER	BFU	DEBUG	BUSY fouled up
FILIO	BIN	STOP	I/O to a negative block
FILUUO	BMR	JOB	Block missing from RIB
COMMON	BNF	HALT	BOOTS not found
FILUUO	BNR	JOB	Block not RIB
FILFND	BNT	DEBUG	Block not there
CORE1	BNZ	DEBUG	Bit not zero
CP1SER	BPS	HALT	Both processors stopped
COMCON	BRC	DEBUG	Bad return from CMPBIT

Table 3   TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
SEGCON	BSN	STOP	Bad segment number
XTCSER	BSY	DEBUG	DA28 busy
FILIO	BWA	JOB	Block went away
COMMON	C#P	DEBUG	CPU n power failed?
CP1SER	C1N	DEBUG	CPU 1 NXM
FILUUO	CAO	DEBUG	Cluster address odd
REFSTR	CAS	HALT	Could not allocate space
COMMON	CD#	STOP	CPU n cache directory parity error
FILIO	CDA	DEBUG	In core copy does not agree
MSGSER	CDD	JOB	Cannot disconnect device
CLOCK1	CFP	JOB	Cannot find PDB
ONCMOD	CGS	HALT	Cannot get STR data block
FHXKON	CIF	DEBUG	RC10 is not FANCY
REFSTR	CIO	DEBUG	CFP is odd
SCNSER	CLO	STOP	Chunk links to 0
FILFND	CME	DEBUG	CFP modulo error
VMSER	CMS	DEBUG	CORE1 must skip
SEGCON	CMU	STOP	Core messed up
SCHED1	CNA	STOP	Core not available
FILUUO	CNE	DEBUG	Cluster not even
FILUUO	CNF	DEBUG	In core copy not found
KILOCK	CRW	STOP	CA resource wrong
COMCON	CSA	DEBUG	Cannot set access allowed
FILIO	CSE	STOP	Checksum error
SEGCON	CSP	JOB	Cannot store path
NETSER	CWN	DEBUG	Core allocation went negative
FILIO	DBZ	DEBUG	DEPLPC bit zero
FILUUO	DCR	DEBUG	DELRIB CPOPJ return
FILUUO	DDS	DEBUG	DELRIB did not skip
FILUUO	DER	DEBUG	DELRIB error return
COMNET	DFU	DEBUG	Device unrecognized
FILIO	DHA	DEBUG	Do not have AU
FILIO	DHB	DEBUG	Do not have buffer
FILIO	DHD	DEBUG	Do not have DA
FILIO	DND	DEBUG	Drive not dual-ported
DTESER	DNE	STOP	Count not even
FILUUO	DNF	DEBUG	DDB not found
DTESER	DNH	STOP	Driver not hungry
DTESER	DNI	STOP	DTE not ready
FILUUO	DNR	DEBUG	DELRIB nonskip return
FILUUO	DNS		
COMCON	DPL	DEBUG	Directory page lost
COMCON	DPN	DEBUG	Directory page nonexistent
VMSER	DSS	DEBUG	DLTSP skipped
DTESER	EFI	STOP	Illegal function code
ERRCON	EPO	DEBUG	Exec PDL overflow
REFSTR	ERB	DEBUG	Error reading BAT block
ONCMOD	ERD	DEBUG	Error refreshing disk
TAPSER	ERF	STOP	ERP really fouled up
REFSTR	ERH	DEBUG	Error reading HOME.SYS
ONCMOD	ERM	DEBUG	Error reading MFD
REFSTR	ERP	HALT	Too many retrieval pointers
ONCMOD	ERS	DEBUG	Error reading SAT
FILFND	ESS	JOB	Empty system search list
ERRCON	EUE	DEBUG	Exec UUO error
REFSTR	EWB	DEBUG	Error writing block
REFSTR	EWH	DEBUG	Error writing HOME blocks
ONCMOD	EWR	DEBUG	Error while refreshing
FILUUO	FAD	DEBUG	File already dormant
VMSER	FCZ	DEBUG	Funny core bit zero
FILIO	FDP	DEBUG	Fixed head device positioned
NETSER	FFU	STOP	F fouled up
VMSER	FIP	DEBUG	Free page in use
SCNSER	FLE	STOP	Free list empty
DTESER	FNG	STOP	Illegal function code

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
KILOCK	FPF	STOP	Page on free list is not free
KISER	FPI	STOP	Free page in use
KLSER	FPI	STOP	Free page in use
KILOCK	FPN	STOP	Free page not found
REFSTR	HBE	DEBUG	Error reading HOME blocks
XTCSER	HDS	STOP	?????????
FILIO	HIF	DEBUG	Hole in file
ONCE	HNF	HALT	High segment not found
FILIO	HWU	JOB	Hard wrong unit
CLOCK1	IBI	JOB	Intercept block illegal
FILIO	IBZ	JOB	I/O to block zero
SEGCON	ICN	DEBUG	Incore count negative
ONCMOD	IDC	HALT	Impossible drum condition
KISER	IEZ	DEBUG	IOWD equals 0
KLSER	IEZ	DEBUG	IOWD equals 0
TAPSER	IFI	STOP	Illegal function at interrupt
NETSER	IFU	DEBUG	Interrupt flag unrecognized
FILIO	IIP	STOP	I/O in progress error
KISER	IME	JOB	Illegal memory reference from exec
KLSER	IME	JOB	Illegal memory reference from exec
DTESER	IPA	STOP	No post address
VMSER	IPF	DEBUG	In use page free
VMSER	IPM	DEBUG	Illegal pointer in MEMTAB
VMSER	IPN	DEBUG	HIPC page not found
FILUUO	IUN	DEBUG	Invalid unit number
UUOCON	JAC	DEBUG	Job data area clobbered
ONCMOD	JDJ	DEBUG	JFFO did not jump
SYSINI	JIT	HALT	Job in transit
CORE1	JJW	STOP	Job's JDA is wrong
FILIO	JNC	DEBUG	Job not in core
CLOCK1	JNE	STOP	JBTADR not equal to CORTAL
DPXKON	KDS	DEBUG	KONEC2 did not skip
SYSINI	KID	HALT	Controller is down
XTCSER	KNF	STOP	Control not free
D85INT	KR3	STOP	Message too large
TAPSER	KSW	DEBUG	Controller status wrong
TAPUUO	LDN	DEBUG	Tape label DDB not found
ERRCON	LN1	STOP	Line not there
QUESER	LNF	DEBUG	Lock not found
FILIO	LNP	DEBUG	Last pointer not a pointer
SCNSER	LNS	STOP	Line not set up
ERRCON	LNT	STOP	Line not there
FILUUO	LPU	JOB	Last pointer unit change
CP1SER	MAU	DEBUG	Master already unlocked
NETSER	MBE	DEBUG	Monitor buffer exists
METCON	MCM	DEBUG	MCDB is missing
FILFND	MCN	DEBUG	Mount count negative
DTESER	MDM	STOP	Master DTE missing
FILIO	MHB	DEBUG	Must have buffer
ONCE	MIW	STOP	Memory interleaving wrong
VMSER	MIZ	DEBUG	MEMTAB is zero
ERRCON	MMN	HALT	Monitor memory NXM error
ERRCON	MMP	HALT	Monitor memory parity error
KILOCK	MMR	STOP	Moving monitor page not requested
FILIO	MNA	DEBUG	Monitor buffer not available
SYSINI	MNM	STOP	Monitor in nonexistent memory
KILOCK	MPN	STOP	Monitor page not found
REFSTR	MSR	HALT	No second RIB
NETSER	MY1	STOP	Incorrect just gave some back
NETSER	MY2	DEBUG	Already checked this in FEKINT
NETSER	MY4	DEBUG	Garbage
NETSER	MY5	DEBUG	Garbage
FILUUO	NAP	JOB	Not address pointer
CLOCK1	NCA	STOP	No core assigned
ONCMOD	NDC	STOP	No DF10C code

Table 3   TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
SCNSER	NDJ	DEBUG	No DDB for job
CLOCK1	NDP	DEBUG	Not DDB pointer
CLOCK1	NDS	STOP	Null job did SAVEGET
ONCE	NED	HALT	No exec DDT
FILUUO	NER	DEBUG	No extended RIB
UUOCON	NEV	STOP	No exec virtual memory
FEDSER	NFB	STOP	No front-end device block
DTESER	NFC	STOP	No free core
RPXKON	NFD	DEBUG	No front-end drive
VMSER	NFS	DEBUG	No first slot
SYSINI	NFU	DEBUG	No first unit
DTESER	NIS	STOP	DTE in wrong state
TAPUUO	NIV	STOP	Null interrupt vector
FILIO	NMB	DEBUG	Need monitor buffer
ONCMOD	NMC	HALT	No more core
NETSER	NMF	DEBUG	No monitor buffer
REFSTR	NMU	DEBUG	No more units
FILUUO	NNF	DEBUG	NMB not found
FILUUO	NNR	JOB	No next RIB
ONCMOD	NNU	DEBUG	Not new unit
SCNSER	NOT	DEBUG	No operator terminal
SCHED1	NPC	STOP	No PDB in core
FILIO	NPD	DEBUG	No pointer in DDB
KILOCK	NPF	STOP	Next page free
KLSER	NPI	HALT	Not parity instruction
DATMAN	NPJ	DEBUG	No PDB for job
KISER	NPN	STOP	Nonexistent page not free
KLSER	NPN	STOP	Nonexistent page not free
KISER	NPP	STOP	No PI in progress
KLSER	NPP	STOP	No PI in progress
ERRCON	NPU	STOP	Null PDL underflow
VMSER	NRF	DEBUG	SWPLST not really fragmented
FILUUO	NRM	JOB	Next RIB missing
ONCMOD	NRS	DEBUG	No RIB in SAT
VMSER	NSE	DEBUG	No SWPLST entry
FILFND	NSL	JOB	No such search list
REFSTR	NSS	DEBUG	No space for SAT
FILIO	NSU	DEBUG	No such unit
SCHED1	NTE	STOP	Not processor queue error
COMNET	NTF	STOP	NT resource mixed up
FILFND	NUB	JOB	No UFB block
FILUUO	NUE	DEBUG	No UFB error
XTCSER	NUI	DEBUG	Nonexistent unit interrupt
FILUUO	NUN	DEBUG	NMB use count negative
FILUUO	NUP	DEBUG	No unit change pointer
VMSER	NUS	DEBUG	No unit for swapping
NETSER	NVP	STOP	Not a valid PCB
DTESER	NWD	STOP	No doorbell
FILIO	NXU	DEBUG	Nonexistent unit
VMSER	O1F	DEBUG	Only one fragment
D8SINT	OIP	DEBUG	Output on progress
FILUUO	ONC	DEBUG	Odd-numbered cluster
VMSER	P2L	STOP	Page too low
COMCON	PAO	STOP	Page already out
DTESER	PCI	STOP	Function code illegal
IPCSER	PCN	DEBUG	Packet count negative
NETSER	PCW	STOP	PCB count wrong
FILIO	PDA	DEBUG	Pointers with different addresses
VMSER	PEW	DEBUG	PAGTAB entry wrong
KISER	PEZ	STOP	PAGPTR=0
KLSER	PEZ	STOP	PAGPTR=0
KILOCK	PFA	STOP	Page free already
VMSER	PFC	STOP	Page on free core list
COMCON	PGL	STOP	Pages got lost
ERRCON	PIE	STOP	Priority interrupt error

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
VMSER	PIF	DEBUG	Page is free
VMSER	PIN	DEBUG	Page in working set
KISER	PIP	STOP	PI in progress
KLSER	PIP	STOP	PI in progress
VMSER	PIW	DEBUG	Page is not in working set
CLOCK1	PJ0	DEBUG	Requeue JOB 0
FILIO	PLP	DEBUG	Past last pointer
KISER	PMU	STOP	PAGTAB is messed up
KLSER	PMU	STOP	PAGTAB is messed up
FILIO	PNE	DEBUG	Pointers not equal
FILFND	PNM	DEBUG	Physical name mismatch
KILOCK	PNP	STOP	Page not present
VMSER	PNW	DEBUG	Page not in working set
CLOCK1	POP	STOP	PI on progress
SEGCON	POR	STOP	Process out of range
FILIO	PQE	DEBUG	Position queue empty
KISER	PSF	STOP	Page in segment free
KLSER	PSF	STOP	Page in segment free
KLSER	PTH	HALT	Parity trap halt
DTESER	PTL	STOP	Packet too large
KLSER	PTP	HALT	Page table parity
CORE1	PTT	DEBUG	Past top of table
SEGCON	PUF	JOB	Path UUO failed
FILUUO	PUN	DEBUG	PPB use count negative
DTESER	QEF	STOP	Queue entry full
SCNSER	QWC	DEBUG	On wrong CPU
SCHED1	RBQ	STOP	Requeuing to beginning of queue
SCNSER	RCC	STOP	Range checked chunk
FSXKON	RDP	DEBUG	RS04 does not position
SEGCON	RDS	STOP	Remap did not skip
ERRCON	REH	HALT	Recursion in error handler
TAPSER	RFU	STOP	Recovery fouled up
FILIO	RHN	DEBUG	Reread HOME block count negative
XTCSER	RIE	DEBUG	Remote interrupt error
DPXKON	RIF	DEBUG	RP10 is not fancy
D8SINT	RIP	DEBUG	Read in progress
SCHED1	RJZ	STOP	Requeue JOB zero
ONCMOD	ROU	HALT	Ran out of units
ONCMOD	RPM	DEBUG	Retrieval pointer mismatch
VMSER	RPZ	STOP	Returning page zero
ERRCON	SAC	DEBUG	Strange APR condition
CP1SER	SAU	DEBUG	Slave already unlocked
COMMON	SB#	STOP	CPU n SBus error
FILUUO	SBT	DEBUG	Should not be truncating
VMSER	SBW	DEBUG	SWPLST bits wrong
XTCSER	SCB	DEBUG	Spurious CONI bit
SEGCON	SCR	DEBUG	Segment could not be read
FILUUO	SER	JOB	SETDDO error return
FILUUO	SFI	JOB	STR free count inconsistent
FILIO	SFU	DEBUG	Swapper fouled up
VMSER	SIN	DEBUG	SWPCNT is negative
VMSER	SLF	DEBUG	SWPLST full
FILUUO	SLM	DEBUG	Search list missing
VMSER	SLZ	DEBUG	SLECNT is zero
SCHED1	SMU	DEBUG	SWPCNT messed up
SCHED1	SMU	DEBUG	Try to recover from error
KILOCK	SNF	STOP	Segment not found
SWPSER	SNI	DEBUG	Swapping not in progress
SCHED1	SOD	STOP	Space on disk
ERRCON	SOR	STOP	Segment out of range
FILUUO	SPM	JOB	Second pointer missing
CP1SER	SPS	HALT	Second processor stopped
ONCMOD	SRE	DEBUG	SAT read error
SWPSER	SRO	STOP	Space ran out
SWPSER	SSD	STOP	Swap space disappeared

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
KILOCK	SSO	STOP	Segment swapped out
SWPSER	SWN	DEBUG	SQREQ went negative
DTESER	T1E	STOP	TO11 error
XTCSER	TC0	DEBUG	?????????
XTCSER	TC1	STOP	?????????
XTCSER	TC2	DEBUG	?????????
XTCSER	TC3	DEBUG	?????????
XTCSER	TC4	DEBUG	?????????
XTCSER	TC5	DEBUG	?????????
XTCSER	TC6	DEBUG	?????????
XTCSER	TC7	STOP	?????????
FILUUO	TCI	DEBUG	Truncation check inconsistent
FILIO	TMP	DEBUG	Too many pointers
REFSTR	TMR	HALT	Too many retrieval pointers
ONCMOD	TMU	HALT	Too many units
TSKSER	TND	DEBUG	Tasks not defined
DTESER	TNI	STOP	DTE not idle
DTESER	TQP	STOP	Found queue point
DTESER	TXE	STOP	TO10 error
FILIO	UDE	DEBUG	Unit does not exist
FILUUO	UDM	JOB	UFD data missing
FILUUO	UFI	STOP	Unit free count inconsistent
D8SINT	UID	DEBUG	Unexpected input done
ONCMOD	UIF	HALT	Unit already in file STR
ERRCON	UIL	STOP	UUO at interrupt level
XTCSER	UIP	DEBUG	Not a unique interrupt
FILUUO	UNF	DEBUG	UFB not found
COMMON	UNJ	DEBUG	Illegal null job UUO
VMSER	UNL	DEBUG	UPMP not last
D8SINT	UOD	DEBUG	Unexpected output done
FILUUO	UPC	JOB	Unit change pointer clobbered
KLSER	UPF	HALT	Unexpected page fail
FILIO	UPI	DEBUG	Unit pointer illegal
TAPSER	USW	DEBUG	Unit status wrong
VMSER	WAD	DEBUG	WSBTBL and AABTBL discrepancy
DTESER	WCN	STOP	Wrong CPU number
KLSER	WPT	HALT	Wrong parity trap
SCHED1	XTH	DEBUG	XJOB too high
REFSTR	ZBC	DEBUG	Zero blocks per cluster

GENERAL INFORMATION

DDT (Dynamic Debugging Technique) is a utility program for on-line checkout, testing, and control of MACRO and FORTRAN programs. A modified version of DDT is always loaded with the 10-based 10 diagnostic routines. Many of these diagnostics use DDT for command interpretation and test dispatching (e.g., a diagnostic which uses an $G following a test identification (FRTEST$G) is actually using a DDT feature to dispatch to the starting address of the test). DDT supports many commands which are useful for controlling diagnostics during maintenance.

DDT<CR>	KLDCP and DIAMON command to start DDT
<CR><LF>	PROMPT - DDT uses a carriage return followed by a line feed to indicate it is ready to accept a command.
$G	Exit DDT - Begin execution of main (diagnostic) program.
NOTES	1. This module summarizes the most commonly used DDT commands. Refer to the Software Notebooks for a complete list of commands.
	2. Use symbolic location PATCH for building special test routines or patching the main program.

DATA AND COMMAND FORMATS

DDT has two primary data formats: symbolic and halfword.

```
     SYMBOLIC:  CAT+2/ MOVE 3,500
     HALFWORD:  CAT+2/ 200140,,500
```

Table 1 describes the data format field delimiters.

Table 2 summarizes the DDT commands.

Table 3 summarizes DDT error messages.

Table 1   DDT Field Delimiters

Delimiter	Description
space	A space delimits the op-code field.
,	A comma delimits the AC field.
( )	Parentheses delimit the index field.
@	The @ symbol indicates indirect addressing.
,,	Double commas delimit halfwords.

Table 2 DDT Command Summary

Command	Description
Special Editing Commands	
rubout	The rubout key will cause the last character typed to be deleted.
↑U	(Control U) Delete line.
↑W	(Control W) Delete last word, back to delimiter.
↑R	(Control R) Retype last line.
Arithmetic Operations	
+	117+123<CR> Addition
-	51-17<CR> Subtraction
*	15*12<CR> Multiplication
'	256'16<CR> Division
Radix	
$nR	$8R Set the base radix to n.
Address Modes	
$A	Set address mode to absolute numeric.
$R	Set address mode to relative symbolic.
Printout Modes	
$H	Set printout mode to halfword.
$S	Set printout mode to symbolic.
$T	Set printout mode to ASCII text.
6$T	Set printout mode to sixbit text.
Searching	
a<b>c$W	2000<2050>MOVE$W Search for the key word "c." Begin the search at address "a" and end the search at address "b."
Symbols	
.	A period represents the symbolic value of the position pointer.
$Q	Represents the last quantity typed
@	Represents the indirect bit
name$:	MAIN$: Opens local symbol table for use by DDT. Name equals the name specified in the MACRO-10 title statement. For most diagnostics the title is MAIN.
sym:	CAT: Insert a new symbol in the symbol table. Use the current value of the pointer.
n<sym:	2017<CAT: Insert a symbol in the symbol table. Use the value specified by n.
sym$$K	CAT$$K Delete the specified symbol from the symbol table.

Table 2 DDT Command Summary (Cont)

Command	Description
Breakpoints	
adr$B	4000$B Set a breakpoint at the specified address. Symbolic address may be used.
$P	Proceed from the breakpoint.
n$P	5$P Set the proceed counter to n and proceed from the breakpoint.
$$P	Proceed always
$B	Remove all breakpoints.
0$nB	0$2B Remove the breakpoint specified by n.
Instruction and Program Execution	
inst$X	MOVE 3,CAT+3$X Execute the specified instruction once.
$X	$X Execute the instruction pointed to. Print the operands and increment the pointer (PC).
n$X	4$X Repeat the $X command n times, printing the operands and incrementing the pointer each time.
n$$X	4$$X Repeat the $X command n times. The operands are printed for the last executive only.
$G	Start the program at the normal starting address (JOBSA).
adr$G	2050$G<CR> Start the program at the specified address.
Input Formats	
inst	MOVE AC4, CAT+3 Format for inputting a symbolic instruction.
#,,#	777000,,000777 Format for inputting half words.
#	14 Format for inputting octal digits.
#.	94. Format for inputting decimal digits.
#.#	273.5 Format for inputting a floating point number.
"/A/	"/THIS IS A MESSAGE/ Format for inputting ASCII text.
"A$	"Y$ Format for inputting one ASCII character.
$"/A/	$"/THIS IS A MESSAGE/ Format for inputting sixbit ASCII text.
$"A$	$"Y$ Format for inputting one sixbit ASCII character.

Table 2 DDT Command Summary (Cont)

Command	Description
Examine and Modify Locations	
adr/	CAT/<CR> Print contents at address and leave open for modification.
adr!	CAT!<CR> Open address for modification but do not print current contents.
adr[	MASK[<CR> Print contents of address as a numerical value. Leave open for modification.
adr]	Print symbolic contents of address. Leave open for modification.
^(BACKSPACE)	Examine address location minus one
TAB	Examine location specified by address
$<	A patch is made by opening an address, typing (ALTMODE) (ANGLE-BRACKET). This saves the current contents of the address and opens the patch area for new instructions. After the new instructions are entered, the patching is closed by typing (ALTMODE) (ANGLE-BRACKET). The original contents are then placed in the patch area followed by two jump instructions which will return to the original address +1 or +2, depending on whether the last instruction in the patch skips or not. Example: ADDRESS/contents $< PATCH/new instruction PATCH +1/new instruction $> PATCH +2/contents PATCH +3/jump 1, ADR +1 PATCH +4/jump 2, ADR +2
line feed	Typing a line feed will close the current address and cause the contents of the next sequential address to be printed. The address will be left open for modification. Up arrow will cause the contents of the last address specified minus one to be printed. The address is left open for modification.
Carriage return	Typing a carriage return will clear the currently open address. If modifications were made the new contents are inserted.
Repeating Printouts in Modes Other Than Prevailing or Temporary	
=	Typing the = symbol following a symbolic printout will cause the printout to be repeated in halfword format.
-	Typing a dash (-) following a halfword printout will cause the printout to be repeated in symbolic format.
/	Typing the / symbol will print out the location pointed to but will not change the pointer.
[	Typing the [symbol will print out the location pointed to as a numeric value.
]	Typing the] symbol will print out the location pointed to as a symbolic instruction.

Table 2 DDT Command Summary (Cont)

Command	Description
Clear Memory	
adr<adr$$Z	PATCH<PATCH+20$$Z Clear memory from address to address.

Table 3 DDT Error Messages

Error	Description
U	Indicates the user typed an undefined symbol which cannot be interpreted by DDT. Everything typed by the user since the last DDT printout is ignored.
?	Indicates an illegal DDT command has been typed or a location outside of the user's assigned memory area has been referenced.

GENERAL INFORMATION
PIP (Peripheral Interchange Program) is a utility program which is used to transfer files between standard peripheral devices. PIP can also perform editing and magtape control functions during file transfers.

R PIP <CR>	Monitor commmand to load and start PIP
*	Prompt - indicates PIP is ready to accept commands
↑C	Exit PIP - return to monitor command mode

Notes

1. This module is a summary of PIP intended for use by field engineers. Refer to the Software Notebooks for a complete description.
2. Wild characters, the asterisk (*) and question mark (?) may be used in filename and extension construction.
3. Octal constants may be used in filenames and extensions. The octal constant must be preceded by a pound sign (#) and delimited by a nonoctal digit or a character.
4. Including the "/X" switch in a command string will cause PIP to transfer each file separately (file by file) to the destination device.
5. Excluding the "/X" switch from the command string will cause PIP to combine (concatenate) the specified source files into one large file on the destination device.

COMMAND CONVENTIONS AND SWITCHES
PIP command conventions and switches are described in the following tables.

PIP Command String Format
A PIP command string consists of two fields separated by an equal sign (=) and terminated by a carriage return <CR>.

A PIP command string which is used to transfer files between I/O devices has the following format:

```
DESTINATION = SOURCE <CR>

dev:file.ext/s/s[p,pn]<nnn>↑ident↑=dev:file.ext[p,pn]<CR>
```

A PIP command string which does not transfer files (i.e., move magtape) has the following format:

```
DESTINATION = <CR>

MTA3:(MU)=<CR>
```

The equal sign delimiter and a terminator are still required in commands formatted in this manner despite the fact that only the DESTINATION portion of the command is used.

The DESTINATION portion of a PIP command describes the device and file(s) which is to receive the transferred data. This portion of a command consists of one file specification.

The SOURCE side of the command describes the device from which the transferred data is to be taken. This portion of a command may contain one or more file specifications.

PIP command strings may be of any length; both upper and lower case characters may be used. PIP commands are normally terminated and the requested operation initiated by a carriage return. However, an ALTMODE, ESC, line feed, vertical TAB, or form feed can also be used as a command terminator.

Table 1 PIP Command Conventions

Convention	Description
dev:	Either a physical or a logical device name. Refer to Table 3.
[directory]	The identifier of a specific directory (i.e., UFD or MFD) within the system. This identifier may consist of a project, programmer number pair and Sub File Directory (SFD) names.
.ext	A 1 to 3 character alphanumeric extension assigned to the name of a file either by the user or by the system.
file	A 1 to 6 character alphanumeric identification which is either to be assigned to a new file (when on the destination side of the command) or which identifies an existing file (when on the source side of the command).
↑ident↑	A 1 to 6 character name which is to be given to the contents of a DECtape reel mounted on a specified DECtape unit.
<nnn>	A 3-digit protection code which is to be assigned to either one or more destination files or to a specified User File Directory. Refer to Table 4 and Table 5 respectively.
/s	Switches which affect the transfer. All switches in a PIP command string must be preceded by a slash - e.g., /sw/sw - or enclosed in parentheses - e.g., (sw/sw). Refer to Table 6 for a summary of PIP switches.

Table 2 PIP Command String Delimiters

Delimiter	Use and Description
:	The colon delimiter follows and identifies a device name. For example, the device DTA1 is specified as DTA1: in PIP commands.
[]	Square brackets are used to enclose the user DIRECTORY numbers and SFD names (if SFDs are used). For example [40,633] or [40,633,SFD1,SFD2...SFDn] represent the manner in which DIRECTORY numbers can be written.
<>	Angle brackets must be used to enclose a protection code (e.g. <057>) which is to be assigned to either a file or a user file directory (UFD).
,	Commas are used to separate user project and programmer numbers, and file specification groups. For example: dev:[40,633]=dev:file.ext,file.ext<CR>
↑↑	A name to be assigned as an identifier to a DECtape is enclosed within a set of up-arrows (e.g. ↑MACFLS↑).
.	A period delimiter must be the first character of a filename extension. The form on an extension is (.ext).
#	A number symbol is used as a flag to indicate the presence of an octal constant in a filename or a filename extension.
!	An exclamation symbol may be used to delimit a file specification. When used, the ! symbol causes control to be returned to the monitor from PIP and the specified file (or program) to be loaded and run. This function is provided as a user convenience to eliminate the need for several control entries.

Table 2 PIP Command String Delimiters (Cont)

Delimiter	Use and Description
=	The equal sign must be used to separate the destination and source portions of a PIP command.
()	Parentheses are used to enclose magnetic tape options, PIP control switches, and one or more PIP function switches. The form of a command employing parentheses to enclose a series of switches is: dev:file.ext(sw1sw2..swn)=...<CR>

Table 3 PIP Acceptable Device Mnemonics

Mnemonic	Device
CDP	Card Punch
CDR	Card Reader
CTY	Console TTY
DTA	DECtape
DSK	Disk
DPx	Packs
FXx	Fixed-Head
DIS	Display
LPT	Line Printer
MTA	Magnetic Tape
OPR	Operator Terminal
PTP	Paper Tape Punch
PTR	Paper Tape Reader
PLT	Plotter
PTY	Pseudo-TTY
SYS	System Library
TTY	Terminal
TMP	Pseudo-device TMPCO

Table 4 File Protection Codes

Code	Permitted Operations
0	Change protection, rename, write, update, append, read, execute.
1	Rename, write, update, append, read, execute.
2	Write, update, append, read, execute.
3	Update, append, read, execute.
4	Append, read, execute.
5	Read, execute.
6	Execute only.
7	No access privileges. File may be looked up if the UFD permits.

Table 5 UFD and SFD Protection Codes

Code	Permitted Operations
0	Access not permitted.
1	The directory may be read as a file.
2	CREATEs are permitted.
3	The directory may be read as a file and CREATEs are permitted.
4	LOOKUPs are permitted.
5	The directory may be read as a file and LOOKUPs are permitted.
6	CREATEs and LOOKUPs are both permitted.
7	The directory may be read as a file and both CREATEs and LOOKUPs are permitted.

Table 6 PIP Control Switch Summary

Switch	Description
/DX	Copy all but specified files
/F	List disk or DTA directory (filenames and ext. only).
/G	Ignore I/O errors.
/H	Image binary processing (mode)
/I	Image processing (mode)
/J	Punch cards in ASCII (output device must be CDP) or convert control characters on terminal output.
/L	List directory.
/N	Delete sequence numbers.
/O	Same as /S switch, except increment is by 1.
/P	FORTRAN output conversion assumed. Convert format control character for line printer listing. /B/P FORTRAN binary.
/Q	Print (this) list of switches and meanings.
/R	Rename file.
/S	Resequence, or add sequence number to file; increment is by 10.
/T	Suppress trailing spaces only.
/U	Copy block 0 (DTA).
/V	Match and count angle brackets (<>).
/W	Convert TABs to multiple spaces.
/X	Copy specified files. (The DX switch tells PIP to copy all but specified files.)
/Y	DECtape to paper tape - If extension is: RMT - A RIM10B paper tape (with terminating transfer word) is produced; RTB - A RIM10B paper tape (with RIM loader and terminating transfer word) is produced; SAV - A RIM10B paper tape is produced (with neither RIM loader nor terminating transfer word)
/Z	Zero out directory

Table 7 PIP Magtape Switch Summary

Switch	Description
(M2)	Select 200 BPI density.
(M5)	Select 556 BPI density.
(M8)	Select 800 BPI density.
(MA)	Advance MTA one file.
(M#nA)	Advance MTA n files.
(MB)	Backspace MTA one file.
(M#nB)	Backspace MTA n files.
(MD)	Advance MTA one record.
(M#nD)	Advance MTA n records.
(ME)	Select Even Parity.
(MF)	Mark EOF.
(MP)	Backspace MTA one record.
(M#nP)	Backspace MTA n records.
(MT)	Skip to logical EOT.
(MU)	Rewind and unload MTA or DTA.
(MW)	Rewind MTA or DTA.

Examples
The following are examples of commonly used PIP command strings:

EX1 - PIPing an ASCII file from the DISK to the line printer

```
LPT:=DSK:ERROR.SYS<CR>
```

EX2 - Combines two files on disk into one file on DECtape:

```
DTA1:FILCOM.MAC=DSK:FILA.MAC,FILB.MAC<CR>
```

EX3 - Copies a paper tape

```
PTP:=PTR:<CR>
```

EX4 - Specifies that the DECtape on DTA3 be given the identifier "MYFILE" and receive a copy of each file on DTA1.

```
DTA3:↑MYFILE↑/X=DTA1:*.*<CR>
```

TOPS-10 SYSERR

GENERAL INFORMATION
SYSERR is the report-generating part of the TOPS-10 error detection, recovery, and reporting program. When the operating system detects an error, both hardware and software error information is recorded and stored in the system error file (ERROR.SYS). SYSERR is the user mode program that reads, formats, and prints the contents of the ERROR.SYS file.

LOADING AND STARTING PROCEDURE

.R SYSERR<CR>	Typed at monitor command level
FOR HELP, TYPE "/HELP"	Standard SYSERR message
*	Prompt, indicates SYSERR is ready to accept a command

SYSERR COMMAND FORMAT
SYSERR uses the following command format.

```
*dev:file.ext[p,pn] = dev:file.ext[p,pn]/s/s/.../s<CR>

(output, destination) = (input, source)
```

The user may use all or any part of the following SYSERR default command string.

```
DSK:ERROR.LST = SYS:ERROR.SYS/ALLSUM<CR>
```

For example:

*=<CR>	Uses the entire default command string.
*TTY:=<CR>	Changes the output device from the disk to terminal.
*=/DEV:RP04/DETAIL:<CR>	Produces a detailed report on all RP04s.
*TTY:=/BEGIN:-1D<CR>	Produces a summary report of the last day entries.

NOTE
The name of the output file (default = ERROR.LST) will change if a primary switch is used. The name of the output file will be the same as the name of the first primary switch specified. For example, if the /MASALL switch is used, then the name of the output file will become MASALL.LST.

SYSERR CONTROL SWITCHES
The content of the SYSERR report is controlled by the switches appended to the SYSERR command string. There are two types of control switches: primary switches and secondary switches.

Primary Switches - The primary switches determine the type of report that SYSERR will generate. Refer to Table 1. Note that a single command string may have any number of primary switches.

Secondary Switches - The secondary switches are used to limit the report to a particular device, group of devices or date and time period. Secondary switches also control the level of detail reported. Refer to Table 2.

Table 1 SYSERR Primary Switches

Switch	Description
/ALL	*=/ALL<CR> List all entries in the input file.
/allNXM	*=/ALLNXM<CR> List all entries that pertain to nonexistent memory conditions (NXM).
/allPAR	*=/ALLPAR<CR> List all entries that pertain to parity errors.

Table 1 SYSERR Primary Switches (Cont)

Switch	Description
/allPER	*=/ALLPER<CR> List all performance-related entries.
/allSUM	*=/ALLSUM<CR> List a summary of each entry in the input file. This is the default switch.
Central Processor Switches	
/cpuALL	*=CPUALL<CR> List all CPU, main memory, and system-related entries. The front-end subsystem for the KL10-based system will also be listed.
/cpuCHK	*=/CPUCHK<CR> List all continued STOPCD (stop code) entries.
/cpuNXM	*=/CPUNXM<CR> List all entries that may have been caused by a CPU-detected nonexistent memory (NXM).
/cpuPAR	*=/CPUPAR<CR> List all entries that pertain to CPU-detected parity errors.
/cpuPER	*=/CPUPER<CR> List all CPU and system-related performance entries.
/cpuRLD	*=/CPURLD<CR> List all system reload entries.
Channel Switches	
/chnALL	*=/CHNALL<CR> List all entries that pertain to the data channels (both internal and external).
/chnNXM	*=/CHNNXM<CR> List all entries that were caused by a channel-detected nonexistent memory (NXM).
/chnPAR	*=/CHNPAR<CR> List all entries that pertain to data channel-detected parity errors.
Communication Subsystem Switches	
/comALL	*=/COMALL<CR> List all entries that pertain to DL10-based communications subsystem.
Disk Subsystem Switches	
/dskALL	*=/DSKALL<CR> List all (non-Massbus) fixed and moving head disk entries.
/dskBTH	*=/DSKBTH/DEV:DPA<CR> List both the device and data channel entries for the device specified. The secondary switch /DEV: should be used with this switch.

Table 1 SYSERR Primary Switches (Cont)

Switch	Description
/dskPAR	*=/DSKPAR<CR> List all (non-Massbus) fixed and moving head disk entries that pertain to parity errors.
/dskPER	*=/DSKPER<CR> List all disk performance entries. These entries will not be recorded in ERROR.SYS unless DAEMON is assembled with the FTUSCN switch turned on.
Magtape Subsystem Switches	
/magALL	*=/MAGALL<CR> List all (non-Massbus) magtape entries.
/magPAR	*=/MAGPAR<CR> List all (non-Massbus) magtape entries that pertain to parity errors.
/magPER	*=/MAGPER<CR> List all magtape-related performance entries.
Massbus Controller Switches	
/masALL	*=/MASALL<CR> List all entries that pertain to Massbus devices.
/masBTH	*=/MASBTH/DEV:MTB4<CR> List both the Massbus and the data channel entries for the Massbus device specified. The secondary switch /DEV: should be used with this switch.
/masNXM	*=/MASNXM<CR> List all Massbus entries that may have been caused by a nonexistent memory.
/masPAR	*=/MASPAR<CR> List all Massbus device entries that pertain to parity errors.
Unit Record Equipment Switches	
/UNDALL	*=/URDALL<CR> List entries that pertain to unit record equipment on the I/O bus. Currently only the LP100 is reported.

Table 2 SYSERR Secondary Switches

Switch	Description	Cross Ref.
/BEGIN	*=/BEGIN:MM-DD-YY:HH:MM:SS<CR> or *=/BEGIN:-7D<CR> List only those entries that were recorded after the date and time specified.	1
/BRIEF	*TTY:=/BEGIN:-7D/BRIEF:72<CR> Print the sequence number, date and time, error type, and a brief summary of each entry. The number specifies the terminal page width (from 72 to 132 columns). See the /SEQUENCE switch.	
/DETAIL:	*=/DEV:DPA7/DETAIL:<CR> List all information, including Massbus controller and device register data, for each entry.	

Table 2 SYSERR Secondary Switches (Cont)

Switch	Description	Cross Ref.
/DEV:	*=/DEV:DPA<CR> or *=/DEV:MTA4<CR> or *=/DEV:RP06<CR> List only entries for the logical or physical device specifed. Devices currently accepted are: CD20 KLINIK RP02 RP06 11CPU DH11 LP20 RP03 RS04 KLCPU RD10 RP04 TU16 KLERR RM10 RP05 TU45	
/END:	*=BEGIN:MAY-6-79:00:01/END:MAY-12-79:24:00<CR> Do not list any entries recorded after the end date and time specified.	
/ID	*=/ID:SCRATC<CR> List only entries that pertain to the disk pack or magtape reel identification specified.	
/NDEV:	*/NDEV:DPB3<CR> or *=/NDEV:RP04<CR> Do not include entries in the report that pertain to the logical or physical device specified. This switch may be used with disk, magtape, and Massbus device primary commands.	
/RETRY:	*=/DEV:MTB7/RETRY:5<CR> List only those entries that have a retry count greater than the value specified. This switch only applies to disk, magtape, and Massbus entries.	
/SEQ:	*=/SEQ:17<CR> or SEQ:(10,216,-4,517) This switch is used after the /BRIEF: switch to list the entries for the sequence number(s) specified.	
/SRE:	*=/SRE:10<CR> For reporting purposes, change the soft read error threshold to the value specifed. The default is 4.	
/STR:	*=/STR:DSKB0<CR> List only entries for the structure specified.	
/SWE:	*=/SWE:3<CR> For reporting purposes, change the soft write error threshold to the value specified. The default is 7.	

SYSERR COMMAND DESCRIPTIONS (Cross Reference)

1 /BEGIN:APR-16-1979:12:30:00<CR> or BEGIN: -7D<CR> - SYSERR also recognizes relative dates in the form -nD:HH:MM:SS where -nD specifies the number of days in the past. This is particularly useful for BATCH control files. For example:

/BEGIN:-7D<CR> would list only those entries that occurred during the last week.

BEGIN:-12<CR> would list only those entries that occurred during the last twelve hours.

BEGIN:-0:30<CR> would list only those entries that occurred during the last thirty minutes.

Table of Contents

TOPS-20 SYSTEM PROGRAM LIBRARY
The programs in TOPS-20 System Program Library are listed and described in Table 1.

Table 1 TOPS-20 System Program Library

Program	Description
ACCTPR	ACCTPR translates the binary records in the old System Accounting File FACT.BIN to ASCII records which may be processed by report-generating programs written in higher level languages, e.g., COBOL. ACCTPR is documented in the TOPS-20 Operator's Guide.
ACCT20	ACCT20 generates accounting reports from the data in the System Accounting File, FACT.BIN. It serves as an example of relevant techniques for customers wishing to develop reporting programs tailored to their own installation. ACCT20 will not process the new format USAGE files produced by TOPS-20 Release 3. Refer to the program CONV20 for information on converting USAGE files to FACT file format.
ACTGEN	ACTGEN is an account generator program used to create and install an account validation data base for use by TOPS-20 in validating accounts. It is intended primarily for use by the system manager and operator. Wheel or operator capabilities must be enabled to run ACTGEN. ACTGEN is documented in the DECSYSTEM-20 System Manager's Guide.
BOOT	BOOT is used to load the TOPS-20 monitor from disk into KL10 memory. On normal system startup, BOOT is automatically loaded and started by RSX20F, and will load the TOPS-20 monitor without operator intervention. BOOT is also responsible for dumping KL10 memory after system malfunction, for later analysis. BOOT is documented in the following documents: DECSYSTEM-20 Software Installation Guide DECSYSTEM-20 Operator's Guide
CHECKD	CHECKD checks TOPS-20 disk file structure and bit table for consistency. In the process of checking the directory structure, CHECKD finds all disk space which is in use; this allows CHECKD to compute the disk pages lost. CHECKD can optionally release this lost space. CHECKD can also be used to completely rebuild the disk bit table or to scan the directory structure for a specified disk address. CHECKD may also be used to create new file structures. CHECKD is documented in the DECSYSTEM-20 Operator's Guide.
CHKPNT	CHKPNT has three major functions: 1. Compile account statistics on disk space utilization 2. Set the monitor checkpoint interval 3. Copy system-generated accounting data to the accounting file. CHKPNT is documented in the TOPS-20 Operator's Guide.
CREF	CREF takes the modified listing files produced by the language processors and produces a final, printable listing with cross reference tables appended. CREF is documented in the DECSYSTEM-20 User's Guide.
DDT	DDT is a symbolic assembly language debugger. DDT allows up to 8 breakpoints as well as symbolic patching and manipulation of various datatypes.

Table 1 TOPS-20 System Program Library (Cont)

Program	Description
DLUSER	DLUSR is a program which obtains identifying information about each directory on a system and places it in a file. The program can then use this file to create the same directories later, in the event of a system rebuild. DLUSER is documented in the DECSYSTEM-20 Operator's Guide and DECSYSTEM-20 System Manager's Guide.
DUMPER	DUMPER is a program for saving and restoring disk files using magtape. It is used by operations personnel for file system maintenance, and may be employed by users who wish to keep certain files on magtape and/or transfer them between systems.
EDIT	EDIT is a line-oriented editor which is used to create and edit text files. It resembles the TOPS-10 editor SOS in function and command structure.
FE	FE is a utility for file transfers between the TOPS-20 file system and the FILES-11 file system. It handles protocol for the FE device such that FE: can be addressed as a FILES-11 device, usually through 11 PIP. CAUTION The FE device is intended for use only in software development and updating procedures by knowledgeable people. Use without proper caution may produce unpredictable results. FE depends on the existence of the RSX-20F task T20ACP, which should reside on the -11 file system as T20ACP.TSK. Use of FE and file conversion procedures are described in the Guide To Using the FE Device, USEFE.MEM.
FILCOM	The FILCOM program compares two files and outputs the differences between them. With FILCOM you may compare both ASCII files and binary files. FILCOM compares ASCII files line by line and binary files word by word.
FORMAT	FORMAT provides the mechanism for formatting and/or verifying RP04, RP05, RP06 disk packs that are configured to RH20s. FORMAT produces a pack in the identical format to one that was created using the diagnostic, DDRPI. FORMAT runs during timesharing only, while DDRPI can FORMAT in stand-alone mode only.
GALAXY	GALAXY is the Batch and Spooling Subsystem for the DECsystem-10 and DECSYSTEM-20. GALAXY comprises all the software (excluding operating systems software) necessary to do batch processing and input and output spooling and all queue management and task scheduling required for those functions. GALAXY Release 3 consists of the following programs: Program — What It Does QUASAR — Central queue manager, task scheduler, and GALAXY system controller BATCON — Batch job processor LPTSPL — Lineprinter output spooler (unspooler) SPRINT — Card reader input stacker/spooler QUENCH — Timesharing users' interface to the GALAXY system QMANGR — Interface module for FOROTS, BASIC, etc.
LINK	LINK is the linking loader for the DECSYSTEM-20. OVRLAY is the overlay handler for the DECSYSTEM-20.

Table 1 TOPS-20 System Program Library (Cont)

Program	Description
	LINK and OVRLAY are documented in the DECSYSTEM-20 User's Guide and in the DECSYSTEM-20 LINK User's Guide.
MAIL	MAIL is a program which allows users to send messages to other users. Messages sent by MAIL are stored in the receiver's disk directory so that they may be referenced when convenient.
	MAIL depends on the programs INFO and MAILER to perform its stated tasks. Also, the program RDMAIL is used by message recipients to read messages.
	MAIL is documented in the TOPS-20 User's Guide.
MAKLIB	MAKLIB is used to update and index .REL files. MAKLIB will insert, delete or replace modules. It is also used to index FORLIB.REL and LIBOL.REL to speed up the loading process.
	MAKLIB is documented in the DECSYSTEM-20 User's Guide.
MAKRAM	MAKRAM is a program to generate LP20 translation RAM files. MAKRAM commands are described in MAKRAM.HLP.
MAKVFU	MAKVFU is a program to generate LP05 Direct Access Vertical Format files. MAKVFU commands are described in MAKVFU.HLP.
OPLEAS	OPLEAS is the program that enables the operator to talk to users running PLEASE. Requests for contact with the operators are queued; thus the user can type a request for operator action and know that the request will be received even if the operator is currently busy. OPLEAS also handles structure and tape mount requests submitted via the EXEC TMOUNT and SMOUNT commands.
	OPLEAS is documented in the TOPS-20 User's Guide.
PA1050	RA1050 is the TOPS-10 UUO simulator produced from the file PAT.MAC. It gets mapped into the address space of any program that executes a TOPS-10 UUO. Its function is to intercept all TOPS-10 UUOs and simulate them with the appropriate TOPS-20 JSYSs.
PLEASE	PLEASE provides a facility for one user at a time to talk to an operator. Requests for contact with the operator are queued; thus the user can type a request for operator action and know that the request will be received even if the operator is currently busy.
	PLEASE runs in conjunction with OPLEAS.
	PLEASE is documented in the TOPS-20 User's Guide.
PTYCON	PTYCON is a pseudoteletype (PTY) controller. It allows a user multiple job control from a single terminal. PTYCON provides the means to converse with a number of subjobs and to control the manner and times when output is received from the subjobs.
RDMAIL	RDMAIL is a program which allows a user to read the messages which have been sent to him. It always reads the messages from the file MAIL.TXT.
	RDMAIL is documented in the DECSYSTEM-20 User's Guide.
RSXFMT	RSXFMT is a utility program to convert files from TOPS-20 and/or DOS-11 file formats to RSX-11 formats.
	Use of RSXFMT and file transfer procedures are described in the Guide To Using the FE Device, USEFE.MEM. RSXFMT commands are described in RSXFMT.HLP.

Table 1 TOPS-20 System Program Library (Cont)

Program	Description
RUNOFF	RUNOFF is a text-processing program. RUNOFF will format input text, generate tables, build lists, handle page and section numbering. RUNOFF allows a user to make all sorts of changes to the text of a document and still produce a clean, well-formatted result. RUNOFF is documented in _Getting Started with Runoff_.
SETSPD	SETSPD is a privileged system program which processes the 3-CONFIG.CMD file and, in so doing, sets many initial parameters about the system such as initial line speeds, system logical names, and magtape logical to physical correspondences.
SYSERR	SYSERR is a program used to list the contents of the system error file. It is the report generating portion of the DECSYSTEM-20 Error Detection, Recovery, and Reporting package. Documentation on how to run SYSERR and descriptions of report formats may be found in the _DECSYSTEM-20 System Error Detection, Recovery, and Reporting Reference Manual_.
SYSJOB	SYSJOB is a program for controlling system background programs. It is normally started only by job 0, and it creates additional processes and jobs as necessary. An operator or other privileged job may pass commands to SYSJOB via an exec command (↑E) SPEAK to affect the status of the background programs. SYSJOB is documented in the _DECSYSTEM-20 Operator's Guide_ under the (↑E) SPEAK command.
ULIST	ULIST provides a mechanism for listing user and directory information. The listing may be directed to the printer, the user's terminal, or to a file. ULIST will provide information on user and directory groups, directory numbers, quotas, and protections, and will list user passwords if desired.
WATCH	WATCH is a system program which provides a list of various system statistics and job run times upon request. A user can thus periodically check system performance with this utility.

TOPS-20 COMMAND LANGUAGE
The TOPS-20 Operating System supports approximately 70 basic commands. These commands are described in Table 2.

Special symbols and control characters used by TOPS-20 are described in Table 1.

COMMAND FORMAT
TOPS-20 commands use the following format.

```
COMMAND$(guide word)ARG$(guide word)ARG$(...<CR>
```

The base command and each argument is delimited by an altmode (ESCAPE KEY). The command string is terminated by a carriage return <CR>.

ERROR MESSAGES
Table 3 lists and describes many of the most commonly used BUGCHKS and BUGHLTS associated with a TOPS-20 operating system. The list was taken from TOPS-20 BIG SYSTEM, TOPS-20 MONITOR 3A (2013). A complete list for any given TOPS-20 operating system may be printed by typing

```
PRINT PS:<SYSTEM>BUGSTRING.TXT<CR>
```

Table 1 TOPS-20 Symbols and Control Characters

Character	Description
↑C↑C	Two control C characters will return the terminal to monitor command level.
@	Prompt - A single @ sign indicates the monitor is at command level and ready to accept commands.
,<CR>	A command and carriage return typed following a command name causes the monitor to enter subcommand level for the command named.
@@	Prompt - A double @@ sign indicates the monitor is at a subcommand level and ready to accept subcommands only.
<CR>	A single carriage return terminates a command or subcommand.
<CR><CR>	A double carriage return terminates a subcommand and returns the monitor to command level.
?	A question mark typed at the command level or subcommand level will cause the monitor to print a list of the available commands. A question mark typed following a partially typed command will cause the monitor to print a list of all commands or subcommands which begin with the characters typed. A question mark typed following a guide word will cause the monitor to print a list of the possible arguments. A question mark printed by the monitor indicates the user has made an error in typing a command.
$ (altmode) (ESCAPE)	If there is no ambiguity in a partially typed command, pressing the ESCAPE key will cause the remaining characters and the first guide word of the command to be printed. If a partially typed command is ambiguous pressing the ESCAPE key will cause the terminal bell to ring. The ESCAPE key is also used to terminate an argument and causes the next guide word to be printed.

Table 1 TOPS-20 Symbols and Control Characters (Cont)

Character	Description
RUBOUT DELETE	The RUBOUT or DELETE key will cause the last DELETE character typed to be deleted.
↑W	Typing a control W will cause the last field typed to be deleted.
↑U	Typing a control U will cause the entire command line to be deleted.
↑R	Typing a control R will cause the current command line to be reprinted.
↑O	Typing a control O will stop the current printout.
!	The exclamation mark is used to delimit text following a command. This is useful for sending messages during a KLINIK linkup.

Table 2 TOPS-20 Command Summary

Command	Description
System Access Commands	
ATTACH	Connects your terminal to a designated job. See also: DETACH, UNATTACH
DETACH	Disconnects your terminal from the current job without affecting the job. See also: ATTACH, UNATTACH
DISABLE	Returns a privileged user to normal status. See also: ENABLE
ENABLE	Permits privileged users to access and change confidential system information. See also: DISABLE
LOGIN	Gains access to the TOPS-20 system. See also: LOGOUT
LOGOUT	Relinquishes access to the TOPS-20 system. See also: LOGIN
UNATTACH	Disconnects a terminal from a job; it does not have to be the terminal you are using. See also: ATTACH, DETACH
Information Commands	
DAYTIME	Prints the current date and time of day.
INFORMATION	Provides information about your job, files, memory, errors, system status, and many other parameters.
SYSTAT	Outputs a summary of system users and available computing resources.
Terminal Commands	
ADVISE	Sends whatever you type on your terminal as input to a job connected to another terminal. See also: BREAK, RECEIVE, REFUSE, TALK
BREAK	Clears terminal links and advising links. See also: ADVISE, RECEIVE, REFUSE, TALK
RECEIVE	Allows your terminal to receive links and advice from other users. See also: ADVISE, BREAK, REFUSE, TALK
REFUSE	Denies links and advice to your terminal. See also: ADVISE, BREAK, RECEIVE, TALK
SET	Declares certain action to be taken when errors are detected in TOPS-20 commands.

Table 2 TOPS-20 Command Summary (Cont)

Command	Description
TAKE	Accepts commands from a file, just as if you had typed its contents on your terminal.
TALK	Links two terminals so that each user can observe what the other user is doing, yet does not affect the other user's job. See also: ADVISE, BREAK, RECEIVE, REFUSE
TERMINAL	Declares the hardware type of terminal you have, and lets you inform TOPS-20 of any special characteristics of the terminal.
Device Handling Commands	
ASSIGN	Reserves a device for use by your job. See also: DEASSIGN, DEFINE
BACKSPACE	Moves a magnetic tape drive back any number of records or files. See also: REWIND, SKIP, UNLOAD
DEASSIGN	Releases a previously assigned device. See also: ASSIGN
EOF	Writes an end-of-file mark on a magnetic tape.
REWIND	Positions a magnetic tape backward to its load point. See also: BACKSPACE, SKIP, UNLOAD
SKIP	Advances a magnetic tape one or more records or files. See also: BACKSPACE, REWIND, UNLOAD
UNLOAD	Rewinds a magnetic tape until the tape is wound completely on the source reel. See also: BACKSPACE, SKIP, REWIND
File Systems Commands	
ACCESS	Grants ownership and group rights to a specified directory. See also: CONNECT, END-ACCESS
APPEND	Adds information from one or more source files to an existing disk file. See also: EDIT
CLOSE	Closes a file or files left open by a program.
CONNECT	Removes you from your current directory and connects you to a specified directory.
COPY	Duplicates a source file in a destination file.
CREATE	Starts EDIT for making a new file. See also: EDIT
DELETE	Marks the specified file(s) for eventual deletion (disk files only) or deletes the specified files (all other devices). See also: EXPUNGE, UNDELETE
DEFINE	Associates a logical name with one or more file names. See also: ASSIGN
DIRECTORY	Lists the names of files residing in the specified directory and information relating to those files. See also: FDIRECTORY, TDIRECTORY, VDIRECTORY
EDIT	Starts EDIT for changing an existing file. See also: APPEND, CREATE
EXPUNGE	Permanently removes any deleted files from the disk. See also: DELETE, UNDELETE

Table 2 TOPS-20 Command Summary (Cont)

Command	Description
END-ACCESS	Relinquishes ownership rights to a specifie directory. See also: ACCESS
FDIRECTORY	Lists all the information about a file or files. See also: DIRECTORY, TDIRECTORY, VDIRECTORY
LIST	Prints one or more files on the line printer with or without formatting. See also: PRINT, TYPE
PRINT	Lists one or more files on the line printer. See also: LIST, TYPE
QUEUE	Places an entry into or examines a specified queue, for example, the line printer output queue.
RENAME	Changes one or more descriptors of an existing file specification.
SDISMOUNT	Notifies the system that the given structure is no longer needed. See also: SMOUNT, SREMOVE
TDIRECTORY	Lists the names of all files in the order of the date and time they were last written. See also: DIRECTORY, FDIRECTORY, VDIRECTORY
SMOUNT	Requests that a structure be made available to the user. See also: SDISMOUNT, SREMOVE
TYPE	Types the specified files on your terminal. See also: PRINT, LIST
SREMOVE	Makes a structure unavailable and requests its removal. See also: SDISMOUNT, SMOUNT
UNDELETE	Restores one or more disk files marked for deletion. See also: DELETE, EXPUNGE
TMOUNT	Requests that a magnetic tape be made available to the user.
VDIRECTORY	Lists the names of all files, as well as their protection, size, and date and time they were last written. See also: DIRECTORY, FDIRECTORY, TDIRECTORY
Program Control Commands	
COMPILE	Translates a source program using the appropriate compiler. See also: DEBUG, EXECUTE, LOAD, MERGE
CONTINUE	Resumes execution of a program interrupted by a control C. See also: REENTER, START
CREF	Runs the CREF program which produces a cross-reference listing and automatically sends it to the line printer.
CSAVE	Saves the program currently in memory so that i may be used by giving a RUN command. The progra is saved in a compressed format. See also: SAVE
DDT	Merges the debugging program, DDT, with the current program and then starts DDT. See also: DEBUG, MERGE
DEBUG	Takes a source program, compiles it, loads it with DDT and starts DDT. See also: COMPILE, DDT, MERGE

Table 2 TOPS-20 Command Summary (Cont)

Command	Description
EXECUTE	Translates, loads, and begins execution of a program. See also: COMPILE, LOAD
FORK	Makes the TOPS-20 language work for a particular address space.
GET	Loads an executable program from the specified file. See also: LOAD
LOAD	Translates a program and loads it into memory. See also: EXECUTE
MERGE	Loads an executable program into memory and merges it with the current contents of memory. See also: DEBUG
POP	Stops a copy of the TOPS-20 Command Language and returns control to the previous copy of the Command Language. See also: PUSH
PUSH	Starts a new copy of the TOPS-20 Command Language. See also: POP
R	Runs a system program. See also: EXECUTE, GET, LOAD, RUN, START
REENTER	Starts the program currently in memory at an alternate entry point specified by the program. See also: CONTINUE, START
RESET	Clears the job to which your terminal is currently attached.
RUN	Loads an executable program from a file and starts it at the location specified in the program. See also: EXECUTE, GET, LOAD, START
SAVE	Copies the contents of memory into a file in executable format. If memory contains a program, you may now execute the program by giving the RUN command with the proper file specification. See also: CSAVE
START	Begins execution of a program at the location specified in the entry vector. See also: CONTINUE, EXECUTE, GET, LOAD REENTER
Batch Commands	
SUBMIT	Enters a file into the Batch waiting list. When it is your job's turn, the commands contained in the file are executed.

Table 3 TOPS-20 BUGCHKS and BUGHLTS

Name	Type	Description
ABKSKD	HLT	Address break from scheduler context
ADDONF	HLT	ADDOBJ - LLLKUP failed
APRNX1	HLT	NXM detected by APR
APRNX2	HLT	NXM detected by APR
ASAASG	CHK	DSKASA - Assigning already assigned disk address
ASGBAD	CHK	DSKASA - Assigning bad disk address
ASGBPG	CHK	INIBTB - Failed to assign bad page(s)
ASGREP	CHK	Illegal priority given to ASGRES
ASGREQ	CHK	Illegal pool number given to ASGRES
ASGSW2	HLT	SWPOMG - Cannot assign reserved drum address
ASGSWB	CHK	SWPINI - Cannot assign bad address
ASOFNF	HLT	DELFIL: ASOFN gave fail return for long file XB
ASTJFN	HLT	GETFDB: Called for JFN with output stars
BADBAK	CHK	FILIN2 - Backup copy of root directory is not good
BADBAT	CHK	BAT blocks unreadable
BADBTB	HLT	NIC - Illegal reference to bit table
BADDAC	HLT	INSACT - Null account string seen
BADDIS	CHK	TAPE: Inconsistent state code
BADIDX	CHK	IDXINI: Partially unsuccessful index table rebuild
BADREC	HLT	FILINI - Reconstruction of root directory failed
BADROT	HLT	FILIN2: Root directory is invalid
BADTAB	CHK	VERACT - Spurious hash table encountered
BADTTY	HLT	Transfer to nonexistent terminal code
BADTYP	HLT	Bad label field description
BADXT1	HLT	Index table missing and cannot be created
BADXT2	CHK	Index table missing and was created
BADXTB	HLT	FILIN2: Could not initialize index table
BKUPDF	HLT	BKUPD - Bad CST1 entry or inconsistent CST
BLKF1	CHK	BYTINA: BLKF set before calling service routine
BLKF2	CHK	BYTOUA: BLKF set before call to service routine
BLKF3	CHK	CLZDO: BLKF set before call to service routine
BLKF4	CHK	.GDSTS: BLKF set before call to device routine
BLKF5	CHK	.MTOPR: BLKF set before call to device routine
BLKF6	CHK	.SDSTS: BLKF set before call to device routine
BOOTCR	HLT	GETSWM - Not enough core for SWPMON
BOOTER	HLT	GETSWM - Error loading SWPMON
BOOTLK	HLT	GSMDSK - Failed to lock needed pages
BOOTMP	HLT	GSMDSK - Cannot map bootstrap pages
BTBCR1	HLT	FILINI - No bit table file and unable to create one
BTBCRT	HLT	FILINI - Could not initialize bit table for public structure
CDILVT	HLT	Illegal device type
CKDFRK	HLT	JOB 0 CFORK failed
CKLBLK	CHK	CKLERR: Close and abort blocked
CLZABF	CHK	CLZFFW: Service routine blocked on an abort close
CLZDIN	INF	NETCLZ - Could not send DI
CPYUF1	CHK	CACCT: Impossible failure of CPYFU1.
CRDBAK	CHK	CRDIR3: Could not make backup copy of root directory
CRDBK1	CHK	CRDIR4: Could not make backup copy of root directory
CRDNOM	CHK	CRDIR - Failed to make MAIL.TXT file
CRDOLD	CHK	CRGDGB: Old format CRDIR is illegal
CRDSDF	CHK	CRDIR1: SETDIR failed on new directory
CRSPAG	CHK	VERACT - Account data block crosses a page boundary
CST2I1	HLT	Page table core pointer and CST2 fail to correspond
CST2I2	HLT	MVPT - CST2 inconsistent
CST2I3	HLT	Page table core pointer and CST2 fail to correspond
DEABAD	CHK	DSKDEA - Deassigning bad disk address
DEAUNA	CHK	DEDSK - Deassigning unassigned disk address
DELBDD	INF	DELDIR: Bad directory deleted. Rebuild bit table
DELNDF	HLT	DELNOD - LLLKUP failed
DEQMDF	CHK	DEQUE: Internal monitor DEQ failed
DEVUCF	CHK	DEVAV - Unexpected CHKDES failure
DGUTPG	HLT	DIAG - Locked page list page locked at DIAG UNLOCK
DGZTPA	HLT	DIAG - Locked page list page was zero
DIRACT	CHK	ACTBAD: Illegal format for directory account block in directory:
DIRB2L	CHK	RLDFB2: Directory free block too large in directory:

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
DIRB2S	CHK	RLDFB1: Directory free block too small in directory:
DIRBAD	CHK	SETDI4: Smashed directory number:
DIRBAF	CHK	RLDFB5: Block already on directory free list in directory:
DIRBCB	CHK	RLDFB3: Directory free block crosses page boundary in directory:
DIRBLK	CHK	BLKSCN: Illegal block type in directory:
DIRDNL	CHK	ULKDIR - Directory not locked, directory number:
DIREXT	CHK	EXTBAD: Illegal format for directory extension block in directory:
DIRFDB	CHK	Illegal format for FDB in directory:
DIRFKP	CHK	SETDIR - Directory page 0 belongs to fork in directory:
DIRFRE	CHK	FREBAD: Illegal format for directory free block in directory:
DIRIFB	CHK	RLDFB4: Illegal block type on directory free list in directory:
DIRNAM	CHK	NAMBAD: Illegal format for directory name block in directory:
DIRPG0	CHK	DR0CHK: Illegal format for directory page 0 in directory:
DIRPG1	CHK	DRHCHK: Directory header block is bad in directory:
DIRRHB	CHK	RLDFB6: Attempting to return a header block in directory:
DIRSY1	CHK	DELDL8: Directory symbol table fouled up for directory:
DIRSY2	CHK	MDDNAM: Symbol table fouled up in directory:
DIRSY3	CHK	LOOKUP: Symbol search fouled up in directory:
DIRSY4	CHK	NAMCM4: Directory symbol table fouled up in directory:
DIRSY5	CHK	SYMBAD: Illegal format for directory symbol table in directory:
DIRSY6	CHK	RBLDST: Prematurely ran out of room in symbol table in directory:
DIRULK	CHK	ULKMD2: Attempt to unlock illegally formatted directory, directory number:
DIRUNS	CHK	UNSBAD: Illegal format for directory user name block in directory:
DLDEF	INF	Logical name define failed for front-end console terminal
DMPRLF	CHK	DMPREL - Failed to release page
DN20ST	INF	DTESRV - DN20 stopped
DRMFUL	HLT	Drum completely full
DRMIBT	HLT	DRMASN - Bit table inconsistent
DRMNFR	HLT	DRMAM - Cannot find page when DRMFRE non-0
DSKBT1	CHK	DSK bit table fouled, cannot find free page on TRK with non-0 count
DSKBT3	CHK	Disk bit table already locked at LCKBTB
DST2SM	HLT	SWPINI - DST too small
DTECAR	HLT	DTESRV - Carrier function with no line number present
DTECDM	INF	DTESRV - TO10 counts do not match
DTEDAT	CHK	TAKTOD - Illegal format for time/date
DTEDEV	HLT	LINEAL - Illegal device
DTEDIN	INF	DTESRV - TO10 in progress on doorbell
DTEDME	INF	DTESRV - Zero Q count
DTEERR	CHK	DTESRV - DTE device error
DTEIDP	HLT	DTESRV - Indirect pointer with garbage packet
DTEIFR	HLT	DTESRV - Illegal function request from 11
DTELPI	INF	DTECHK - DTE lost PI assignment
DTEMCC	HLT	DOFRGM - MCB disagrees with count
DTEODD	CHK	TAKLC - Odd byte count for line characters
DTEP2S	CHK	TO10DN - Packet too small
DTEPGF	CHK	DTE transfer page fail
DTEPNR	INF	DTESRV - Incorrect indirect setup
DTETIP	CHK	DTETDN - TO10 DONE received with no transfer in progress
DTETTY	HLT	TAKLC - Non-TTY device on function code 4
DTEUIF	HLT	DTESRV - Unimplemented function from 11

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
DVCHRX	CHK	DVCHR1 - Unexpected CHKDES failure within .DVCHR
DX2DIE	CHK	PHYX2 - DX20 halted
DX2FGS	CHK	PHYX2 - Fail to get sense bytes
DX2FUS	CHK	PHYX2 - Fail to update sense bytes
DX2IDM	CHK	PHYX2 - Illegal data mode at DONE interrupt
DX2IDX	INF	PHYX2 - Illegal retry byte pointer
DX2IEC	CHK	PHYX2 - Illegal error class code
DX2IFS	CHK	PHYX2 - Illegal function at start I/O
DX2IRF	INF	PHYX2 - Illegal function during retry
DX2MCF	CHK	PHYX2 - DX20 microcode check failure
DX2N2S	INF	PHYX2 - More TU70s than table space, excess ignored
DX2NRT	CHK	DX2ERR - IS.NRT set on successful retry
DX2NUD	CHK	PHYX2 - Channel done interrupt but no unit active
DX2NUE	CHK	PHYX2 - No active UDB and DX20 composite error set
DX2RFU	CHK	PHYX2 - Error recovery confused
DX2UNA	INF	PHYX2 - Attention interrupt and UDB not active
DX2UPE	CHK	PHYX2 - Fail to update sense bytes during initialization
EFACF1	CHK	EFACT: CLOSF failed to close FACT file
EFACF3	CHK	EFACT: Failed to write into FACT file
ENQMLF	CHK	ENQUE: Internal ENQ of a monitor lock failed
EXPAFK	HLT	EXPALL: JOB 0 CFORK failed
EXPRCD	CHK	EXPALL: RCDIR failure
FATAPE	HLT	Fatal address parity error
FATCDP	HLT	Fatal cache directory parity error
FATMPE	HLT	Fatal parity error
FEBAD	CHK	FEHSD - Wrong front end
FEBFOV	CHK	FEHSD - Buffer overflow
FEOCPB	CHK	FEFSYS - Failed to back up root directory
FEUSTS	CHK	FESSTS - Unknown status
FILBAK	CHK	FILCRD: Could not create backup of root directory
FILBOT	CHK	Could not create BOOTSTRAP.BIN file
FILBTB	HLT	Unable to write bit table file
FILCCD	CHK	Could not create directory
FILFEF	CHK	Could not create front-end file system
FILHOM	CHK	Unable to rewrite HOME blocks in WRTBTB
FILIRD	HLT	FILINW: Could not initialize the root directory
FILJB1	CHK	FILCRD: No room to create standard system directories
FILMAP	HLT	FILIN2: Could not map in root directory
FILRID	HLT	FILINW: Index table already set up for root directory
FIXBAD	CHK	Could not rewrite HOME blocks to point to front-end filesystem
FIXBDB	CHK	Could not rewrite HOME blocks to point to BOOTSTRAP.BIN
FKWSP1	CHK	LOADBS - Unreasonable FKWSP
FLKNS	CHK	FUNLK - Lock not set
FLKTIM	CHK	FLOCK - Timeout
FRKBAL	CHK	AGESET - Fork not in BALSET
FRKNDL	CHK	Fork not properly deleted
FRKNPT	HLT	FKHPTN - Fork has no page table
FRKPTE	HLT	BADCPG - Fatal error in fork PT page
FRKSLF	HLT	SUSFK - Given self as argument
GLFNF	HLT	GLREM - Fork not found
GTFDB1	CHK	DSKINS: GETFDB failure
GTFDB2	HLT	NEWLFP: GETFDB failure for open file
GTFDB3	HLT	DSKREN - GETFDB failure for open file
GTFDB6	HLT	CRDI0A: Cannot do GETFDB on root directory
HARDCE	CHK	Hard cache errors - cache deselected
HSHERR	CHK	VERACT - Hash value out of range
HSYFRK	HLT	HSYS - JOB 0 CFORK failed
IBCPYW	HLT	COPY - Write pointer in index block
IBOFNF	HLT	FILINI: ASOFN failure for root directory IB
IDFOD1	CHK	AT MENTR - INTDF overly decremented

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
IDFOD2	CHK	AT MRETN - INTDF overly decremented
IDXNOS	HLT	FILINI - Could not assign free space for IDXTAB
ILAGE	HLT	Bad age field in CST0
ILBOOT	HLT	GETSWM - Illegal value of BOOTFL
ILCHS1	HLT	PHYSIO - Illegal channel status at SIO
ILCHS2	HLT	PHYSIO - Illegal channel state at STKIO
ILCNSP	HLT	PHYSIO - Illegal call to CONSPW
ILCNST	HLT	PHYSIO - Illegal call to CONSTW
ILCST1	HLT	Illegal address in CST1 entry, cannot restart
ILDEST	HLT	Illegal destination identifier to SETMPG or SETPT
ILDRA1	CHK	DASDRM - Illegal or unassigned drum address
ILDRA2	HLT	DRMIAD - Illegal drum address
ILFPTE	HLT	ILLFPT: Illegal section number referenced
ILGDA1	HLT	GDSTX - Bad address
ILGDA2	HLT	GDSTX - Bad address
ILIBPT	CHK	Bad pointer type in index block
ILIRBL	HLT	PHYSIO - IORB link not null at ONFPWQ
ILJRFN	CHK	JFKRFH - Bad JRFN, ignored
ILLDMS	CHK	BADDMS: Illegal DMS JSYS from monitor context
ILLIND	HLT	Illegal indirect
ILLSTR	INF	NSPTSK - Illegal initialization message
ILLTAB	CHK	TABLK2: Table not in proper format
ILLUUO	CHK	KIBADU: Illegal UUO from monitor context
ILMADR	HLT	Illegal address reference in monitor
ILOFN1	HLT	MSCANP - Illegal identification
ILOKSK	HLT	OKSKED when not NOSKED
ILPAG1	HLT	SWPOT0 - Invalid page
ILPAGN	HLT	MRKMPG - Invalid page number
ILPDAR	HLT	PHYSIO - Illegal disk address in PAGEM request
ILPID1	CHK	CREPID: Attempt to create illegal PID
ILPID2	CHK	DELPID: Validated PID turned illegal
ILPLK1	HLT	MLKPG - Illegal arguments
ILPPT1	HLT	UPDOFN - Bad pointer in page table
ILPPT2	HLT	UPDPGS - Bad pointer in page table
ILPPT3	HLT	Bad pointer in page table
ILPSEC	CHK	Illegal section number
ILPTN1	HLT	MRPACS - Illegal PTN
ILRBLT	HLT	PHYSIO - IORB link not null at ONF/STWQ
ILRFPD	HLT	PDL - OV in illegal page reference
ILSPTH	HLT	SETPT - SPTH inconsistent with XB
ILSPTI	HLT	Illegal SPT index given to SETMXB
ILSRC	HLT	Illegal source identifier given to SETPT
ILSTP3	HLT	VERLUK: Impossible skip return from EXTLUU
ILSWPA	HLT	SWPIN - Illegal swap address
ILTWQ	HLT	PHYINT - TWQ or PWQ incorrect
ILTWQP	HLT	PHYSIO - PWQ or TWQ tail pointer incorrect
ILULK1	HLT	MULKPG - Tried to unlock page not locked
ILULK2	HLT	Tried to unlock page not locked
ILULK3	HLT	MULKMP - Illegal monitor address
ILULK4	HLT	MULKCR - Illegal core page number
ILUST1	HLT	PHYSIO - Unit status inconsistent at SIO
ILUST2	CHK	PHYSIO - Unit status inconsistent at SPS
ILUST3	HLT	PHYSIO - SCHSEK - Impossible unit status
ILUST4	HLT	PHYSIO - Controller active at SPS
ILUST5	HLT	PHYSIO - Illegal unit or channel state at STKIO
ILWRT2	HLT	Attempted write reference to protected monitor
ILXBP	HLT	SETPT - Bad pointer in XB
IMPUUO	HLT	Impossible MUUO
INDCNT	INF	DTESRV - Bad indirect count
INVDTE	HLT	DTEQ - Invalid DTE specified
IOPGF	HLT	I/O page fail
IPCFKH	CHK	CHKPDD: Could not find local fork handle
IPCFRK	CHK	PIDINB: Cannot create forks for IPCF
IPCJB0	CHK	PIDINI: Not in context of JOB 0
IPCMCN	CHK	MESREC: Message count went negative

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
IPCOVL	HLT	PIDINI: PIDS and free pool overlap, IPCF will not work!
IPCSOD	CHK	GETMES: Sender's count overly decremented
JONRUN	HLT	JOB 0 not run for too long, probable swapping hangup
JSBNIC	HLT	SETPPG - JSB not in core
JTENQE	HLT	JTENQ with bad NSKED
KLIOVF	CHK	DTESRV - KLINIK data base too large
KPALVH	HLT	Keep alive ceased
LCKDIR	HLT	Attempt to lock directory twice for same fork
LNGDIR	CHK	Long directory file in directory:
LNMILI	CHK	LNMLUK: Illegal value of logical name table index
LUUMN0	HLT	LUUO in monitor context
LUUMON	HLT	.LBCHK: Illegal LUUO from monitor context
MAP41F	HLT	MAPF41 failed to skip
MAPBT1	HLT	OFN for bit table is zero
MDDJFN	HLT	GETFDB: Called for non-MDD device
MNTLNG	HLT	MNTBTB - Bit table is a long file
MONPDL	HLT	Overflow or PDL overflow trap in monitor
MPEUTP	HLT	PFCDPE - Unknown trap on test reference
MPIDXO	CHK	MAPIDX - No OFN for index table file
MTANOA	CHK	IRBDN2: IRBDON called for an active IORB
MTANOI	CHK	GETUBF: No queued IORBs for input
MTANOQ	CHK	IRBDN1: IRBDON called for non-queued up IORB
MTAORN	CHK	MTDIR0: Magtape IORB overrun
MTARIN	HLT	MTAINT: Interrupt received for nonactive IORB
MTFCNX	HLT	MTLFCN: Function code too large
NEWBAK	HLT	FILRFS - NEWIB failure for backup root directory
NEWROT	HLT	FILRFS - NEWIB failure for root directory
NOACB	HLT	MENTR - No more AC blocks
NOADXB	HLT	RELOFN - No disk address for XB
NOALCM	CHK	ALCMES: Cannot send message to allocator
NOBAT1	CHK	Failed to write primary BAT block
NOBAT2	CHK	Failed to write secondary BAT block
NOBTB	CHK	FILINI - Unable to open bit table file
NOBTBN	HLT	FILINI - Unable to get size of BOOTSTRAP.BIN file
NOCTY	HLT	Unable to allocate data for console terminal
NODIR1	CHK	SPLMES: DIRST failed on existing directory name
NOFEFS	HLT	FILINI - Unable to get size of front-end file system
NOFNDU	HLT	FNDUNT - Cannot find device for JFN
NOFRSP	CHK	TTSPST - Could not get a free block
NOINTR	CHK	ITRAP and previous context was NOINT
NOIORB	HLT	SETIRB - Missing IORB
NOLEN	HLT	UPDLEN: No length information for OFN
NOMHDR	CHK	Illegal message with no header
NOPGT0	HLT	OPNLNG: No page table 0 in long file
NOPID	CHK	PIDKFL: PID disappeared
NORSXF	HLT	Failed to get space for master DTE
NOSEB2	HLT	PGMPE - No SYSERR buffer available
NOSERF	CHK	Cannot GTJFN error report file
NOSKTR	CHK	ITRAP from NOSKED context
NOSLNM	CHK	SLNINI: Cannot create system logical name
NOSPLM	CHK	RELJFN: Could not send spool message to QUASAR
NOTOFN	HLT	UPDOFO - Argument not OFN
NOUTF1	CHK	SPLOPN: NOUT of directory number failed
NOUTF2	CHK	SPLMES: NOUT of generation number failed
NPWQPD	CHK	PHYSIO - Null PWQ at position done
NRFTCL	CHK	PHYSIO - No requests found for cylinder seeked
NSKDIS	HLT	Dismiss while NOSKED or with non-resident test address
NSKDT2	CHK	PGRTRP - Bad INTDF
NSPFRK	HLT	NSPINI - CFORK failed
NSPRTH	CHK	NSPTSK - Invalid routing header
NULQTA	HLT	QCHK - No quota information setup
NWJTBE	CHK	No free JTB blocks
OFFSPE	HLT	OFFSPQ - Page not on SPMQ
OPOPAC	HLT	MRETN - Tried to over-pop AC stack
OVFLOW	HLT	ASOFN - Allocation table overflow

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
OVRDTA	INF	PHYSIO - Overdue transfer aborted
P2RAE1	CHK	PHYH2 - RH20 register access error reading register
P2RAE2	CHK	PHYH2 - Register access error writing register
P2RAE3	CHK	PHYH2 - Register access error on DONE or ATN interrupt
PAGLCK	HLT	DESPT - Page locked
PAGNIC	HLT	GETCPP - Page not in core
PGNDEL	HLT	REMFPB - Page not completely deleted
PH2DNA	INF	PHYH2 - Done interrupt and channel not active
PH2IHM	CHK	PHYH2 - Illegal HDW mode - word mode assumed
PH2PIM	CHK	PHYH2 - RH20 lost PI assignment
PH2WUI	HLT	Wrong unit interrupted
PHYCH1	HLT	PHYSIO - Home block check IORB already on TWQ
PHYCH2	INF	PHYSIO - Home block check IORB timed out
PHYCH3	INF	PHYSIO - Home block check IORB timed out but was not on TWQ
PHYICA	HLT	PHYINI - Illegal argument to core allocation
PHYICE	INF	PHYINI - Failed to assign resident STG
PHYLTF	HLT	PHYSIO - SCHLTM - Unexpected LATOPT failure
PHYNIR	CHK	PHYSIO - Null interrupt routine at operation done
PHYP0E	HLT	PHYALZ - Page 0 storage exhausted
PI1ERR	CHK	Unexpected unvectored interrupt on channel 1
PI2ERR	CHK	Unexpected unvectored interrupt on channel 2
PI4ERR	CHK	Unexpected unvectored interrupt on channel 4
PI5ERR	CHK	Unexpected unvectored interrupt on channel 5
PI6ERR	CHK	Unexpected unvectored interrupt on channel 6
PIDFLF	CHK	/CREPID: Free PID list fouled up
PIDOD1	CHK	MUTCHO: PID count overly decremented
PIDOD2	CHK	DELPID: Overly decremented PID count
PIITRP	HLT	Instruction trap while PI in progress or in scheduler
PISKED	HLT	Entered scheduler with PI in progress
PITRAP	HLT	Pager trap while PI in progress
PM2SIO	CHK	PHYM2 - Illegal function at start I/O
PRONX2	HLT	NXM detected by processor
PSBNIC	HLT	SETPPG - PSB not in core
PSINSK	CHK	PSI from NOSKED context
PSISTK	HLT	PSI storage stack overflow
PTAIC	HLT	SWPIN - PT page already in core
PTDEL	HLT	DESPT - PT not deleted
PTMPE	HLT	Page table parity error
PTNIC1	HLT	SWPIN - Page table not in core
PTNON0	HLT	SETPT0 - Previous contents non-0
PTOVRN	HLT	UPDPGS - Count too large
PVTRP	HLT	Proprietary violation trap
PWRFL	HLT	Fatal power failure
PWRRES	CHK	Power restart
PYILUN	HLT	PHYSIO - Illegal unit number
RELBAD	CHK	RELFRE - Bad block being released
RELRNG	CHK	RELFRE: Block out of range
RESBAD	CHK	RELRES: Illegal address passed to RELRES
RESBAZ	CHK	RELRES: Free block returned more than once
RESBND	CHK	RELRES: Releasing space beyond end of resident free pool
RFILPF	CHK	Refill error page fail
RH2ICF	HLT	PHYRH2 - Invalid channel function
RP4FEX	HLT	PHYP4 - Illegal function
RP4IF2	HLT	PHYP4 - Illegal function at STKIO
RP4IFC	HLT	PHYP4 - Illegal function at CNV
RP4ILF	HLT	PHYP4 - Illegal function on interrupt
RP4LTF	HLT	PHYP4 - Failed to find TWQ entry at RP4LTM
RP4PNF	HLT	PHYP4 - Disk physical parameters not found
RP4SSC	CHK	PHYP4 - Stuck sector counter
RP4UNF	HLT	PHYP4 - Unit type not found:

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
RPGERR	HLT	BADCPG - Fatal error in resident page
RSMFAI	HLT	RESSMM - Failed to assign swap MON page
SBSERF	INF	SBSERR - Could not get error block
SEBISS	CHK	SEBCPY - Insufficient string storage in block
SEBUDT	CHK	SEBCPY - Unknown data type
SECEX1	HLT	SETMPG - Attempt to map nonexistent section
SECG37	HLT	ILSCN - Section number greater than 37
SECGT1	HLT	PGRT3 - Section number greater than MAXSEC
SECNX	HLT	Creating page table for non-0 section
SERFOF	CHK	Cannot OPENF error report file
SERFRK	HLT	SERINI - Cannot create SYSERR fork
SERGOF	CHK	SETOFI - Cannot GTJFN/open SYSERR file
SHRNO0	HLT	DESPT - Share count non-0
SHROFD	HLT	DWNSHR - OFN share count underflow
SHROFN	HLT	UPSHR - OFN share count overflow
SKDCL1	HLT	Call to scheduler when already in scheduler
SKDCL2	HLT	Call to scheduler when already in scheduler
SKDMPE	HLT	MPE in scheduler or PI context
SKDPF1	HLT	Page fail in scheduler context
SKDTRP	HLT	Instruction trap while in scheduler
SNPIC	CHK	SNPFN3: Instruction being replaced has changed
SNPLKF	CHK	SNPFN0: Cannot lock down page into monitor
SNPODB	CHK	SNPF4C: Count of inserted breakpoints overly decremented
SNPUNL	CHK	SNPF5A: Cannot unlock SNOOP page
SPTFL1	HLT	SPT completely full
SPTFL2	HLT	SPT completely full
SPTPIC	HLT	SWPIN - SPT page already in core
SPTSHR	HLT	UPSHR - SPT share count overflow
SPWRFL	CHK	Spurious power fail indication
SRQOVF	CHK	SCDRQ - Scheduler request queue overflow
STKOVF	HLT	Monitor stack overflow
STRBAD	HLT	ASOFN - Illegal structure number
STZERO	HLT	FILINI: STRTAB entry for PS is 0
SUMNR1	CHK	AJBALS - SUMNR incorrect
SUMNR2	CHK	SUMNR incorrect
SWPASF	CHK	CHKBAT - Failed to assign bad swapping address
SWPFPE	CHK	Swap error in sensitive file page
SWPIBE	CHK	Swap error in index block
SWPJSB	CHK	Swap error in JSB page
SWPMNE	HLT	Swap error in swappable monitor
SWPPSB	HLT	Swap error in PSB page
SWPPT	HLT	Swap error in unknown PT
SWPPTP	HLT	Swap error in unknown PT page
SWPUPT	HLT	Swap error in UPT, or PSB
SYSERF	CHK	LOGSST - No SYSERR storage for restart entry
TM2CCI	CHK	PHYM2 - TM02 SSC or SLA will not clear
TM2HER	CHK	TM2ERR - IS.HER set on successful retry
TM2IDM	CHK	PHYM2 - Illegal data mode at done interrupt
TM2IDX	INF	PHYM2 - Illegal retry byte pointer
TM2IF2	CHK	PHYM2 - Illegal function on command done
TM2IRF	INF	PHYM2 - Illegal function during retry
TM2N2S	INF	PHYM2 - More drives than table space, excess ignored
TM2NUD	CHK	PHYM2 - Channel done interrupt but no unit active
TM2RFU	CHK	PHYM2 - Error recovery confused
TM2UNA	INF	PHYM2 - Done interrupt and UDB not active
TRPSIE	CHK	No monitor for trapped fork
TTBAD1	HLT	Bad device designator for terminal at ATACH2
TTDAS1	HLT	HLTJB: Unable to deassign controlling terminal
TTICN0	HLT	TCI - No buffer pointer but count non-0
TTILEC	CHK	TTSND - Unrecognized escape code
TTNAC1	CHK	Line not active at PTYOPN
TTNAC3	HLT	CTY not active at FSIPBO
TTNAC4	HLT	CTY not active at FSIPBI
TTNAC5	HLT	CTY not active at FSIINI
TTNAC7	CHK	Deallocating inactive line

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
TTNAC8	HLT	Cannot assign terminal at DEVINI
TTOCN0	HLT	TTSTO - No buffer but count non-0
TTONOB	HLT	TTY OUTPUT - No buffer but count non-0
TTYBBO	CHK	TTYSRV - Big buffer overflow
TTYNTB	CHK	Ran out of TTY buffers
TWQNUL	HLT	PHYSIO - PWQ or TWQ was null at a seek or transfer completion
UBANXM	HLT	I/O NXM from Unibus device
UIONIR	HLT	UDSKIO - No IORB for NOSKED fork
ULKBAD	CHK	Unlocking TTY when count is 0
ULKSTZ	CHK	Overly decremented structure lock
UNBFNF	CHK	UNBLK1 - Fork not found
UNPGF1	HLT	MEMPAR - Parity error during memory scan
UNPGF2	HLT	Unknown page failure type
UNPIRX	CHK	UNPIR - No PSI in progress
UNTRAP	HLT	Unknown trap instruction
UNXMPE	HLT	PFCDPE - Unexpected parity error trap
USGHOL	INF	Lost page(s) in usage file
UXXCKP	HLT	Could not create checkpoint file
UXXCL1	CHK	Unable to create new usage file
UXXCL2	CHK	Unable to open new usage file
UXXCL3	CHK	Unable to close usage file
UXXCRE	HLT	Cannot create usage file
UXXFAI	CHK	Usage JSYS failure
UXXFIT	INF	Checkpoint file not in correct format for this system, rebuilding.
UXXILL	HLT	USGMES: Illegal function code
UXXMAP	HLT	USGMAP: Call to JFNOFN failed
UXXOPN	HLT	Unable to open usage file
UXXWER	CHK	Write error in usage file
WRTBT4	CHK	ASOFN on bit table file failed
WRTCPB	CHK	WRTBTB - Failed to back up root directory
WRTLNG	HLT	WRTBTB - Bit table is a long file
WSPNEG	CHK	SOSWSP - WSP negative
XBWERR	CHK	UPDOFN - Disk write error on XB
XSCORE	HLT	CST too small for physical core present

SYSERR COMMAND DESCRIPTIONS
This section describes in detail the commands and switches summarized in Table 2.

1 /BEGIN:APR-16-1979:12:30:00<CR> or BEGIN: -7D<CR> - SYSERR also recognizes relative dates in the form -nD:HH:MM:SS where -nD specifies the number of days in the past. This is particularly useful for BATCH control files. For example:

/BEGIN:-7D<CR> would list only those entries that occurred during the last week.

/BEGIN:-12<CR> would list only those entries that occurred during the last twelve hours.

/BEGIN:-0:30<CR> would list only those entries that occurred during the last thirty minutes.

GENERAL INFORMATION

SYSERR is the report-generating part of the TOPS-20 error detection, recovery, and reporting program. When the operating system detects an error, both hardware and software error information is recorded and stored in the system error file (ERROR.SYS). SYSERR is the user mode program that reads, formats, and prints the contents of the ERROR.SYS file.

LOADING AND STARTING PROCEDURE

.@ SYSERR<CR>	Typed at monitor command level
FOR HELP, TYPE "/HELP"	Standard SYSERR message
*	Prompt, indicates SYSERR is ready to accept a command

SYSERR COMMAND FORMAT

SYSERR uses the following command format.

```
*dev:<DIRECTORY>file.ext = dev:<DIRECTORY>file.ext/s/s/.../s<CR>

    (output, destination) = (input, source)
```

The user may use all or any part of the following SYSERR default command string.

```
DSK:ERROR.LST = PS:<SYSTEM>ERROR.SYS/ALLSUM<CR>
```

For example:

*=<CR>	Uses the entire default command string.
*TTY:=<CR>	Changes the output device from the disk to terminal.
*=/DEV:RP04/DETAIL:<CR>	Produces a detailed report on all RP04s.
*TTY:=/BEGIN:-1D<CR>	Produces a summary report of the last day entries.

NOTE

The name of the output file (default = ERROR.LST) will change if a primary switch is used. The name of the output file will be the same as the name of the first primary switch specified. For example, if the /MASALL switch is used, then the name of the output file will become MASALL.LST.

SYSERR CONTROL SWITCHES

The content of the SYSERR report is controlled by the switches appended to the SYSERR command string. There are two types of control switches: primary switches and secondary switches.

Primary Switches - The primary switches determine the type of report that SYSERR will generate. Refer to Table 1. Note that a single command string may have any number of primary switches.

Secondary Switches - The secondary switches are used to limit the report to a particular device, group of devices or date and time period. Secondary switches also control the level of detail reported. Refer to Table 2.

Table 1 SYSERR Primary Switches

Switch	Description
/ALL	*=/ALL<CR> List all entries in the input file.
/allNXM	*=/ALLNXM<CR> List all entries that pertain to nonexistent memory conditions (NXM).
/allPAR	*=/ALLPAR<CR> List all entries that pertain to parity errors.

Table 1 SYSERR Primary Switches (Cont)

Switch	Description
/allPER	*=/ALLPER<CR> List all performance-related entries.
/allSUM	*=/ALLSUM<CR> List a summary of each entry in the input file. This is the default switch.
Central Processor Switches	
/cpuALL	*=CPUALL<CR> List all CPU, main memory, and system-related entries. The front-end subsystem for the KL10-based system will also be listed.
/cpuCHK	*=/CPUCHK<CR> List all BUGCHK, BUGINF, and BUGHLT entries.
/cpuPAR	*=/CPUPAR<CR> List all entries that pertain to CPU-detected parity errors.
/cpuPER	*=/CPUPER<CR> List all CPU and system-related performance entries.
/cpuRLD	*=/CPURLD<CR> List all system or front-end reload entries.
Massbus Controller Switches	
/masALL	*=/MASALL<CR> List all entries that pertain to Massbus devices.
/masNXM	*=/MASNXM<CR> List all Massbus entries that may have been caused by a nonexistent memory.
/masPAR	*=/MASPAR<CR> List all Massbus device entries that pertain to parity errors.
Network and Front-End Switches	
netALL	*=/NETALL<CR> List all entries pertaining to networks.
netHDW	*=/NETHDW<CR> List all network hardware entries.
netOPR	*=/NETOPR<CR> List all network operator and network PDP-11 report entries
netPER	*=/NETPER<CR> List all DN64 statistics and line counter entries
System Switches	
sysLOG	*=/SYSLOG<CR> List system configuration status changes and system log messages.

Table 2 SYSERR Secondary Switches

Switch	Description	Cross Ref.
/BEGIN	*=/BEGIN:MM-DD-YY:HH:MM:SS<CR> or *=/BEGIN:-7D<CR>	1
	List only those entries that were recorded after the date and time specified.	
/BRIEF	*TTY:=/BEGIN:-7D/BRIEF:72<CR>	
	Print the sequence number, date and time, error type, and a brief summary of each entry. The number specifies the terminal page width (from 72 to 132 columns). See the /SEQUENCE switch.	
/DETAIL:	*=/DEV:DPA7/DETAIL:<CR>	
	List all Massbus controller and device register information, for each entry.	
/DEV:	*=/DEV:DPA<CR> or *=/DEV:MTA4<CR> or *=/DEV:RP06<CR>	
	List only entries for the logical or physical device specifed. Devices currently accepted are:	
	CD20 FE DEV RP04 TU70	
	CLOCK KLCPU RP05 TU71	
	CTY KLERR RP06 TU72	
	DH11 KLINIK TU16 TU77	
	DLSCAN LP20 TU45 11CPU	
	DL11C RM03	
	To indicate a specific disk drive (DP) or magtape drive (MT) by /DEV:name use the form DPabc or MTabc, where	
	a = the logical controller address	
	b = the logical MASSBUS address	
	c = the logical slave address for MT and 0 for DP.	
	The first summary listing will provide the logical addresses.	
/END:	*=BEGIN:MAY-6-79:00:01/END:MAY-12-79:24:00<CR>	
	Do not list any entries recorded after the end date and time specified.	
/NDEV:	*/NDEV:DPB3<CR> or *=/NDEV:RP04<CR>	
	This switch performs the opposite function of /DEV. Using /NDEV: device name will generate a listing of all entries except those which involve the named device.	
/RETRY:	*=/DEV:MTB7/RETRY:5<CR>	
	List only those entries that have a retry count greater than the value specified.	
/SEQ:	*=/SEQ:17<CR> or SEQ:(10,216,-4,517)	
	This switch is used after the /BRIEF: switch to list the entries for the sequence number(s) specified.	

MAINTENANCE SOFTWARE

Maintenance software is one of three major categories of software. Refer to Figure 1.

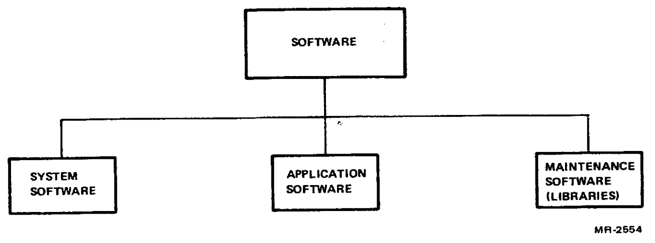


Figure 1   Three Major Categories of Software

Maintenance software has two primary uses: during preventive maintenance it is used to verify the operational status of the hardware, and during corrective maintenance it is used to detect and isolate (diagnose) hardware malfunctions.

Maintenance Libraries

Maintenance software is organized into maintenance libraries. The libraries are identified by the base processor that executes the program and the type of system the programs are designed to maintain. This guide, for example, describes four maintenance libraries. Refer to Figure 2.

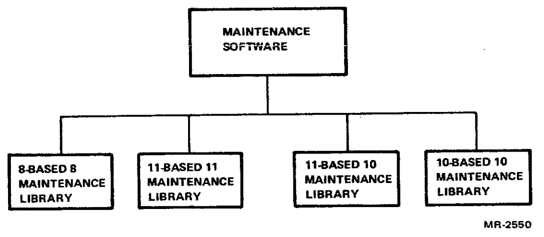

MAINTENANCE SOFTWARE
8-BASED 8 MAINTENANCE LIBRARY
11-BASED 11 MAINTENANCE LIBRARY
11-BASED 10 MAINTENANCE LIBRARY
10-BASED 10 MAINTENANCE LIBRARY
MR-2550

Figure 2   Four Maintenance Libraries

8-Based 8 Maintenance Library - Written in PDP-8 machine language and used to diagnose faults in PDP-8 based subsystems

11-Based 11 Maintenance Library - Written in PDP-11 machine language and used to diagnose faults in PDP-11 based subsystems

11-Based 10 Maintenance Library - Written in PDP-11 machine language, executed by the KL10 console front-end subsystem and used to diagnose faults in the KL10 processor, memory, and channels

10-Based 10 Maintenance Library - Written in PDP-10 machine language and used to diagnose faults in the PDP-10 processor, memory, and I/O subsystems

Maintenance libraries consist of utility programs, control files, and diagnostic programs. Refer to Figure 3.

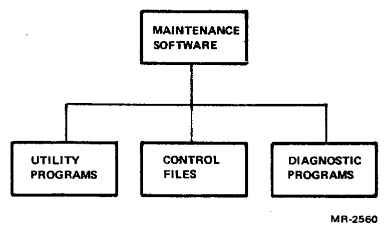


Figure 3 Component Parts of Maintenance Software

The utility programs and control files are used primarily to simplify loading, running, and maintaining the diagnostic programs and diagnostic storage media. The diagnostic programs have several uses which are listed below.

1. During the prototype stages of hardware design, diagnostics are used to determine if the system, subsystem, or unit under development functions properly and as expected. During this time both the hardware and the diagnostic(s) may undergo any number of changes or revisions.

2. During manufacturing and final system integration, diagnostics are used to assure that the product functions properly before shipment.

3. During installation, diagnostics are used to double-check manufacturing and to correct any malfunctions which may have occurred during shipment.

4. During customer acceptance, diagnostics are used by the installation team to demonstrate to the customer that the hardware is operating properly.

5. During preventive maintenance, diagnostics are used to assure that the system is fully (100%) operational. Diagnostics are also used at this time to identify potential problems and problems which can be deferred to a later, more convenient, date for correction.

6. During corrective maintenance, diagnostics are used to detect and isolate the cause of a hardware malfunction.

7. Following corrective maintenance, diagnostic programs are used to verify that all problems have been identified and resolved and that the system is fully (100%) operational.

Utility Programs
Basically, there are six types of utility programs.

1. Bootstrap Loaders - A bootstrap loader is a two-part program. The first part either resides permanently in a read-only memory (ROM) or must be manually deposited into memory. The second part resides in the boot block (block 0) of an input device. The first part consists of a few instructions which, when executed, will read in the remainder of the program from a predetermined input device. Once in memory, the bootstrap loader can be used to load and start a larger, more complex program such as a diagnostic or diagnostic monitor.

2. Downline Loaders - A downline loader is a program that enables the user to transfer a utility or diagnostic program from the host system to another system for execution.

3. Diagnostic Monitors - A diagnostic monitor is a special purpose loader that enables the user to:

   a. Manually (via commands) load and run a single utility or diagnostic program

   b. Automatically (via a control file) load and run a sequence or hierarchy of diagnostic programs.

4. Program Maintenance Utilities - This type of utility program is used to generate and patch or update control files and diagnostic programs.

5. File Maintenance Utilities - This type of utility program is used to generate and update diagnostic (disk and tape) storage media.

6. Hardware Utilities - A hardware utility is a utility-type program that supports maintenance or diagnostic functions. For example, the 11-Based 10 Maintenance Library uses utilities of this type to configure the main memory subsystem and to read and change the internal status of the CPU.

Utility programs are relatively easy to use because they support individual command sets which can be summarized in tables similar to the one following.

Table 1 TRACON Control Function Summary

Command	Description	Cross Ref.
A	A<CR> Auto insert - automatically builds an internal command file as commands are typed.	1
E	E<CR> Edit or create a command buffer. Refer to Table 2.	2

The table lists the base commands in alphanumeric order, provides an example which illustrates the proper command format, and briefly describes the task the command performs. The cross reference is used only when a more detailed description is necessary. For example, the following command description corresponds to the table illustrated above.

TRACON COMMAND DESCRIPTION
This section describes in detail each of the commands summarized in Table 1, Table 2, and Table 3.

1. A<CR> - The A command opens the command buffer for input. All commands typed following an A command are entered into the buffer until a KA command is typed. The commands in the buffer are executed via the X, L or M command. The buffer may be saved for future reference with the DC command.

Control Files
A control file or script is an ASCII text file which is generated using a standard editor program and which contains a list of commands to be executed by a program, usually a utility. When directed to do so, the program will read the control file into a memory buffer and begin executing the commands it contains as though they were entered directly from the user's terminal.

The following is a typical example of a control file. The file in the example is part of the B command file executed by KLDCP.

Example:

```
;B.CMD, KL10 PROCESSOR DIAGNOSTIC BOOT, 9-JUN-78
;PROCESSOR HARDWARE

;DTE20 INTERFACE
P DGDTE.A11
SED 2
;EBOX PART 1
P DGKAA.A11
SED 1
;EBOX PART 2
P DGKAB.A11
SED 1
;MBOX BASIC
etc.
```

In this example, the utility program (KLDCP) is directed to run two passes of DGDTE, one pass of DGKAA, one pass of DGKAB, etc. Note that command lines preceded by a semicolon are considered text and are ignored by the program processing the control file.

During preventive and corrective maintenance, control files are used to automatically direct the diagnostic monitor to sequence through a series (hierarchy) of diagnostic programs. Control files used in this manner assure that the proper programs are run in the correct order and for the prescribed period of time.

Diagnostic Programs
Diagnostic programs are divided into two major categories; exercisers, which include system exercisers and subsystem exercisers, and operability tests, which include functional tests and hardware (FRU) tests. Refer to Figure 4.

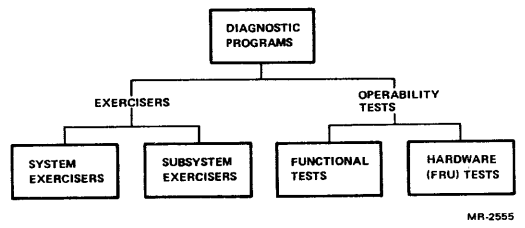


Figure 4 Major Categories and Types of Diagnostic Programs

Exercisers are designed to generate maximum system interaction by operating the CPU, storage and/or I/O subsystems at or near full capacity. In this respect exercisers are used to:

1. Localize a system malfunction to the failing subsystem or unit within that subsystem

2. Troubleshoot (intermittent) reliability, priority arbitration, and/or subsystem interaction faults which only occur when the system is operating at or near full capacity.

Operability tests are designed to systematically test each individual instruction, operation or function the hardware is capable of executing. Primarily, operability tests are used to detect solid hardware faults and isolate the cause to the failing function or field replaceable unit (FRU).

Loading and Starting Procedures - Each maintenance library has a standard procedure for loading and starting utility and diagnostic programs. Typically, the procedure involves a bootstrap or ROM loader, a primary loader (e.g., a diagnostic monitor), a set of standard program starting and restarting addresses, and a set of standard program control switches. The procedure generally describes how to:

1. Boot the system and load the primary loader

2. Direct the primary loader to load the desired utility diagnostic, or control file

3. Sets the desired program control switches

4. Direct the primary loader to start the program or execute the control file.

Operational Control - There are two principal methods of controlling the operation of a diagnostic program. The first is control switches and the second is operator dialogue. Many diagnostics use a combination of both.

Switch control is usually implemented through the console data switches. Each switch is assigned a fixed program control function (e.g., inhibit printouts, loop on test, halt on error, etc.). The switches are read at the start of the program and may be read periodically by the program while it is running. The program uses the state of the control switches to determine operating parameters as well as how to react to various error conditions.

Operator dialogue takes place between the diagnostic and the user's terminal. It enables the diagnostic to query the user and enables the user to specify program parameters such as the subsystems and/or units to be tested, the type of test to be run and the data patterns to be used during testing. The flexible control offered by operator dialogue complements the more rigid control supplied by switches.

Thus many diagnostics, particularly those in the exercise category, use the switches to specify standard control functions and the operator dialogue to specify variable test parameters.

Internal Structure - Although there is no rule stating that all diagnostic programs must use a standard internal structure, most do. The structure consists of four parts: a program initialization and control routine, a main diagnostic segment, a set of service routines, and a storage area for fixed and variable data. Refer to Figure 5.

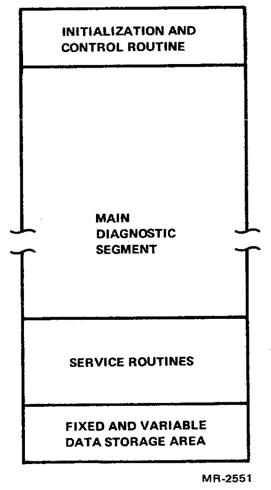

INITIALIZATION AND CONTROL ROUTINE
MAIN DIAGNOSTIC SEGMENT
SERVICE ROUTINES
FIXED AND VARIABLE DATA STORAGE AREA
MR-2551

Figure 5   Internal Structure of Diagnostic Programs

Program Initialization and Control Routine - This routine initializes the system, reads and stores the state of the program control switches, performs operator dialogue and dispatches to the main diagnostic segment of the program.

Main Diagnostic Segment - The main diagnostic segment consists of a set or series of exerciser or test routines. The routines that make up exercisers are inextricably intertwined (complex to say the least). This is particularly true at the machine language level. On the other hand, the series of test routines that constitute operability tests are relatively easy to understand. This is true both at the conceptual level and at the machine language level. The organization and structure of operability tests will be discussed later.

Service Routines - Service routines are used by both the program initialization and control routine and the routines that make up the main diagnostic segment of the program. They are used to handle keyboard input and printer output, generate data patterns, load and unload memory buffers, and format error messages. Separating service routines from the main program in this manner and making them available to both the initialization and test routines simplifies programming, eliminates redundancies, helps standardize the diagnostics and makes it significantly easier to read the program listing should that become necessary.

Fixed and Variable Data Storage Area - The fixed storage area is used to store program constants such as fixed ASCII messages and test operands. The variable data storage area provides temporary buffer areas for storage of terminal input and output, program stack operations, various test data patterns, etc.

Exercisers - As mentioned earlier, system exercisers and subsystem exercisers are similar both in design and in intended use. System exercisers are designed to generate maximum system interaction by operating the CPU, storage, and I/O subsystems at or near full capacity. Subsystem exercisers are designed to generate maximum subsystem interaction by operating the control unit, data channel (if present) and one or more storage or I/O devices at or near full capacity.

System exercisers are intended to:

1. Localize a system malfunction to the subsystem or device causing the failure

2. Troubleshoot (intermittent) reliability, priority arbitration and subsystem interaction faults.

Subsystem exercisers are intended to:

1. Localize a subsystem failure to the control unit, channel or device causing the failure

2. Troubleshoot (intermittent) reliability, internal priority arbitration,and device and device bus interaction faults.

There are three important points to keep in mind when using exercisers.

1. Exercisers are designed to troubleshoot the so-called intermittent class of failure. They should never be used to troubleshoot solid faults or faults that can be detected and isolated using an operability test.

2. Exercisers operate on the assumption that certain hardware error checking logic is functional (e.g., parity checking networks, time-out logic, etc.). In many cases they use this logic to indicate that an error has occurred. Therefore, before using an exerciser for troubleshooting purposes, use the appropriate operability test to verify that no solid hardware malfunctions exist in the error checking logic.

3. Exercisers typically provide extensive operator dialogues and comprehensive error message reports. This is done because the internal structure and control of exercisers tends to be complex. When an exerciser must be used for troubleshooting, use the flexibility of the operator dialogue, all the information provided by each error message and deductive reasoning to determine the cause of the problem. Avoid using troubleshooting techniques that require analyzing the exerciser at the machine language level.

Operability Tests - Functional tests and hardware (FRU) tests are also similar in design and intended use. The major difference between the two is their degree of isolation or fault resolution. Functional tests isolate faults to the failing function. From that point the technician must analyze the failing test or subtest and determine which component or field replaceable unit is causing the failure.

Hardware (FRU) tests, on the other hand, attempt through a logical analysis of the symptoms, to isolate the specific field replaceable unit (FRU) causing the failure. Thus, in most cases, the need to analyze hardware (FRU) tests at the machine language level is eliminated. The exception, of course, is when the test detects a fault but is unable to identify the failing FRU. In this case an analysis of the test would be required.

The main diagnostic segment of an operability-type diagnostic (as mentioned earlier) consists of a series of individual test routines. Each test routine is designed to check one specific hardware function. When a function can be checked in more than one way (e.g., a multiplexer with several data and gating inputs) then the test for that function will be divided into subtests. Each subtest will in turn test one aspect of the function until the function is completely tested. Refer to Figure 6.

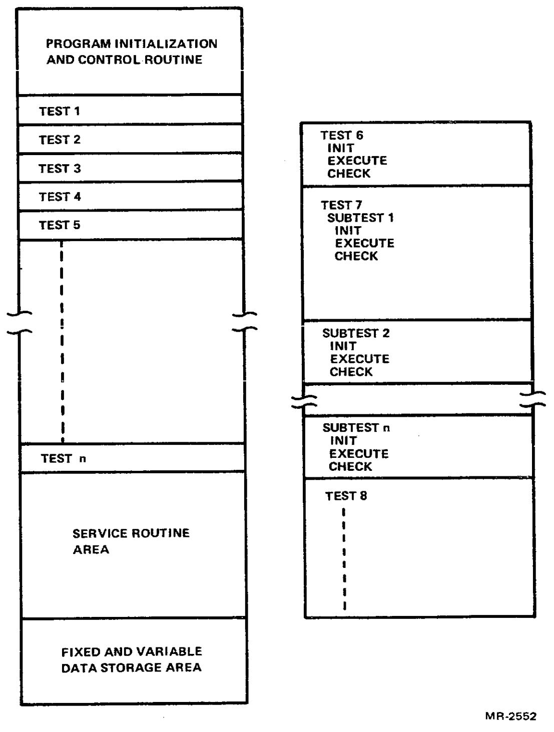


Figure 6 Internal Structure of Operability Tests

Both test and subtest routines have three major sections of machine language code.

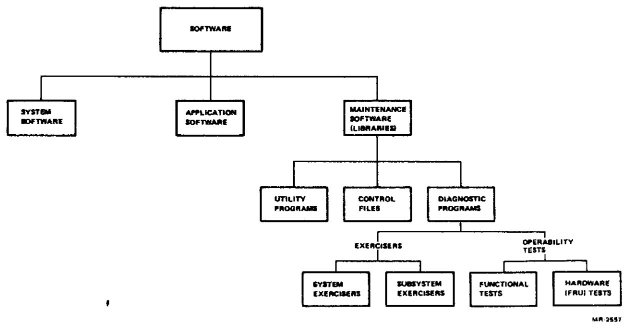
1. INIT - The INIT or initialization code preconditions the hardware for testing. This may involve tasks such as resetting the hardware, checking to assure that no error conditions already exist, setting and clearing status bits generating test operands, and building channel command lists and data buffers. Service routines are frequently used to accomplish many of the tasks involved in test initialization.

2. EXECUTE - The execute code usually consists of one machine language instruction which will cause the function or operation under test to activate. Occasionally two or more instructions are required for this purpose.

3. CHECK - The checking code determines if the hardware responded properly. Most often this involves reading hardware status registers and checking buffers for correct data. Typically, an error checking service routine is used for this purpose. The expected results of the test are passed to the error checking routine by the test initialization code. The actual test results are passed to the routine by the checking code. The error checking routine compares the expected results with the actual results and determines if an error has been detected.

If no error is detected, program control will generally proceed to the next test or subtest in the sequence. The program will continue in this manner until each hardware function or FRU has been completely tested for solid failures. If, on the other hand, the test does detect an error, the program may respond in a number of ways. For example, it may print an error message, ring the bell on the user's terminal, halt, go on to the next test or loop on the failing test. How a program responds to an error depends largely upon the initial operator dialogue and the state of the program control switches.

SUMMARY
The following diagram and 13 points summarize the organization and structure of maintenance software.



1. Maintenance software is one of three major categories of software.

2. Maintenance software is divided into specific maintenance libraries.

3. Maintenance libraries include utility programs, control files, and diagnostic programs.

4. Utility programs (excluding hardware utilities) simplify using and maintaining the diagnostic programs and maintenance library storage media. They are not diagnostics and should never be used as such.

5. Hardware utilities support functions that can be used for diagnostic purposes.

6. Control files are ASCII text files which contain a list of commands to be executed by a utility program. They too simplify using and maintaining the maintenance library.

7. There are two major categories of diagnostics; exercisers and operability tests.

8. During preventive maintenance both types are used to verify the operational status of the hardware.

9. During corrective maintenance

   a. Exercisers are used to localize malfunctions and troubleshoot intermittent (reliability) problems.

   b. Operability tests are used to detect solid faults and isolate the cause to the failing function or field replaceable unit.

10. Most diagnostics use a common structure consisting of

    a. a program initialization and control routine
    b. a main diagnostic segment
    c. a set of service routines
    d. a storage area for fixed and variable data.

11. The internal organization of exercisers is complex. Troubleshooting techniques that require analysis of them at the machine language level should be avoided.

12. The internal organization of operability tests is basically straightforward. They consist of individual tests and subtests.

13. Tests and subtests have three major parts.

    a. INITialization
    b. EXECUTE
    c. CHECK

Table of Contents

8-BASED 8 MAINTENANCE LIBRARY

The 8/8 Maintenance Library is included in this volume of the KL10 Maintenance Guide because a PDP-8A minicomputer is used as a microcontroller for the TU70 Series magtape subsystems. The following tables and procedures are used to describe the library and its utilization.

Table 1 8/8 Utility Programs

Utility	Description
BINARY	A diagnostic loader for paper tapes having a PB suffix (e.g., DJKAA-C-PB1)
RIM	A manually deposited readin mode loader for paper tapes having PM suffix (e.g., DJKAA-C-PM1). There are two versions of the RIM loader. Refer to Table 3. The high-speed version is used with a high-speed paper tape reader. The low-speed version is used for low-speed paper tape readers (e.g., ASR-33s).

Table 2 8/8 (PDP-8A) Processor and Memory Diagnostic Hierarchy

Diagnostic	Description
DJKAA.A8	PDP-8A CPU Test
DJMSA.A8	1K to 4K MS8-A MOS Memory Test
DJEXA.A8	1K to 32K Random Memory Reference Instruction Exerciser.

Table 3 8/8 Low-Speed and High-Speed RIM Loaders

Memory Location	Low Speed	High Speed	Comments
0156	6032	6014	RIM loader starting address
0157	6031	6011	
0160	5357	5357	
0161	6036	6016	
0162	7106	7106	
0163	7006	7006	
0164	7510	7510	
0165	5357	5374	
0166	7006	7006	
0167	6031	6011	
0170	5367	5367	
0171	6034	6016	
0172	7420	7420	
0173	3776	3776	
0174	3376	3376	
0175	5356	5337	

Procedure 1   8/8 Loading RIM Formatted Paper Tapes

Step	Procedure
1	Manually deposit the appropriate RIM loader into memory. Refer to Table 3.
2	Place paper tape to be loaded in the paper tape reader and turn the reader ON.
3	Load address 0156, press key CLEAR, then START. The tape will automatically read in.
4	Press key STOP after the last data on tape has read in but before the blank tape trailer has passed through the reader.
5	Refer to the specific program summary for further operating information.

Procedure 2   8/8 Loading Binary Formatted Paper Tapes

Step	Procedure
1	Read in the binary loader program. Refer to Procedure 1.
2	Place the paper tape to be loaded in the paper tape reader. Turn the reader ON.
3	Load addresses 7777. Press key CLEAR.
4	Reset console data switch 0 (i.e., 3777) if a high-speed paper tape reader is being used. If a low-speed reader is used, leave the switches set at 7777.
5	Press key START. The tape will read in automatically and the processor should halt with the AC display equal to all 0s. If the processor halts before the tape has read in or if the AC is non-0, a checksum error has occurred. Correct the problem and reload the tape.
6	Refer to the specific program summary for further operating information.

GENERAL INFORMATION

Code	DJEXA.A8
Title	1K to 32K Random Memory Reference Instruction Exerciser

Abstract

The 1K to 32K random memory reference instruction exerciser program is a test to check the execution of the memory reference instruction (AND, TAD, ISZ, DCA, JMP and JMS) of the processor for all addressing modes in a 1K to 32K PDP-8E/8A series computer. The program rolls up and rolls back a page at a time in a 2K, 3K or 4K memory and also swaps up and down between memory fields if more than 4K is available.

The program initially occupies locations 0 to 1776. Addresses 0 to 177 are used for program loading and initialization. They will be destroyed once the program has been started. If the program is allowed to relocate, it will roll up and back through a memory field and relocate between memory fields if more than 4K. All locations outside of the area taken up by the program, in any field, are used as a test area and they are filled with halts after every 4096 instructions have been tested.

Normally all testing is done randomly, but for troubleshooting purposes, the program can be constrained to absolute addresses, instructions, and constant data.

Hardware Required

PDP-8A, PDP-8E, PDP-8F or PDP-8M processor/1K to 32K of core memory/load device/programmer's console (optional)

Preliminary and Associated Programs

Refer to diagnostic hierarchy (8/8 STD module).

Restrictions

1. Each time the programmer's console is installed or removed the program must be reloaded and bit 0 of location 21 reinitialized.

2. Before each program start, location 21 (bits 7-11) in the program field "must be" initialized for the amount of memory to be tested. Location 21 initially is preset to 0 (no front panel switch register and 1K of memory). Refer to Table 2.

3. Once the program has rolled into another area of memory, the memory size cannot be decreased below the 1K segment that the program is located in.

Notes

1. If a programmer's console is installed the program should be stopped by setting the switch register to 0400. This will assure that the program is not in the process of relocating. For those systems without a front panel, it is best to reload the program.

2. DJEXA is run automatically under DDDXA, a PDP-10 TU70 diagnostic.

3. DDDXA supports code which simulates the programmer's console.

4. If running in a 1K system, the test instructions are executed in page 0 of field 0 only and the program will not roll.

5. Constraining - The programmer's console must be installed to constrain the program. The program is normally set up to select instructions, addresses, addressing modes and test fields (if any) randomly. For troubleshooting purposes, the operator has the capability of constraining the program to the following specific areas.

   a. Memory and AC data

   b. Memory field(s)

   c. Instructions (AND, DAC, ISZ, JUMP, JMS, or TAD)

   d. Addressing modes (direct, indirect, and auto index)

   Refer to the listing on microfiche for a description of the constraining procedure.

Loading and Starting Procedure

Via TU70 Diagnostic
DJEXA is run automatically under DDDXA.A10. Refer to the 10-Based 10 Maintenance Library.

Via Paper Tape
DJEXA is available in standard RIM format. Refer to the 8/8 STD module for loading procedures.

Console Switches

Refer to Table 1. If the program was initialized to run without the programmer's console, location 20 in the program field "must be" set for the desired switch register setting (normally 0s) before each program start. Location 20 initially is preset to 0.

OPERATIONAL CONTROL
With the program loaded and the CPU stopped, initialize locations 20 and 21 using either the programmer's console or the console program built into DDDXA. Refer to Table 1 and Table 2 respectively.

LOAD ADDRESS to address 0200, set the switch register, and press CLEAR, then CONTINUE. The program will now run until an error is encountered or until switch register 3 is set to a 1.

To restart the program if it was stopped by setting switch register 3, proceed as follows.

1. Examine location 1 of the 4K field in which the program is located.

2. Reset the switch register.

3. Set the extended memory address field equal to bits 6, 7 and 8 of location 1.

4. Set switch register bits 0 through 5 equal to bits 0 through 5 of location 1.

5. Press LOAD ADDRESS, set the switch register (refer to Table 1), press CLEAR, then CONTINUE.

PROGRAM DESCRIPTION
This program is quite powerful to the extent that it checks the memory reference instructions for the following.

1. Correct execution of instructions in random addresses.

2. Correct execution of instructions in random fields.

3. Correct execution of instructions for the following addressing modes:

   a. Direct and indirect addressing

   b. Same page and page 0 addressing

   c. Auto index addressing

4. Correct random memory data after execution of instruction.

5. Correct random AC data after execution of instruction.

All the above operations are selected randomly but they can be constrained. The program can test 1K to 32K of memory in 1K increments. Testing in a 1K memory is limited to testing instructions only on page 0.

The program has the capability of relocating through memory for the purpose of testing random instructions in the addresses where the program was previously located. If the program is limited to 1K, there will be no relocation. The program rolls up through memory after every 4096 instructions have been tested. The program rolls up by relocating the program by one memory page after each roll memory is filled with halts. When the program has rolled up to the upper limit of this field and after 4096 instructions have been executed, the program is then rolled back by one memory page. After every 4096 instructions, the program is rolled back until the program resides in addresses 0200 to 1777. At this time the program will swap up to another field if selected or start rolling up if limited to a 2K to 4K memory. If program was relocated by swapping up to another field, all of memory outside of program area is filled with halts. The program will now roll up and roll back in this field. This rolling up and rolling back and swapping up is done until the upper test field is reached. When this field is reached, the program will roll up and roll back in this field and then swap down by 1 memory field. This rolling up and down and swapping down is done until field 0 is reached and then the sequence is repeated over. After every roll and swapping of the program, memory is filled with halts. All relocating of the program is checked for errors.

Location 1 of the program field contains the starting address and the field that the program is in. The contents of location 1 is in the following format: SAF0. By adding 00 onto SA, the number obtained (SA00) will be the new starting address for the program. F should equal the field that the program is in.

ERROR SUMMARY
DJEXA halts upon detecting an error. Also at start time and after each program roll and field swap, the program fills all memory not occupied by the program with halts.

To determine why a particular halt occurred proceed as follows.

1. Examine location 1 of the 4K field that the program is located in. Record this value unless location 1 contains 7402 (HLT). If location 1 contains 7402 the halt occurred outside the program area. This usually indicates an addressing problem. In such a case continue to examine location 1 at page boundary (i.e., 200, 400, 600 etc.) until a value other than 7402 is obtained. Record this value.

2. Set the extended memory address field equal to bits 6, 7, and 8 of the word stored in location 1.

3. Add 11 to bits 0-5 of the word stored in location 1 and set switch register bits 0-5 equal to the result.

4. Set switch register bits 6-11 equal to 31.

5. Press LOAD ADDRESS, reset the switch register, press CLEAR, then CONTINUE. The CPU will halt at location XX50 where XX equals the even page boundry of the program location.

6. Location XX50 is a common halt used to display the information described in Table 3. Record the value of the AC for each halt (1 through 12). Use key CONTINUE to proceed through the halts. Press key CONTINUE after the last error halt (12) to loop on the error. This assumes that the switch register is set up for error looping.

Example:
To determine the type of error, the operator must understand the test instruction setups. The JMS indirect addressing mode setup is included here as part of the example. Setups for the remaining instructions are described in the listing on microfiche.

JMS Indirect Addressing Setup

1. Instruction setup is put in some random field.
2. Location 0 of this field contains the return pointer to the program.
3. The contents of the AC contain some random number.
4. The program JUMPS to the instruction address.
5. Instruction address = the test JMS indirect instruction.
6. Reference address = indirect address.
7. Indirect address should contain instruction address +1 after execution of instruction.
8. Indirect address +1 = CIF to program field.
9. Indirect address +2 = JMS I 0 return to program.
10. Indirect address +3 = JMS I 0 return to program.

Following is a list of error information which was recorded as a result of a JMS-type error. For a complete description of the error catagories refer to Table 3.

Halt	AC	Halt	AC
1 EXP FIELD	0050	7 REF ADD	5343
2 RET FIELD	0040	8 INDIRECT ADD	7004
3 EXP PC	7007	9 INIT MEM DATA	7415
4 RET PC	7007	10 FINAL MEM DATA	5214
5 INST ADD	5213	11 INIT AC DATA	5117

The setup for the above example is as follows.

1. The test instructions were supposed to go into field 5.
2. The AC before execution of the test instruction was 5117.
3. The program JUMPS to location 5213 in field 5 and executes the following code.

Location	Contents	
5213	4743	instruction address and instruction
5343	7004	reference address and indirect address
7004	XXXX	indirect address and some unknown number
7005	6202	change I.F to program field
7006	4400	return to program
7707	4400	return to program

NOTE
Initial memory data is N/A.

It can be seen from the table that the JMS worked but instead of putting the instructions in field 5, they were put into field 4. Therefore, it may be concluded that bit 8 was not loaded into the instruction field register.

Table 1 DJEXA Location 20 or Console Switch Summary

Switch	State	Description
0	0	Normal operation
	1	Inhibit error halt
1	0	Normal operation
	1	Loop on test conditions
2	0	Normal operation
	1	Inhibit program relocation
3	0	Normal operation
	1	If switch 1 = 0 halt after execution of 4096 test instructions

Table 2 Location 21 Program Test Field Codes

Memory	LOC21	Memory	LOC21	Memory	LOC21	Memory	LOC21
1K	0000	9K	0010	17K	0020	25K	0030
2k	0001	10k	0011	18k	0021	26k	0031
3k	0002	11k	0012	19k	0022	27k	0032
4K	0003	12K	0013	20K	0023	28K	0033
5K	0004	13K	0014	21K	0024	29K	0034
6K	0005	14K	0015	22K	0025	30K	0035
7K	0006	15K	0016	23K	0026	31K	003
8K	0007	16K	0017	24K	0027	32K	0037

NOTE

Bit 0 of Location 21 = 0 indicates the programmer's console is not installed.

Bit 0 = 1 indicates the programmer's console is installed.

Table 3   Error Halt Information

Halt No.	AC Contains
1	The field that program put instruction in.
2	The field the program returned from. N/A if program halted outside of program area.
3	The expected PC return from test instruction.
4	The actual PC return from test instruction. N/A if program halted outside of program area.
5	The address of the test instruction. The program goes to this address minus one to do a CIF for AND, TAD, ISZ, DCA before executing the test instruction.
6	The test instruction that was executed.
7	The address which the test instruction will reference or, if instruction is indirect, this address will contain the indirect address.
8	The indirect address which the test instruction will reference. N/A if test instruction is a direct address.
9	The memory data which is put into reference address or indirect address if instruction is direct or indirect. N/A for a JUMP or JMS instruction.
10	The contents of reference address 'or indirect address after execution of instruction. N/A if program halted outside of program area. NOTE For a JMP instruction this number should be equal to a CIF X and for a JMS instruction this number should equal the instruction address plus 1 location.
11	The contents of the AC before the execution of the instruction.
12	The contents of the AC after execution of the test instruction. N/A if program halted outside of program area.

GENERAL INFORMATION

Code	DJKAA.A8
Title	PDP-8A CPU TEST
Abstract	The PDP-8A instruction test is designed to test all logic on the 8A CPU board that is testable by the use of programmed instructions. This includes the power fail option if it is installed in the CPU. The 8A instruction test uses locations 0000 through 3777. The 4K version of this test will run in any 4K memory field, as long as locations 00000 through 00177, and location 017777 exist and are read/write memory. Special RIM-format tapes are available to allow running the test in 1K of memory, by executing the test in two consecutive 1K segments. In this case addresses 0000 through 1777 are used.
Hardware Required	PDP-8A, PDP-8E, PDP-8F or PDP-8M mainframe/1K to 32K of memory/PDP-8A I/O simulator (optional)
Preliminary and Associated Programs	Refer to diagnostic hierarchy (8/8 STD module).

Restrictions

1. This cannot be run on PDP-8, PDP-8I, PDP-8L, PDP-8S and PDP-12 mainframes.

2. Interrupts - No interrupts except power fail and those from the DATA BREAK/INTERRUPT simulator are permitted.

3. M873 - If the CPU has a timeshare option (M873 only), the option must be disabled as it may cause unexpected interrupts.

4. Device Code 77 - In order to completely test ROM H, it is necessary to execute an IOT instruction with bit 3 of the IOT (e.g. 64XX, 65XX, 66XX, 67XX). The IOT instruction used by the test is 6770. If this IOT conflicts with a device on the system under test, disconnect the device from the machine while running part 2 of this test.

5. Programmer's Console - To run part 2 of this test (using the I/O simulator), in a machine with a programmer's console installed, the display must be set for the MD, STATE, or STATUS or an error halt will occur.

6. 1K to 3K machines - Special RIM-format paper tapes are required to run this test in machines with less than 4K of memory. The instruction test is segmented into two 1K segments that are punched on two RIM format paper tapes labeled 08-DJKKA-PM1. The second 1K segment is only for use with an 8A I/O simulator. If no simulator is available, the second 1K segment should not be run.

Notes

1. Untested Logic - Due to certain hardware restrictions, some logic on the PDP-8A CPU board is not tested by this program. Below is a list of logic that is known to be untested.

```
ROM A:
  addresses 00 - 03 (extended load address 7)
  addresses 04 - 07 (extended load address)
  addresses 14 - 17 (load address)
  addresses 20 - 23 (examine)
  addresses 24 - 27 (deposit)
```

Auto restart logic (restart after power fail)

Next time state stall logic

Load address line (DR2) and associated logic

2. Versions - There are several versions of DJKAA.

Field Service/XOR 4K Version	MAINDEC-08-DJKKA-C-PB1
Field Service 1K Segment Part 1	MAINDEC-08-DJKKA-C-PM1
Field Service 1K Segment Part 2 (I/O simulator required)	MAINDEC-08-DJKKA-C-PM2
APT-8A 4K Version	MAINDEC-08-DJKKA-C-PB2
APT-8A 1K Segment Part 1	MAINDEC-08-DJKKA-C-PB3
APT-8A 1K Segment Part 2	MAINDEC-08-DJKKA-C-PB5
ACT-8E Version	MAINDEC-08-DJKKA-C-PB5

The XOR variation of the field service version, the APT-08A and ACT-8E versions are documented in the listing on microfiche.

3. Errors - It is recommended that the program be reloaded after an error has been detected.

Loading and Starting Procedure

Via TU70 Diagnostic
DJKAA is run automatically under DDDXA.A10. Refer to the 10-Based 10 Maintenance Library.

Via Paper Tape
DJKAA is available on binary-formatted paper tape. Refer to 8/8 STD module for loading procedures.

The field service 1K versions of DJKAA are on RIM-formatted paper tape. Refer to 8/8 STD module for loading procedure.

Control Switches

Refer to Table 1. If a programmer's console is not available, deposit the desired state of the console switch register into memory location 20.

OPERATIONAL CONTROL

1. If a programmer's console is installed, initialize memory location 21. Refer to Table 2. If no initialization is performed, the program will assume that no programmer's console or I/O simulator is available.

2. If running the 4K version, or the first 1K segment, initialize location 0221 to a 7000, to prevent the initial halt (step 6).

3. Load address 200.

4. Set the control switches. Refer to Table 1.

5. Press INIT, then CONTINUE.

6. The CPU should halt with MA = 0222, AC = 7777 and LINK = 1. If they are incorrect refer to the ERROR SUMMARY.

7. Set front panel indicator switch to MD, STATE, or STATUS position.

8. Press CONTINUE. Do not press INIT.

9. The program will run continuously unless an error is detected or unless control switch 3 is set.

TEST SUMMARY

During the first portion of the test, all basic Group 1 and Group 2 operate instructions are tested. No combined operate instructions are tested except CLA CLL. Then all MRI instructions are tested, using both direct and indirect addressing. During this section the adder is tested by the use of TAD instructions and IAC. Finally basic Group 3 operate instructions are tested.

ROMS D and F on the CPU module are tested by executing all Group 1, 2, and 3 operate instructions of the form: 7XX0, 7XX1. For each instruction thus tested, 8 different data combinations of AC, MQ, and LINK are used. The sequence of ROM testing is as follows:

1. The AC, LINK, and MQ are set to specified values.
2. The instruction to be tested is executed.
3. The AC, LINK, and MQ are saved, and whether the instruction skipped is noted.
4. The AC, LINK, and MQ are set to the same values as in step A.
5. The instruction is simulated using only those instructions that were tested during the first part of the test (basic operate and MRI instructions).
6. The results of the simulation are compared to the results of the actual instruction. Any differences result in an error halt.

The logic on the CPU board concerned with data breaks and interrupts is tested by the use of a special simulator. I/O SKIPS, AC transfers, interrupts, indicate logic, and data breaks are tested if the simulator is available. All Omnibus lines are tested either directly or indirectly with the exception of NEXT TIME STATE STALL (BR2) and the unused Omnibus line (BS2).

Table 1 DJKAA Location 20 or Control Switch Summary

Switch	State	Description
0-2		Not used
3	0 1	Loop on complete test Halt at end of test (1 pass)
4-11		Not used

Table 2 DJKAA Location 21 Bit Summary

Bit	State	Description
0	1	Indicates a programmer's console is installed.
3	1	Indicates a CPU I/O simulator is installed.
5	1	Run the XOR version of DJKAA.
6	1	Indicates the mainframe is a PDP-8E, PDP-8F or PDP-8M.

ERROR SUMMARY

All program errors are indicated by means of error halts. The program listing contains a brief explanation of the error to the right of each halt in the listing. Use the PC contents after the error halt to find the error information in the listing.

NOTE

If running the second 1K segment in a 1K CPU, add 2000 to the address of any error halt before consulting the listing.

The following errors have been included here because they require further explanation:

PC	Explanation
0036	This indicates that an unexpected interrupt was received by the CPU. Location 0000 will contain the address +1 of where the program was interrupted. For example, if location 0000 contains 2635, then the computer was interrupted after the instruction at location 2634.
0221	This is not an error halt unless the AC is not equal to 7777 or the LINK is not equal to 1. If the AC or the LINK is incorrect, load address 0200, press the HALT key, and press CLEAR momentarily. The program may now be executed one instruction at a time by pressing the CONTINUE key. The operator should check the contents of the AC and LINK against the expected contents given in the program listing after executing each instruction. When an instruction is executed and the computer registers no longer agree, the failing instruction has been found.
1631	A skip error occurred during the test of ROMS D and F. The instruction in the AC was executed, and resulted in a skip when none was expected, or did not skip when it was expected to skip. Make a note of the instruction, then press CONTINUE to get the contents of the AC, MQ and LINK at the time of the failure. To execute the failing instruction again, press CONTINUE. To execute the next instruction (or next data pattern with same instruction), make a note of the AC, LINK, and MQ contents for reference, then load address 1673, press CLEAR, then CONTINUE. If further error halts occur, the error information should be recorded for use in determining a pattern for the failure (e.g., CLL CML combination skips, SPA SNA does not skip if LINK is set, etc.).
1650	A data error occurred during the test of ROMS D and F. The instruction in the AC was executed, and resulted in incorrect contents of the AC, MQ, or LINK. Make a note of the instruction, then press CONTINUE to get the expected contents of the AC, LINK, and MQ. Make a note of the expected contents, then press CONTINUE to get the contents of the AC, MQ, and LINK as they were found after the instruction was executed. To execute the same instruction again, press CONTINUE. To execute the next instruction, or the next data pattern for same instruction, load address 1673, CLEAR and CONTINUE. If further errors occur, the error information should be recorded for use in finding a possible pattern in the error; e.g., CLL CMA CML combination does not work correctly.
All Others	Refer to program listing under proper PC.

Loop on Error

To loop on a failing instruction, (other than ROM D and F test), it is necessary to deposit a JUMP instruction in place of the error halt that is occurring. The JUMP instruction should cause the program to jump back to the point where the failing instruction is executed.

NOTE

If special conditions are required (e.g. AC and LINK must be clear, AC must be equal to 7777, etc.). The operator will have to deposit the proper setup instructions to cause these conditions previous to the failing instruction, and make the JMP after the failing instruction jump accordingly.

Example:

```
Address    Contents    Mnemonic

0361       1024        TAD K1        AC to 0001 link = 1
0362       1054        TAD K7777     AC to 0000 link to 0
0363       7450        SNA           Should not skip if AC = 0000
0364       7430        SZL           Should skip if link = 0
0365       7402        HLT           Carry failed to propagate
                                     through adder
```

If the program is halting at address 0365 with the AC non-0, one or both of the TAD instructions are failing. In order to loop on the failing instructions, a JUMP to the first TAD instruction (location 0361) could be placed at location 0365, but the AC would not be clear, and the LINK would not be set when the first TAD is executed. In order to loop correctly, the following patch is required to set the AC and LINK to the proper values. An asterisk indicates instructions that were deposited for looping purposes.

```
Address    Contents    Mnemonic

0360*      7320        CLA CLL CML    (Clear AC, set link)
0371       1024        TAD K1
0362       1054        TAD K7777
0363       7450        SNA
0364       7430        SZL
0365*      5360        JMP .-5        (Failed, do again)
0366*      7402        HLT            (Did not fail)
```

The loop should be executed the first time doing one instruction at a time with HALT/SS selected, to ensure that the instruction is still failing, and that any instructions inserted (in the above example the JMP .-5, and the CLA CLL CML) are not also failing.

In order to run the program again after repairs have been made, a reload of the program is required.

GENERAL INFORMATION

Code	DJMSA.A8
Title	1-4K MS8-A MOS Memory Test
Abstract	The 1-4K MS8-A MOS memory test is a program that will test MOS memories from 1K up to 4K. It consists of an address selection test, a floating 1s and 0s test and a worst-case data test. This program provides CPU-XOR, ACT-8A, ACT-8E and stand-alone capabilities. This diagnostic fits in four pages of a 1K segment. It will test the upper four pages plus up to 3K of RAM memory above. The program then relocates back to repeat the cycle continuously. The modified RIM loader described in Table 2 loads the diagnostic into the desired 1K segment. This becomes the lowest 1K segment to be tested. Before running the program the last address to be tested is deposited in location 23. The program writes over the RIM loader in operation, so it must be toggled in again if desired after the program runs the three tests on the RAM memory.
Hardware Required	PDP-8A mainframe/MS8-A MOS RAM memory (minimum 1K maximum 4K)
Preliminary and Associated Programs	Refer to diagnostic hierarchy (8/8 STD module).
Restrictions	1. The assumption is made that RAM will never go before ROM memory. 2. The 1K RAM in which the diagnostic is loaded will be considered the lowest 1K segment when test begins. 3. The program will have to be reloaded whenever the user wishes to change to another test environment (stand-alone, CPU-XOR, ACT-8A, ACT-8E).
Notes	1. ACT-8A Capability (manufacturing and depot only) ACT-8A hooks have been provided in the program to replace all error halts with exit returns to host when running program on the ACT-8A system. 2. CPU-XOR Capability (manufacturing and depot only) XOR hooks have been provided in the program to loop on XOR errors when running on the CPU-XOR tester. 3. ACT-8E Capability (manufacturing and depot only) ACT-8E hooks have been provided in the program to allow it to be run on the ACT-8E system.
Loading and Starting Procedure	Via TU70 Diagnostics DJMSA is automatically loaded under DDDXA.A10. Refer to the 10/10 Maintenance Library. Via Paper Tape: 1. The diagnostic is in RIM format. Toggle in the modified RIM loader as described in Table 2.

2. Read in the 1-4K MS8-A MOS memory test from the tape. Deposit the last address of the highest address range to be tested into location 23 of the range the diagnostic was loaded into. The default is 1777 which signifies a 1K memory starting at location 0.

3. Set switch register to location 200 of the range the diagnostic was loaded into. Press LOAD ADDRESS.

4. Set switch register according to Table 1.

5. Press CLEAR and then CONTINUE.

Control Switches

If you do not have a front panel, deposit the desired switch register contents in location 21 of the range the program is using. If you do have a front panel, deposit 4000 in location 21 of that same 1K segment. Refer to Table 1.

OPERATIONAL CONTROL
In order to load the diagnostic into a 1K range other than 0-1777 the modified RIM loader is used. Otherwise the standard RIM loader is satisfactory.

The user will modify X777 to the address range desired: 0000, 2000, 4000, 6000.

TEST SUMMARY
The individual tests performed by this diagnostic are described in Table 3.

ERROR SUMMARY
All errors are reported by error halts. All tests use the same error report routine. Error halts will occur at either location 737 or location 1737, depending on which portion of MOS RAM the program has entered.

Error halts should be handled as follows.

1. AC will display PC of test in error.
2. Press CONTINUE.
3. AC will display location being tested.
4. Press CONTINUE.
5. AC will display contents of location.
6. Press CONTINUE.
7. AC will display expected contents.
8. Set switch register for recovery as desired according to Table 1.

Program swap or program relocation errors will halt at either location 264 or location 1264. This type of error is fatal and indicates that the MOS RAM is faulty and that the program should not be continued.

Table 1 DJMSA Control Switch Summary

Switch	State	Description
0	0	Halt on error
	1	Do not halt on error
1	0	Normal operation
	1	Loop on error
2	0	Do all three tests
	1	Loop on test
3	0	Run continuously
	1	Halt on pass completion

Table 2 Modified RIM Loaders

Low RIM Loader (Modified)			High RIM Loader (Modified)		
Address	Code	Mnemonic	Address	Code	Mnemonic
X753	6032	KCC	X753	6014	RCF
X754	0375	AND X775	X754	0375	AND X775
X755	1377	TAD X777	X755	1377	TAD X777
X756	3376	DCA X776	X756	3376	DCA X776
X757	6031	KSF	X757	6011	RSF
X760	5357	JMP .-1	X760	5357	JMP .-1
X761	6036	KRB	X761	6016	RCC
X762	7106	CLL RTL	X762	7106	CLL RTL
X763	7006	RTL	X763	7006	RTL
X764	6031	SPA	X764	7510	SPA
X765	5367	JMP X757	X765	5354	JMP X754
X766	6034	RTL	X766	7006	RTL
X767	7420	KSF	X767	6011	RSF
X770	5367	JMP .-1	X770	5367	JMP .-1
X771	6034	KRS	X771	6016	RCC
X772	7420	SNL	X772	7420	SNL
X773	3776	DCA I X776	X773	3776	DCA I X 776
X774	5353	JMP X753	X774	5354	JMP X754
X775	1777	MASK	X775	1777	MASK
X776	0		X776	0	
X777	0	ADDRESS RANGE	X777	0	ADDRESS RANGE

Table 3 DJMSA Test Summary

Test	Description
1	ADDRESS SELECTION TEST Each location being tested has its address written into itself. They are then all tested scanning backward. Then, scanning backward, each location has the complement of its address written into itself. The memory is then tested scanning forward.
2	FLOATING 1S AND 0S TEST Through each location being tested a word is written, then tested. The word consists of the twelve ways a 1 can be floated (from 1, 2, 4 through 4096) and the twelve ways a 0 can be floated or the complement of the last 12 words. This test checks for interaddress bit shorts.
3	WORST-CASE DATA PATTERN TEST a. Write worst-case data pattern through RAM and test each location. Repeat with complement of pattern. b. Vary pattern and continue at step a. c. After a maximum of 12 patterns the test ends. The pattern will be varied from X,X',X,X' ... to X,X,X',X' ... to X,X,X,X, X',X',X',X' to etc. The last pattern in this progression will have the first half of the RAM contain X and the rest contain X'. The default for this test is X = 2525 and X' = 5252.

GENERAL INFORMATION

Code	DXMPA.A8
Title	DX10 I/O Processor Microcode
Abstract	The DX10 microcode is a PDP-8 program which manipulates the DX10 hardware to allow that hardware to function as an independent I/O processor. This processor, the DX10, has the ability to interface to the PDP-10 via both the I/O and the memory bus. It also interfaces to an STC magtape controller, the TX01, via an IBM-type channel bus. The logic within the DX10 is manipulated by a series of special IDTs. Basically, they allow the setting and clearing of flip-flops, the selection of registers, and the loading and reading of these registers. The microcode will perform these sequences in a prescribed manner which will allow it to fetch commands from PDP-10 memory and process these commands. The commands will typically direct the microcode to read and write magtape.
Hardware Required	KA10, KI10 or KL10 mainframe/32K of core minimum/TU70 magtape subsystem
Preliminary and Associated Programs	This microprogram is used in conjunction with DDTUA and DDTUB
Restrictions	None
Notes	None
Loading and Starting Procedure	DXMPA is automatically loaded by DDTUA and DDTUB.
Control Switches	None

OPERATIONAL CONTROL
Normally DXMPA is controlled by the program that loaded it. The PDP-8A may also be started at one of the following addresses.

Address	Function
200	Normal cold start address
201	Normal restart address
202	Start without diagnostics enabled
203	Reserved
204	Reserved
205	Start diagnostics only (implicit loop on error)
206	Reserved
207	Diagnostic error halt location

When the microcode is started at 205, only the diagnostics will run. If an error occurs, the error code is loaded into register number 17 and the diagnostic will loop on the failure.

The normal logout status is stored into ICPC+1, +2, and +3. The format is:

```
ICPC+1/ DSR<0:7>,CSR<8:19>,SEQCOD<22:27>,DAR<28:35>
ICPC+2/ BYTE COUNTER <0:13>, CPC<14:35>
ICPC+3/RECORD LENGTH<12:35>
```

DXMPA TEST SUMMARY
The microcode is loaded by a PDP-10 program using the DX10 hardware feature which allows loading and reading the memory of the PDP-8A with DATAO/DATAI commands. It is then started at PDP-8 memory location 200. The PDP-8 will first initialize the DX10, then perform a system reset on the magtape controller. A set of diagnostics will be run to ensure the hardware is functional. Then the PDP-8 enters an idle loop were it waits for further direction from the PDP-10. This direction comes from the channel command register. The PDP-10 can put the following four basic commands into this register.

1. If the PDP-10 sets both CLEAR and CONTINUE in the command register, this tells the PDP-8 to start the channel program at the address specified in the ICPC register.

2. If the PDP-10 sets only CONTINUE, this tells the PDP-8 to continue the channel program at the address specified in the ICPC register.

3. If the PDP-10 sets only CLEAR, this tells the PDP-8 to perform a systems reset.

4. If the PDP-10 sets the status request bit, this tells the PDP-8 to perform a store status operation. The address of the status buffer comes from the address specified in the ICPC register.

Once a channel program is started it continues running until either the tape system encounters an error or a channel command is fetched with the GO bit off.

If an error is stopping the channel program, the microcode will clear the CONTINUE bit, perform a store status operation, generate a PDP-10 interrupt, and finally return to the idle loop for further instruction from the PDP-10.

Microdiagnostics
The microcode will test portions of the DX10 hardware upon initial startup and periodically while it is in the idle loop. To ensure good response to PDP-10 commands, the PDP-8 checks the CONTINUE flip-flop prior to each test. The tests are described in Table 1.

If the microcode detects a diagnostic error it will halt at 207 after first loading IBus number 17 with the error code. This same code is available in PDP-8 memory location 5.

Hardware Readin
The hardware readin microcode is buffered in a 128-word ROM. This ROM is block-transferred into the last page of PDP-8 memory when the hardware readin key is struck on the processor. This page of code will read the first record on the first ready drive into PDP-10 memory and then start the PDP-10 at location 100.

The code flows as follows:

1. Initialize DX10
2. Set up rewind command in CMD
3. Clear device address = 0
4. Perform rewind
5. If error-increment device address and go to step 4
6. Here when ready unit found
7. Change command to read data
8. Set up byte counter and data address registers
9. Start read command
10. Load channel bus control register
11. Wait for block done
12. Start PDP-10 at location 100.

Table 1 Microdiagnostic Test Summary

Test	Description
1-10	Test the special IOT's ability to clear the AC
11	Test the 8R load and read commands
12	Test the 8R selection logic
13	Test the silo logic in dump mode
14	Test the silo logic in dump mode with slow clock
15	Test the silo logic in byte mode
16	Test the silo logic in byte mode with slow clock
17	Test the silo logic in ASCIZ mode
20	Test the silo logic in ASCIZ mode with slow clock
21	Test the silo logic in SIXBIT mode
22	Test the silo logic in SIXBIT mode with slow clock
23	Test the block done logic (including interrupt)
24	Test the CPC register
25	Test the DAC register
26	Test the channel bus interface by performing a sense command to device address 0. NOTE Test 26 is not run when the diagnostics are called from the idle loop.

ERROR SUMMARY

Microcode
If the microcode detects one of the following errors it will set the correct bit in the CSR and store the error status.

Selection Error - This error will occur if the channel program sends the address of a device that is not present.

Sequence Error - The microcode will encode the type of sequence error into the code field of ICPC+1, halt the channel program, perform a systems reset, and return to the idle loop.

Code	Description
00	Channel program is structured incorrectly
02	Bus in parity error
26	A read or write command was not followed by transfers
36	Not all of the desired sense bytes were transferred in extended store of status
43	Initial selection error reading sense bytes

Device Parity - This error indicates a channel bus parity error occurred while reading or writing data.

Length Error - This error means that the record just read was either shorter or longer than expected.

OPI Error - Operation incomplete error sets if a transfer fails to complete within 10 seconds. (Each transfer command word is timed.)

If the DX10 encounters a nonexistent memory error or a memory parity error, the PDP-8 is halted and the PDP-10 is interrupted.

If the microcode detects an error while processing a command and the error is not a sequence or selection error, it will attempt to store extended status.

If ICPC+3 contains a status pointer it will do the extended store.

If ICPC+3 does not contain the pointer, no extended status is stored.

If ICPC+3 contained - D24B13+300 (777200000300) then the following information would be stored starting at location 300.

```
ICPC+1/DSR<0:7>,CSR<8:18>,SEQCOD<22:27>,DAR<28:35>
PCPC+2/BYTE COUNTER<0:13,>CPC<14:35>
ICPC+3/777200,,300

300/record length<12:35>
301/tag lines,,bus lines
302/dac<14:35>
303/version<0:5>,edit<6:17>,FR<18:35>
304/sense bytes 0, 1, 2, 3
305/sense bytes 4, 5, 6, 7
306/sense bytes 8, 9, 10, 11
307/sense bytes 12, 13, 14, 15
310/sense bytes 16, 17, 18, 19
311/sense bytes 20, 21, 22, 23
```

NOTES

1. The number of bytes stored is controlled by the byte count field of the word in ICPC+3. The format is byte count <0:13>; address <14:35>.

2. If a sequence error is detected while reading, then ICPC+1 and ICPC+2 is overwritten with the error information.

3. Forcing an extended store on rewind initiation could cause the loss of the completion interrupt.

Error Recovery
This microcode will automatically attempt recovery from tape errors if bit 3 of the device command instruction is set. Errors that can be recovered are: data check, DX10 silo data parity errors, and intermittent bus timeout checks.

Write errors are retried by backspacing over the record just written, reading in reverse until a record with no error is found or load point is found, then spacing over the good record, erasing 4.2 inches of tape and writing the record again. If the error persists, the sequence is repeated up to 75 times, each time erasing an additional 3.6 inches of tape.

Read errors are retried by attempting to reread the record in the same direction 30 times, moving the tape past the tape cleaner blade after every fourth read. If these rereads fail, the record is read in the opposite direction up to a maximum of 30 times. The read in opposite direction is not attempted if the record is longer than the length requested or if the record cannot be read into the same position in memory because of hardware restrictions. The read opposite is accomplished by building a channel transfer list in PDP-8A memory. Three pages are reserved for this list, so a list of up to 42 transfer words can be read.

Microdiagnostic Error Codes

Code	Description
100	LBO did not clear AC.
101	STM did not clear AC.
102	INT did not clear AC.
103	L8S did not clear AC.
104	LCB did not clear AC.
105	L8C did not clear AC.

106	L8B did not clear AC.
107	L8A did not clear AC.
110	8R byte A was read bad (L8A and G8A).
111	8R byte B was read bad (L8B and G8B).
112	8R byte C was read bad (L8C and G8C).
113	8R2 selection bad.
114	8R1 selection bad.
115	8R0 selection bad.
116	Wraparound selection of 8R3 bad.
117	Wraparound selection of 8R0 bad.
120	INT did not select 8R0.
121	Dump mode silo test failed.
122	Dump mode silo test (slow clock) failed.
123	Byte mode silo test failed.
124	Byte mode silo test (slow clock) failed.
125	ASCIZ mode silo test failed.
126	ASCIZ mode silo test (slow clock) failed.
127	SIXBIT mode silo test failed.
130	SIXBIT mode silo test (slow clock) failed.
131	TBD did not skip on overflow.
132	No interrupt on overflow.
133	Resetting overflow did not clear interrupt.
134	CPC data test failed.
135	DAC data test failed.
136	Control unit did not respond to selection sequence.
137	Control unit did not respond to device address 0.
140	ADDRESS IN did not set in selection sequence.
141	Bad parity on bus in with address in presented.
142	Not 0 address received when attempting to select unit 0.
143	ADDRESS IN did not drop during selection sequence.
144	STATUS IN did not set during selection sequence.
145	Bad parity on BUS IN with status byte presented.
146	Initial status byte was non-0 in sense command.
147	STATUS IN did not drop when service out was set during presentation of status byte.
150	SERVICE IN did not set presenting a sense byte.
151	Parity error on BUS IN while presenting a sense byte.
152	SERVICE IN did not drop in response to SERVICE OUT during presentation of sense byte.
153	STATUS IN did not set after presenting 24 sense bytes.
154	Parity error in status byte after sense command.
155	STATUS IN did not drop after presenting ending STATUS IN sense command.
156	OPERATIONAL IN did not drop after sense command.

157	REQUEST IN never set after short burst sequence.
160	ADDRESS IN did not set in CU-initiated sequence.
161	Parity error on BUS IN while reading address in CU-initiated selection sequence.
162	Non-0 address presented in CU-initiated selection sequence.
163	ADDRESS IN did not drop in CU-initiated selection sequence.
164	STATUS IN did not set in CU-initiated selection sequence.
165	STATUS IN did not drop in CU-initiated selection sequence.
166	OPR IN did not drop after CU-initiated selection sequence.
167	SERVICE OUT did not set while transferring data in tests 13-23.
170	SERVICE OUT did not clear while transferring data in tests 13-23.

Table of Contents

11-BASED 11 STANDARD INFORMATION
This module summarizes standard information and procedures which are common to many of the programs in the 11-Based 11 Maintenance Library.

PROGRAM IDENTIFICATION CODE

```
DZRXA.BIC
||| |
||| |    .BIN extension used to store program in ABS format.
||| |    .SAV extension used to store program in core image format.
||| |    .BIC extension indicates ABS format chainable program.
||| |
||| └──►Unique program identifier
|||
||└────►Major program class
||       CD = card reader
||       DH = DH11
||       RX = RX11
||       etc.
||
|└─────►Processor type

         A = 11/05, 15, 20
         B = 11/40
         C = 11/45
         Z = all processor types
         X = nonstandard program
|
└──────►Diagnostic
```

STANDARD CONSOLE CONTROL SWITCHES
Table 1 lists and describes the standard functions of the console data switches used to control the operations of most 11-based 11 diagnostics.

Table 1 Standard Console Control Switch Summary

Switch	State	Function
15	0	Normal operation
	1	Halt on error
14	0	Normal operation
	1	Loop on test. Loop on the test currently being executed.
13	0	Normal operation
	1	Inhibit error printouts. Error messages will not be printed on console terminal.
12	0	Normal operation. Set T bit on alternate passes of program.
	1	Inhibit trace traps. The T is not set.
11	0	Normal operation
	1	Inhibit iterations. Each test will be executed only once.
10	0	Normal operation
	1	Ring bell on error.
09	0	Normal operation
	1	Loop on error. The program will loop on the test that detected the error. If the error goes away during the looping, the program will proceed to the next test.
08	0	Normal operation
	1	Loop on test in SWR<07:00>. The program will loop on the test number specified by switches 07 through 00.
07-00	0	Normal operation
	1	Test no. Loop on test specified by bits 07 through 00 if switch 8 is set.

STANDARD PROGRAM STARTING ADDRESSES

Table 2 lists the standard starting addresses used by 11-based 11 maintenance programs.

Table 2 Standard Program Starting Addresses

Address	Function
000200	Diagnostic program starting address

UTILITY PROGRAMS

Table 3 lists the utility programs associated with the 11-Based 11 Maintenance Library.

Table 3 11-Based 11 Utility Programs

Utility	Description
XXDP.BIN	A universal PDP-11 diagnostic monitor used to load and sequence diagnostic and utility programs.
COPY.BIN	A program used to transfer 11-based 11 maintenance programs from one storage medium to another.
UPD1.BIN	A 4K program used to add, delete, rename, or patch 11-based 11 maintenance programs.
UPD2.BIN	An 8K enhanced version of UPD1.
XTECO.BIN	An editor program used to generate and edit ASCII text files.
Special-Purpose Loaders and Monitors	
DGQDD.A11	DN87S Loader
DGQDE.A11	DLDP/DL11-E Monitor
DGQDF.A11	DN2x Front-End Loader
DGQDG.A11	DN2x Secondary Front-End Monitor
DGQEA.BIN	DN2x Bootstrap Loader

DIAGNOSTIC PROGRAM HIERARCHIES

The following tables describe the 11-Based 11 Maintenance Library diagnostic hierarchies.

Table 4	System Exercisers
Table 5	PDP-11/34 Processor and Memory Management Diagnostic Hierarchy
Table 6	PDP-11/40 Processor Diagnostics
Table 7	KMC11 Microprocessor Diagnostic Hierarchy
Table 8	PDP-11 Memory and Internal Option Diagnostics
Table 9	Disk Subsystem Diagnostic Hierarchy
Table 10	DECtape Subsystem Diagnostics
Table 11	Hard Copy Equipment Diagnostics
Table 12	Interprocessor Buffer Diagnostics
Table 13	Data Communications Subsystem Diagnostic Hierarchies

Table 4 System Exercisers

Diagnostic	Title
DFQAA.BIN	1080 Front-End Subsystem Exerciser
SY2040.BIN	2040 Front-End Subsystem Exerciser
DEC/X11	Universal System Exerciser (user configured)

Table 5 PDP-11/34 Processor and Memory Management Diagnostic Hierarchy

Diagnostic	Title
DFKAA.BIC	PDP-11/34 CPU Test
DFKAB.BIC	PDP-11/34 Trap Test
DFKAC.BIC	PDP-11/34 EIS Instruction Test
DFKTH.BIN	PDP-11/34 Memory Management Diagnostic

Table 6 PDP-11/40 Processor Diagnostics

Diagnostic	Title
DBQEA.BIC	PDP-11/40 CPU Test

Table 7 KMC11 Microprocessor Diagnostic Hierarchy

Diagnostic	Title
DZKCA.BIN	KMC11 CPU Microdiagnostic
DZKCC.BIN	Basic W/R and Microprocessor Test
DZKCD.BIN	Main Memory, JUMP CRAM Test of Microprocessor
DZKCE.BIN	DDCMP - Mode Line Unit Test
DZKCF.BIN	BITSTUFF - Mode Line Unit Test

Table 8 PDP-11 Memory and Internal Options Diagnostics

Diagnostic	Title
DZQMC.BIC	Memory and Parity Test
DZBMD.BIN	BM873 ROM Test
DZKAQ.BIN	Power Fail Test
DZKWA.BIC	KW11-L Line Clock Test

CZQMCF.BIN

CCMFAF

Table 9 Disk Subsystem Diagnostic Hierarchies

Diagnostic	Title
RH11-RP04/05/06 Disk Subsystem	
DZRJG.BIC	Diskless Controller Test (Part 1)
DZRJH.BIC	Diskless Controller Test (Part 2)
DZRJI.BIC	Functional Controller Test (Part 1)
DZRJJ.BIC	Functional Controller Test (Part 2)
DZRJA.BIC	Mechanical and Read/Write Test
DZRJD.BIC	Performance Exercises
DZRJB.BIN	Formatter Program
DQRJC.BIN	Head Alignment and Verification Program
RX11-RX01 Floppy Disk Subsystem	
DQRXB.BIC	RX11 Floppy Test
DQRXA.BIC	RX11 Floppy Reliability Test

Table 10 DECtape Subsystem Diagnostics

Diagnostic	Title
DZTCB.BIC	TC11 DECtape Test
YPTC.BIN	PDP-11 DECtape Formatter YPTCBØ.BIN

Table 11 Hard Copy Equipment Diagnostics

Diagnostic	Title
DXLPB.BIN	LP20 Line Printer Diagnostic
DZCDB.BIN	CD11/CD20 Card Reader Diagnostic

Table 12 Interprocessor Buffer Diagnostics

Diagnostic	Title
DGDTE.BIN	DTE20 Diagnostic
DZITA.BIN	Interprocessor Test Program

Table 13 Data Communications Subsystem Diagnostic Hierarchy

Diagnostic	Title
DH11 Communication Multiplexer	
DZDHM.BIC	Basic Test
DZDHB.BIN	Memory Test
DZDHD.BIN	Speed Selection Logic Test
DZDHK.BIC	Modem Control Multiplexer Diagnostic
DZDHL.BIN	Overlay for Interprocessor test
DQDHN.BIN	Reliability Test
DL11 Data Line Interface	
DZDLC.BIC	DL11-E Test
DZDLD.BIC	DL11-W Test
DQ11 Data Line Interface	
DZDQA.BIN	Basic Logic Test (Part 1)
DZDQC.BIN	Interrupt Logic Test
DZDQD.BIN	Receiver and Transmitter Test
DZDQE.BIN	Miscellaneous RX/TX/BBC Test
DZDQG.BIN	DQ11 Trail Program (Parameter Input)
DUP11 Data Line Interface	
DZDPB.BIC	Basic DPU11 and Off Line SDLC Transmitter Test
DZDPC.BIC	DUP11 Offline SDLC Receiver, Modem Control and Interrupt Test
DZDPD.BIC	DUP11 SDLC Data, Functions and DEC Mode Test
DZDPE.BIC	DUP11 Confidence Test
DZDPF.BIC	ITEP Overlay for DUP11
DZ11 Communciation Multiplexer	
DZDZA.BIC	DZ11 8-Line Async Multiplexer Test
DZDZB.BIN	DZ11 Overlay for ITEP

STANDARD LOADING AND STARTING PROCEDURES

The following tables and procedures describe the various methods of loading and starting 11-Based 11 Maintenance Library programs.

Procedure 1 Standard ROM Bootstrap Procedure

Step	Procedure
1	Halt the CPU.
2	Determine the type of ROM used by the CPU.
3	Mount the program storage medium on unit 0 of the device to be used for bootstrapping. Make the unit ready and on-line.
4	Load the appropriate ROM starting address. Refer to Table 14.
5	Set the switch register as specified if BM792 ROM is being used; otherwise, clear the switch register.
6	Set ENABLE. Press START.
	The monitor will automatically load and start. Refer to the XXDP Summary.
	EXCEPTION: The paper tape absolute loader is not self-starting. Refer to Procedure 3, Step 6.

Table 14 Standard Bootstrap ROM Starting Addresses

Device	BM873-YB	BM873-YA	MR11-DB	BM792
RK11	773030	773010	773110	773100 (SWR=777406)
TC11	773070	773030	773120	773100 (SWR=777344)
TM11	773110	773050	773136	
TU16	773150			
TA11	773524	773230		773300
RP11 (RP03)	773350	773100	773154	
RH11 (RP04)	773320			
RH11 (RS03)	773000			
RF11	773136	773000	773100	
RC11	773212	773144	773220	
PC11	773620	773312		
KL11/DL11	773510			

Procedure 2 M9301 Bootstrap Procedure

Step	Procedure
1	Remove the Unibus terminator and insert the M9301 bootstrap module.
2	Load address: 173000 for system without memory management 773000 for systems with memory management.
3	Mount the program storage medium on the unit to be used for bootstrapping (usually unit 0).
4	Set ENABLE. Press START. The M9301 will respond by printing a dollar sign ($).
5	Type the appropriate M9301 command. Refer to Table 15. If a unit other than 0 is to be used for loading, append the command with the unit number (e.g., if DECtape unit 4 is to be used the command would be DT4<CR>). The monitor will automatically load and start. Refer to the XXDP Summary. EXCEPTION: The paper tape absolute loader is not self-starting. Refer to Procedure 3, Step 6.

Table 15 M9301 Bootstrap Module Command Summary

Command	Bootstrap DEVICE
CT	TA11 magtape cassette
DB	RJP04 disk pack
DK	RK11 disk cartridge
DM	RK06 disk
DP	RP11 RP02/RP03 disk pack
DS	RJS03/04 fixed head disk
DT	TC11 DECtape
DX	RX11 diskette
MM	TJU16 magtape
MT	TM11 800 BPI magtape
PR	PC11 paper tape
TT	DL11 ASR-33 terminal (paper tape)

Procedure 3 PC11/DL11 Paper Tape and Absolute Loader Bootstrap Procedure

Step	Procedure
1	Halt the CPU.
2	Select an address prefix which corresponds to the available memory from the chart below. Memory size Prefix 4K 017 8K 037 12K 057 16K 077 20K 117 24K 137 28K 157
3	Deposit: Address Data (xxx = the prefix) xxx 744 016 701 xxx 746 000 026 xxx 750 012 702 xxx 752 000 352 xxx 754 005 211 xxx 756 105 711 xxx 760 100 376 xxx 762 116 162 xxx 764 000 002 xxx 766 xxx 400 xxx 770 005 267 xxx 772 177 756 xxx 774 000 765 xxx 776 177 560 if load device is terminal reader. 177 550 if load device is PC11.

Procedure 3 PC11/DL11 Paper Tape and Absolute Loader Bootstrap Procedure (Cont)

Step	Procedure
4	Place the absolute loader paper tape in the reader.
5	Load address xxx 744 (xxx = prefix). Set ENABLE. Press START. The paper tape will automatically read in.
6	Initialize absolute loader to use either the hardware or the software switch register. Deposit: Address Data (xxx = prefix) xxx 516 777570 if HW SWR is used 000176 if SW SWR is used
7	Place diagnostic paper tape in paper tape reader.
8	Load address xxx 500 (xxx = prefix). Set ENABLE. Press START. The paper tape will automatically read in.
9	Set the HW or SW switch register to control the diagnostic execution.
10	Load address 000200. Set ENABLE. Press START. The diagnostic will begin execution. Refer to the diagnostic summary or listing on microfiche. NOTE To load subsequent diagnostics repeat steps 7 through 10. This assumes the current diagnostic has not overwritten the absolute loader.

Procedure 4 RP11-RP03 Disk Pack Manual Bootstrap Procedure

Step	Procedure
1	Halt the CPU.
2	Mount the disk pack on drive 0. Make the drive ready and on-line.
3	Deposit: Address Data 001 000 012 705 001 002 176 716 001 004 012 715 001 006 177 400 001 010 012 745 001 012 000 005 001 014 105 715 001 016 100 376 001 020 005 007
4	Load address 001000. Set ENABLE. Press START. The monitor will automatically load and start. Refer to the XXDP Summary.

Procedure 5 RH11-RP04 Disk Pack Manual Bootstrap Procedure

Step	Procedure	
1	Halt the CPU.	
2	Mount the disk pack on drive 0. Make the drive ready and on-line.	
3	Deposit: Address	Data
	010 000	012 700
	010 002	176 700
	010 004	012 710
	010 006	000 023
	010 010	005 060
	010 012	000 034
	010 014	005 060
	010 016	000 006
	010 020	012 760
	010 022	177 400
	010 024	000 002
	010 026	012 710
	010 030	000 071
	010 032	105 710
	010 034	100 316
	010 036	005 007
4	Load address 010000. Set ENABLE. Press START. The monitor will automatically load and start. Refer to the XXDP Summary.	

Procedure 6
RX11/RXV11-RX01 Floppy Disk Manual Bootstrap Procedure

Step	Procedure	
1	Halt the CPU.	
2	Mount the floppy disk on unit 0. Make the unit ready and on-line.	
3	Deposit: Address	Data
	001 000	005 000
	001 002	012 701
	001 004	177 170
	001 006	105 711
	001 010	001 776
	001 012	012 711
	001 014	000 003
	001 016	005 711
	001 020	001 776
	001 022	100 405
	001 024	105 711
	001 026	100 004
	001 030	116 120
	001 032	000 002
	001 034	000 770
	001 036	000 000
	001 040	005 000
	001 042	000 110
	001 044	000 000
	001 046	000 000
	001 050	000 000
4	Load address 001000. Set ENABLE. Press START. The monitor will automatically load and start. Refer to the XXDP Summary.	

Procedure 7 TC11-TU56 DECtape Manual Bootstrap Procedure

Step	Procedure
1	Halt the CPU.
2	Mount the DECtape on transport 0. Make the transport ready and on-line.
3	Deposit: Address Data 177342 004003 The transport should rewind and the REMOTE indicator should remain lit.
4	Examine the current location.
5	Deposit 000001 in the current locations. The REMOTE indicator should go out.
6	Deposit: Address Data 000216 012737 000220 000005 000222 177342 000224 000777
7	Load address 000216. Set ENABLE. Press START. The monitor will automatically load and start. Refer to the XXDP Summary.

STANDARD ERROR FORMAT
Most errors occur as either halts or messages printed on the console terminal. Error halts should be looked up in the listing.

The standard format for an error message is:

```
MESSAGE

TEST          PC          H/W (optional)
```

Where:

MESSAGE is a description of the type of error.

PC is the address in the diagnostic where the failure was detected.

H/W is the status of the hardware under test when the failure was detected. This is optional.

If an output console is unavailable, the error information will be stored in a location in core. This address may be examined after the processor is halted. The listings and program summary should be consulted for the address of the error information.

GENERAL INFORMATION

Code	COPY.A11
Title	XXDP Copy Utility Program
Abstract	The COPY utility enables the user to duplicate XXDP storage media. The COPY program allows only copying on the same media. The program will not copy anything other than XXDP material. It is not a general-purpose copy program.
Hardware Required	PDP-11 mainframe/8K of memory (minimum)/input and output devices.
Preliminary and Associated Programs	COPY assumes the basic instructions and the input and output devices are operational.
Restrictions	The input and output devices must be of the same type.
Notes	COPY will overwrite the KLDCP area of core.
Loading and Starting Procedure	Via XXDP monitor type: R COPY<CR> Via KLDCP type: P COPY$
Control Switches	None

OPERATIONAL CONTROL
Once started COPY will print

```
DZQUQ-A-COPY-XXDP COPY PROGRAM 21-JUL-78
```

```
DATE:
```

Type the date according to the following format, followed by <CR>. Dashes must be included.

DD-MMM-YY (e.g., 07-AUG-78)

The program echoes back the date and then types:

```
RESTART: NNNNNN       PROGRAMS RESTART ADDRESS.
```

Command Summary
COPY commands are summarized in Table 1.

Table 1 COPY Command Summary

Command	Description	Cross Ref.
COPY	COPY DX0:=DX1:<CR> or COPY DX0:=DX1:/NEW<CR> (out)(in) (out)(in) Copy the entire input medium over to the output medium. The /NEW switch will cause the copy to take place on a file-by-file basis. Verification is automatic.	1
VERIFY	VERIFY DX0:DX1<CR> (out)(in) Verify that the output medium is identical to the input medium.	2
BOOT	BOOT DT0:<CR> This command causes block 0 of the device specified to be read into core and started (i.e., return to the XXDP monitor).	N/A
DIR	DIR DX1:*.All<CR> Print the directory of the specified device on the console terminal. Universal "field and character" substitutes (*,?) are permitted.	N/A
DIRLP	DIRLP DT1:*.BI?<CR> Print the directory of the specified device on the line printer. Universal "field and character" substitutes (*,?) are permitted.	N/A
FILL	FILL<CR> Set the console fill count.	N/A

Command Descriptions
The following is a detailed description of the commands listed in Table 1.

1. COPY DX0:=DX1:<CR> or COPY DX0:-DX1:/NEW<CR>
   Copy is the basic command to copy XXDP software. The source and destination must be on the same medium.

   *MAKE OUTPUT READY, TYPE<CR> WHEN READY.

   This is to inform the user that the output device must be powered up, ready and write-enabled. When all these requirements are met, type <CR> to start the copy process.

   When the copy is completed a verification pass is made. This pass is started when the program types:

   *STARTING VERIFICATION.

   When the verify pass is complete the program types:

   VERIFY COMPLETE, COPY COMPLETE.

   When the optional /NEW switch is used, the COPY program will copy disks or DECtapes on a file-by-file basis rather than on a block- by-block basis. This feature is useful when it is known that a master disk contains one or more bad blocks which are not being used in the files contained in the disk, and the user wishes to copy all files to another disk.

2. VERIFY DT0:DT1<CR>
   The verify command will only do a verification of an XXDP medium. The program then types:

   *STARTING VERIFICATION.

   The verification has now begun.

   When the verification is complete the program types:

   *VERIFICATION COMPLETE.

ERROR SUMMARY
The following is a list of COPY error messages and their meanings.

NEXFIL	File not found.
DEVERR	Device error, check for READY, ON LINE etc.
DEVFUL	Device full, no more room for files.
INVCMD	Invalid command, check last command string.
INVNAM	Invalid name, for file or command.
INVDEV	Invalid device, check device table.
INVADR	Invalid address, address should be even.
CKSMER	Load (CHECKSUM) error.
EOM	End of medium error, reached end of medium before end of file.
DELOLD	Delete old file first.
DELERR	Delete error.
INVCOR	Core error.
INVSW	Invalid switch.
POFLOW	Program overflow error, not enough core.

GENERAL INFORMATION

Code	XTECO.A11
Title	Text Editor
Abstract	The XTECO text editor program enables the user to create and edit ASCII text files. All editing can be done by using a few simple commands. XTECO is a character-oriented editor. One or more characters in a line can be modified without retyping the rest of the line. XTECO does not require that line numbers or other extraneous information be associated with the ASCII text. XTECO operates on ASCII data files. A file is an ordered set of data on some peripheral device. In the case of XTECO, a data file is some type of document. An input file may be a named file on any directory device (disk, magtape, DECtape, cassette). An output file can be written onto any of the same devices. The input file for a given editing operation is the file to which the user wishes to make changes. If the user is using XTECO to create a new file, there is no input file. The output file is either the newly created file, or the edited version of the input file. In general, the editing process proceeds as follows. The user specifies the file he wishes to edit, and then a block of text is read into core. The user modifies the text by using the various editing commands. He then appends additional blocks of text and edits them until the entire file has been edited, at which point he outputs the edited file and closes it.
Hardware Required	PDP-11 mainframe/8K of core (minimum)/file input and storage device (DECtape, RX11, etc.)
Preliminary and Associated Programs	The editor assumes that the basic instructions and the selected input and output devices are operational
Restrictions	None
Notes	None
Loading and Starting Procedure	Via XXDP monitor type: R XTECO<CR> Via KLDCP type: P XTECO$ **NOTE** **XTECO overwrites KLDCP, therefore KLDCP must be reloaded after running XTECO.**
Control Switches	None

OPERATIONAL CONTROL
After XTECO is started it will print the following message:

```
    DZQUG-E XTECO - XXDP TEXT EDITOR
    DATE:
```

Type the date according to the following format, followed by <CR>. Dashes must be included.

```
    DD-MMM-YR e.g., 28-JUL-78
```

The program echoes back the date and then types:

```
    RESTART: 005730
```

Command Summary
XTECO supports two types of commands.

Table 2 Lists and describes nonedit-type commands.
Table 3 Lists and describes edit commands.

Since XTECO is a character-oriented editor, it is very important that the user understand the concept of the buffer pointer. The position of the buffer pointer determines the effect of many of the editing commands. For example, insertion and deletion always takes place at the current position of the buffer pointer.

The buffer is the current text contents in core, from the first character, up to and including the last character.

The buffer pointer is simply a movable position indicator. It is always positioned between two characters in the buffer, or before the first character in the buffer, or after the last character in the buffer. The pointer may be moved forward or backward over any number of characters.

XTECO commands may be given one at a time. However, it is usually more convenient to type in a single command string, several commands that form a logical group. An example of a command string is shown below.

```
*IHEADING$NTAG:$2LT$$
```

This command string inserts the word "heading", searches for string "tag:", moves pointer forward 2 lines and types line pointed to.

A command string is typed after XTECO indicates its readiness by printing an asterisk. Command strings are formed by merely typing one command after another, and are terminated by typing two consecutive altmodes.

Execution of the command string begins only after the double altmode has been typed. At that point, each command in the string is executed in turn, starting at the left. When all commands have been executed, XTECO prints another asterisk, indicating readiness to accept another command.

If some command in the string cannot be executed because of a command error, execution of the command string stops at that point, and an error message is printed. Commands preceding the bad command are executed. The bad command and those following it are not executed.

Table 1 XTECO Special Characters

Character	Description	Cross Ref.
*	The asterisk is the prompt for edit mode.	N/A
<CR>	A carriage return is used to terminate all nonedit commands.	N/A
<ALT>	A single altmode echoes as a $ and is used to terminate a edit command.	N/A
<ALT><ALT>	A double altmode echoes as $$ and causes execution of the edit command or command string.	N/A
↑C	Control C is used to exit out of any command mode. It will cause an open output file to be closed. The user must be careful not to type ↑C unless he wishes to abort his operation. This is especially important when editing a file, as all work would be lost.	N/A
↑O	Control O is used to stop printing on the console terminal, as when typing multiple lines of text when editing a file.	N/A

Table 1 XTECO Special Characters (Cont)

Character	Description	Cross Ref.
↑U	Control U is used to empty out contents of keyboard buffer, as when the user wishes to start typing his command sequence all over again.	N/A
RUBOUT or DELETE	The RUBOUT or DELETE key is used to remove one or more characters typed from command or text string. One depression of the RUBOUT key removes one character.	N/A

Table 2 XTECO Nonedit Command Summary

Command	Description	Cross Ref.
EDIT	EDIT DK0:file.ext-DK1:file.ext<CR> (output) (input) EDIT is a general-purpose command for editing an existing text file. It permits the user to edit an input file in one type of device and to output the edited file to a different type device. All editing commands are available when under the EDIT command.	N/A
TECO	TECO DX0:file.ext<CR> The TECO command is a specialized version of the EDIT command. Under the TECO command the input and output device/drive must be the same, and must be random access type devices (disk, DECtape). The edited output file takes its name from the name and extension of the input file, and the input file is renamed to a .BAK extension (for backup). All editing commands are available.	N/A
TEXT	TEXT DX0:file.ext<CR> The TEXT command is used when the user wishes to create a new text file. The TEXT command does not require an input file, only an output file. All editing commands are available with the exception of the A (APPEND) command which becomes a no-op command when no input file exists.	N/A
BOOT	BOOT DT0:<CR> This command causes block 0 of the device specified to be read into core and started (i.e., restart the XXDP monitor).	N/A
DELETE	DELETE DK0:file.ext<CR> This command causes the file specified to be deleted from the directory.	N/A
DIR	DIR DT0:<CR> Print a directory of the specified device on the console terminal. Universal "field and character" substitutes (*,?) are permitted.	N/A
FILL	FILL<CR> Set the console fill count	N/A
PRINT	PRINT RX0:file.ext<CR> Print the file specified on the line printer. Universal "field and character" substitutes (*,?) are permitted.	N/A
RENAME	RENAME dev:file.ext-dev:file.ext<CR> (new) (old) Rename the old file. The devices must be the same. This command is not allowed on magtape or cassette.	N/A
TYPE	TYPE dev:file.ext<CR> Type the file specified on the console terminal. Universal "field and character" substitutes (*,?) are permitted.	N/A

Table 3 XTECO Edit Command Summary

Command	Description	Cross Ref.
C	C$ or 9C$ or -10C$ Move the pointer forward or back one or more characters.	1
L	L$ or 0L$ or 6L$ or -9L$ Move the pointer forward or back one or more lines.	2
J	J$ Move the pointer to the beginning of the buffer, immediately before the first character in the buffer.	N/A
ZJ	ZJ$ Move the pointer to the end of the buffer, just after the last character.	
A	A$$ or 3A$$ Append one or more text blocks to the end of the buffer.	3
D	D$ or 5D$ Delete one or more characters.	4
I	I ASCII TEXT$ Insert ASCII TEXT into buffer.	5
K	K$ or 8K$ Delete (KILL) one or more text lines.	6
T	T$ or 16T$ Type one or more text lines beginning at the pointer.	7
S	S ASCII TEXT STRING$ Search the buffer for the specified character string.	8
N	N ASCII TEXT STRING$ Search the buffer and the remainder of the input file for the specified character string.	9
EX	EX$$ Output edited file to output device and close output.	10

Command Descriptions
The following is a detailed description of the commands listed in Table 3.

1. C$ or 5C$ or -10C$ - The C command moves the pointer one character in the buffer. The C command may be preceded by a (decimal) numeric argument. The command nC moves the pointer forward over n characters. The command -nC moves the pointer backward over n characters. (The pointer cannot be advanced beyond the ends of the buffer.)

2. L$ or 0L$ or 6L$ or -9L$ - The L command is used to advance the buffer pointer or move it backward, on a line-by-line basis. The L command takes a numeric argument, which may be positive, negative, or 0, and is understood to be 1 if omitted.

Suppose the buffer pointer is positioned at the beginning of line B or at some position within line B.

The command L or 1L advances the pointer to the beginning of line B+1.

The command nL, where n>0, advances the pointer to the beginning of line B+n.

The command -0L moves the pointer to the beginning of line B. If the pointer is already at the beginning, nothing happens.

The command -L or -1L moves the pointer back to the beginning of line B-1.

The command -nL moves the pointer back to the beginning of line B-n.

3. A$ or 3A$$ - The A command reads in the next block of text from the input device and adds it to the contents of the text buffer in core.

The A command accepts numeric arguments. Example: 3A$$. However, it does not execute any other commands following it in the command string. It is meant to be used singly in a command string. When not enough core is available to satisfy an A command, XTECO outputs part of the text buffer onto the output device until the requirements of the A command are satisfied.

NOTE

Execution of the A (append) command does not change the position of the buffer pointer.

4. D$ or 5D$ - Individual characters are deleted by using the D command. The command D deletes the character immediately following the buffer pointer. The command nD, where n>0 deletes the n characters immediately following the pointer.

5. IASCII TEXT$ - The only insertion command is the I command. The ASCII text that is to be inserted into the buffer is typed immediately after the letter I. The text to be inserted is terminated by an altmode.

Any ASCII character except NULL, ALTMODE, RUBOUT, ↑C, ↑O, and ↑U may be included in the text to be inserted.

If a carriage return is typed in an insertion, it is automatically followed by a line feed. The text to be inserted is placed in the buffer at the position of the buffer pointer; i.e., between the characters. At the conclusion of the insertion command the buffer pointer is positioned at the end of the insertion.

Any number of lines may be inserted with a single I command. However, it is recommended that no more than 10 to 20 lines should be inserted with each I command.

6. K$ or 8K$ - Lines are deleted by using the K command. The K command may be preceded by a numeric argument, which is understood to be a 1 if omitted. The command nK (n>0) deletes everything from the current position of the buffer pointer through the nth line feed character following the pointer.

7. T$ or 16T$ - Various parts of the text in the buffer can be typed out for examination by use of the T command. Just what is typed out depends on the position of the buffer pointer and the argument given. The T command never moves the buffer pointer.

The T command types out everything from the buffer pointer through the next line feed. Thus, if the pointer is at the beginning of a line, the T command causes that line to be typed out. If the pointer is in the middle of a line, T causes the portion of the line following the pointer to be typed.

The command nT (n>0) is used to type out n lines; i.e., everything from the buffer pointer through the nth line feed following it.

The user, especially one new to XTECO, should use the T command often, to make sure the buffer pointer is where he thinks it is.

During execution of any T command, the user may stop the terminal output by typing the ↑O (control O) character. The typeout stops and execution of the remainder of the command string is aborted. Therefore, lengthy typeouts should be restricted to single command, command strings.

8. SASCII TEST STRING$ - The S command is used to search for a character string within the buffer. The string to be searched for is specified as an alphanumeric argument following the S command. This argument must be terminated by an altmode.

Execution of the S command begins at the position of the buffer pointer and continues to the end of the buffer. If the specified string is not found, an error message is printed and the buffer pointer is set to the location where the search began.

9. NASCII TEST STRING$ - The N command is similar to the S command. The difference is that an S command ends at the end of the buffer, whereas the N command does not. An N search begins like an S search, but if the character string is not found in the current buffer, an automatic A (append) command is executed and the search continued until the search is successful or the input file exhausted.

If the N command finds the specified string, the pointer is positioned at the end of the string found. If the string is not found, an error message is printed and the pointer is set at the beginning of the buffer. Since a good part of the file may already have been output to the output device, the user may have no other choice than to exit via the EX: command, and to reopen the file and try the N search again with a character string that can be found.

NOTE

When attempting to search it is very easy to overlook an occurrence of the search string preceding the one the user desires. For example, he may want to move the pointer after the word "and" but erroneously position the pointer after a preceding occurrence of a word like "thousand." For this reason, the user is strongly urged to execute a T command to ascertain the position of the pointer after each search command.

10. EX$$ - The only output command available with XTECO is the EX (EXIT) command. The EX command is used to conclude an editing job with a minimum of effort. XTECO will rapidly move all of the rest of the input file to the output file, close the file, and return to command mode so that the user may give other nonedit-mode commands.

ERROR SUMMARY

The following is a list of possible XTECO error messages.

CKSMER	Checksum error during LOAD command.
DELERR	Bit map error during delete operation on DECtape or disk. Not usual unless medium has been wiped out. Transfer files to other medium.
DELOLD	Delete old file before giving command that would create file with same name.
DEVERR	Device error on either input or output device. Check that output device is write-enable.
DEVFUL	Device full. Applies to DECtape and disk. No more file storage available. Delete unnecessary files and try again, or use another medium.
DIRERR	Invalid name in device directory.
DUMP ERROR	ACT mode only. Occurs during DUMP command when data dumped on output device does not match data in core.
EOM	End of medium. Occurs during input operations when the program attempts to input and the file is at an end. Serious problem. File in storage is probably wiped out.
INVADR	Invalid address. Must be even, within existing LOCORE and HICORE limits, and must not be within UPDATE program.
INVCMD	Invalid command. Check command just given.
INVCOR	High core limit lower than lower core limit. Correct core limits. Occurs during DUMP command.
INVDEV	Invalid device specified for command given.
INVNAM	Invalid file name. No special characters allowed. A through Z, and 0 through 9 are the only valid characters. Also occurs if * or wild character construction filenames are specified to a command that does not allow them.
INVSW	Invalid switch specified in command string.
NEXFIL	Nonexistent file. File does not exist in device directory.
NOTRDY	Paper tape device is not ready. Make it ready.
POFLOW	Program too large to load within existing core space.

GENERAL INFORMATION

Code	XXDP.BIN
Title	11-Based 11 Diagnostic Monitor
Abstract	The XXDP monitor is the preferred method of loading and running 11-based 11 utility and diagnostic programs. The XXDP monitor can be directed to execute a command file containing a list of programs to be run, or it can be directed to execute single programs directly from the user's terminal.
Hardware Required	PDP-11 mainframe/4K of core (minimum)
Preliminary and Associated Programs	The monitor assumes the basic instructions and load device are operational.
Restriction	None

Notes

1. When running dignostics that test the XXDP medium, care must be taken to ensure the medium is not accidentally destroyed. The best method for doing this is to either write-protect or remove the medium.

2. When running individual programs directly from the terminal, the monitor must be manually restarted between each of the programs.

3. XXDP is the monitor designation where:

   RBDP is the RH11-RP04 monitor
   RKDP is the RK11 monitor
   RMDP is the RK06 monitor
   RPDP is the RP11 monitor
   RSDP is the RH11-RS04 monitor
   RXDP is the RX11 monitor
   TADP is the TA11 monitor
   TCDP is the TC11 monitor
   TMDP is the TM11-TM02 monitor

Loading and Starting Procedure	Standard (Refer to the 11/11 STD module.)
Control Switches	None

OPERATIONAL CONTROL
After the diagnostic monitor is started, it will print the following typical message:

```
    DZQUC-E 21 JULY-78 TCDP-TC11 MONITOR nnK
    RESTART ADDR: xxxxxx
    BOOTED VIA UNIT#:0
```

This message is followed by a short help file. A period (.) indicates the monitor is ready to accept commands.

COMMAND SUMMARY
All XXDP monitors use the same command set. These commands are summarized in Table 1.

Table 1 XXDP Diagnostic Monitor Command Summary

Command	Description	Cross Ref.
C	C file<CR> Execute command file specified by file.ext Switches: C file/QV<CR> Execute the command file in quick-verify mode.	1
D	D<CR> Print directory of load medium on console terminal Switches: D/F<CR> Print abbreviated directory D/L<CR> Print directory on line printer	2
E0	E0<CR> Enable drive 0 (TADP monitor only)	3
E1	E1<CR> Enable drive 1 (TADP monitor only)	3
F	F<CR> Set console fill count	4
L	L file.ext<CR> Load program specified by file.ext	5
R	R file.ext Load and start program specified by file.ext Switches: R file.ext/#<CR> Run the program the number of passes specified by #.	6
S	S<CR> Start program loaded via the L command	7
	S addr<CR> Restart the processor at the address specified by addr.	8

Command Description
This section describes in detail each of the commands summarized in Table 1.

1. C CPU<CR> - The C command reads into core and executes the specified command file (CPU.CCC). The use of a command file permits a series of programs to be run without operator intervention.

Use of the /QV switch will cause each program in the command file to be executed <u>once</u> regardless of the pass count specified in the file.

NOTE

1. The command file must have a .CCC file extension; however, the extension should not be included in the command format.

2. To prematurely terminate the execution of a command file, repeatedly type ↑C (control C) characters.

3. Information about constructing a command file follows this section.

2. D<CR> - The D command reads and prints the file directory of the load medium. The following switches may be used to modify the D command.

/F will cause the directory to be printed in an abbreviated form.

/L will cause the directory to be printed on the line printer. A line printer must be on the system. No check is made for it.

/L/F - These switches can be used in combination to print an abbreviated directory on the line printer.

The directory is printed in the following format:

```
12-JAN-76
ENTRY:               FILNAM.EXT      DATE          LENGTH      START
000001               1               2-AUG-78      14          000105
000002               2               2-AUG-78      12C         000172
000003               3               2-AUG-78      12C         000206
000004               5               2-AUG-78      12C         000222
FREE FILES: 444
```

DATE: the date the file was created

LENGTH: the number of blocks occupied by the file

C: indicates the files are stored contiguously

START: the octal address of the first block in the file. Note that this is not the starting address of the program.

NOTE

1. The D command accepts universal filename, extension and character substitutes.
2. If there is no line printer on the system the /L switch will cause a trap condition.

3. E0<CR> and E1<CR> - These commands only pertain to the TADP version of the XXDP monitor. They are used to select for input drive 0 and drive 1 respectively.

4. F<CR> - The fill command allows the number of fill characters following a carriage return to be changed. Fill characters prevent overprinting due to the mechanical delay of the carriage return mechanism. The default value (14) is set up for LA30s. For other terminals this value may be both time-consuming and annoying. To change the fill count, type F<CR>. The program will respond by printing the current filler count value. Type the desired value followed by a carriage return.

Example:

```
F<CR>

000014 1
```

The 000014 is typed by the program and indicates the current filler count. The 1 indicates the user typed a filler count of 1.

5. L DZLPB.BIC<CR> - The L command loads but does not start the specified program (DZLPB.BIC). This load-only command allows the user to set switches or insert code changes before starting the program.

6. R DZLPB<CR> - The R command loads and starts the specified program (DZLPB). A file extension is not necessary for BIC and BIN formatted programs. The program specified must be of the self-starting type.

If the R command was terminated with an altmode ($) the program will be automatically started at location 200 (octal).

7. S <CR> - The S command starts the program loaded at that program's starting address (usually 200 octal).

8. S addr<CR> - This form of the S command starts the processor at the specified octal address (addr). It can be used for special program starts or to restart the XXDP monitor.

Command File Construction

A command file is an ASCII file created with an editor program (XTECO) and used to control the XXDP monitor. A command file should be used for running programs during preventive maintenance. This will allow the field engineer to make other routine checks and thus shorten the preventive maintenance period.

During corrective maintenance a command file can be used to run a set of functional and reliability diagnostics while the field engineer performs a visual/mechanical inspection of the system.

The following are rules for constructing a command file.

1. A command file may contain any commands or switches supported by the XXDP monitor. The C command, however, will cause a new command file to be read in on top of existing command file. For this reason the C command should be limited to the last command in the command file.

2. Comments may be incorporated into a command file but they must be preceded by a semicolon (;).

3. All programs to be executed by a chain file _must_ have a .BIC extension. The BIC extension, however, should _not_ be included in the command string. The directory command is an exception to this rule.

4. The command file itself must have a CCC extension.

5. All programs and the command file itself must be located on the same physical medium.

Following is an example of a typical command file.

```
;CPU.CC
;THIS COMMAND FILE EXERCISES THE XYZ PROCESSOR WITH T1-T13..
;
R DOAA/1000        ;RUN T1 1000 TIMES<CR>
R DOBA/1000        ;RUN T2 1000 TIMES<CR>
R DOCA/1000        ;RUN T3 1000 TIMES<CR>
R DODA/1000        ;RUN T4 1000 TIMES<CR>
R DOEA/1000        ;RUN T5 1000 TIMES<CR>
R DOFA/1000        ;RUN T6 1000 TIMES<CR>
R DOGA/1000        ;RUN T7 1000 TIMES<CR>
R DOHA/1000        ;RUN T8 1000 TIMES<CR>
R DOJA/1000        ;RUN T9 1000 TIMES<CR>
R DOKA/1000        ;RUN T10 1000 TIMES<CR>
R DOLA/1000        ;RUN T11 1000 TIMES<CR>
R DOMA/1000        ;RUN T12 1000 TIMES<CR>
L DONA             ;LOAD T13<CR>
S/1000<CR>         ;START IT, RUN 1000 TIMES<CR>
C CPU              ;RESUBMIT COMMAND FILE AGAIN.
```

XXDP Monitor Error Summary
Monitor errors are summarized in Table 2.

Table 2 XXDP Monitor Error Summary

Error Messages	Description
CKSMER	Checksum error during LOAD command.
DEVERR	Device error on input device.
EOM	End of medium. Occurs during input operations when the program attempts to input and the file is at an end. Serious problem. File in storage is probably wiped out.
INVADR	Invalid address. Must be even within existing LOCORE and HICORE limits, and must not be within update program.
INVCMD/SW	Invalid command and/or switch. Check command.
INVNAM	Invalid character typed for file name.
NEXFIL	Nonexistent file. If using a command file the extension program to be run does not have .BIC

DXLPB

GENERAL INFORMATION

Code	DXLPB.BIN
Title	LP20 Line Printer Diagnostic
Abstract	This diagnostic is designed to test the LP20(s) and line printer(s) connected to the PDP-11/40 console front-end subsystem.
Hardware Required	PDP-11/40 mainframe/16K of memory (minimum)/up to 2 LP20s/and any of the following line printers: LP05, LP07, LP10, and LP14.
Preliminary and Associated Programs	Refer to the diagnostic hierarchy (11/11 STD module).
Restrictions	LP05 and LP07 line printers should be set to 6 lines per inch.
Notes	1. Set control switch 0 (1) to test the line printer; otherwise only the LP20 will be tested. 2. Set control switch 1 (1) to shorten DAVFU testing and save paper. 3. Execution time is approximately 10 seconds without a line printer and 1.5 minutes with a line printer. 4. Hammer Alignment Test - Refer to Table 3, Test 144.
Loading and Starting Procedure	This program is designed to run under the XXDP diagnostic monitor (Refer to the XXDP Summary module). Starting Addresses: 200 - Standard; 204 - Restart diagnostic without reconfiguring; 210 - Perform manual intervention tests
Control Switches	Refer to Table 1

OPERATIONAL CONTROL

In addition to the control switches, DXLPB supports the following operator-selectable service routines.

Starting Address	Routine Name	Description
214	ORAMST	Set Up Core RAM Buffer
220	ODISLR	Discrete Load RAM
224	ODMALR	DMA Load RAM
230	ODMPRM	Dump RAM to Core
234	OCHRLD	Load Core Print Buffer from TTY
240	OTSTPT	Print in Test Mode
244	OLPTPT	Print on LPT
250	OVFULD	Load VFU
254	OPGTST	Test Page Counter

Refer to the listing on microfiche for a detailed description of these routines.

DXLPB TEST SUMMARY

DXLPB consists of two sets of tests. The first set tests the LP20(s) only. (Refer to Table 2.) The second set tests the line printer(s) and is only run when control switch 0 is set (1). Refer to Table 3.

ERROR MESSAGE SUMMARY

All error typeouts are in a format similar to the following.

```
ERROR  CSRA    CSRB    REG      GOOD     ACTUAL
PC     STATUS  STATUS  ADDRESS  DATA     DATA
12340  000000  000000  175400   0000200  000000
```

Checksum Errors
On all DMA transfers, the checksum will be computed and written at location OGDSUM. The checksum as read from the LP20 is written at location OLPSUM.

Premature DONE Errors
If DONE occurs abnormally during a DMA transfer and the program is not looping, one of the following messages will be printed.

E	followed by three digits indicates an error. The digits correspond to the low-order byte of the CSRB register.
P	indicates the paper counter equals 0.
U	indicates an undefined character was typed.

After any of these messages, type one of the following keys.

$	(Altmode) Exits
<CR>	(Carriage return) Prints the contents of the registers
SPACE	Resets the error, sets GO and continues to wait for DONE

Table 1 DXLPB Control Switch Summary

Switch	State	Description
15	1	Halt on error.
14	1	Loop on test.
13	1	Inhibit error printouts.
12	1	Inhibit END OF PASS message.
11	1	Inhibit iterations.
10	1	Ring bell on error.
09	1	Loop on error.
08	1	Loop on test specified in switches <00:07>.
<07:00>		Specifies the test to loop on. See switch 8.
01	1	Shorten DAVFU testing. Switch 08 must be reset (0).
00	1	Test line printer after testing LP20. Switch 08 must be reset (0).

Table 2 LP20 (Only) Test Summary

Test	Description
1	REGISTER ACCESS TEST
2	(LPCSRA) REGISTER R/W TEST
3	(LPCSRA) HI-BYTE TEST
4	(LPCSRA) LO-BYTE TEST
5	(LPCSRA) CLEAR TEST (LOINIT)
6	(LPCSRA) CLEAR TEST (RESET)
7	(LPCSRB) REGISTER R/W TEST
10	(LPCSRB) HI-BYTE TEST
11	(LPCSRB) LO-BYTE TEST
12	(LPCSRB) CLEAR TEST (LOINIT)
13	(LPCSRB) CLEAR TEST (RESET)
14	(LPBSAD) REGISTER R/W TEST

Table 2 LP20 (Only) Test Summary (Cont)

Test	Description
15	(LPBSAD) CLEAR TEST (LOINIT)
16	(LPBSAD) CLEAR TEST (RESET)
17	(LPCTR) REGISTER R/W TEST
20	(LPCTR) CLEAR TEST (LOINIT)
21	(LPCTR) CLEAR TEST (RESET)
22	(LPPCTR) REGISTER R/W TEST
23	(LPPCTR) CLEAR TEST (LOINIT)
24	(LPPCTR) CLEAR TEST (RESET)
25	(LPRAMD) REGISTER R/W TEST
26	(LPCCTR) REGISTER R/W TEST
27	(LPCCTR) CLEAR TEST (LOINIT)
30	(LPCCTR) CLEAR TEST (RESET)
31	(LPCCTR) LO-BYTE ADDRESSING TEST
32	(LPCCTR) HI-BYTE ADDRESSING TEST
33	(LPTDAT) HI-BYTE ADDRESSING TEST
34	(LPTDAT) LO-BYTE TEST
35	(LPCSRA) DISTURB TEST
36	(LPCSRB) DISTURB TEST
37	(LPBSAD) DISTURB TEST
40	(LPCTR) DISTURB TEST
41	(LPPCTR) DISTURB TEST
42	(LPRAMD) DISTURB TEST
43	(LPCCTR) DISTURB TEST
44	(LPTDAT) DISTURB TEST
45	(RAM) BASIC ADDRESS TEST
46	(RAM) BASIC PARITY NETWORK TEST PR=1
47	(RAM) BASIC PARITY NETWORK TEST PR=0
50	(RAM) BASIC COMPLEMENT ADDRESS TEST
51	(RAM) FLOATING 1S and 0S ADDRESS TEST
52	(RAM) ADVANCED ADDRESS TEST
53	(RAM) FLOATING 1S and 0S ADDRESS TEST
54	(RAM) BASIC DATA TEST
55	(RAM) ADVANCED DATA TEST
56	(RAM) PARITY GENERATOR NETWORK
57	(RAM) PARITY GENERATOR NETWORK (RAMTST) MODE
60	(RAM) DISABLE LOAD TEST
61	(RAM) DISABLE READ TEST
62	(LPPCTR) DECREMENT TEST
63	(LPPCTR) DECREMENT CARRY TEST
64	(LPPCTR) DECREMENT TEST (GOERR)

Table 2 LP20 (Only) Test Summary (Cont)

Test	Description
65	(LPPCTR) DECREMENT TEST (PAGTST)
66	(RAM) LOAD VIA DMA (DONE TEST)
67	(RAM) LOAD VIA DMA DATA TEST
70	SYNTST MASTER SYNC (SYNTIM)
71	SYNTST MASTER SYNC (ERROR)
72	DEMTST TEST (DEMTIM)
73	DEMTST TEST CLEAR (LOINIT)
74	DEMTST TEST (ERROR)
75	MEMTST TEST (MEMPAR)
76	MEMTST TEST CLEAR (LOINIT)
77	MEMTST TEST (ERROR)
100	GO ERROR TEST SET (GO ERR)
101	GO ERR OR CLEAR FAILED (GO ERR)
102	INTERRUPT TEST
103	LPC9 SEL BYT CTR (CLEAR DONE)
104	LPR2 BYTE CNTR ZERO (SET DONE)
105	LPD4 PAGE CNTR EMPTY L (SET PAGE ZERO)
106	LP8C CLEAR 1 L (CLEAR PAGE ZERO)
107	LPC9 SEL PAG CTR L (CLEAR PAGE ZERO)
110	LP20CK CHECKSUM LOGIC TEST
111	LP20CK CHECKSUM LOGIC TEST
112	LP20CK CHECKSUM LOGIC TEST
113	DELIMITER HOLD TEST
114	RAM STATIC TRANSLATE DATA TEST
115	RAM INT AND TRANS COMBINATION STATIC TEST This test ensures that if the RAM INT and TRANS bits are up, the character will be sent straight to the printer without translation.
116	RAM INT AND TRANS COMBINATION INTERRUPT TEST This test ensures that if the INT and TRANS bits are on for a character, an illegal character interrupt will only occur if the DELIM HOLD flip-flop is set. The BYTCNT indicates the error type. BYTCNT ERROR -4 Transfer never started (GO error, etc.). -3 Illegal character INT occurred with DELIM HOLD not set. -2 INT occurred on the delimiter (should not happen). -1 Correct termination. 0 Illegal character INT did not occur at all.
117	COLUMN COUNTER INIT TEST This test loads 177400 into the LPCCTR register and does a LOCAL INIT. The high byte (column counter) should be clear.
120	COLUMN COUNTER INCREMENT AND LINE FEED TEST This test does a LOCAL INIT and increments the column counter from 0 through 204 character-by-character by doing transfers in test mode, each time checking the column counter. Then a line feed is done and the column counter should clear.

Table 2 LP20 (Only) Test Summary (Cont)

Test	Description
121	COLUMN COUNTER INCREMENT AND FORM FEED TEST This test increments the column counter to 177, then does a form feed in test mode. The column counter should be clear after the form feed.
122	LINE OVERFLOW TEST This test transfers 133 decimal characters in test mode and expects to see the column counter byte be 001 (8 bits). The test then transfers the character 002 with the RAM PI and TRANS bits on. The column counter should reset.
123	TAB AND LINE OVERFLOW TEST This test first transfers one "printing" character followed by 15 tabs. The column counter is checked after each tab.
	This test completes the LP20 controller-only testing. Set switch 0 to 1 to test the line printer.

Table 3 Line Printer (Only) Test Summary

Test	Description
124	VFU LOAD TEST This test will only be run on printers with loadable VFUs.
125	BASIC FORM FEED AND PAGE COUNTER TEST
126	PAGE COUNTER OVERFLOW TEST
127	BASIC LINE FEED TEST This test exercises the "SLEW n LINES" commands of the VFU. The SLEW command is done once per printed page, surrounded by text describing the command done. All 16 commands are exercised unless switch 1 is set in which case just one command is done to save time and paper.
130	DAVFU CHANNEL SLEW TEST This test exercises the DAVFU channel slew commands. Each command is done enough times to take up one page. The command is done just before each text printing "THAT WAS CHANNEL n". This test exercises only one channel if switch 1 is set.
131	DAVFU SPECIAL LOAD OF 143. Partitions This test will be skipped if the printer has an optical VFU.
132	DATA TRANSFER PATHS TEST This test checks the data transfer paths (with alternating 1s and 0s), from the LP20, through the line printer input register, and into the printer's buffer. An alternating string of "*" and "U" characters ("NOT EQUAL" and "DOWN ARROW" for the APL charaband) are transmitted to the printer.
133	ALTERNATE LINES ALL CHARACTERS AND ALL ILLEGAL 7-BIT CHARACTERS This test is designed primarily to test the line printer character generator and comparator logic, and its ability to detect and act upon both printable and illegal (nonprinting) characters. NO LOWER CASE: 0- 37 ILLEGAL; 40-137 LEGAL; 140-177 ILLEGAL LOWER CASE: 0-37 ILLEGAL APL CHARABAND: 0- 37 ILLEGAL; 40-177 LEGAL; 240-267 LEGAL
134	OVERPRINT TEST
135	MULTIPLE LINE ADVANCE TEST This test checks the multiple line advance of the line printer. A line of numbers is printed, then the paper is advanced that number of lines by a line feed. Thus the number printed will indicate the number of blank lines following that line. The number varies between 2 and 7.

Table 3 Line Printer (Only) Test Summary (Cont)

Test	Description
136	DRUM PATTERN CHARACTER TEST
137	SLIDING PATTERNS - LEFT
140	SLIDING PATTERNS - RIGHT

141	SINGLE CHARACTER, ALL COLUMNS TEST
	This test is designed as an endurance test of the line printer as well as a character check of the drum. 132 columns of each of the legal characters are transmitted to the line printer and printed in rotation. A sample of the printout follows.
	`?????----------?????`
	`@@@@@----------@@@@@`
	`AAAAA----------AAAAA`
	`BBBBB----------BBBBB`
	`----------`
	`----------`
	`ZZZZZ----------ZZZZZ`

142	SPURIOUS HAMMER FIRING TEST - RIGHT TRIANGLES This test is designed to detect spurious hammer firings and defective hammer drivers during operation of the line printer. The patterns which are produced are right- and left-side wedges, each composed of 132 lines of print. Any print outside of the wedge will be caused by a hammer misfire or hammer bounce.
143	SPURIOUS HAMMER FIRING TEST - LEFT TRIANGLES Same as Test 142 however with left triangles.
144	HAMMER ALIGNMENT TEST This routine is designed to be used as a driver for manual hammer alignment and intensity adjustments. The test prints a full 132-column line of E characters for 63 lines.
145	NON-LP07 SHUTTLE POSITIONING TEST This test checks the hammer shuttle for correct operation. Full lines of Es are printed by printing a pair of Es at a time, then overprinting those Es to the line until the line is completed. A total of 16 lines are printed during this test. This test is not done on LP07s because LP07s do not allow more than 6 overprints per line.
146	CHARACTER CODE TABLE This test prints a list of all characters (except 000, 001, 012, 014, 015, 212, 214, 215) in a handy table form.
147	LINE OVERFLOW TEST
150	TAB TEST (With Overflow)
Control Tests and Operator Interactive Tests	
151	MANUAL LPT ERROR GENERATION TEST The printer-ready line continuously monitors the following conditions within the printer. Its true state at the control electronics interface is conditional upon none of these conditions existing: A. Paper Out or Torn B. Drum Gate Open C. Ribbon Stall Condition D. Power Supply Fault E. Hammer Bank Fault F. DAVFU Error (if available) G. Switched Off Line.

Table 3 Line Printer (Only) Test Summary (Cont)

Test	Description
	The manual-interactive test sequence which follows is designed to test the proper operation of the ready line as it appears at the interface electronics with respect to the above items that are testable (i.e., A, B, F and G). Initial manual test sequence is as follows. 1. After "POWER ON - TURN ON-LINE" has been displayed, power-up the line printer and turn on line, making sure that the paper is in place in the tractors and that the drum gate is closed. 2. Press CONT. "READY SET OK - TRY TORN PAPER SWITCH" will be displayed if printer is on-line and no errors exist. 3. Tear the paper off below the printer drum gate and use the manual TOP OF FORM switch to drive all the paper out of the printer. Observe that the printer READY light goes out and the paper ERROR light goes on at the printer control panel. Attempt to place the printer on-line. The ON-LINE and READY lights on the printer control panel should remain off. 4. Press CONT. An error message (error count 2) will occur if the printer-ready line remains high at the interface electronics. 5. After successful completion of steps 3 and 4, the message "ERROR SET OK - TURN ON LINE" will be displayed. Restore paper to the tractors, close the drum gate and place the printer in the ready on-line state. Observe that both the ON-LINE and READY label lights illuminate on the printer control panel. 6. Press CONT. An error message (error count 4) will occur if the printer-ready line does not go high at the interface electronics. 7. Drum Gate - After successful completion of steps 5 and 6, the message "READY SET OK - TRY, DRUM GATE SWITCH" will be displayed. Open the printer drum gate and observe that the ON-LINE and READY lights go out and the drum gate error light goes on on the printer control panel. 8. Press CONT. An error message (error count 5) will occur if the printer ready line appears to remain high at the interface electronics. 9. Ready - After successful completion of steps 7 and 8, the message "ERROR SET OK - TURN ON LINE" will be typed. 10. Press CONT. and continue with the next test.
152	PRINT SPEED TIMING TEST This test is designed to time the printer for one full minute. During this time the printer will print the diagonal of the drum pattern so that only two hammers maximum fire at any given instant and maximum print time is used for each line. If a KW11-L or KW11-P is available, it will be used to time the printer. If both are available, the KW11-L will be used. If neither are available, manual timing will be used. When manual timing is used, type instructions on the terminal. To start the timing, place switch 0 in the up position. At the end of one full minute, place switch 0 in the down position to stop the timing. If using an internal clock for timing, place switch 0 in the up position before starting the test. Whichever method of timing is used, at the end of one full minute the approximate print speed will be typed on both the teleprinter and line printer.

Table 3 Line Printer (Only) Test Summary (Cont)

Test	Description
153	DAVFU INCOMPLETE DATA TEST This test attempts to load the direct access vertical format unit (DAVFU) with incomplete data (an odd number of data words) between the start load and stop load commands. This should cause a format error to occur in the line printer. Failure to cause an error in the line printer will cause an error typeout to occur. Then the message "VFU DAT INC TEST DONE - CLEAR AND TURN ON LINE" will be typed. Clear the format error in the printer and place the printer in the read-on-line state. The VFU will then be loaded normally. This test will not be executed on printers with optical VFUs.
154	DAVFU NO 1s IN CHANNEL 1 TEST This test attempts to load the VFU with 1s in all channels except channel 1. Failure to cause an error in the line printer will cause an error typeout to occur. Then the message "VFU 0S IN CHAN 1 TEST DONE - CLR, TURN ON LINE, HIT CONT" will be typed. Clear the format error in the printer and put on-line. The VFU will then be loaded normally. This test is not done on printers with optical VFUs.
155	DAVFU LOAD 144. Partitions This test attempts to load 144 VFU partitions. Failure to cause an error in the line printer will cause an error typeout to occur. Then the message "VFU 144 LOAD TEST DONE - CLR, TURN ON LINE, HIT CONT" will be typed. Clear the format error in the printer and put on-line. The VFU will then be loaded normally. This test is not done on printers with optical VFUs.
156	LP07 LPI SET TEST This test ensures that the lines-per-inch is program-selectable. The program will select both 6 and 8 lines per inch and ask you to check the LPI-indicating LEDs in the printer. One full page is printed at 8 LPI. This test is done only on LP07s.
157	LP07 PARITY TEST This test ensures that the LP07 parity detection logic works correctly. Sending even parity to the LP07 causes an error condition, and lighting of the PARITY indicator. Bit 05 should set in the LPCSRB register.
160	LP07 PARITY INTERRUPT TEST This test checks that a parity error in the LP07 will cause an interrupt. Bit 15 should set in the LPCSRA register.

Table of Contents

Summary	Microfiche Classification	Version
11/10 STD	NA	0.0
DIACON	NA	NA
KLDCP	KL10(DGQDA)	0.17
KLDCPU	KL10(DGQDA)	0.17
MEMCON	KL10(DGQFB)	1.6*
TRACON	KL10(DGQFA) (DHQFA)	0.11/0.3
DGDTE	KL10	0.11*
DGKAA	KL10	0.11
DGKAB	KL10	0.12*
DGKBA	KL10	0.11
DGKBB	KL10	0.11*
DGKBC	KL10	0.10
DGKBD	KL10	0.13
DGKBE	KL10	1.10
DGKCA	KL10	0.10
DGMCA	KL10	1.10
DGMCB	KL10	1.10
DGMMA	KL10	0.3
DGQDA	KL10(KLDCP)	0.17
DGQDD	DN87S	0.1
DGQDE	DL11-E	0.1
DGQDF	DN2X	0.2
DGQDG	DN2X	0.1
DGQEA	DN2X	0.1
DGQFA	KL10(TRACON)	0.11
DGQFB	KL10(MEMCON)	1.6*
DHKAA	KL10	0.2
DHKAB	KL10	0.3*
DHKBA	KL10	0.3
DHKBB	KL10	0.2*
DHKBC	KL10	0.1
DHKBD	KL10	0.5
DHKBF	MF20	0.3
DHKBG	MF20	0.3
DHKCA	KL10	0.1
DHMCA	KL10	1.2*
DHMCB	KL10	0.2*
DHQFA	KL10(TRACON)	0.3

*Program revised, code change only.

PROGRAM IDENTIFICATION CODE
The coding scheme used to identify 11-Based 10 Maintenance Library programs is the same as that used for the 10-Based 10 Maintenance Library. Refer to the 10/10 STD module.

STANDARD PROGRAM STARTING ADDRESSES
Table 1 lists the standard program starting addresses for the 11-Based 10 Maintenance Library programs.

Table 1 Standard Program Starting Addresses

Address	
3000	Standard starting address for all 11-based 10 diagnostics
100000	KLDCP starting address
173000	Boots RSX-20F if KLAD pack is mounted

UTILITY PROGRAMS
Table 2 lists and briefly describes the utility programs associated with the 11-Based 10 Maintenance Library.

Table 2 11-Based 10 Utility Programs

Utility	Description	
KLDCP.BIN	A program which runs in the PDP-11/40 console front-end subsystem; supports console operations, and both 10/10 and 11/10 diagnostics. It also interfaces the CTY to either the TOPS-10 or TOPS-20 operating system during timesharing.	
	STANDARD CONTROL FILES	
	B.CMD	Runs 11/10 and 10/10 mainframe diagnostics on KL10 model PA processors
	BB.CMD	Runs 11/10 and 10/10 mainframe diagnostics on KL10 model PV processors
	BT.CMD	Initializes KL10(PA) mainframes to run 10/10 maintenance programs
	BBT.CMD	Initializes KL10(PV) mainframes to run 10/10 maintenance programs
	DIAGA.RAM	Microcode file for KL10-PA; used by most DG-series diagnostics
	DIAGB.RAM	Microcode file for KL10-PV; used by most DH-series diagnostics
	EBOXA.RAM	Microcode file for KL10-PA; used by DG-series EBox diagnostics
	EBOXB.RAM	Microcode file for KL10-PV; used by DH-series EBox diagnostics
	U.RAM	Microcode file for KL10-PA; used by 10/10 processor functional diagnostics
	UB.RAM	Microcode file for KL10-PV; used by 10/10 processor functional diagnostics
	KLLD.CCL	Loads the SUBRTN package and KLDDT

Table 2 11 Based 10 Utility Programs (Cont)

Utility	Description
	CONFG.CCL Memory configuration file
	X1.CCL External memory 1-way interleave
	X2.CCL External memory 2-way interleave
	X4.CCL External memory 4-way interleave
	CONFG1.CCL Internal memory 1-way interleave
	CONFGR.CCL Internal memory relocation file
	HIMARC.CCL Internal memory set high current margins
	LOMARC.CCL Internal memory set low current margins
	HIMARS.CCL Internal memory set high strobe margins
	LOMARS.CCL Internal memory set low strobe margins
	HIMART.CCL Internal memory set high threshold margins
	LOMART.CCL Internal memory set low threshold margins
	MGNOFF.CCL Internal memory clear margins
	MC.CCL Clear memory
	WRMEM.CCL AC memory test
KLDCPU.A11	A program for copying files and maintaining the KL10 maintenance library.
MEMCON.A11	A program for configuring internal and external memory.
TRACON.A11	A program for tracing and reporting the internal status of the EBox, MBox and channels.
Bootstrap Loaders	
KLADBT.BIN	A bootstrap loader for loading KLDCP from a KLAD-10 pack.
KLDTBT.BIN	A bootstrap loader for loading KLDCP from a DECtape.
KLRXBT.BIN	A bootstrap loader for loading KLDCP from a floppy disk.

STANDARD PROGRAM CONTROL SWITCHES
The switches used to control the operation of 11-based 10 diagnostics are determined by XORing the physical PDP-11/40 console data switches with the switch data entered via the KLDCP ES command.

Table 5 describes the standard effect each switch has on operation of the program. Exceptions to the standard switches are noted in the individual program summaries.

Table 5 Program Control Switches

ES Digit	Switch	Mnemonic	State	Description
1st	15	ABORT	0 1	Normal operation Abort at end of pass
2nd	14	RSTART	0 1	No function Print totals and restart
	13	TOTALS	0 1	No function Print totals and continue
	12	NOPNT	0 1	Normal printout Inhibit all but forced printouts
3rd	11			Not used
	10	DING	0 1	No function Ring TTY bell on error (forced output)
	9	LOOPER	0 1	Proceed to next test Enter scope loop on that error
4th	8	ERSTOP	0 1	No Function Halt on test error. In user mode exit test.
	7	PALERS	0 1	Print only first error in loop Print all errors
	6	RELIAB	0 1	Quick verify mode Reliability mode
5th	5	TXTINH	0 1	Print full error message Inhibit comment portion of error message
	4	INHPAG	0 1	KI10 and KL10 - allow full 256K/4096K addressing KI10 and KL10 - inhibit paging, treat memory as 112K minus 1 max.
	3	MODDVC	0 1	No function Enter dialogue to change device codes
6th	2	INHCSH	0 1	KL10 - allow cache use KL10 - inhibit cache use
	1	OPRSEL	0 1	Use default operations (DIACON) Operator test selections (DIACON)
	0	CHAIN		Reserved - used by DIAMON and other programs to control chain operations

✓ = Run By B String

DIAGNOSTIC HIERARCHIES

Table 3 lists the diagnostic hierarchy for KL10 model PA processors.

Table 4 lists the diagnostic hierarchy for KL10 model PV processors.

Table 3 KL10 (PA) Processor and Memory Diagnostic Hierarchy

Diagnostic	Title
✓ DGDTE.A11	DTE20 Interface Test
✓ DGKAA.A11	EBox Test (Part 1)
✓ DDKAB.A11	EBox Test (Part 2)
✓ DGKBA.A11	MBox Basic Test
✓ DGKBB.A11	MBox Control and Memory Test
NOTE ✓	Memories must be configured from this point on. Use CONFG.CCL control file.
✓ DGKBC.A11	Paging Logic Test
Cache (Optional)	
DGMCA.A11	MBox Cache Option Test (Part 1)
DGMCB.A11	MBox Cache Option Test (Part 2)
Channels (Optional)	
DGKBD.A11	MBox Channel Loopback Test (Part 1)
DGKBE.A11	MBox Channel Loopback Test (Part 2)
✓ DGKCA.A11	Meter Board Test
DGMMA.A11	Memory Reliability and Margining Test

Table 4 KL10 (PV) Processor and Memory Diagnostic Heirarchy

Diagnostic	Title
DGDTE.A11	DTE20 Interface Test
DHKAA.A11	EBox Test (Part 1)
DHKAB.A11	EBox Test (Part 2)
DHKBA.A11	MBox Basic Test
DHKBB.A11	MBox Control and Memory Test
MOS Memory (Optional)	
DHKBF.A11	MF20 Test (Part 1)
DHKBG.A11	MF20 Test (Part 2)
NOTE	Memories must be configured from this point on. Use CONFG.CCL control file
DHKBC.A11	Paging Logic Test
Cache (Optional)	
DHMCA.A11	MBox Cache Option Test (Part 1)
DHMCB.A11	MBox Cache Option Test (Part 2)
Channels (Optional)	
DHKBD.A11	MBox Channel Loopback Test (Part 1)
DGKBE.A11	MBox Channel Loopback Test (Part 2)
DHKCA.A11	Meter Board Test
DGMMA.A11	Memory Reliability and Margining Test

STANDARD LOADING AND STARTING PROCEDURES

Procedure 1 describes how to bootstrap load the console front-end subsystem from a KLAD-10 or KLAD-20 disk pack.

Procedure 2 describes how to load KLDCP from a KLAD pack using RSX-20F.

Procedure 3 describes how to load and start programs using KLDCP.

Procedure 1 Bootstrap Loading the Console Front-End Subsystem

Step	Procedure
1	Mount the program storage medium. Make the unit ready and on line.
2	Set the PDP-11 console switches if they are to be used to direct the bootstrap operation. Refer to Table 6. EXAMPLES: 1. SW = 203 Input device = RP04/5/6. Load RSX-20F, (KL10 down). 2. SW = 207 Input device = RP04/5/6. Run through dialogue to load KLDCP. 3. SW = 003 Boot from floppy. Load RSX-20F, (KL10 down).
3	Press and hold the ENABLE switch.
4	Momentarily press the boot device switch (DECTAPE, FLOPPY, DISK or SWITCHES).
5	Release the ENABLE switch. Either RSX-20F or KLDCP will automatically load and start. Refer to Procedure 2, Procedure 3 or the appropriate program summary for further program loading instructions.
7	Reset the console switches if they were used for the bootstrap operations.

Table 6 PDP-11 Bootstrap Switch Summary

Switch	State	Description
15	1	Infinite retry - The ROM will continue retrying a bootstrap operation on error until aborted by manual intervention.
	0	The ROM will retry on error a finite number of times.
14-11		Selects the line number within the DH11 or DL11 group to be used as the console terminal.
10-8		If bits 6-3 are in the range of 0-2, then bits 10-8 represent the unit number of the bootstrap device selected by bit 7.
		If bits 6-3 are in the range of 3-17, the unit number defaults to 0 and bits 10-8 represent the DH11 unit number of the console terminal.
7	1	Use the RP04/5/6 unit selected as the boot device.
	0	Use as available, one of the RX11 floppies, TC11 DECtapes or DL11 async lines as the boot device. See switches 10-8.

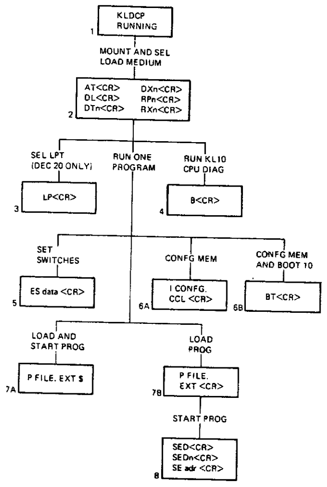
Table 6 PDP-11 Bootstrap Switch Summary (Cont)

Switch		Description
6-3		These bits select the speed and fill class for the console terminal.
	Bits 6-3	**Function**
	0	Ignore bits 14-11. Use bootstrap device built into the software.
	1 or 2	Fill class for DL11 line selected by bits 14-11.
	3-17	ROM defaults to unit 0. Bits 6-3 equal speed and fill class for DH11 line selected by bits 14-11.
2-1		Load Select
	Bits 2-1	**Function**
	0 0	Complete reload of TOPS-10 or TOPS-20 monitor.
	0 1	Load RSX-20F. KL10 is assumed to be down.
	1 0	Load RSX-20F. KL10 is assumed to be up.
	1 1	Run through dialogue questions to control loading.
0	1	Interpret switches as specified above.
	0	Interpret switches 15-1 as an address, transfer control to that address after saving R0-R7 in memory locations 50-56.

Procedure 2 Loading KLDCP from RSX-20F

Step	Procedure
1	This procedure assumes RSX-20F is running in the console front-end subsystem. Refer to Procedure 1.
2	If at KLI command level (KLI>), type: EXIT<CR>
3	Mount the KLAD pack and make unit ready and on-line.
4	To RSX-20F type: (↑\\) This will put the CTY in PARSER command mode.
5	To PARSER type: MCR BOO<CR> This directs PARSER to load and start the BOO TSK file.
6	To BOO type: DBOOT<CR> This directs BOO to load and start KLDCP.

Procedure 3 Loading and Starting Programs via KLDCP
The following diagram illustrates the steps involved in loading and starting programs via KLDCP. Each step is described in detail following the diagram.



Procedure 3 Loading and Starting Programs via KLDCP

Step	Procedure
1	KLDCP must be running in the console front-end processor. Refer to Procedure 1 or 2.
2	Mount the storage medium containing the programs to be run and select the load device by typing the appropriate KLDCP command. AT<CR> selects the APT10 DL<CR> selects the DL11 DTn<CR> selects DECtape transport n DXn<CR> selects floppy unit n RPn<CR> selects disk drive unit n RXn<CR> selects floppy unit n
3	If the line printer is to be used for output (DECSYSTEM-20s only), To KLDCP type: LP<CR>
4	If the operational status of the KL10 mainframe is questionable, To KLDCP type: B<CR> This command causes KLDCP to run the 11-based 10 and the 10-based 10 KL10 processor functional diagnostics. The command requires approximately twenty minutes for execution and should only be used when the operational status of the KL10 mainframe is in question.

Procedure 3   Loading and Starting Programs via KLDCP (Cont)

Step	Procedure	
5	If a single diagnostic is to be run, the console control switches may need to be set. Table 1 describes the standard control switches. Exceptions to the standard are described in the individual diagnostic summaries. To KLDCP type: ES data<CR> where "data" is six octal digits and represents the desired switch setting. Note that KLDCP exclusive ORs the physical 11/40 console switches with the "data" specified in the ES command. Therefore, to use only the "data" specified in the ES command, the console switches must be reset. Conversely, to use the physical console switches the ES command must specify a switch setting of 0s (e.g., ES0<CR>).	
6	If a single diagnostic is to run, it may require that the KL10 memories be configured. Refer to the diagnostic hierarchy listed in Table 4 or 5. If configuration is necessary, To KLDCP type either:	
6A	I CONFG.CCL<CR>	This command directs KLDCP to configure the KL10 memories.
	or:	
6B	BT<CR>	This command causes KLDCP to configure memory, load the microcode, load the 10/10 SUBRTN package and KLDDT.
7	If a single program is to be run, To KLDCP type one of the following.	
7A	P file.ext$	This will cause the file specified to load and start.
7B	P file.ext<CR>	This will cause the file specified to be loaded but not started. This opportunity may be used to insert changes or otherwise modify the program.
8	There are three commands which may be used to start an 11-based 10 program once it is loaded. To KLDCP type one of the following.	
	SE adr<CR>	This will start the program at the address specified.
	SED<CR>	This will start the program at its normal starting address (3000).
	SEDn<CR>	This will start the program and run it for the specified number of passes(n).

STANDARD ERROR MESSAGE FORMATS
11-based 10 diagnostics generally use one of three methods for reporting errors. Some diagnostics, however, use a combination of the methods. The objective in all cases is the same: to provide the user with as much information about the problem as possible.

First Method
The first and most common method of reporting errors is "actual data" vs "expected data." The following example illustrates this method.

```
Test Number 27     Subtest 5     PC = 10244

Source

Actual Data      101 010 010 010 000 011 111 001 101 000 100 111
Expected Data    100 XXX XXX XXX 0X0 1XX XXX XXX XXX XXX XXX 111
Difference         1                 1
```

Test Number and Subtest - The test and subtest numbers provide an index to the test and subtest description in the diagnostic summary.

PC - The PC provides an index into the test code which detected the failure. Refer to the diagnostic listing on microfiche.

Source - The information provided in this section of the error message indicates the source of the "actual data." The source printed may be a register name, memory location, diagnostic function, etc.

Actual Data - The actual data is the machine state which occurred as a result of the test.

Expected Data - The expected data is the mask used by the test to determine if the actual data is correct. Bits marked by an "X" are not checked by the test [i.e., they have no significance to the test and may be in a set (1) or reset (0) state].

Difference - The difference indicates the bits in "actual data" word which are in error; i.e., these bits are the symptoms which indicate there is a malfunction in the hardware.

Second Method
The second method used by 11 based 10 diagnostics to report errors is to print the failing test number and PC and then list the names of the signals which are in error.

Third Method
The third method of error reporting is to print a text file which describes the nature of the failure and most probable causes. this method is most often used when it is difficult or impossible to determine the exact cause.

GENERAL INFORMATION

Abstract	DIACON is the executive controller which is assembled with each 11-based 10 diagnostic.
Notes	Console commands may be executed directly from DIACON command mode. If a naming conflict occurs, a period preceding the command will ensure that the console will act on it. The sync point for scope loops is generated by DIACON at A36E1 on the CPU backplane.
Loading and Starting Procedure	DIACON is automatically loaded as part of each 11-based 10 diagnostic.

OPERATIONAL CONTROL

DIACON has two modes of operation. Normal operation (control switch 1 reset) is transparent to the user. In this mode DIACON runs the diagnostic, performs fault convergence or cataloguing, reports fault symptoms, loads and runs TIC files and/or isolation routines, and prints the most probable cause (board callout), all without operator intervention.

The second mode of operation (control switch 1 set) enables a set of commands which permit user intervention. These commands are described in Table 1. The commands described in Table 2 are only supported if they are listed by the /H command.

Table 1 DIACON Command Summary

Command	Description
.	Commands beginning with a period are passed to KLDCP for execution.
H	H<CR> Print a list of the DIACON commands currently in effect.
HE	HE<CR> Print a list of all commands supported by DIACON.
/H	/H<CR> Prints a list of special switches defined by the diagnostic programmer. User switches are supported only if the programmer specified this option. Refer to Table 2.
↑C	Control C interrupts the execution of the diagnostic. Control is returned to KLDCP.
HC	HC<CR> The HC command continues from an error halt.
$	$ES 0<CR> Altmode interrupts the execution of the diagnostic for one KLDCP command line.
TS	TS 16<CR> The TS (Test Start) command starts the diagnostic beginning at the test number specified.

Table 1 DIACON Command Summary (Cont)

Command	Description
TL	TL 14,37<CR> The TL (Test Loop) command loops between the first test number and the second test number. A carriage return instead of a second test number will cause the first test specified to be looped on. The abort switch (15) will return control to the console. The TL command can be used to report the first error in every test. This can be done by setting the print all errors switch (07) and the abort switch (15) and specifying a test range of the entire diagnostic,.
PS	PS<CR> The PS (Print Symptoms) command causes DIACON to report unreported errors or to repeat its last error report. Calling of isolation routines is also permitted. No symptom will be printed if the test has been restarted or no fault has occurred.

Table 2 DIACON User Implemented Command Summary

Command	Description
FB	FB<CR> The FB (set Function Breakpoint) command solicits a diagnostic function, bit, and polarity which, if detected, will cause a break to occur. Only one function breakpoint is permitted at a time.
FC	FC<CR> The FC (Function breakpoint Continue) command restores the PDP-11 registers and continues from the last function breakpoint.
RB	RB(CR> The RB (Remove Breakpoint) command removes the function breakpoint.
RG	RG<CR> The RG (print PDP-11 registers) command prints the contents of R0 through R7 saved at the last function breakpoint. This command is primarily for use in debugging programs.

GENERAL INFORMATION

Code	DGQDA BIN
Title	DECSYSTEM Diagnostic Console Program
Abstract	KLDCP resides in the console front-end processor and supports KL10 based systems at the following three levels.

1. At the console level, KLDCP supports KL10 and PDP-11/40 console functions.
2. At diagnostic run time, KLDCP loads, starts, and provides subroutine services for 11-based 10 and 10-based 10 diagnostic and utility routines.
3. At the timesharing level, KLDCP provides an interface between the CTY or KLINIK terminal and the TOPS-10 or TOPS-20 monitor.

Note	The DTE20 must be in privileged mode.
Loading and Starting Procedure	Refer to the 11/10 STD module.

OPERATIONAL CONTROL
KLDCP is controlled via commands entered at the CTY or KLINIK terminal. The commands consist of two or three characters followed by one or more arguments. The conventions used to illustrate the KLDCP commands are described in Table 1. The commands supported by KLDCP are described in Table 2.

Table 1 KLDCP Command Conventions

Convention	Description
adr	An octal address
data	An octal data field
file.ext	Any legal file name from one to six characters followed by a dot and an extension of zero to three characters
<CR>	Standard command string terminator
$	When used to terminate a P command, the $ (altmode) will cause the file specified to be loaded and started.
#	An octal argument
:	Separates the address and data fields in examine and deposit commands
?	Precedes error message printouts
C	Control C aborts program, returns control to KLDCP from 10 memory.
↑T	Control T must be used as a terminator for commands that are to be interpreted by programs running in the KL10.
↑X	Control X interrupts the program running in 10 memory for one KLDCP command.
;	When a semicolon precedes local comment, the text following it is only printed on the terminal. Messages are sent between the CTY and KLINIK terminal using the semicolons.

Table 2 KLDCP Command Summary

Command	Description
General Commands	
R	R MR, EX inst, PL15<CR> Repeat commands following. Inhibit machine-state printouts.
RP	RP MR, EX inst, PL15<CR> Repeat commands following. Do not inhibit data printout.
TD #	R MR, EX inst, PL10, TD5, PL5<CR> Perform specified (#) time delay.
TF #	TF 0<CR> Set terminal fill count. 0 - 110 baud 3 - 600 baud 1 - 150 baud 4 - 1200 baud 2 - 300 baud 5 - 2400 baud
TW #	TW 132<CR> Set terminal page width (10 min. - 132 max.).
TP #	TP 60<CR> Set terminal page length.
LP	LP<CR> Select line printer for output.
KLINIK	Enable/disable KLINIK line
PDP-11 Console Commands	
ES	ES<CR> Print present 11 switch register.
ES data	ES 103452<CR> Set 11 switch register to data specified.
E36 adr	E36 5000<CR> Examine specified 11 address for a 36-bit word.
EE adr	EE 3000<CR> Examine specified 11 word address.
EB adr	EB 2001<CR> Examine specified 11 byte address.
D36 adr:data	D36 5000:252525 252525<CR> Deposit 36-bit data specified into 11 address specified.
DE adr:data	DE 3000:103452<CR> Deposit 16-bit data into 11 address specified.
DB adr:data	DB 2001:377<CR> Deposit 8-bit byte into 11 byte address.
ZE adr,adr	ZE 100, 200<CR> Clear the 11 memory from address to address.
SE adr	SE 3000<CR> Start 11 at address specified.
SED	SED<CR> Start 11 diagnostic.
SED #	SED 100<CR> Start 11 diagnostic and run number (#) of passes.
BP adr	BP 3150<CR> Set a breakpoint at 11 address specified.

Table 2 KLDCP Command Summary (Cont)

Command	Description
BC	BC<CR> Continue from breakpoint.
RB	RB<CR> Remove breakpoint.
RG	RG<CR> Print registers saved at breakpoint (R0-R7).
PA, Clock, and, Cache Commands	
PA	PA<CR> Establish a fixed core address for the KL10 communication region. A second PA will make the communications region relative to the EBR.
CS	CS<CR> Print clock source code.
CS #	CS 1<CR> Select specified (#) clock source. 0 normal clock 1 speed margin clock 2 external clock DEC20 + KL10R KW20
CR	CR<CR> Print clock rate code.
CR #	CR 1<CR> Select specified (#) clock rate. 0 normal 1 divide by 2 2 divide by 4 3 divide by 8
CE	CE<CR> Print current cache enable code.
CE #	CE 10<CR> Select cache enables according to number code specified (#). 1 (0001) enable cache 3 2 (0010) enable cache 2 4 (0100) enable cache 1 10 (1000) enable cache 0 (default is 17 - all four caches)
CI	CI<CR> Executes cache invalidate instruction.
CF	CF<CR> Executes cache flush instruction.
Microcode CRAM/DRAM Commands	
MM adr	MM 150<CR> Set sync mark (bit 34) at microcode address specified.
MMA adr	MMA 150<CR> Set sync marks from address 0 up to and including address specified.
MU	MU 150<CR> Clear sync mark at microcode address specified.
MUA adr	MUA 100<CR> Clears sync marks from address 0 up to and including address specified.
SM	SM<CR> Start 10 microcode running.

Table 2 KLDCP Command Summary (Cont)

Command	Description
EC adr	EC 112<CR> Examine CRAM at address specified.
DC adr:data	DC 112:123456 123456 123456 123456 123456 12<CR> Deposit data specified at CRAM address specified.
RC adr	RC 123<CR> Examine CRAM address specified using diagnostic functions.
ED add	ED 776<CR> Examine DRAM at address specified.
DD adr:data	DD 776:7 6 1 1234<CR> Deposit data specified at DRAM address specified. NOTE The DD command will prompt for odd address data.
Diagnostic Functions	
FX FUNCT	FX 11<CR> Execute diagnostic function specified (00-37).
FW funct:data	FW 77:252525 252525<CR> Write data specified to diagnostic function address specified (40-77).
FR funct	FR 100<CR> Read and print the contents of the diagnostic function address specified (100-177).
FR functl, functX	FR 100,150<CR> Read and print the contents of each diagnostic function beginning at functl and ending with functX.
FS	FS<CR> Generate a sync pulse at 4A36E1
R/W Major Registers	
DA data	DA 123456 654321<CR> Deposit data specified into AR register.
XX	AR<CR> Read and print the contents of the register specified by XX. XX = ALL - Print all CRAM and registers AD - adders ADX - extended adders ADB - address break AR - arithmetic reg ARX - extended AR BR - buffer reg BRX - extended BR ERG - EBus reg FM - fast memory reg MQ - multiply/quotient reg PC - program counter PI - priority interrupt VMA - virtual memory address VMH - VMA held

Table 2 KLDCP Command Summary (Cont)

Command	Description
KL10 Console Commands	
RI	RI<CR> Reinitialize console program.
MR	MR<CR> Master reset.
HC	HC<CR> Continue from program halt or error.
SW	SW<CR> Print present 10 switch register.
SW data	SW 123456 654321<CR> Set the 10 switch register to the data specified.
EM adr	EM 2000<CR> Examine 10 core at address specified.
EN	EN<CR> Examine next sequential 10 address.
DM adr:data	DM 2000:123456 654321<CR> Deposit data specified into 10 address specified.
DN data	DN 123456 654321<CR> Deposit data specified into next sequential 10 address.
MZ adr,#	MZ 100,50<CR> Clear the number of address specified by # beginning at the 10 address specified.
EX inst	EX 201000 777777<CR> Execute 36-bit instruction specified.
EXP inst	EXP 201000 777777<CR> Executes instruction specified and prints machine state changes at each clock tick.
EXT inst	EXT 201000 777777<CR> Sets up the instruction specified to be executed by the TRACON or TRACE program.
SP	SP<CR> Stop 10, clear run flip-flop.
RN	RN<CR> Start 10, set run flip-flop.
SI	SI<CR> Single instruct, push continue button.
SI #	SI 5<CR> Single instruct the specified number (#) of times.
SIP	SIP<CR> Single instruct and print machine state changes.
PL	PL<CR> Pulse clock one tick.
PL #	PL 21<CR> Pulse clock specified number (#) of ticks.
BU	BU<CR> Burst clock once.
BU #	BU 3<CR> Burst clock the number (#) of times specified.

Table 2 KLDCP Command Summary (Cont)

Command	Description
KL10 CPU Setup Commands	
AC BLK	AC BLK<CR> Print current AC block number.
AC BLK #	AC BLK 7<CR> Select AC block specified (#).
PE	PE<CR> Print KL10 parity enable codes.
PE #	PE 1<CR> Enable KL10 parity options according to code specified (#). 1 (00001) field service probe 2 (00010) DRAM parity 4 (00100) CRAM parity 10 (01000) FM parity 20 (10000) AR/ARX page fail (default is 16)
PD	PD<CR> Disable all KL10 parity options.
File and Device Selection Commands	
FV	FV<CR> Select files-11 media type.
FE	FE<CR> Select secondary front-end load mode.
DL	DL<CR> Switch to DL-DN87S load mode.
AT	AT<CR> Switch to APT10 load mode.
XX #	DT 1<CR> Select specified device type (XX) and unit number (#) for input. XX = DT DECtape DX diskette RP RP04 RX floppy
KL10 Start Commands	
ST adr	ST 4000<CR> Start 10 at address specified.
ST	ST<CR> Start 10 at previously supplied address.
STD	STD<CR> Start 10 diagnostic (EPT adr = 440).
STD #	STD 100<CR> Start 10 diagnostic and run the number of passes specified (#).
EP #	EP 10<CR> Set EOP (end-of-pass) interval count.
STL	STL<CR> Start 10 loader - DIAMON, MAGMON or D20MON (EPT adr = 442).
DDT	DDT<CR> Start DDT (EPT adr = 441).

Table 2 KLDCP Command Summary (Cont)

Command	Description
STM	STM<CR> Start 10 monitor - TOPS-10 or TOPS-20 (EPT adr = 443).
MC	MC<CR> Continue 10 monitor.
RSX	RSX<CR> Boot RSX-20F from KLAD pack.
BT	BT<CR> Boot system to run diagnostics with KLDCP.
B	B<CR> Boot system and run all KL10 diagnostics.
LI	LI<CR> Log in.
LO	LO<CR> Log out.
File Load and Execute Commands	
I file.ext	I X2.CCL<CR> Execute specified indirect file.
J file.ext	J DHDIAG.CMD<CR> Execute specified double indirect file.
JR	JR<CR> Repeat last J command.
JC	JC<CR> Continue interrupted double indirect command file.
P file.ext	P DHKAA.A11<CR> Load specified file.
LE file.ext	LE TRACON.A11<CR> Load PDP-11 .A11 file.
LB file.ext	LB XTECO.BIN<CR> Load PDP-11 .BIN file.
LR file.ext	LR EBOX.RAM<CR> Load microcode .RAM file.
LT file.ext	LT DFDTE.A10<CR> Load KL10 .A10 file.
GO	GO<CR> Go start program just loaded.
File Verify, Write, and Rename Commands	
V file.ext	V DHKAA.A11<CR> Verify - compare specified file against file in core.
CD	CD CHAN.TST 700000,100000<CR> Write the contents of 11 core beginning at the first address specified and ending at the last address specified. The core contents will be written into the file specified on the selected output device. The file specified must already exist on the output device. (See KLDCPU ALLOC command.)
CDA	CDA CRASH.A11<CR> Write entire contents of 11 core to output device using file name specified. The file must already exist on the output device. (See KLDCPU ALLOC command.)

Table 2   KLDCP Command Summary (Cont)

Command	Description
WF file.ext	WF DHKBB.All<CR> Write - copy file specified from DECtape or floppy to RP04.
RENM	RENM file.ext filel.ext<CR> Rename RP04 file from file.ext to filel.ext.
Miscellaneous Commands	
H	H<CR> Print KLDCP help file.
H file.ext	H TRACON.All<CR> Print help file for file specified.
T	T<CR> Print time
C msg	C mount the KLAD pack <CR> Send message specified from console terminal to KLINIK terminal and vice versa.

KLDCP ERROR MESSAGE SUMMARY
The following are standard KLDCP error messages listed in alphanumerical order. The notes are referenced in the text.

NOTES

1. This error message could have occurred because of a faulty deposit or examine command. Try a TRACON deposit or examining command. TRACON does not use the PI system.

2. These error messages include the value of the PC at the time of error. The PC allows the field engineer to look up the failing code in the KLDCP listing and determine what combination of instructions caused the fault to occur.

3. These error messages are associated with the APT10 and should never occur in a system installed in the field. The APT10 is an automatic processor tester used by manufacturing to check out the KL10 CPU.

4. These error messages apply to the internal format of the program being loaded. Most likely, these errors will occur as a result of a bad copy of the program or a faulty I/O device.

? ADR - An improper address parameter was used with the command. It may be that the address is nonexistent or inappropriate for the device being addressed. For example, an odd starting address supplied with an SE command would cause an ? ADR error. Check the address parameter of the command.

? APT10 - An APT10 command was issued but no APT10 was selected. See Note 3.

? APT10 ENQ - KLDCP made a service request to the APT10 but the APT10 was unable to perform that service. See Note 3.

? BLK # FLOPPY ERR - A nonexistent block number (#) was used in addressing the floppy disk.

? BP ERR - KLDCP supports the insertion of up to eight breakpoints. Should this number be exceeded, the message ?BP ERR will be printed.

? BUS TIMEOUT - This error occurs as a result of a Unibus timeout condition. That is, a slave sync pulse has not occurred within 15 microseconds after a master sync pulse is issued. The cause of this error depends on the I/O device being serviced at that time. In most cases, however, the cause will turn out to be either the eleven memory or the DTE20. Note that the 15 microseconds timeout delay may vary depending on the characteristics of the I/O devices connected to the Unibus. Some require the delay to be extended.

? CKSUM ERR: ECT - Load line checksum error occurred. See Note 4.

? CLK ERR AT # - The clock logic in the KL10 will not respond to single pulsing. See Note 2.

? COMM ERR # CODE - This is a general APT10 error message. More specific information can be found by looking up the error (#) code. See Note 3.

? CAN'T LOAD - Indicates that the error retry count has been exceeded and the requested file cannot be loaded.

? CAN'T CONT - This message is associated with breakpoints. The breakpoint function uses the stack (R7) to store the return address. If the contents of the stack are changed after a breakpoint has occurred and the operator attempts to continue from the breakpoint by typing BC, the message ? CAN'T CONT will be printed. If it is important to restart the program try an SE command using the address of the breakpoint plus 1.

? CKSUM ERROR - The binary file just read had a data checksum error. The problem could be due to a bad copy of the binary file or a faulty I/O device. See Note 4.

? DF ERR - A diagnostic function parameter error has been detected. For example, a FX120 would cause this message because 120 is not within the acceptable range for a diagnostic function execute. Check the parameters of the diagnostic function.

? DF TIMEOUT AT # - A diagnostic function was executed but there was no response from either the KL10 or DTE20 within a reasonable period of time (a few microseconds). Check power to the DTE20 and the clock in the KL10 - the DTE20 diagnostic should catch this problem. See Note 2.

? DIAMON XFER - DIAMON was unable to transfer a file or part of file to the KL10.

? DM ERR AT # - KLDCP is unable to deposit in the KL10. Try the TRACON deposit command because it does not use the PI system. See Note 2.

? EB PAR - An EBus parity error has occurred. Check the source and direction of the EBus transfer.

? EM ERR AT # - This message occurs as a result of an incomplete examine operation (i.e., the KL10 or DTE20 did not respond properly to the command). Try the TRACON examine command. TRACON does not use the PI system. See Note 2.

? EOF - An unexpected End of File was detected. See Note 4.

? F11 FIND - The specified file cannot be found in the files-11 directory. The directory may have been destroyed. Always write-protect the KLAD Pack.

? F11 LOG BLK - The logical block number given to address files-11 formatted media is nonexistent.

? FATAL - This error message does not pertain to KLDCP. It is a condition reported to KLDCP by a program (usually a diagnostic), running in conjunction with KLDCP. This message occurs when such a program encounters an error condition it was not designed to handle. For example, if, while running an MBox Diagnostic, the EBox fails and the MBox diagnostic cannot recover on its own, it will request KLDCP to print the message "? FATAL." There are several ways to approach this problem.

1. Check for outstanding MCOs.
2. Try a different copy of the program.
3. Load it from a different I/O device.
4. Review the diagnostic hierarchies to determine if preliminary programs should be run.

? FATAL INTR - Fatal Vector Interrupt. KLDCP uses FLAG MODE to keep track of I/O devices. KLDCP does not use the PDP-11 priority interrupt system. Therefore, any vector interrupt is unexpected and considered fatal. Should this error occur, run the PDP-11 priority interrupt diagnostics.

? FORMAT ERR: ECT - The format of the load line is incorrect. See Note 4.

? HARD DTA ERROR - A hard (nonrecoverable) error has occurred in the DECtape subsystem. This is generally a controller- or transport-type problem.

? HARD FLOPPY ERROR - A hard (nonrecoverable) error has occurred in the floppy subsystem. A problem of this type usually indicates a controller or device error.

J CMN - The "common area" of the DRAM data does not match. Check the common area and retype the command.

? J SIZE - The size of the DRAM J field is too large. Check the size and retype the command.

? KL10 CLOCK ERROR STOP - This message occurs as a result of an error stop condition - FM parity, CRAM parity, DRAM parity, or FS probe. The reason for the error stop is reported along with the error message.

? KL10 HALTED - The KL10 executed a halt instruction. Check the KL10 PC and refer to the program listing.

? KL10 RUNNING ECT - Certain console commands cannot be executed while the KL10 is running. They are as follows.

Diagnostic functions
Internal EBox resistor reads
Pulsing the clock
Clock rate source changes
Microcode mark and unmark functions
CRAM and DRAM deposits and examines
Cache invalidate and flush
Clearing KL memory
AC block selection

Before executing any of these commands, stop the KL10 by typing the SP command.

KLDCP CKSUM - The KLDCP code has been changed since the last checksum operation was performed. If the code was not deliberately modified to patch around a problem or execute a slightly different operation, this could mean any of the following.

1. The last command executed somehow, inadvertently, changed the code.
2. The console front-end system has developed a problem.
3. The DTE may have a fault that caused data to be written into the wrong area of 11 core.

? LINE TOO LONG - The internal file data line is too long (in excess of 132 characters). See Note 4.

? LOAD CHR ERR: ECT - The load line identification character is invalid. See Note 4.

? LP ERR - KLDCP has detected an error status coming from the line printer.

LPT OFF - The line printer appears to be off-line.

? MZ ERR AT # - This message occurs as a result of an incomplete MZ deposit operation. Try the TRACON deposit command. It does not use the PI system. See Note 2.

? NAME EXT - An invalid file name or file extension was used.

? NO LPT - There is no detectable line printer.

? NO MASTER DTE - KLDCP will not run with the DTE20 in restricted mode. This is because a restricted DTE20 will not allow the execution of the diagnostic functions. If this error message is printed, check the switch on the DTE20. Other possibilities are that the DTE will not respond to the Unibus address, or the DTE has lost power.

NON-EX FILE - KLDCP could not find the file as specified. Try a directory command DI.

? PARAM - As soon as a command is entered, KLDCP checks to assure that the typed-in parameters of the command fall within acceptable boundaries. If the parameters are outside the boundaries for that command, the error message "? PARAM" is printed. For example, a nine (9) entered in an octal field would cause a ? PARAM error message to be printed. Check the parameters of the line and retype the command.

? RES INST - There are certain PDP-11 operation codes that are not implemented by the hardware. These are referred to as reserved instructions and should never be executed. Execution of a reserved instruction will cause a trap to address 10 and KLDCP will print "? RES INST." This type of error usually indicates that some portion of the core was destroyed. Try reloading. If that does not correct the problem, run the PDP-11 processor and memory diagnostics, including the diagnostic that checks the reserved instructions.

? RESPONSE - The APT10 has failed to respond within a reasonable amount of time. See Note 3.

? REV DTA ERROR - KLDCP allows for three reversals in tape motion during a search. If that number is exceeded, the error message ? REV DTA ERROR is printed.

? RP04 ERROR # CODE - This error message occurs as a result of an RP04 error. The number code corresponds to one of the following:

1. Unit number incorrect
2. Drive not available
3. Drive unit error 1
4. Drive unit error 2
5. Drive unit error 3
6. Home block read error
7. Not home block
10. Incorrect file system name
11. No index file
13. Reading past EOF
14. Blk size position error
15. Read error
16. Attempt to change allocation
17. Buffer size
20. Current position
21. Insufficient allocation for write
22. Directory rewrite error
23. Data block write failure
24. End of file
25. Rad50 conversion error

? SEL ERR - KLDCP cannot select the requested AC block. KLDCP uses the AR data path and diagnostic functions to select the AC block. A select error usually indicates a faulty data path or diagnostic function.

? SOFT DTA ERROR - This message indicates that a soft (recoverable) data error occurred on the DECtape. This is usually a media problem.

? ST UNFLO - Stack underflow. This error occurs any time the software attempts to POP more entries off the stack than were originally pushed onto it. This error indicates that the KLDCP code was destroyed. Reload KLDCP. If that doesn't correct the problem, run the PDP-11 Processor and Memory Diagnostics.

? UCODE HUNG - The microcode is not in the halt loop. This may indicate that the KL10 is not set up properly to execute the command (i.e., the ucode is not loaded) or that the ucode did not return to the halt loop. It may be hung up waiting for a memory response.

? - KLDCP does not recognize the command as typed. Check for proper format and retype the command.

? 10 CLK OP - The KL10 uses the clock in the PDP-11 to keep track of time. This error message indicates that the PDP-11 cannot notify the KL10 that a clock tick has occurred.

? 10 CMD ERR - The program running in the KL10 has issued an illegal command to KLDCP.

? 10 SW - KLDCP is unable to notify the KL10 of a change in the data switches. See Note 1.

? 10 TTI - KLDCP is unable to send a teletype character to the KL10. See Note 1.

? 11 PARITY - An 11 parity error has been detected. Run 11 memory NPR device and DTE20 diagnostics.

GENERAL INFORMATION

Code	DGQDB.A11
Title	DECSYSTEM Diagnostic Console Utility Program
Abstract	KLDCPU is a console utility program which resides in the lower half of 11/40 core and extends the KLDCP command set to include file maintenance service. The utility is capable of performing operations on DECtape, floppy and KLAD packs. The utility has single file manipulations capability and also facilities for handling groups of files.

Notes

1. KLDCP will perform a validity check of the utility portion and will request that the operator load KLDCPU.A11 if it is not resident when any utility command is performed.

2. Any command which is not one of the utility commands is automatically passed to KLDCP for processing. This allows all the KLDCP commands to be performed from the utility command process.

3. The SAVRSX and KLADBT commands are used to change the hardware boot on the disk so that KLDCP is booted when the disk button is pushed.

   RSX-20F must have been installed on the KLAD-10 disk so that the proper exchange takes place.

   RSX-20F is then booted when required by the KLDCP RSX command. This command reads the RSX-20F boot block from the RSXBT.ZRO file, installs it in memory starting at zero and starts it as though a switch register disk boot were done.

4. Wild characters - the asterisk (*) and question mark (?) - may be used in file name construction.

Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	None

OPERATIONAL CONTROL
KLDCPU commmands may be entered either directly via the CTY or KLINIK link or indirectly via a control file.

The conventions used to illustrate KLDCPU commands are described in Table 1. KLDCPU switches which may be used to modify the commands are described in Table 2. The commands supported by KLDCPU are summarized in Table 3.

Table 1 KLDCPU Command Conventions

Convention	Description
↑C	Control C returns to command mode, aborting the operation in progress.
↑Z	Control Z exits TEXT mode.
:	Delimits device specification.
=	Delimits input and output file specifications.
DTn:	DECtape unit n.
RXn:	Floppy unit n.
DXn:	Floppy unit n.
RPn:	RP04/06 unit n. The RP04/RP06 disk is a read-only device, as the file structure is maintained via the TOPS-10 or TOPS-20 systems. The disk may be either the KLAD-10 or the KLAD-20 format; selection of the disk will automatically select the proper processing operations.

Table 2 KLDCPU Software Switch Summary

Switch	Description
/F	DIR DT0:/F<CR> Print the directory in abbreviated format.
/N	FILE DT0:DT1:FILE.EXT/N<CR> Do not list each file name as it is transferred.
/H	/H<CR> Print the help message.

Table 3 KLDCPU Command Summary

Command	Description	Cross Ref.
REMOTE	REMOTE<CR> Select remote terminal.	1
RI	RI<CR> Reinitialize console (KLDCP).	2
BOOT	BOOT RX0:<CR> Load and start the bootstrap loader from the device specified.	3
SVBOOT	SVBOOT DT0:=RP0:KLDTBT.BIN<CR> Create the specified boot file and write it to the boot block of the specified output device.	4
KLADBT	KLADBT<CR> Write KLADBT.ZRO to the boot block of the KLAD-10 pack.	5
SAVRSX	SAVRSX<CR> Transfer the boot block of a KLAD-10 disk to the file RSXBT.ZRO.	6
DIR	DIR DT0:<CR> Print the directory for the specified device.	7

Table 3 KLDCPU Command Summary (Cont)

Command	Description	Cross Ref.
FID	FID RP0:DGMMA.*<CR> Print the specified files, file identification line.	8
RENAME	RENAME RX0:DGKAA.A11=RX0:DGKAA.OLD<CR> Rename the specified file to a new name.	9
DEL	DEL RP0:DHKAA.A11<CR> Delete the specified file from the device specified.	10
ZERO	ZERO DT1:<CR> Clear the directory of the device specified.	11
ASG	ASG RP0:=MASTER:<CR> Assign the specified logical name to the physical device specified. Acceptable logical names are: IN, OUT, MASTER and NEW.	12
DATE	DATE: 31-OCT-77<CR> Change the date used by KLDCPU format = DD-MMM-YY.	13
ALLOC	ALLOC DT0:CRASH.DMP/100<CR> Allocate an empty file (having 100 blocks) for future use on DECtape or floppy disk.	14
PIP	PIP RX0:file.IN=DT0:file.OUT<CR> Transfer the file specified from the input device to the output device.	15
FILE	FILE DT0:=DT1:*.A11<CR> Perform bulk file transfers between the input and output devices.	16
FILET	FILET DT0:*.*<CR> Test files specified for error.	17
DTCOPY	DTCOPY<CR> Copy all the files from one DECtape to a second DECtape.	18
RXCOPY	RXCOPY<CR> Copy all the files from one floppy disk to a second floppy disk.	19
TAPT	TAPT DHKCA.A11<CR> Transfer the file specified from the APT10 to the KLAD-10 disk pack.	20
TEXT	TEXT RP0:CPU.CCL<CR> Build an ASCII command file and write it to the device specified using the filename and extension specified.	21
DO	DO CPU.CCL<CR> Execute the specified command file.	22

COMMAND DESCRIPTIONS

This section describes the commands summarized in Table 3.

1. REMOTE<CR> - The REMOTE command selects remote terminal operation over the APT10 communication link.

2. RI<CR> - The RI command is passed to KLDCP where it reinitializes the diagnostic console and returns control to KLDCP.

3. BOOT RX0:<CR> - The BOOT command causes block 0 of the device to be read into memory starting at location 0. Block 0 is assumed to contain a bootstrap loader. The utility then transfers control to the boot just read in at location 0.

4. SVBOOT DT0:=RP0:KLDTBT.BIN<CR> - The SVBOOT command reads the specified binary file (KLDTBT.BIN) and writes it out to the specified output device (DT0) in the boot block and to the core image boot blocks. (The file specified must have a bootstrap format.)

5. KLADBT<CR> - The KLADBT command copies the KLDCP bootstrap loader file (KLADBT.BR0) to block 0, cylinder 0 of the KLAD-10 pack.

6. SAVRSX<CR> - The SAVRSX command copies block 0 cylinder 0 of the KLAD-10 to the file RSXBT.ZRO.

7. DIR DT0:<CR> - The DIR command gives a directory of the requested device (DT0). This command will give the entire directory or a partial directory of requested files via use of the wild character or asterisk constructions.

DIR RP0:<CR>	Prints a full directory.
DIR RP0:*.BIN<CR>	Prints a directory of all files with a BIN extension.
DIR RP0:A7????.*<CR>	Prints a directory of all files. Those first 2 characters are A7.

8. FID RP0:DGMMA.*<CR> - The FID command prints a file identification line directory of the requested device (RP0). The file identification line is the first line of an ASCIIzed file which provides internal file identification (i.e., file name, file version and creation date). The FID command provides the same wild character and asterisk constructions as does the DIR command.

9. RENAME RX0:file.new=RX0:file.old<CR> - The RENAME command renames an old file (file.old) to a new file name (file.new).

10. DEL DT0:DHKAA.A11<CR> - The DEL command causes the file specified (DHKAA.A11) to be deleted from the directory of the device specified (DT0).

11. ZERO DT1:<CR> - The ZERO command clears the directory of the device specified (DT1).

12. ASG RP0:=MASTER:<CR> - The ASG command allows the use of logical names in command files. Allowed logical names are: IN, OUT, MASTER, and NEW. A command file may use a logical name such as "MASTER" instead of specifying a physical device. Then, before executing the command file the user can assign the desired physical device to the logical name. This permits the use of any available unit.

13. **DATE:** 31-OCT-77<CR> - The DATE command allows changing the date used by the utility operations. Type the date according to the following format.

    DATE: DD-MMM-YY

    DD is the day of the month.
    MMM is the month: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC.
    YY is the year.

    When transferring files to a new medium the original file's date is used unless the input device is the disk, in which case the typed-in date is used.

14. **ALLOC** DT0:CRASH.DMP/100<CR> - The ALLOC command allows an empty file (CRASH.DMP) to be allocated on either the DECtape or floppy for subsequent use by the CORE DUMP KLDCP file generation command.

    The size is the number of blocks required. If the "/SIZE" is not given in the command, the size will be specifically asked for.

15. **PIP** RX0:file.IN=DT0:file.out<CR> - The PIP command is used to transfer a file (file.out) from one device (DT0) to another device (RX0). The device types may be different and the file name may be changed; however, asterisk and wild character constructions may not be used. The output file name must not exist on the output device.

16. **FILE** DT0:=DT1:*.A11<CR> - The FILE command is used to do bulk transfers (i.e., all files on DT1 with an .A11 extension) from one device to another device. The FILE command is similar to the PIP command except that it can utilize the asterisk and wild character constructions. If a file of the same name already exists on the output device, the file command will delete the old file.

17. **FILET** DT0:*.*<CR> - The FILET command tests all files named by reading them into a buffer to make certain that no device errors occur. Any device errors are listed as they occur.

18. **DTCOPY** - To be supplied.

19. **RXCOPY** - To be supplied.

20. **TAPT** - To be supplied.

21. **TEXT** RP0:CPU.CCL<CR> - The utility includes the facility to execute a sequence of commands contained in ASCII text file. This text file may be created via the TEXT command.

    When the TEXT command is issued the named output file is opened for output and the operator is prompted with a quotation mark (") to indicate readiness to accept text. Any normal ASCII command character may be placed into the file.

    RUBOUT can be used to delete characters on the current line (but not on preceding lines).

    CONTROL C (↑C) will abort the text operation.

    CONTROL Z (↑Z) is the standard terminator for input. It will close out the text file and return to command mode.

22. DO CPU.CCL<CR> - The DO command is used to cause execution of a control file. The file is executed line by line and may contain either utility commands or KLDCP commands. Executable files are created via the TEXT command or via any of the text editors.

KLDCPU ERROR MESSAGE SUMMARY
The following is an alphabetical listing of KLDCPU error messages.

DELERR - A bit map error occurred during a delete operation.

DELOLD - Delete the old file before issuing a command which would create a file with the same name.

DEVERR - A device error occurred on either the input or output device. Check that the output device is write-enabled.

DEVFUL - The output device is full. There is no more file storage room available.

DIRERR - An invalid file name exists in the device directory.

INVCMD - The command issued is invalid. Examine the command for proper format and retype.

INVDEV - The device specified in a command is invalid. Check the command for proper device mnemonic and retype. If the error occurred as a result of a command file, check for logical device assignments.

INVNAM - Invalid name. No special characters are allowed (A through Z and 0 through 9 only). This error will also occur if asterisks or wild character constructions are used with a command which does not support them (i.e., PIP). Check the command file name field.

INVSW - An invalid switch was used in the command string. Refer to Table 2.

NEXFIL - The file specified in the command string does not exist. Check the directory of the device.

GENERAL INFORMATION

Code	DGQFB.A11
Title	KL10 Diagnostic Memory Boot Utility
Abstract	This program provides all of the functions necessary to configure the KL10 memory system when running in the front-end resident, KL10 diagnostic environment. This program runs in the PDP-11 under KLDCP. KL10 memory types handled include external core memory (DMA20), internal core memory (MA20/MB20), and MOS memory (MF20). All reasonable mixtures of these devices can be handled together.

The memory boot procedure goes in three basic steps.

1. **Determining physical resources - RESDET -** Determining physical resources, or "RESDET" for short, is the longest and most involved part of the memory boot procedure. Different procedures occur for different memory types, but basically the program determines what physical memory it has to work with. Listed below are the things the program must do for each memory class.

   Internal Core Memory - MA20 and MB20 Controllers
   Find out which controllers, if any, exist. Find out which storage modules exist on each controller. Determine the set of legal starting addresses and the interleave mode for a controller or controller pair.

   External Core Memory - DMA20 Controller
   Find out if the DMA20 exists. Determine its address response(s) and the size of the response(s). Determine the legal interleave modes available. The address response(s) of external memory are fixed and the program must work around whatever it is.

   MOS Memory - MF20 Controllers
   This is very different from the core memories. In addition to finding out what exists the program must also find out the state of the controllers. Because MOS RAMs fail on a regular basis there is a lot of hardware in the controllers to compensate for these failures. The software closely controls the hardware and it is therefore important that the program knows what has already been done.

   If the controller is already configured (it is at software state 2 or 3), then the program treats it as if the address response could not be changed. In this sense it is treated like external core. However, if the program finds some bad hardware, that hardware is eliminated.

   If the controller is not configured but is otherwise initialized (it is at software state 1), the program merely records what storage it has to work with.

   If the controller has not been initialized at all (is at software state 0), then the program has a considerable amount of initialization to do. The double bit error (DBE) scan is by far the most time consuming part of the memory boot process, taking about 22 seconds per 256K of MOS RAM. Fortunately, once this is done the controller is at software state 1 and the DBE scan does not have to be done again until the next power fail. MOS storage blocks found to be irreparably bad are eliminated.

2. Determination of Logical Configuration - **FITMEM** - In this phase the program determines which configurable resources (MA20, MB20, and software state 1 MF20) will go where in the holes in the address space. Hole locations and sizes are determined by the response of the external core memory, preconfigured (software state 2) MF20 memory, and the absolute bounds of the memory space. This process does not involve the hardware at all; it is purely computational.

   The philosophy behind this algorithm is to maximize storage even at the cost of some interleave factor. No memory is ever thrown away except for certain impossible-to-configure conditions which might arise with MA20s or MB20s.

3. Configuration of the Memory - CONFIG - Here the program takes the logical configuration tables and sets up the hardware to match. After this phase is completed, the KL10 memory system is ready for use.

Notes

Memory controllers are assumed to have passed their respective diagnostics (DGKBB/DHKBB, DHKBF, and/or DHKBG).

It is assumed that the KL10 processor is working and that some valid microcode is already loaded.

There must be master oscillator if MOS memory exists.

When MEMCON is started it will do a start microcode in order to make sure that microcode is loaded and running. Because of this, any special state which may have existed in the CPU will be lost.

Loading and Starting Procedure

Standard (Refer to the 11/10 STD module.)

Control Switches

None

OPERATIONAL CONTROL
Once started, MEMCON will prompt with a > (TAB). The user may then enter commands. There are two classes of commands: those involved with configuring the memory system (Table 1) and those which perform functions ancillary to using the memory system (Table 2). KLDCP commands may be entered directly if no naming conflict occurs. Preceding a command with a period ensures that KLDCP will process it. Example: ".RP0" selects RP04/RP06 drive 0 for KLDCP, whereas "RP0" says to report the physical resources.

ERROR MESSAGE SUMMARY
There are no error messages unique to MEMCON.

Table 1 MEMCON Memory Configuration Command Summary

Command	**Description**
CM	CM<CR> or CMF<CR> etc. Determine, report, and set the configuration. Then clear the memory boot. Memory is now configured and ready for use. All physical resource data has been cleared out. This command will automatically do the physical resource determination if it has not already been done, and is therefore the only essential command for configuring memory. See switches.
DL	DL<CR> Determine logical configuration, report it, but do not set it. This command is useful for seeing what the configuration would be if it were set. See switches.

Table 1 MEMCON Memory Configuration Command Summary (Cont)

Command	Description
DP	DP<CR> Determine the physical resources and report them. This forces the memory boot to start from scratch. Time already spent on MF20 DBE scan is not lost providing the previous scan ran to completion. See switches.
Switches	The switches are the same for DP, DL, and CM commands. Typing no switch will use the switches typed for the previous DP, DL, or CM command. If there was no previous DP, DL, or CM command, then the defaults are as shown below. The switches may be in any order.
0,1,2, or 4	Force MA20/MB20 interleave unless memory loss would result. Force DMA20 bus mode if legal. 0 (default) gives optimal results.
F	Force MF20 address reconfigure. In this mode preconfigured MF20s are always deconfigured before the memory resource fit is done. This is not the default. While the "F" parameter to the CM command is not the default, most of the time while in the diagnostic environment the user will want to use it. The recommended minimum command is therefore "CMF".
K	Keep bad MF20 blocks. Normally MF20 blocks which are irreparably bad to the memory boot can still be used partially by monitor if it marks certain pages as unusable. After a brief power fail, monitor should still have this bad page data intact; therefore it is safe to tell the memory boot to keep bad MF20 blocks. This is not the default. Ignored if "F" switch given.
R	Reverse configuration where possible. This is useful for shuffling memory around for diagnostic reasons. It is not normally used otherwise. The "F" switch should be used if this switch is given.
Sn	Substitute MF20 spare bits for bit n (decimal) in all MF20s. This is useful for fixing MOS array boards. The number n must be followed by a space or <CR>. No parameter says to force no swaps. The value of n is 0-35 for data bits, and is 36-42 for ECC 32, 16, 8, 4, 2, 1, and parity.

Table 2 MEMCON Ancillary Command Summary

Command	Description
↑C	Exit back to KLDCP.
↑Z	Exit back to KLDCP.
CO	CO<CR> Clear all function 0 error flags. Use before first DP, DL, or CM command and after diagnostics which intentionally cause memory errors.
DA	DA<CR> Dump PC, VMA, previous and current AC block numbers, and the contents of AC blocks 0-6. Very useful data to accompany a diagnostic bug report. The code to do this command resides in the overlay DBGOVL.A11.

Table 2 MEMCON Ancillary Command Summary (Cont)

Command	Description
DR	DRx n<CR> Dump the content of the MF20 logic control RAM "x" to the console terminal. The meaning of x is "A" for address response RAM, "B" for bit substitution RAM, "E" for fixed value logic RAM, or "T" for the timing RAM. Note that if refresh is running it may interfere slightly with a timing RAM dump. The value of n is the MF20 controller number in the range 10-17.
IC	IC n<CR> Force an initialization of the specified MF20 n. This performs the minimum initialization required to talk to the MF20. The address response RAM is set up so that address bits 18-21 determine which block is being used.
KP	KP c1, c2 s1, s2<CR> Kill physical resources s1 through s2 in memory controllers c1 through c2. This command is used after the DP command. Its purpose is to get rid of storage resources that are not to be used; (i.e., they do not work). s1 and s2 are storage module numbers for MA20s and MB20s, and octal block numbers (0-13) for MF20s.
MO	MO n<CR> Select master oscillator frequency source, where n: =3 for normal (30 MHz); n = 2 for slow (25 MHz, which is for extending a board); n = 1 for fast (31 MHz, for margining the system); n = 0 is external oscillator. Do not use 0 unless a running VFO has been physically attached to the external oscillator input. Meaningless if there is no master oscillator.
PD	PD<CR> Enter a program patching dialogue where the address and content of that address are typed, and then the value the user types in goes to that address. Typing <CR> causes the data to remain the same. Typing <ESC> causes the patcher to ask for a new address. Typing <ESC> to the address enquiry causes exit. The first address used by the patcher is the first free location. The first free pointer is automatically updated as required. The code to do this command resides in the overlay DBGOVL.A11.
RI	RI<CR> Reinitialize the memory boot. The various switches and control flags are put back the way they were when the program was first loaded.
RP	RP<CR> Report physical resources. This command does not do anything other than report the content of the physical resource tables. It is useful after using the KP command to find out if an error has been made.
SD	SD w<CR> Take the 36-bit word "w" and use it as the "to MEM" word of an SBUS DIAG cycle and type the word sent back "from MEM". If the SBDIAG instruction fails then nothing is typed.
SR	SR<CR> Do an SBUS reset.
TC	TC<CR> Test configuration. This must only be done after the CM command. The test consists of reading words 20-23 on every 16K boundary. The response of all NXMs or no NXMs is then compared to what the program thinks it should have at a given address.

GENERAL INFORMATION

Code	DGQFA.All(KL10-PA) and DHQFA.All(KL10-PV)
Title	TRACON-KL10 Diagnostic Console Signal Tracer
Abstract	TRACON resides in the lower half of the 11/40 core. It extends the console command set of KLDCP and aids in troubleshooting KL10 central processor, channel and memory faults. TRACON commands primarily control the CPU clock, and detect and display changes in registers and control signals.
Notes	1. TRACON commands prompt for missing arguments. Responding to a prompt with an altmode ($) will abort the command. 2. KLDCP commands may be executed from TRACON by preceding the command with a period (.). 3. System standard or diagnostic microcode must be loaded in the KL10.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	None

OPERATIONAL CONTROL

TRACON commands may be entered directly from the CTY or KLINIK link or indirectly from a control file.

TRACON commands are divided into two groups: control functions which are described in Tables 1 and 2, and extension commands which are described in Table 3.

Control functions affect TRACON's mode of operation and should not be used in control files. Extension commands are intended for general use and may be included in control files.

ERROR MESSAGE SUMMARY

There are no error messages unique to TRACON.

Table 1 TRACON Control Function Summary

Command	Description	Cross Ref.
A	A<CR> Auto insert - automatically builds an internal command file as commands are typed.	1
E	E<CR> Edit or create a command buffer. Refer to Table 2.	2
ML	ML<CR> Mark loop "starting point"	3
FB	FB 162,31,1<CR> Set function breakpoint at the diagnostic function, bit, and polarity specified.	4
FC	FC<CR> Function break continue	4
CB	CB<CR> Clear function breakpoint	4
RG	RG<CR> Print function breakpoint registers (R0 through R7)	4
KA	KA<CR> Kill (terminate) auto insert; also resets loop marker to line 1	5
T	T<CR> Type contents of command buffer	6
X	X<CR> Execute command buffer	7
L	L<CR> Loop on command file	8
M	M<CR> Multi-burst, step, and trace the command buffer	9
DC	DC CHAN.TST<CR> Write command buffer to an existing file	10
LC	LC CHAN.TST<CR> Load specified control file	11
K	K<CR> Kill command buffer (confirm with a K)	12
H	H<CR> Print command summary	13
/	/<CR> Enter switch dialogue	14

Table 2 TRACON Edit Command Summary

Command	Description	Cross Ref.
E	E<CR> Enter lines into buffer.	15
D #	D 5<CR> Delete specified line (#) from command buffer.	16
I # text	I 7 SET CHAN 3<CR> Insert text before specified line number (#).	17
R # text	R 14 SC 2, START<CR> Replace text at specified line number.	18
K	K<CR> Kill the command buffer (confirm with a K).	19
T	T<CR> Type the contents of the buffer.	20
↑C	↑C CTRL C - Exit from edit mode; return to TRACON command mode.	21

Table 3 TRACON Extension Command Summary

Command	Description	Cross Ref.
SET mode	SET EBR 3,CHAN 1<CR> Set: CACHE EN, PMA, EBR # and/or CHAN #.	22
CLR mode	CLR CACHE,ERB<CR> Clear: CACHE EN, PMA, and/or CHAN #.	23
RM	RM<CR> Reset MBox (force halt loop and set cache look and load if cache is enabled).	24
CE chan,ccw	CE 2,100<CR> Configure EPT for channel specified.	25
SC chan,cmd	SC 1,STA,RES<CR> Simulate CBus command for channel and command or for EBus data specified. Commands are: START, RESET, CTOM, DONE, STORE and SLOW.	26
QC chan,cmd	QC 1,STA,RES<CR> Queue CBus command for memory trace.	27
T1	T1<CR> Trace and print one memory reference (used with the QC command).	28
TM	TM<CR> Trace and print all memory references (used with the QC command).	29
CH	CH<CR> Print default channel number.	30
NC	NC<CR> Next channel (increment the default channel number by 1).	31
CU	CU<CR> Cache refill load (standard).	32
C #	C 3<CR> Cache refill load (use only Cache specified: 0, 1, 2, or 3).	33
IC	IC<CR> Invalidate Cache (use after refill load).	34
VC	VC<CR> Validate core from cache.	35
I	I<CR> Initialize the tick counter to 0.	36
B #	B 29<CR> Burst specified number (#) of clock ticks and report change.	37
C	C<CR> Continue advancing clock.	38
F #	F 14<CR> Find the clock tick # specified.	39
G	G<CR> Go - reset tick counter, stop the clock and print machine state changes.	40
S	S<CR> Single-pulse the clock and report machine state changes.	41

Table 3 TRACON Extension Command Summary (Cont)

Command	Description	Cross Ref.
P	P<CR> Print EBus activity summary since last P or D command.	42
R	R<CR> Read and print machine state changes since they were last reported.	43
D	D<CR> Print the current state of the machine.	44
W filename	W CRASH<CR> Write a crash dump - must specify an existing file.	45
D filename	D CRASH<CR> Print the machine state saved by the W command.	46
EM addr	EM 2000<CR> Examine KL10 address (does not use PI system and the KL10 must be halted).	47
EN or EM	EN<CR> EM:<CR> Examine next sequential KL10 address.	47
D addr:data	D 2000:254000,020000<CR> Deposit data into KL10 address (does not use PI system and the KL10 must be halted).	47
DN:data	DN:254000,001472<CR> Deposit data into next sequential KL10 address.	47
EX inst.	EX 201000,777777<CR> Execute KL10 instruction.	48

TRACON COMMAND DESCRIPTION
This section describes in detail each of the commands summarized in Table 1, Table 2, and Table 3.

1. A<CR> - The A command opens the command buffer for input. All commands typed following an A command are entered into the buffer until a KA command is typed. The commands in the buffer are executed via the X, L or M command. The buffer may be saved for future reference with the DC command.

   NOTES
   1. KLDCP commands may be used in the command buffer.

   2. Commands are automatically parsed before they are entered in the command buffer. For this reason it may be necessary to reconstruct a command for inspection.

2. E<CR> - The E command enters edit mode. The editor may be used to create or edit the command buffer. Edit commands are summarized in Table 2.

3. ML<CR> - The Mark Loop command requests a line number for use with the L command.

4. Function Breakpoint Command - A function breakpoint is a mechanism which permits detection of an event (signal) in the KL10. The clock will be stopped when the leading edge of the event is detected. The event is specified by entering a diagnostic function code, a bit number and a 1 or 0 to select the polarity desired. Once set, the KL10 clock will be stopped and the user notified each time the signal specified transitions to the state selected. Only one function breakpoint may be set at a time. Since this mechanism depends on single-pulsing the clock through the function being performed, only extension commands are affected.

   FB 166,30,1<CR> - Set a breakpoint for diagnostic function 166 bit 30 on a 1 (MEMRQ 1 H). The clock will be stopped on the leading edge of MEMRQ 1 H. Other commands can now be used to read the state of the machine.

   FC<CR> - Continue the command execution until either the next detection of the break condition, the end of the current extension command, or the end of the command buffer.

   CB<CR> - Clear the function break condition set by the FB command.

   RG<CR> - Print the contents of the function break registers R0 through R7.

5. KA<CR> - The KA command performs two functions: it terminates auto insert (A), and it resets the loop marker (LM) to line 1.

6. T<CR> - The T command prints the contents of the command buffer.

   NOTE

   The commands in the buffer are automatically parsed. Therefore the commands may be printed in a slightly different format.

7. X<CR> - The X command executes the contents of the command buffer once.

8. L<CR> - The L command repeatedly executes (loops on) the commands in the buffer. After the first execution of the buffer, execution begins at the line specified by the loop marker (ML). If no ML command has been executed the loop marker defaults to line 1.

9. M<CR> - The M command:

   a. clears the tick counter
   b. burst-executes the command buffer
   c. prints the state of the machine, and
   d. increments the tick counter by 1.

   Steps b through d are repeated until the user interrupts by typing an altmode ($), or until the EBox enters the halt loop, or until no change in machine state is detected for a prespecified number of ticks (refer to TRACON Switches Number 14). The M command, in effect, allows the command buffer to be executed at full speed while printing the machine state at each tick.

   NOTES

   1. The first commands in the buffer must initialize the CPU to an exact known state. Otherwise, the reported changes will be garbaged beyond usefulness.

   2. The C command may be used to continue the trace if it was stopped with an altmode ($).

10. DC CHAN.TST<CR> - The DC command writes the contents of the command buffer to the specified file (CHAN.TST). The file must already exist on the output device.

    NOTE
    A temporary file can be generated using the KLDCPU ALLOC command.

11. LC CHAN.TST<CR> - The LC command loads the specified control file (CHAN.TST) into the command area of core.

12. K<CR> - The K command clears the command buffer. TRACON requires the K command be confirmed by typing a second K.

13. H<CR> - The H command prints a summary of TRACON commands.

14. /<CR> - The / command allows the user to specify groups of registers and signals to be traced. Each group is divided into subgroups which may be turned on or off. The groups are as follows.

    Signals

    EBOX - PI, MCL, CLK, DIA, CTL, CON, MTR, SCD, VMA, CRA

    MBOX - CSH, CHX, MBC, MBX, MBZ

    CHAN - CCL, CH, CCW, CRC

    CYCLIC - Any signals which change frequently. The current list is as follows.

    ```
    EBUS CLK
    SBUS CLK
    EBOX SOURCE
    SYNC
    EBOX CLK
    A CHANGE COMING A
    B CHANGE COMING
    PHASE CHANGE COMING
    ```

    Registers

MICRO	DRAM ABJP, CRA LOC, CR ADR, SBR RET, CRAM NN, DISP, IR, AC, TRAP MIX
DATA/ADDR	PIH, PIO, PI GEN, VMAH OR PC, CLK BURST, FM BLOCK & ADR, AR, ARX, BR, BRX, AD, ADX, FM, MQ, SC, FE, VMA, VMAH, ADR BRK, PC, EBUS REG
METER	CACHE COUNT, EBOX COUNT, INTERVAL, PERF COUNT, PERIOD, TIME
CHAN ADDR	CCW CHA, CH BUF ADR

    The NO CHANGE LIMIT may also be altered with the / command. The limit is used to stop a trace after a specified number of clock ticks with no observed changes in machine state. The current limit is output and the user may enter a new number or a carriage return if the limit is satisfactory.

15. !E<CR> - Enters lines into buffer. The user types E<CR> and the editor outputs a line number at the left margin. After an initial load or an editor K command, the first number output will be 1. If the buffer contains information, the next free line's number is output. After each number, the user enters any extension or console command. Prompting is enabled. No validity checking occurs for console commands. To terminate entry, type an altmode following the line number output.

Example:

```
* E<CR>
!E<CR>
1       RM<CR>                  ;Reset MBox
2       SET EBR 3<CR>           ;Set executive base
                                ;register to 3
3       CE 0,200000 100<CR>     ;Condition channel 0 EPT
4       .DM 100:0<CR>           ;Put a CHLT in command list
5       QC 0.START/RESET<CR>    ;Start channel 0
6       TM<CR>                  ;Watch it fetch a halt
7       $<ALTMODE>              ;Exit-enter command mode
```

16. !D # - Deletes the line # and renumbers all the lines which follow it. (Line numbers are not "sticky;" if needed, use the T command to type all line numbers and their current contents.)

17. !I # <TEXT> - Insert text before line number (#). All lines starting from # are moved down and the text inserted in the resulting hole. As in the D command, lines are renumbered.

18. !R # <TEXT> - Replace text at line # with new text.

19. !K - Kill the buffer. Resets the line count to 0 and recovers the buffer storage space. Confirm with K.

20. !T - Type out the buffer. Types line numbers and text.

21. !↑C - Exit from editor mode to TRACON command mode.

22. SET - The set command alters the operating mode of TRACON, and modifies the performance of the RM command so that the function(s) set is repeated each time the RM command is executed.

    SET CHAN #<CR> - Sets the default channel number to (#) for the CE, SC, and QC commands. Once a channel number has been set, prompting for channel numbers will not occur.

    SET CACHE EN<CR> - Sets cache look and load.

    SET PMA<CR> - Forces the PMA (physical memory address) to the error address register.

    SET EBR #<CR> - Loads the executive base register with the number (#) specified.

    NOTE

    Channel diagnostics always set the EBR to 3.

23. CLR - The CLR command is the complement of the SET command.

    CLR CHAN #<CR> - Eliminates the default channel and reinstates channel number prompting.

    CLR CACHE EN<CR> - Disables cache look and load.

    CLR PMA<CR> - Discontinues the forcing of the PMA to the error address register.

    CLR EBR - Not implemented.

24. RM<CR> - The RM command performs a master reset, clears the diagnostic CRAM address register, and performs 35 MBox clocks. The RM command is similar to the KLDCP SM command except the clock is not left running.

    NOTE

    Functions set by the SET command are also performed each time the RM command is executed.

25. CE 2,100<CR> - The CE command deposits the "initial command word" specified (100) in the executive page table (EPT) for the channel specified (2). The location will be the executive base register (EBR) location specified by a SET command plus four times the channel number. The next location, STATUS 1 will be cleared.

26. SC 1,START<CR> - The SC command uses the diagnostic function write 70 (FW 70 DATA) to simulate a command from the RH20. The data to be used for the write function may be specified as a 36-bit word (DIAG FUNCT 70) or as the signal mnemonic.

EBus Bit	Mnemonic
06	RESET
07	START
09	DONE
10	CTOM
11	STORE
12	SLOW REQ

Channel timing is synchronized to the proper scan point as a function of the SC command.

NOTE
The SC command should not be used in conjunction with the TM command.

27. QC 1,STA,RES<CR> - The QC command sets up a list of CBus commands for later execution. The purpose of the QC command is to defer CBus activity until a T1 or TM command is executed. (The memory reference trace feature provided by this command may miss printing some memory references unless the timing of the channel scan is coordinated with the memory trace.)

NOTE
The QC command accepts the same arguments as the SC command.

28. T1<CR> - The T1 command traces and prints memory references one at a time so that timing synchronization of CBus requests may be provided.

NOTES
1. **The T1 command causes the timing to revert to single-pulse mode.**
2. **The T1 command is normally used in conjunction with the QC command.**

29. TM<CR> - The TM command traces and prints the condition of memory requests and the physical memory address at each SBus address hold time.

NOTE
Notes 1 and 2 under the T1 command apply to the TM command as well.

30. CH<CR> - The CH command prints the default channel number selected by the SET command. Prints NO DEFAULT if prompting for the channel is in effect.

31. NC<CR> - The NC command updates the default channel selected. If no channel default has been set, the default will be channel 0; otherwise, the channel will be incremented. An error indication will be typed if an attempt is made to default to a channel greater than 7.

32. CU<CR> - The CU command uses the standard cache look and load algorithm (least recently referenced data is overwritten). All four caches are loaded.

NOTE
The CU command should be immediately followed by an IC command.

33. C #<CR> - The CU command uses the standard cache look and load algorithm (least recently referenced data is overwritten). However, only the specified cache (#) is loaded.

    NOTE
    The C# command should be immediately followed by an IC command.

34. IC<CR> - The IC command invalidates the contents of cache (clears cache valid bit).

35. VC<CR> - The VC (validate core) command writes the contents of cache to core.

36. I<CR> - The I command sets the clock step (tick) counter to 0.

37. B 29<CR> - The B command bursts the clock the specified number of times (29) and prints the difference between the initial and final state of the machine.

38. C<CR> - The C command continues the clock and prints the machine state changes at each tick. This is accomplished by single-stepping (if the trace was initiated by a G command) or by incremental bursting (if the trace was initiated by an M command). In both cases, the initial state of the machine is assumed to be that stored from the last interrupted G or M command.

    NOTE
    Typing an altmode ($) during a trace printout will stop the printout at the end of the current line. Typing a C command will continue the printout.

39. F 14<CR> - The F command single-clocks the CPU the specified number of ticks (14) and prints the difference between the initial state of the machine and the state of the machine after the final (14th) tick.

40. G<CR> - The G command:

    a. resets the tick counter to 0

    b. reads the initial state of the machine

    c. steps the clock once

    d. reads the new state, and

    e. compares the previous state against the new state and prints the difference.

    Steps 3 through 5 are continuously repeated until the user interrupts by typing an altmode ($), or until the EBox transitions to a halted state, or until no changes are detected within a specified number of ticks. (Refer to TRACON Switches Number 14.)

41. S<CR> - The S command single-pulses the clock and prints the machine state changes.

42. P<CR> - The P command prints an EBus bit activity summary and resets the EBus bit activity accumulator registers. Two accumulators are kept for each group of eight diagnostic read functions (i.e., 100-107, 110-117, 120-127, etc.). One accumulator maintains a logical AND for that group; the other, a logical OR. The P command prints out all these accumulators by diagnostic "read function group" plus a total accumulation for all 64 diagnostic functions. In the AND word, if a bit is a 1, then it was always high; in the OR word a 0 bit was always low. (This should be the case with any bits not assigned to a diagnostic function read group.)

43. R<CR> - The R command reads the current state of the machine, compares it against the previously stored state, and prints the difference. The R command allows the user to execute KLDCP commands and then monitor the machine state change (i.e., execute a KLDCP command followed by an R command).

44. D<CR> - The D command reads and prints the current state of the machine. It also prints the EBus bit statistics and resets the accumulators as in the P command.

45. W CRASH<CR> - The W command writes a crash file for later use. The file specified (CRASH) must already exist on the output device.

46. D CRASH<CR> - The D command reads in and prints the file (CRASH) saved by the W command.

47. Examine and Deposit Commands - Unlike the KLDCP examine deposit commands, the TRACON examine/deposit commands do not use the PI system and do require that the KL10 be halted. They are implemented by executing instructions from the AR which load the ACs and move data to and from memory.

NOTE

Because prompting is in force, a second carriage return is required to reexamine the last address used.

48. EX instruction<CR> - The EX command causes the instruction specified to be placed in the AR and executed by the KL10. This command is similar to the EX command supported by KLDCP; however, breakpoint function may be used with the TRACON version.

GENERAL INFORMATION

Code	DGDTE.A11
Title	DTE20 KL10 to PDP-11 Front End Interface Diagnostic
Abstract	DGDTE is the basic DTE20 KL10 to PDP-11 front end interface test. The program can test any one of four DTE20s (device codes 174400-174436, 174440-174476, 174500-174536, or 174540-174576) connected to a single PDP-11. The test is incremental in nature. It requires no stimulus from the KL10, and executes approximately three passes per second.
Hardware Required	KL10-PA or -PV mainframe
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	During the first pass, the program interrogates the operator for a few test parameters. It also provides periodic progress reports during the first pass as it attacks critical logic areas. On subsequent passes the operator interrogation and progress reports are inhibited.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGDTE TEST SUMMARY
The individual tests peformed by this program are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGDTE Test Summary

Test	Description		
Address Testing			
1	ADDR3 takes the address to be tested (as determined by SELAD2) and individually complements the high-order nine bits. Using this modified address, the program expects an instruction timeout. Failures in this test are caused by malfunctions of the selection logic on the DPS1 print at B-2 B-5 (DSP1 UB ADR SEL H).		
Reset Testing			
2	REST1 performs a reset instruction, reads the 10/11 status register into actual, and checks that the following status flip-flops are clear.		
	DPS5 11 TO10 NORM TERM	(DPS5 C-4)	bit 15
	DPS5 11 TO 10 ERR TERM	(DPS5 C-7)	bit 13
	DPS5 PAR ERR	(DPS5 D-6)	bit 9
	DPS5 REQ 10 INT	(DPS5 B-5)	bit 8
	DPS5 11 TO11 NORM TERM	(DPS5 C-5)	bit 7
	DPS5 NULL STOP	(DPS5 D-4)	bit 5
	DPS5 EBUS PARITY ERROR	(DPS5 B-2)	bit 4
	DPS5 11 TO11 ERR TERM	(DPS5 C-6)	bit 1
	Massive failures in this test probably indicate a failure of the master clear signal; individual failures may be stuck flip-flops.		
Diagnostic Register Testing (Basic)			
3	DIA1 is similar to the previous test except this time diagnostic word 1 is checked for 0s after the reset instruction. The tested logic is contained mostly on the CNT3 print. This test primarily checks the CNT3 Unibus transceivers.		
4	ADDR4 - Knowing that it is possible to address the DTE20 status register without an instruction timeout, ADDR4 next tries to address diagnostic word 1.		
5	ADDR5 is similar to ADDR4 except that it attempts to address diagnostic word 2.		
6	ADDR6 is similar to ADDR4 and ADDR5 except it tries to address diagnostic word 3.		
Status Register Checking (Basic)			
7	STAT1 tries to set the following status flip-flops.		
	DPS5 11 TO10 NORM TERM	(DPS5 C-4)	bit 15
	DPS5 11 TO10 ERR TERM	(DPS5 C-7)	bit 13
	DPS5 10 REQ INT	(DPS5 D-7)	bit 11
	DPS5 REQ 10 INT	(DPS5 B-5)	bit 8
	DPS5 11 TO11 NORM TERM	(DPS5 C-5)	bit 7
	DPS5 EBUS PARITY ERROR	(DPS5 B-2)	bit 4
	DPS5 11 TO11 ERR TERM	(DPS5 C-6)	bit 1
	DPS5 11 INTERRUPT EN	(DPS5 B-4)	bit 0
	Individual failures may be caused by the flip-flops stuck at 0 or Unibus transceiver faults. Massive failures are probably caused by faulty selection or Unibus control signals.		
8	Knowing that the seven status flip-flops can set, STAT2 tries to clear them with a reset instruction.		
9	STAT3 compares the effect of the diagnostic reset function to that of the PDP-11 reset instruction. Since the reset instruction clears all seven flip-flops, only one of them needs to be tested (DPS5 11 TO10 ERR TERM).		
10	STAT4 tries to clear TO10 DONE, TO10 ERROR, 10 REQ INT, 11 to 10 NORM TERM, 11 TO10 ERR TERM, 10 REQ INT, 11 TO11 NORM TERM, and 11 INTERRUPT ENABLE via the individual reset gates with bits 14, 12, 10, 06, 03, and 00.		

Table 1 DGDTE Test Summary (Cont)

Test	Description
Restricted/Privileged Mode Testing	
11, 12 13, 14	RM1-4 checks the restricted mode bit found in the status register. RM1 reports the condition of the bit in the first pass. If the tested DTE20 is privileged, the program does a limited amount of diagnostic bus testing leading up to the issuance of a master reset in test RM4. If the tested DTE20 is restricted, the four tests are dummies.
Interrupt Testing	
15	INT1 verifies that the DTE20 can interrupt the PDP-11.
16, 17 18, 19, 20	Once INTI1 has shown that it is possible to get a vectored interrupt from the 10/11 interface, INT2A-INT2E test to make sure that each of the functions which can cause an interrupt is operating properly. 1 TO10 DONE 2 10 REQ INT 3 TO11 DONE 4 TO10 ER 5 TO11 ER
21	DIAG2 tries to set and clear 10/11 diagnostic mode.
Clock Testing	
22	This is the beginning of the clock testing. CLK1 simply reads diagnostic word 1 200(8) times and expects to see each of the major state flip-flops set at least once.
23	CLK2 sets 10/11 diagnostic mode, which should set CNT4 INH CLK. CNT4 INH CLK should inhibit gated clock pulses, which will prevent the major state flip-flops from cycling.
24	CLK3 sets 10/11 diagnostic mode, detects which major state is on, and single-pulses the clock to see if the major states advance correctly. Since it is already known that the major states all set, one single pulse is adequate to check the single-pulse logic.
25	CLK4 checks the clock state hold flip-flop. The program sets 10/11 diagnostic mode and DS05. It next single-pulses the clock and checks to see that the major state flip-flops do not advance.
26	CLK9 makes sure that each major state is present by itself, and that the major state counter advances correctly from DEX to TO11 transfer, to TO10 transfer, and back to DEX.
RAM Testing	
27	RAM testing begins here. Up to this point, the program has not addressed any of the twelve active RAM locations. A fair amount of untested logic (principally the CNT1 functions leading to CNT1 RAM CYC) is used for the first time. RAM1 tries to address all twelve RAM locations and uses an instruction timeout as an error indicator.
28	RAM11 simply tries to write all 0s into the delay counter and read them back. Note that this is the first time we try to read anything from the RAM.
29	RAM2 writes 0s in all active RAM locations and reads them back. Two errors are possible: RM2ER1 if non-0s are returned, and RM2ER2 if 0s are returned but the function RFM=0 (available in status register) is returned false.
30	RAM22 is similar to RAM11 except this time all 1s are written into the delay counter.
31	RAM3 is similar to RAM2 except this time all 1s are written into the twelve RAM locations.

Table 1 DGDTE Test Summary (Cont)

Test	Description
32	RAM4 makes sure that CNT1 SWAP H is not stuck true. It loads the delay count with 0000377 and verifies that it does not read as 177400.
33	RAM5 is the first serious RAM data test. It fills RAM with all 1s. It then reads 1s from each location and writes 0s back. Addressing problems will cause 0s to appear where 1s are expected.
34	RAM6 is a typical memory address test. The first RAM location (the delay count) is loaded with four 4-bit bytes of 0s, the second location (DEX word 1) gets four 4-bit bytes of 2, the third 4, etc. By using four 4-bit bytes, a measure of isolation to the four individual 7489 RAM chips is provided.
35	RAM7 is a simple test of the RAM 0 detection logic (RAM=0). RAM2 has already checked to see that RAM=0 will set when all 0s are read from the RAM. RAM7 will make sure that it does not set with a floating 1 coming out of the RAM.
State Count Testing	
36	SC1 locks the 10/11 interface in DEX mode and loads the state count with a binary count of 0-4. It reads the RAM address associated with each of the DEX minor states.
37	SC20 locks the interface in DEX state, loads a state count of 17(8), and single-pulses the clock. It then checks the RAM address bits to ensure that the single-pulse logic works, and that the state counter can produce the correct DEX minor state. Failures in the test usually indicate failures in the 74193 up-counter chip.
38	SC21 locks the interface in DEX mode, forces a minor state count of 00, pulses the clock, and checks (via reading the RFM AD bits from DIAG3) to see if the count has advanced to 01. If this works, the program again pulses the clock and reads the RFM address bits to make sure the count does not advance to 02 (the advance should be inhibited since INT1 BUS COMP(0) should hold CNT4 INH CLK SET).
39	SC21A - Having shown that the 74193 state counter can advance to minor state DEX ADR2, the program will next attempt to single-pulse to minor state DEX WD1. The incrementation of the minor state counter should be inhibited since the INT1 BUS COMP flip-flop being reset should hold the CNT4 INH CLK flip-flop set.
40	SC22 locks the interface in DEX mode and forces minor state DEX ADR2 true with the EB DONE SET bit true. The test then pulses the clock to see if the minor state will advance to DEX WD1. If this works, the routine continues pulsing the clock, checking for decodes DEX WD2 and DEX WD3.
41	SC3 is similar to SC1 except this time, the program locks the interface in the TO10 major state and checks the RAM addressing.
42	SC4 is similar to SC2 except that this time the program single-steps through a TO10 transfer. The routine bypasses a count of 6 (which causes an untested NPR request) and increments to 10(8).
43	SC5, like SC1 and SC3, locks the interface in a major state (this time TO11 transfer) and applies various input to the minor state counter, observing the RAM address bits via diagnostic word 3.
44	SC6 checks the incrementation of the minor state counter from 10(8) TO11(8) during a TO11 transfer.

Table 1 DGDTE Test Summary (Cont)

Test	Description
Operation Initiation Testing	
	The next two tests (OPR1 and OPR2) try to initiate DEX and TO11 transfer operations (the initiation of a TO10 transfer is checked late in test EBH1).
45	OPR1 locks the interface in 10/11 diagnostic mode, single-pulses into the DEX major state, loads DEX address 2 (to set CNT5 DEX start), pulses the clock, and checks to see that the major state has not advanced.
46	OPR2 is similar to OPR1 except that this time the program tries to initiate a TO11 transfer. The routine puts the interface in the TO11 transfer major state, loads the TO11 address and byte count (which should set CNT5 TO11 RDY), pulses the clock and checks to see if the interface remains locked in the TO11 major state.
ABC Register Testing	
47	ABC1 checks to see if the ABC register can hold 0s. The routine proceeds as follows. a. Load 1s into the TO11 11 address. b. Load 0s into the delay count. c. Lock the interface in the TO11 transfer major state. d. Set minor state TO11 DLY RD. e. Pulse the clock (transfer delay count to ABC register). f. Set minor state TO11 ADR INC. g. Pulse the clock (transfer ABC to TO11 byte count). h. Compare TO11 11 address for all 0s.
48	ABC2 is similar to ABC1 except that it checks to see if the ABC register can hold 1s.
49	ABC3 checks ABC register incrementation. Using a data pattern of 00, 01, 03, 07, 17, etc., the program loads the ABC register. After incrementation, the program reads the incremented count and checks for outputs of 01, 02, 04, 10, 20, etc.
50	The previous test (ABC3) used the TO11 input to CNT1 DLYINC to increment the ABC register. ABC4 checks the TO10 input to CNT1 DLY INC.
51	ABC5 checks the CNT4 TO11 ADR ADD input to CNT1 ABC INC. The program loads the TO11 11 address location in the RAM with 1 (to set TO11 word), increments, and checks to see if the 1 is still there.
52	ABC6 is similar to ABC5 but checks the CNT4 TO10 EBUF FILL H input to CNT1 ABC INCL.
NPR Testing	
53	NPR1 tests the basic NPR interrupts circuit.
54	NPR2 loads the TO11 byte count with minus 1 (which sets CNT5 TO11 BC LD). It performs a master clear, loads the TO11 11 address, pauses for a while, and checks to see that the TO11 address is not modified. If it has been incremented, it means that the NPR occurred. This implies that CNT5 TO11 BCLD was not cleared by master clear.
55	NPR3 does another NPR transfer (again with a byte count of minus 1) and checks the TO11 addresss to determine if it is incremented during minor state TO11 FILE READ. Since bit 0 of the TO11 11 address is 1, the extra increment of the ABC register at minor state TO11 ADR ADD should not occur.

Table 1 DGDTE Test Summary (Cont)

Test	Description
56	NPR3A does another single-byte TO11 transfer. The program clears the NPR target address, loads the TO11 data word with sixteen 1s (-1), and steps through a TO11 transfer. At the completion of this operation, the data in the NPR target address is compared to a word of 377(8) (eight 0s and eight 1s).
57	NPR3B is similar to NPR3A. This time the routine does a full-word transfer and checks that two 8-bit bytes are delivered to PDP-11.
58	NPR3C tries transferring 0s during a TO11 byte transfer. The routine is the same as NPR3A except that a byte of eight 0s is dumped on top of a background of 1s (rather than 1s dumped on top of 0s).
59	NPR3D is the first test of the byte swap logic during the TO11 byte transfer. It clears the NPR target address, loads the TO11 data word with all 1s, sets byte mode, and initiates the transfer to an odd address. When all of this is done it expects to find a 177400 in the NPR location.
60	NPR3E is the last of the TO11 byte mode tests. It is the effective complement of test NPR3D. It tries to deliver eight 0s to the left side of an NPR location by initiating a byte transfer with an odd address.
61	NPR4 loads a byte count of 00, performs a 1-byte TO11 transfer, and checks the TO11 done flag.
62	NPR5 checks the TO11 null stop logic. The routine is as follows. a. Load a TO11 byte count with the null stop bit true. b. Load a word of all 0s into the TO11 data word. c. Start an NPR transfer at minor state TO11 FILE READ. d. Wait for TO11 done. e. Check for DPS5 null stop.
63	NPR6 is the first test of the TO10 NPR. New logic checked is primarily in the area of setting the CNT4 REQ flip-flop via minor state CNT4 TO10 FL WR.
64	NPR7 checks ABC register incrementation during minor state TO10 E-BUF FILL and during TO10 FILE WR.
65	NPR8 tests the bus timeout logic. The routine starts an NPR to a nonexistent 11 address and expects the nominal 50 microsecond bus timeout delay to complete the Unibus cycle and set TO11 error.
66	NPR9 is similar to NPR8 except it does a TO10 NPR from a nonexistent 11 address and expects the TO10 error status flip-flop to set.
Miscellaneous Testing of TO10 Transfer	
67	T10B checks the byte swap logic. It loads the TO10 address with an odd address (bit 00=1) and forces a TO10 E B REQ CYC. It then checks the TO10 data word for a swap.
68	Having shown that it is possible to swap eight 1s, TO10B1 tries to perform the complement of that operation (i.e., swap eight 0s).
69	TO10C is similar to TO10B except a word of 17777(8) should be swapped into the TO10 data word as 000377(8).
Miscellaneous Testing of TO11 Transfer	
70	TO11A is complex. The routine checks the swap of the E buffer into the RAM. It proceeds as follows. a. Load RAM location DEX address 1 with a word of 0s. b. Perform a diagnostic reset to allow the clock to free-run.

Table 1 DGDTE Test Summary (Cont)

Test	Description
	c. As major state DEX comes up, the 0s from 10 address word 1 are read into the E buffer. d. Lock the DTE20 in TO11 major state. e. Load the TO11 data word with all 1s for a background. f. Load the TO11 address with an odd number. g. Set TO11 byte mode and a byte count of 1. h. Force minor state TO11 shift. i. Pulse the clock once, which brings up CNT1 SWAP H and advances to minor state TO11 EBUF STOR. j. Pulse the clock again, which transfers the swapped E buffer into RAM location TO11 data word. k. Compare the TO11 data word for 0s. If all of this fails, good luck! NOTE This test will produce an erroneous error printout if the ECO which corrects the missing etch on the E buffer swap logic is not installed.
E Buffer Testing	
71	This begins testing of the E buffer. EBUF1 loads 10 ADR WORD1 with 0s and transfers it to the E buffer bits 20-35. It then shifts the 0s into E buffer bits 4-19 and transfers these bits back into DEX WORD2.
72	EBUF2 is similar to EBUF1 except that it shifts 1s.
73	EBUF3 tests the ability of E buffer bits 0-4 to hold 0s.
74	EBUF4 is identical to EBUF3 except that this time the test tries to put 1s in EBUF 00-03.
75	EBH1 tries to initiate a TO10 transfer by using a special decode of the TO11 minor state count decoder. The program does the following. a. Load the TO10 address to set CNT5 TO11 ADR LD (CNT5 C-3). b. Lock the interface in TO11 transfer major state. c. Send a count of 16(8) to the state counter, which should bring up decode CNT4 EBH SET L (which sets CNT5 TO10 BC LD). d. Send bit 0 only to diagnostic word 1, which leaves the interface in 10/11 diagnostic mode but resets CNT4 STATE HOLD (CNT4 C-2). e. Pulse the clock so that TO10 is true. f. Single-pulse the clock until TO10 transfer comes true. g. Move to the TO10 major state again (TO10 ready is not set until EBH CYC goes false). h. Pulses the clock once more to make sure that the interface is locked in the TO10 transfer major state. NOTE There are similarities between this test and previously executed tests OPR1 and OPR2.

Table 1 DGDTE Test Summary (Cont)

Test	Description
76	EBH2 checks the ability of the EBus hold (EBH) register to hold 0s. The program first generates master clear (which should clear the register). It next loads the TO10 byte count with all 1s. The EB HOLD is next transferred to the TO10 byte count location in RAM, where a test for 0s is performed. Since the only way to load the EBus hold register is via a DATAO instruction from a working KL10 processor, this test is all that can be done.
DS Register Tests	
77	DI0 reads the diagnostic bus enable switch (via bit 3 of the status register) and determines whether or not to disturb the diagnostic and EBuses.
78	DI2 sets KL10 diagnostic mode (this will disturb the KL10) and sends all 0s to the seven KL10 diagnostic functions (DS00 DS06). It then reads diagnostic word 1 and looks for a 1 in bit 1 (KL10 diagnostic mode) and 0s in bits 09-15 (the DS flip-flops).
79	DIA3
80	DIA4
EBus Testing	
81	EB1 is the first test of the EBus. It clears the E buffer, sets EBus loop and reads the E buffer back into RAM where bits 00-19 are checked for 0s.
82	EB2 checks whether bits 20-35 of the EBus will hold 0s.
83	EB3 is similar to test EB1 except that it checks whether EBus bits 00-19 can hold 1s.
84	EB4 is similar to EB2 except that it checks EBus bits 20-35 for 1s rather than 0s.
EBus Cycle Testing	
85	CLK5 forces minor state TO11 I/O FUNCTION and reads the RAM address (which should be 07 - TO11 BYTE CNT). The routine next single-pulses the clock (which should have no effect since CLKS INH CLK is set waiting for UBIC BUS COMP) and reads the RAM address to make sure that it does not advance to 11(8), indicating minor state TO11 shift. Following this, the program forces EBus cycle complete (via bit 14 of diagnostic word 2) and reads the RAM address to see if it has advanced to 11(8) (minor state TO11 SHIFT).
86	CLK6 is similar to but simpler than CLK5. The routine ensures that minor states TO10 E-B REQ and DEX ADR2 can also stop the clock. (In addition, since this is the first time these minor states have been produced, the associated RAM addresses are checked).
87	DEXF1A - This test and the two following will test the EBus using a floating 1/0 pattern; i.e., a single 1-bit will be floated through the EBus and the result checked.
88	DEXF2A - refer to test 87.
89	DEXF3A - refer to test 87.

Table 1 DGDTE Test Summary (Cont)

Test	Description
EBus Parity Checking	
90	This begins checking of the EBus parity logic. EPAR1 starts a single-byte TO11 transfer at minor state TO11 FILE READ. Since no data has been loaded into the E buffer, all 0s should produce a parity error. The routine checks for the absence of TO11 DONE (an abnormal termination) and the presence of both TO11ER (status register bit 1) and BPARER (status register bit 4).
91	EPAR2 marks the beginning of serious testing of the DPS4 parity checking logic. The routine simply reads the DPS4 parity flip-flop after performing a CON state clear to make sure the flip-flop can hold a 0.
92	EPAR3 forces the DPS4 parity checking network to read seven specially-selected test patterns. The first two patterns [00 and 153721(8)] have an even number of 1 bits; therefore, the DPS4 parity flip-flop should remain cleared. The remaining five patterns (175747(8), 62132(8), 42002(8), 70066(8), and 102332(8)) all contain an odd number of 1 bits and should set the parity flip-flop.

GENERAL INFORMATION

Code	DGKAA.All
Title	KL10-PA CPU EBox Diagnostic Part 1
Abstract	This diagnostic is designed to detect and isolate faults in the EBox logic.
Hardware Required	KL10-PA mainframe
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKAA TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. (Refer to the 11/10 STD module.)

Table 1 DGKAA Test Summary

Test	Description
1	(EBUS0): Basic EBus and Diagnostic Function Bus This test attempts to clear the machine and read all 0s, then all 1s across the EBus.
2	(ECLK0): Clock Board Initialization This tests that the diagnostically readable signals on the clock board are properly initialized by a master reset. This test (and those concerned with clock board signals which follow) uses subroutine DCOMP. A table of initially expected data for the clock board appears in the initialization file, EINIT1.P11.
3	(ECRAM0): Command Register Initialization This tests that the microcode data register (command register) is cleared after a reset.
4	(ECTL0): Control Boards Initialization This tests whether the APR, CON, CTL and MCL board logic assumes its expected state following a clear. This test (and the control logic tests which follow) uses a master comparison subroutine, CTLCMP, which in turn uses subroutine DCOMP. A table showing initial expected data and masking may be found in the initialization file, EINIT1.P11.
5	(EFLAG0): Flag Logic Initialization This tests that the flag logic on the SCD board assumes its expected state following a clear. This test (and the flag logic tests which follow) uses a master comparison subroutine, FLGCMP, which in turn uses subroutine DCOMP. A table showing initial expected data and masking may be found in the initialization file, EINIT1.P11.

Table 1 DGKAA Test Summary (Cont)

Test	Description
6	(ESCDO): 10-Bit Arithmetic Initialization This tests that the 10-bit arithmetic on the SCD starts off in its expected state following a clear. Tables of these values may be found in the initialization file, EINIT1.P11.
7	(PIZZAO): PI Board Reset Test This test simply performs a master reset of the KL10 and then issues enough clock which would normally cycle the PI board. Since no requests should be pending, the PI board should not cycle. If the PI board does appear to cycle it could be the PIRO flip-flop stuck, or the EBus PI00 line from the DTE20 stuck, or the PI4, PI2, PI1 lines stuck. Either way, the problem is isolated to the PI board, DTE20, or translator board.
8	(ECLK1): Basic Clock Control Registers This tests the ability of the clock control registers on CLK5 to load and hold data via the E and diagnostic buses. The test floats a 1 through the various control registers at each possible source/rate selection. Thus the fact that clocks are produced at each source/rate combination (as well as the appearance of the correct register bits) is checked.
9	(ECLK2): Single-Step Clock Modes This tests the ability of the clock board to generate single EBox and MBox clocks and conditions EBox clocks at each source/rate combination. Also tested are the EBus and SBus clocks.
10	(ECLK3): Burst Counter This verifies that the burst counter down-counts to 0 at each source/rate. It does not verify that a corresponding number of clocks are produced, as this would require much of the untested EBox logic be working. This test uses a table and subroutine from ECLK1.
11	(ECRAM1): Diagnostic CRAM Address Register This checks the path from the EBus to the diagnostic CRAM address register and back. It uses a set of eight test patterns generated by subroutine PATTY. These test patterns are sufficient to prove that each bit can independently be 1 and 0.
12	(ECRAM2): CRAM Data Paths This test checks that each of five CRAM locations (0, 1, 2, 3, 1024) can be loaded and read back correctly. Fourteen 11-byte test patterns from the pattern generation subroutine are used with each CRAM address to verify that each RAM chip can store a 1 and a 0 and that no two chips interact. This test also checks that the EBus was correctly received and transmitted on each CRAM (and the CRA) board, the diagnostic read and write CRAM functions work, and the data holding register (command register) works.

Table 1 DGKAA Test Summary (Cont)

Test	Description
13	(ECRAM3): CRAM Addressing This test checks the addressing of all CRAM chips and the ability of each individual cell to hold a 0 or 1. To begin with, the entire CRAM has been set to all 0 data by the initialization routine, ICRAM1. Subtest 1 reads location 0 to see that it is indeed all 0s, and then writes all 1s back into location 0. It then steps to location 1, reads 0s and writes 1s and so on until all locations in the CRAM have been tested. This demonstrates that each cell in the RAM can hold a 0 and that there are no address line faults of the sort where writing a location with 1s causes some higher location to be modified. Subtest 2 goes the other way around, reading 1s from the top of the CRAM and writing 0s down to the bottom, checking that each cell holds a 1 and that no address faults propagate 0 data downwards. Subtest 3 begins at the top, reading 0s and writing 1s, and subtest 4 starts low and reads 1s and writes 0s (leaving the entire CRAM once again cleared). Problems can usually be localized to the chip by noting the bit position of the failing data. Since the test uses KLDCP RAM reads and writes, EC and DC commands from the console can be used to verify/track down detected addressing problems.
14	(ECRAM4): CRAM Parity Network This test checks the operation of the CRAM parity network by loading four test microwords. The operation of the logic on the clock board to stop the clock on a CRAM parity error is also checked.
15	(ECLK5): Clock Delay (Microcode T Field). This tests the 31, 62, 93 and 520 nanosecond delay logic on the clock board. The T field microcode bits and the CON DELAY REQ signal are also checked. The test issues single MBox clocks and counts the number between EBox clocks. Both too early and too late EBox clocks cause an error.
16	(EDRAM1): IR Register/DRAM Address (I/O, JRST OFF) This test checks that the 13 bits of the instruction register can each store 1s and 0s and that no two bits interact. This register is read by reading the DRAM address with 7XX addressing turned off (bits 0-8) and reading the AC field with AC decoding turned on (bits 9-12). Consequently these features are also checked by this test. Eight patterns, generated by subroutine PATTY, are used.
17	(EDRAM2): DRAM Address - I/O and JRST Logic This test checks the logic which looks for JRST (OP CODE 254), JRST 0, and 7XX instructions and alters the DRAM address accordingly. The effect of JRST instructions on the DRAM J field and the AC field is also tested.
18	(EDRAM3): DRAM Data Paths This test checks that each DRAM data cell in a pair of adjacent locations (locations 0 and 1 chosen for convenience) can independently store a 1 and a 0 and that no two cells interact. The test uses 12 5-byte patterns generated by subroutine PATTY.

Table 1 DGKAA Test Summary (Cont)

Test	Description
19	(EDRAM4): DRAM Addressing This test checks the address lines to all the DRAM chips by filling the entire RAM with 0s, stepping through the addresses one at a time reading 0s and changing to 1s and then stepping through the addresses in the reverse order, reading 1s and restoring 0s. This leaves the RAM cleared and completes the verification that each cell is uniquely addressable and capable of storing both a 1 and a 0. Address 254 reads IR bits 9-12 in place of J07-J10 (hardware does too.)
20	(EDRAM5): DRAM Parity Network This test checks the DRAM parity network with three test patterns. The logic on the clock board to stop the EBox clock on a DRAM parity error is also checked.
21	(ECTL1): DISP Field Decoding and AR, ARX and MQ Control Logic This tests the decoding of the DISP field on CTL1 and all of the logic on CTL2 for controlling the AR, ARX and MQ multiplexers.
22	(ECTLA2): ADXCRY Logic This test checks the ADXCRY gates on CTL1 and PC+1 INH on CON4.
23	(ECON1): COND Field Decoders This tests the decoders on CON1 and various gates using the decoded signals on the control logic boards. The following decoder signals are not verified, as they will be checked with the logic they control: COND/AD FLAGS, COND/PCF-#, COND/FE SHRT, COND/EBOX STATE, COND/EBUS CTL, CON SKIP EN 60-67, and CON SKIP EN 70-77. The four signals COND/024-COND/027 are not tested because they are not used.
24	(ECON2)/EAPR1: CONO APR, PI, PAG and DATAO Logic This test exercises the flip-flops which are controlled by the CON number FUNC 01X decoding of CON COND/DIAG FUNC and the magic # field. This includes the decoders on CON3, the registers controlled by CONO PI and CONO PAG on CON3, the DATAO APR register on APR3 and the APR error interrupt logic on APR1 and APR2. Note that only the internal control of the error flip-flops is tested here; the response to actual error conditions comes much later.
25	(ECON3): UCODE and Processor State Registers This tests the microcode and processor state registers on the CON board.
26	(EAPR2): EBus CTL, MBox CTL and REG FUNC with # Field Decoding This tests the EBus control register on APR3, the MBox control logic on APR5, and the register function decoding on APR6. All three involve the decoding of a microcode function with the magic # field.
27	(EAPR3): Previous Context and AC Block Registers This test checks the CON LOAD PREV CONTEXT and CON LOAD AC BLOCKS logic on CON3, the previous section register on APR3, and the AC block registers on APR5. The FM block mixer logic is covered in the next test, EAPR4.
28	(EAPR4): Fast Memory Address Mixer and AC+1, 2 and 3 Logic. This test checks the logic which adds 1, 2 or 3 to the AC number and the fast memory address mixer on APR4. The mixer inputs from ARX 14-17 and VMA 32-35 are tested later after the data paths and VMA have been checked.

Table 1 DGKAA Test Summary (Cont)

Test	Description
29	(EMCL1)/EFLAG1: AD Function Logic and VMA Held Flip-Flops This test uses MEM/AD FUNC and AD bits 0-12 to independently set each of the principal flip-flops on the MCL board. These flip-flops are used, in turn, to test VMA held register and the VMA held/PC flags multiplexer. Also, the PC flags are set and checked using SCD LOAD FLAGS and AR bits 0-12 to provide interference patterns for testing the VMA held/PC flags multiplexer.
30	(EMCL2): Memory Request Address Mode Control Logic This test checks the decoding of the microcode MEM FIELD (MCL1), the memory request generation logic (MCL1 and MCL5, CON5), the request-type memory (flip-flops clocked by REQ EN on MCL2 and 6), the SXCT/PXCT/VMAX extension logic on MCL4, the DRAM A field decoding on MCL5 and the PREV SEC TO ARMM gates also on MCL5. It uses 24 patterns and depends on many previously tested machine features (for example: CWSX, AC# and AC reference).
31	(EMCL3): VMA Context Storage Logic This tests the flip-flops set with LOAD VMA CONTEXT on MCL2, 3 and 6 and the USER EN and PUBLIC EN logic on MCL2.
32	(EAPR5): FM Block Selection This test checks the FM block, VMA block and XR block mixers and the VMA block register on APR5. It uses some of the PXCT and previous enable logic on the MCL board.
33	(EFLAG2): Processor Flags This checks all of the processor flag logic on SCD4 and 5 not already tested in EMCL1, except for some gating involving MBox, signals for the private instruction flag and the arithmetic overflow flags (checked in EFLAG2). The trap mixers.
34	(ECRA01): DISP RAM to Control REG, DISP Enables, DISP Parity This test is designed to check the DISP field of the CRAM to the DISPATCH/SPEC field of the control register. It tests all bits of this section of control register and also tests the DISP FIELD enable gates. The test begins by loading a 5-bit pattern into the DISP field of the CRAM. This pattern (pattern list is in Table DCRA01:) is then clocked to the control register. The control register and DISP enable gates are then read back and verified to be correct.
35	(ECRA01): Dispatch Codes 1, 2, 3 and 6 and Also AREAD Logic This test checks DISP field dispatch codes 1, 2, 3, and 6. It assures that when these codes are selected, the appropriate data is multiplexed into the CRAM address. For dispatch code 1, we get DRAM J. For code 2, we get AREAD. For code 3 we get SBR RET, and for code 6 we get CTL NICOND. When checking code 2 and AREAD, the test also runs a selection of patterns through the AREAD network to assure that it is in good condition. The basic test procedure is to load the DISP field with the dispatch code under test. Next the DRAM J field is loaded with a test pattern (if checking code 1) or the complement of the expected data. The diagnostic address register is also loaded with the complement of the expected data. Finally, the CRAM address is read to be sure the dispatch code is selecting what is expected.

Table 1 DGKAA Test Summary (Cont)

Test	Description
36	(ECRA03): Dispatch Codes 30, 32, 33 and 35. Also NORM Logic This test is designed to check the control RAM address board dispatch codes 30 (MQ), 32 (AR,BR,AD SIGNS), 33 (DRAM B), and 35 (NORM LOGIC). It also tests the NORM logic priority encoder found on the IR/DRAM board. The basic test procedure is to load the dispatch code into the CRAM. If checking dispatch code 33 which requires DRAM data, the test will clock the microcode word, just loaded into the CRAM, into the control register, then load the appropriate DRAM data and test the CRAM address to verify that it is correct. If the subtest does not require DRAM data, then it loads the AR with test data and at the same time loads the CRAM dispatch data to the control register. Then it tests the CRAM address for correctness.
37	(ECRA04): J-Field and CRA LOC Register Test. Executed at Burst Speed. This test is designed to check the control RAM address board J-field to CR ADR lines and to test both the CR ADR to CRA LOC register lines and the CRA LOC register itself. All tests take place at burst speed (full speed at the currently selected clock rate using the burst counter). The basic test procedure is to load the J-field test pattern into the J-field of CRAM location 0, and leave all other bits at location 0 at 0 (except for a dispatch code = 10). The CRAM location which would be addressed by the current test pattern is then loaded with all 1s. Next the current CRAM address is set to 0. Finally the test gives a burst of clock ticks to cause two EBox clocks to occur. This should force the J-field test pattern into the control register (on the first EBox clock). On the second EBox clock the all 1s RAM word should be addressed and loaded into the control register. Also, the CRA LOC register should be loaded with the J-field test pattern. Now if the control register is not all 1s, there has been a J-field hardware error. If the CRA LOC register is wrong, it has a hardware failure.
38	(ECRA05): Subroutine Return Register This test checks the subroutine return (SBR RET) register on the CRA board. It assures that data 4 levels can be pushed deep onto the stack and that the data can be retrieved with four pops from the stack. It also assures that none of the 11 SBR RET register data lines interfere with one another. This test runs in burst mode; that is, runs in bursts at full clock speed at the currently selected clock source and rate. The basic test procedure is to set up a 3-microword instruction sequence which issues a subroutine call. Next this sequence is burst with varying data patterns that check for interference between the SBR RET register data lines. Then a sequence which issues subroutine RETURNS is set up and executed four times to pop the data from the bottom of the stack. This data is verified to be correct. Finally, a data pattern which finishes the check for stuck-at-1 and stuck-at-0 is pushed to the stack bottom and popped to the top and verified.

GENERAL INFORMATION

Code	DGKAB.A11
Title	KL10-PA CPU EBox Diagnostic Part 2
Abstract	This diagnostic program is designed to detect and isolate faults in the EBox logic.
Hardware Required	KL10-PA mainframe
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKAB TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. (Refer to the 11/10 STD module.)

Table 1 DGKAB Test Summary

Test No.	Description
1	EEDP01 Data Paths Basic MUX Select Test This test verifies that all mixers on the data path board can independently select any of their individual inputs, that no MUX select lines are stuck either high or low, and that none of the mixer output lines are stuck low. This test also verifies that each register can hold all 1s and all 0s. The test begins by resetting the EBox and then uses the MQ as a source of 1s (MQ reset state = 1s) and passes these 1s from one register to another. Registers are cleared of their 1s to ensure that if a select line fails, the correct register will contain the wrong data. The test begins at microinstruction 1, bursts one EBox clock, and tests the registers. Subtest 2 begins at microinstruction 1, bursts two EBox clocks, and checks. Subtest 3 begins at 1, bursts 3, and so on, until the complete set of transfers has been executed at burst mode.

Table 1 DGKAB Test Summary (Cont)

Test	Description
2	EEDP02 Data Paths Basic Interference Test This test verifies that the output of all mixers and registers on the data board have independent lines; i.e., that the AR has 36 independent lines, that the ADA has 36 independent lines, and so on for all registers and mixers. The test runs in burst mode and first loads the AR with interference data. Next, a burst of EBox clocks is given and the interference pattern is sent on the following journey. from the AR to BR BR through ADB through AD to ARX ARX to BRX BRX through ADXB through ADX to AR AR through ADA through AD to MQ MQ to ARX ARX through ADXA through ADX to AR done
3	ESCD1 SC Register (SC from AR and SC Recirculation) This test verifies that each bit of the shift count (SC) register can store a 1 and a 0 and that all bits except SC00 and SC01 are independent. It also verifies the path from AR bits 18 and 28 to 35 to the SC via the SCM and the recirculation path of SC to SC through the SCM. It uses the standard test patterns of subroutine PATTY and bursted microcode. It precedes the rest of the SC tests because the SC register must work in order for the remaining data path tests to run. Refer to the EBox diagnostic microcode listing at RAM location: START = SCM00 for details of each microword.
4	EEDP03 Shifter Board Test - Burst Speed This test runs 85 patterns through the shifter board and verifies that the board has no errors. It runs at burst speed. At single-step speed the BR and BRX, respectively, are loaded with test patterns destined for the AR and ARX. Next, an SC count is loaded into the AR. The burst is started and the microcode does the following. AR (SC COUNT) to SC BR to AR, and BRX to ARX SH (shift board) to the AR, ARX and MQ The MQ, AR and ARX are verified to have the correct data. Expect all failures in this test to be on either shifter board (M8510) or backplane from AR or ARX to shifter board.
5	EEDP04 Data Path Adder ALU and Carry Skip Network Test This test checks and verifies the correct operation of the KL10 72-bit adder, including the associated carry skip logic (found on the IR/DRAM board). The tests are divided into three groups: a. Tests where only the ARX and BRX require test patterns and only the ADX is tested b. Tests where only the AR and BR require test patterns and only the AD is tested c. Tests where the AR, BR, ARX, and BRX all require test patterns and both AD and ADX are tested, including carrys from ADX to AD.

Table 1 DGKAB Test Summary (Cont)

Test	Description
	There is a separate microcode routine for each of these three groups. After the correct test patterns have been loaded to the AR, BR, ARX, and/or BRX at single-step speed, a burst mode clock is given to actually do the add test at full machine speed.
6	EDP4A -AD=0 Test Using Floating 1s Down the AD This test checks the -AD=0 logic on each of the EDP boards. The test simply floats a 1 down the AD and does a burst of clocks and a skip on -AD=0, in order to ensure that each AD bit individually is capable of causing the condition -AD=0. Failures during this test should be traced to the EDP board which contains the bit under test.
7	EEDP05 Fast-Memory Basic Interference Test - Burst Speed This test detects any interference between the 36 data bits coming from the fast-memory RAMs located on the data path board. The test first loads the AR with the current interference pattern, then loads that pattern into the currently addressed fast-memory location. The output of the fast memory is then checked to see that there is no interference between any of its bits.
8	EEDP06 Fast-Memory RAM Test This test detects any fast-memory RAM addressing problems, or fast-memory cells stuck high or low. The test algorithm consists of the following three segments. a. Fill all RAM locations with 0s. b. Begin at block 1, address 0, read all 0s there, then write all 1s there, and increment to the next address. It should still be 0s. Now read 0s there, then write 1s, and again increment to the next address. Continue to the last address and block. c. This segment is similar to segment b. Begin at address block 7, address 17. Read 1s there, then write 0s. Decrement to the next address. It should still be 1s. Now read the 1s, write 0s and decrement again. Continue to block 0, address 00. The test begins by loading three 36-bit words into several data path registers at single-step speed. Once the four registers are initialized (BRX-41,,0) (BR=777757,,0) (AR=770000,,0) (MG=770000,,0) the rest of the test is run in the microcode at full machine speed (current console selected clock rate). Next, the program waits a finite amount of time until the microcoded test should have completed, then stops the clock and examines what halt loop the test is in (i.e., the normal termination halt loop or the error halt halt loop). Error Processing This microloop could have halted at any of the following three locations. a. Halt at location HALT00 if the RAM did not properly fill with 0s (a stuck-at-1 condition). b. Halt at location HALT20 if an address multiply selects more than one RAM location. c. Halt at location HALT40 if an address multiply selects more than one RAM location in ascending direction, or perhaps a stuck-at-0 condition.

Table 1 DGKAB Test Summary (Cont)

Test	Description
9	EVMA01 VMA and Address Break Registers Data and Interference Test This test verifies that the VMA and address break registers can both hold all 1s and all 0s, and that no register lines interfere with one another. The AR is first loaded with the current test data. Next, the starting address of the microcode for this test is loaded and finally a burst of EBox clocks is given. The microcode that is executed by this burst should load the test pattern from the AR, through the AD, and into the VMA and address break registers. Single discrepancies imply bad register bits; multi-discrepancies imply failure of the registers to load properly. Dropped bits could indicate no terminator.
10	EVMA02 VMA and Address Break Match Logic, Bits 18-35 This test verifies the correct operation of the VMA/address break match logic on print (VMA3), and the VMA 18-31=0 logic. Both registers are loaded with all 1s and all 0s and verified to match, which checks the lines for stuck highs and lows. Next, the address break register is kept loaded with 0s and a single 1 is floated down the VMA register to verify that the XOR gates operate correctly (also checking 18-31=0).
11	EVMA07 PC and VMA Held Register Tests - Static This test checks the diagnostic multiplexers that read out the contents of the PC and VMA held registers. The test loads interference patterns into the PC, then reads the PC, transfers the pattern to the VMA held, and then repeats this for all the interference patterns.
12	EVMA03 PC and VMA Held Register Tests - Burst Speed This test verifies that neither the PC nor VMA held register has any bits stuck at 1 or stuck at 0. It also tests that each register's bits do not interfere with any other bits in that register. The backplane lines are tested between the PC or VMA held 2 mixer and the ADA mixer on the data path board. The microcode for this test begins with the test data in the AR. At burst mode, it loads the VMA, then immediately loads the VMA held register. The microcode steers the PC through the ADA into the ARX. Next it steers the VMA held register through the ADA into the AR. Finally, it loads the PC from the VMA, then tests the results. Because of the loading sequence, the AR should contain the current test pattern as transferred from the VMA held register. The ARX should contain not the current, but the previous test pattern, which was just transferred from the PC. Thus the PC and VMA held registers always contain different patterns, which also enables verification of the PC or VMA held mixer.
13	EVMA04 VMA Register Binary Counter and VMA Adder Tests This test has two phases. The first checks the VMA register binary counter and its ability to increment by 1 and decrement by 1. It ensures that each bit of the VMA register can "carry" into the next bit (VMA INC) and also can "borrow" from the next higher order bit (VMA DEC). It also ensures that the carry from each 4-bit chip is connected and operating correctly. It extends an all 1s pattern across the VMA to be sure that a carry into a next higher bit only carries 1 bits worth (i.e., that the carry lines internal to the chip are not shorted). The second phase of the test uses patterns out of the PC and the CRAM number field to ensure that the VMA ADDER ALUs have no fault either internal or external to the chip.

Table 1 DGKAB Test Summary (Cont)

Test	Description

This section of the test checks the VMA ADDER ALUs on VMA board. First, a test pattern is placed in the PC register; secondly, a test pattern is placed in the CRAM number field. The output of the VMA ADDER is then clocked into the VMA itself and verified to be correct. This test is run at burst speed.

The final phase of VMA ADDER test checks the MCL VMA INC line into bit 35 of the VMA ADDER ALU.

14 EVMA05 VMA AC REF and VMA SECTION 0

This test verifies correct operation of the VMA AC REF logic and VMA SECTION 0 gate. It does this by setting up 10 patterns which drive the VMA AC REF logic. The pattern set up is completely microcoded and is merely driven by the PDP-11 which selects the correct microstarting address and data pattern for the VMA section.

VMA 18-31=0	PAGE UEBR REF	VMA READ	VMA WRITE	VMA EXTENDED	VMA 13-17	EXPECT VMA AC REF
1	0	0	1	1	0	1 ST 1
0	0	0	1	1	0	0 ST 2
1	1	0	1	1	0	0 ST 3
1	0	0	0	1	0	0 ST 4
1	0	1	0	0	37	1 ST 5
1	0	0	1	1	20	0 ST 6
1	0	0	1	1	10	0 ST 7
1	0	0	1	1	4	0 ST 8
1	0	0	1	1	2	0 ST 9
1	0	0	1	1	1	0 ST 10

15 EVMA06 Previous Section REG, ARMM MUX, and VMA IN MUX

This test checks the interference through and the selectability of the ARMM multiplexer. It also completely tests the previous section (PREV SEC) register and the selectability of the VMA IN multiplexer. The basic test sequence is as follows.

a. Load test pattern to PREV SEC register.

b. PREV SEC register through ARMM to AR.

c. PREV SEC register through VMA IN to the VMA. Stop and verify AR and VMA.

d. PC through VMA IN to the VMA. This should clear the VMA. Again stop and verify the VMA contents.

The test continues this loop to float a 1 down the PREV SEC register. Finally, it loads the PREV SEC register with all 1s, then reads the PC of all 0s through the ARMM into the AR, to verify the ARMM selectability.

16 EMCL4 Page Illegal Entry Logic

This test verifies the PAGE ADDRESS COND/PAGE ILL ENTRY logic on MCL3. It uses the VMA and ADR BRK registers on the VMA board, the COMP register on APR3, and the logic used to load them.

17 EFLAG3 Arithmetic Overflow Flags

This test checks the carry and overflow flags on SCD4 with the associated gates on the data path boards. Note that it depends on the following functions working: AD/A, AD/A+B, ADB/AR*4, ADB/BR, SC/A, SC/A+B&SCADA&B/#.

Table 1 DGKAB Test Summary (Cont)

Test	Description
18	ETSAT1 Test-Satisfied Logic This test checks the test-satisfied logic on IR3 and the skip-satisfied logic on MCL 4 and 5. It depends on the IR/DRAM working and the AD and AD carry logic working. Set up AD=0 and AD CRY-02. The four possible states for these variables are realized as follows: AD NOT 0 & NO CARRY : AD/A+B, A=-1, B=0 (CODE 010) AD NOT 0 & AD CRY-02: AD/A-B-1, A=-1, B=0 (CODE 000) AD IS 0 & NO CARRY : AD/A+B, A=0, B=0 (CODE 110) AD IS 0 & AD CRY-02 : AD/A+1, A=-1 (CODE 001)
19	ECON4 GO/START/RUN, I/O LEGAL, and COND ADR 10 Logic This test checks most of the logic on CON2. It requires some flags to be working: MCL VMA FETCH and VMA AC REF. Note that the diagnostic functions SET RUN, CLR RUN and CONTINUE (decoded on CON2) are also tested here. The DTE status bits for EBOX HALTED and RUN are also checked. The last part of this test checks the timing of the start and run synchronizing logic. That is, it checks that START and RUN take three clocks to set from the issuing of the (asynchronous) diagnostic function; that START remains up for only three clock ticks, and that RUN stays set for three clock ticks after the clear function has been issued. The test begins with a diagnostic continue and watches START (COND=71).
20	ECLK6 Clock Board Page Fail Logic This test forces a page fail using APR SET PAGE FAIL and then checks the sequencing of the logic on CLK4 and CLK3 (and the addressing on the CRA board). It runs at burst speed. Each subtest begins with the logic reset and issues a burst of MBox clocks which is one greater than the previous subtest's burst. Thus the page fail sequence is checked clock tick by clock tick, but is always stepped at machine speed rather than single-step.
21	ECLK7 Simulated MBox Response This test checks the MB wait and MB XFER logic on the CON and CLK boards. The test makes two simulated MBox cycles: one an AC reference and one not.
22	ESCD2 Basic 10-Bit Data Paths This test uses the AR to SC input path tested in ESCD1 to send patterns over the following 10-bit arithmetic data paths: SC from SC via the SCAD (checks first position on SCADB multiplexer; SCAD passes B data on A+B and third position on SCM multiplexer), FE from SCAD, FE recirculation and shift right, SC from FE (second position on SCM multiplexer) and AR (upper bits) from SCAD via ARMM (ARMM positions 3 and 4). It uses standard test patterns and bursted microcode. Refer to the EBox diagnostic microcode listing at CRAM location: START = SCM00 for details of each microword.

Table 1 DGKAB Test Summary (Cont)

Test	Description
23	ESCD3 SCAD and ARMM Multiplexers This test checks the SCADA, SCADB and ARMM mixers. The test uses two burst microwords to steer AR and number field patterns through the mixers into the AR, SC and FE registers. The first microword is loaded separately by each subtest based on the stimulus table. The second microword, which is the same for all subtests, simply recirculates the register data and is there to switch the mixers to catch any slow-propagating signals. The test assumes that the SCADDER can pass A data (SCAD/A) or B data (SCAD/A+B, SCADA disabled). The adder is checked in test ESCD5.
24	ESCD4 SC .GE. 36 Logic This test checks the SC .GE. 36 gates on SCD2. It uses 1 microword which loads the magic # field through the SCADB, SCAD and SCM into the SC. Nine magic # patterns are used.
25	ESCD5 SCADDER This test checks the 10-bit adder on the SCD board. It loads the SC and FE registers from the magic # field, performs an arithmetic adder function using SC and FE as inputs, and stores the result back in the SC and FE. An extra microword is provided which switches the adder multiplexers to cut off any slow-propagating signals. The first three microwords, found in the EBox diagnostic listing at CRAM location START5 = SCM10, are modified according to the stimulus tables data. The test runs at burst speed.
26	PIZZA1 PI ON LEVEL Set and Clear, GEN LEVEL Set and Clear, ON, OFF, SYS This test checks the ability of the PI ON LEVEL flip-flops and the PI GEN LEVEL flip-flops to load, hold, and clear. It also checks the PI ACTIVE flip-flop and the ability of PI SYS CLR to clear the PI system. The basic test sequence is as follows. a. Set all PI ON LEVEL flip-flops. b. Individually clear each PI ON LEVEL flip-flops. c. Set all PI GEN LEVEL. d. Individually clear each PI GEN LEVEL. e. Set the PI SYSTEM ACTIVE, clear ACTIVE, set all ON and GEN level flip-flops and clear all with a PI SYS CLR. If subtest 15 fails because HONOR INTERNAL is low, look at print PI5 and check the logic producing PI5 GEN INT. When GEN LEVEL 4 is the highest priority, as in this test, GEN INT should be true and should cause HONOR INTERNAL
27	PIZZA2 CS Lines, Load/Test Ring Counter, PI EBUS REG and EBUS PI This test is divided into two sections. The first section tests the CS lines both for stuck condition and for interference between the CS lines. It does this by loading the IR with interference patterns, dumping the IR onto the CS lines, and then reading the CS lines. The second section of the test checks the EBus request logic and the EBUS PI GRANT flip-flop, and partially checks the PI load/test ring counter. It does this by issuing PI GEN requests to the PI board, then bursting the PI clock part way into the request cycle. The PI board is then verified to be in the correct state. See the expected data table for the test patterns used in this test.

Table 1 DGKAB Test Summary (Cont)

Test	Description
	If subtest 6 fails because TIMER DONE fails to function, then either the problem is in the logic that initializes the TIMER DONE circuitry to an initial count of -15, or else the counter is not counting at all.
28	PIZZA3 PIR Flip-Flops, PIR EN, PI REQ SET, PIH Flip-Flops, PI CLRS, PIR/PIH PRI This test exercises and checks the following PI board logic: the PIR1-PIR7 flip-flops, the PIR/PIH priority encoder pair, PI REQ SET decoder, PI CLR decoder, the PIH1-PIH7 flip-flops and the PIR EN gates. The basic test procedure is as follows. a. Set PI cycle to hold the load/test ring counter. b. Do a CONO PI to set one or several GENs. c. Drop PI cycle to enable the load/test ring counter to advance. This should load the PIR flip-flops. d. Stop and read the state of the PI board to see if the PIRs set. e. Set PI cycle, then do a SPEC/SAVE FLAGS which loads the PIH flip-flops via the PIR EN gates. f. Again read the state of the PI board to check that the PIH flip-flops did load. g. Do a PI DISMISS, then verify that the highest level PIH was set.
29	PIZZA4 PI TIM1-TIM7-PI COMP Ring Counter and Timer Done Counter The following two PI board tests are confusing and difficult tests to understand. The connection between the error symptom printed on the terminal and the actual hardware fault causing the symptom is very difficult to explain even with an excellent understanding of how this part of the PI board is intended to work. This test checks two counters on the PI board. The PI time state counter is a pair of shift registers on print PI2 whose eight outputs are labeled: TIM1, TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, COMP. The timer done counter consists of two binary counters connected serially whose only output is labeled TIMER DONE. These two counters operate independently, but must work together. When a cycle is started the time state counter goes to time state TIM1. It remains static, in TIM1 while the timer done counter begins counting. After timer done counter has counted a specific number of MBox ticks, it sets the flip-flop timer done. This act turns the time state counter on and it advances from TIM1 to TIM2. Again it waits for timer done to count before it can advance. This test operates as follows: it issues one too few ticks to set TIM1 and verifies that TIM1 does not come up. It resets the PI board and issues exactly enough ticks to set TIM1, then verifies that TIM1 does occur. Next, it resets the PI board, issues one too few ticks to set TIM2; verifies that TIM2 does not come up, resets, issues exactly enough ticks to set TIM2, verifies that TIM2 comes up and resets. The same procedure is used for TIM3, TIM4, TIM5, TIM6, and TIM7. Faults in these two counters can be isolated to the PI board, but the logic failure on the board itself is difficult or impossible to call out with software. Only by putting the board on extender can the problem be found.

Table 1 DGKAB Test Summary (Cont)

Test	Description
30	PIZZA5 STATE HOLD Logic, EBUS DEMAND Logic, OK ON HALT Decoder This test checks the combinational logic that produces STATE HOLD, and the combinational logic that produces EBUS DEMAND. It also produces the EBus data lines and other PI board signals that are used with the decoder to cause PI14 OK ON HALT. The basic test procedure is to set up a microcode which would normally take the PI board through all seven time states. The microcode includes the special conditions under test and sets the appropriate flip-flops needed for the specific subtests (examples include: APR EBUS RETURN, APR EBUS DEMAND, CON EBOX HALT, CON EBUS REL, and APR EBUS REQ). Finally, the PDP-11 controls the number of clocks given to the PI board and stops the PI board in the time state desired, then reads the PI board and verifies that the combinational logic under test is working. The next section of this test checks the PI OK ON HALT logic. The microcode for this subtest is similar to the preceding subtests, except that during PI TIM6, data is being put onto the EBus in order to cause the PI OK ON HALT decoder to decode the desired function from the EBus.
31	PIZZA6 Physical Number Flip-Flops, from EBus 00-15 and PHYSICAL NUMBER PRIORITY EN This test checks the 16 physical number flip-flops on print PI2 and also checks the dual-priority encoders which take the physical numbers and produce the signals SEL PHY8, SEL PHY4, SEL PHY2, and SEL PHY1. The test also checks the SEL PHY4X to EBus bits 7, 8, 9, 10 mixer on print PI5. The test uses microcode which runs through all seven timing states of the PI board. The PDP-11 stops the clock during PI TIM3 and examines the state of the PI board to ensure that the correct physical numbers have come up. Next, the test is continued through TIM6, with the AR function AR/EBUS. After TIM6, the PDP-11 again stops the clock and examines the PI board and the AR to be sure the correct signals are being put onto the EBus by the PI to EBus mixer.
32	PIZZA7 APR PIA 04,02,01, APR PHY NO. and PIR EN This test checks the APR PIA 04,02,01 flip-flops, which are set by the CONO APR, to check the APR PIA decoders, and to check the output of the APR PIA decoder to the PI REQUEST flip-flops. The test does this by setting all PI levels ON and PI ACTIVE, then setting the 3-bit APR PIA register with 0 (1-7 on successive subtests), cycling the PI board to TIM3, and checking the state of the PI board for correct results. This test also checks the PI2 APR REQUESTING flip-flop.
33	PIZZA8 MTR PIA 04,02,01, MTR PHY NO. and PIR0 This test checks the MTR PIA 04, 02, 01 flip-flops that are set by the CONO MTR generated on the MTR board. The purpose is to check the MTR PIA decoders and the output of the MTR PIA decoders to the PI request flip-flops. The test also checks the DK20 REQUESTING flip-flop on the PI board. The test sets all PI levels ON and PI ACTIVE. It then sets the MTR PIA register with (1-7), then cycles the PI board to TIM3, which should set the DK20 REQUESTING flip-flop. The cycling to TIM3 should also set a PI request. The second phase of the test checks the DTE20's PI REQ 0 line to the PI board. It tests that the DTE20 can cause a PI request on level 0.

Table 1 DGKAB Test Summary (Cont)

Test	Description
34	EPAR1 Parity Test - APR FM 36 RAM Chip Address and Data Test This test is designed to verify the APR FM 36 128 X 1 RAM chip, both data and addressing operation. First it attempts to fill every RAM location with a 0. This pass of the test will find any stuck-at-1 bits. Next, it begins an addressing test. It reads the 0 in location 0 to ensure it has not changed; writes a 1, then goes to the next address. The sequence is repeated to the last RAM address. The RAM should now be full of 1s. Now the test starts at the last RAM address, verifies that the 1 is there and then writes a 0. It decrements the location, verifies the 1, writes a 0, and continues this process until the whole RAM has been refilled with 0s. This last test phase may fail because of either addressing problems, or bits stuck at 0.
35	EPAR2 AR and ARX Parity Chain, CON AR and ARX Parity Bit Generator and CLK This test checks the AR/ARX parity chain located on the shifter board, the CON AR 36 logic on the CON board used to generate the odd parity bit, and the parity checking logic on the CLK board that stops the clock or causes a page fail on FM, AR, or ARX incorrect parity. The test begins by selecting one fast-memory (FM) location and using it throughout the entire test. It attempts to set the FM-36 bit to a 1 or a 0 as the test requires, and, on each subtest, checks that the FM-36 bit has indeed gone to a 1 or 0. If it has not, the failure could be in that RAM chip. Subtests 1-10 load test patterns to test the CON AR 36 logic and the CON ARX 36 logic. In conjunction with the FM-36 bit, the test is also setting and clearing CON MBOX DATA?, -CON FM DATA, CON FM BIT 36, CON AR LOADED, CON AR FROM MEM, and CON ARX LOADED, all of which are necessary in the CON ARE 36 and CON ARX 36 test patterns. It is possible that CSH PAR BIT A and CSH PAR BIT B could cause these subtests to fail because the EBox has no control over these bits. They should be in their reset state (low). While subtests 1-10 are occurring, the AR and ARX registers are loaded with data that, when combined with CON AR 36 and CON ARX 36, tests the AR and ARX parity chain logic found on the shifter board. Finally, the test uses CON AR LOADED and CON ARX LOADED with SH AR PAR ODD and SH ARX PAR ODD to test the CLK PAGE FAIL logic. Test patterns used are as follows.

	AR From MEM (True)	+WR Even PAR Data H	-MBox Data H FM Data L	FM Bit 36 H
ST1	0	0	1	1
ST2	1	1	1	1
ST3	1	0	1	1
ST4	0	0	0	1
ST5	0	0	1	0
ST6	0	0	1	1
ST7	0	1	1	1
ST8	1	0	0	1
ST9	1	0	1	0
ST10	0	1	1	1

Table 1 DGKAB Test Summary (Cont)

Test	Description
36	EPAR3 AR(ARX) Parity Page Fail, FM Parity Chain, FM Parity Error This test checks that incorrect AR or ARX parity causes CLK PAGE FAILS and that the right page fail address goes to the control RAM address board. It checks the fast memory (FM) parity chain works and ensures that bad fast memory parity causes a CLK ERROR STOP.
37	ECRA06 Microcode Skip Conditions (COND 40-57) and Dispatches This test checks the control RAM address board SKIP conditions and COND FUNCTIONS 40-57. It also picks up several microcode dispatches - the individual inputs to the multiplexers which do the actual microcode dispatching. Each subtest has its own microcode which sets up the desired skip condition or dispatch condition and ends with the skip or dispatch being multiplexed onto the CRAM address line. The actual skip or dispatch is taken, and the CRAM bits which set up the skip or dispatch are left behind in the control RAM while the PDP-11 examines the CRAM location register to verify that the correct dispatch or skip was taken.
38	PIDTE PI Board to DTE20 Interface Test This test checks the basic lines that connect the DTE20 to the EBus and to the EBox PI system. It checks that PI interrupt levels can be assigned to the DTE20 and that the DTE20 can issue an interrupt to the PI board at that assigned level. It checks that a CONI DTE reads what it should, that a CONO DTE sets what it should and that the bits a CONO DTE sets show up in the DTE20's status register. It checks that KL10 HALT LOOP, KL10 RUN FLOP, and EBOX CLK ERR STOP are all readable in the DTE20 register DIAG1. The test also checks that when the DTE20 issues an interrupt, the correct API function type is sent to the EBox (i.e. correct IOP function type is sent on EBus bits 3-5; correct address space specification is sent on EBus bits 0-2; correct qualifier is sent on bit 6, and the correct physical number is decoded and put onto EBus bits 7-10). It tests the DTE20 decoding of the CS lines CS00-CS06 and ensures that the privileged DTE responds only to its own device code.

GENERAL INFORMATION

Code	DGKBA.All
Title	KL10-PA - Basic MBox Diagnostic
Abstract	This diagnostic is designed to test and isolate faults in the KL10-PA MBox logic.
Hardware Required	KL10-PA mainframe/MCA20(optional)
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKBA TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGES
This diagnostic uses the standard error message format. (Refer to the 11/10 STD module.)

Table 1 DGKBA Test Summary

Test	Description
1	Checks MBox clock logic by single-pulsing the clock. The subtest no. is the number of clocks after a master reset. Expected clock states are as follows.

Tick	1	2	3	4	5	6	7	8	9
PHASE CHANGE COMING	T	F	T	F	T	F	T	F	T
A CHANGE COMING	X	F	F	F	T	F	F	F	T
B CHANGE COMING	X	F	T	F	F	F	T	F	F

Test	Description
	Subtest 10 checks clock reliability by free-running the clock and testing for proper synchronization of A CHANGE COMING and the SBus clock.
2	Checks the channel logic timing. Subtests 1-5 check CH MR RESET and CH TO-T3. Subtests 6 and 7 check CH timing block and MB REQ INH. Subtests 8-37 check CBus select logic. Even subtests check the CH board CBus selects, and odd subtests check the CRC CBUS SEL D signals for the proper CBus selects at D-TIME.
3	Checks the state of the channel logic after a MR RESET + 1 tick.
4	Tests for the MBox master reset state. Refer to subroutine RSTCHK for additional information.
5	Checks the channel reset function. See subroutine CHRCHK for additional information. It also checks that no RH20 is interfering with the CBus during a channel scan.
6	Checks that the MBox properly aborts an AC reference and that the microcode can properly reach the halt loop.
7	Checks EBR, UBR, and EBus register for 7s.

Table 1 DGKBA Test Summary (Cont)

Test	Description
8	Checks EBR and UBR for 0.
9	Slides a 0 through the EBR and UBR; then it slides a 1. Test is for shorts.
10	Checks VMA to PMA and special microcode time states.
11	Checks AR to MEM to C mixer to AR for 0s and all 1s.
12	Checks AR to MB to MEM to C mixer to AR and ARX for all 1s. (See test 10 for microcode used.)
13	Checks AR to MB to MEM to C mixer to AR and ARX for all 0s.
14	Checks AR to MB to MEM to C mixer to AR and ARX for a sliding 0 and a sliding 1. (See test 10 for microcode used.)
15	Checks the SBus for all 0s, all 1s, a sliding 0 and a sliding 1. (See test 10 for microcode used.)
16	Checks request logic for 1-word writes via MBox snapshots.
17	Checks the ERA for a write to addresses 777 and 17 777000.
18	Checks SBus DIAG CYC timing and ensures that no controller responds to an SBus diagnostic function 1 with SBus reset true.
19	Checks NXM of a 1-word read and MB data codes for MEM to MBS.
20	Checks for proper NXM of a 1-word write.
21	Check for proper NXM of a read-pause-write.
22	Checks the ability to invalidate the page table by testing that a page refill is initiated for each exec page.
23	Checks control logic for a 4-word read and page refill by performing an NXM of a MAP instruction.
24	Checks generation of memory address parity. Subtests are pattern numbers.
25	Checks MB parity. Subtests 1-15 apply patterns required to test the MB parity tree and bits for normal write parity. Subtests 16-20 check for proper parity error detection.
26	Checks the page table for NXM data.
27	Checks MBox cycle abort.

GENERAL INFORMATION

Code	DGKBB.A11
Title	KL10-PA Memory Systems Diagnostic Test
Abstract	This diagnostic program is designed to detect and isolate all faults related to the operation of the KL10 memory system. It checks all internal and external memory controllers which respond to an SBus diagnostic function. If run in a normal fashion, all will be tested. All internal memory is tested for addressing all 1s and 0s. All of external memory is tested. It is assumed that DGKBA has been run. Memories are left configured for all memory tested.
Hardware Required	KL10-PA mainframe/MA20s/MB20s/DMA20s
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKBB TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGKBB Test Summary

Test	Description
1	Performs a memory system configuration validity check.
MA20/MB20 Controller Tests	
2	Checks the selected MA20/MB20 controller reset function. The setup determines test dispatching for memory controller tests. If the test is started at a controller test, only one controller will be tested. The controller address will then be solicited. If the test is run in a normal fashion, all will be tested. Internal memory tests precede external memory. Internal memories are tested in numerical order.
3	Checks setting of MA20/MB20 flip-flops which may be set on SBus diagnostic function.
4	Checks clearing of MA20/MB20 flip-flops which may be set on SBus diagnostic function.
5	Checks MA20 enables and address boundaries for response conditions. Subtests 1-4 check the requests. Setup loads an instruction in AC10 to support the test. It is as follows: `10:     MOVE     AC17,0(7)`

Table 1 DGKBB Test Summary (Cont)

Test	Description
6	Checks MA20/MB20 address boundaries for no-response conditions. Subtests 1-8 check single-bit-set address boundaries against an address in page 0. SBus diagnostic functions are verified as the boundaries are set. Subtests 9-16 check a 0 boundary condition against each corresponding address bit. Subtest 17 checks the address boundaries for a response.
7	Checks each RQ with no RQ EN to verify a no-response condition.
8	Checks MA20/MB20 address parity for 1-word combinations. Subtests 1-17 check reads from an address pattern list. Subtests 18-22 check proper generation of a parity error for a write and a read. NOTES 1. Tests 9 through 15 check memory modules. If started at a module test, any module may be tested. The module number will then be solicited. If the test is run in a normal fashion, all modules will be tested. 2. Internal memory loopback tests. Note that both test 10 and test 11 leave a fast scope loop running after detection of an error.
9	Setup configures the test controller to respond to its maximum address range, 1-way interleaved, and loads the AC program for loopback cycle checkout.
10	This is the 1s and 0s loopback test to internal memory. Upon detecting an error, it goes into a fast scope loop.
11	Initialization just loads the AC program for doing a sliding patten loopback. The first pattern is 757020,,020757, which is thereafter rotated to the left.
12	Address-tests an MA20/MB20 memory module. The test is performed by executing the following sequence. a. Write 0s to a memory location. b. Read the location just written. c. Read the same location again (to eliminate data paths from consideration). d. Write all 1s to a different location. e. Test that the original location did not change. Solid faults in this test should be generated only by addressing problems. Patterns used correspond to the input combinations for module X, Y drive. A loop is left running in the ACs if a fault is detected. The addresses are left in AC15 and AC16.
13	Internal memory 1s and 0s test. This test ensures that each word in a given storage module can hold all 1s and all 0s. Initialization consists of loading the AC program.
14	Internal memory 1s and 0s test. A loop is left running in the KL10's ACs if an error is detected. The test sends first 1s, then 0s, to a given memory location; then it goes on to the next location.
15	Internal memory parity bit check. This test ensures that each word in a given internal memory has a working parity bit. The initialization just loads the AC program. Internal memory test 15 leaves loop running in ACs upon detecting error.

Table 1 DGKBB Test Summary (Cont)

Test	Description
16	Initialization is a dispatch modifier for internal memory module tests.
17	This is the internal memory module select test. It ensures that the MA20/MB20 controller is properly selecting storage modules in 1-way interleave mode, and that these storage modules are responding correctly. It also provides a more rigorous test of the modules than the 1s/0s test. All of this is accomplished via the address/tag test. The initialization is used only to set up the table required by the test. The only solid errors which should show up here are address inteference errors between storage modules. See the ADRTAG subroutine for more information regarding failure modes.
18	This is the internal memory read-pause-write test. It is performed once for each controller using the address left over from the last previous module test. Initialization consists of loading the AC program. The internal memory test does two things. First, it verifies that the DATA VALID X OUT H signals are set during a read-pause-write cycle. Then, it checks to see if a read-pause-write does, in fact, work. If the test fails here, a scope loop is left running in the ACs.
19	This is the internal memory incomplete-cycle test. It attempts to force an incomplete cycle by stopping a read-pause-write in the middle and then doing a read. It then checks to see if the flag bit is set, and to ensure that the flag can be cleared.
20	This is the internal memory 4-word read/1-way interleave test. Initialization consists of setting up constants and loading the AC program. See subroutine FWDRDT for comments on this test.
21, 22	The initialization for tests 21 and 22 consists of two basic parts: first, checking for compatibility between the test controller (TSTCON) and its alternate (ALTCON=TSTCON, XOR.1), and setting up the SBus diagnostic functions necessary; and second, loading the data for the canned 4-word read test. This is done by FWDRDI. The internal memory (MA20 and MB20) 4-word read/interleave 2 and 4 tests work in the following manner: the SBus diagnostic functions necessary and the AC program are set up by the initialization. The test then resets memory, and configures just TSTCON and ALTCON. The even request enables are first assigned to ALTCON, then later to TSTCON. The actual test is the canned 4-word read test FWDRDT. Note that a hard loop is left running upon the detection of an error.
DMA20 Diagnostic Tests (Do Not Go to MEM)	
23	There is no test number 23. NOTE Tests 24 through 26 check the "setability" of the DMA20 SBus diagnostic function bits.
24	Checks the master reset function with respect to the DMA20 diagnostic function bits.
25	Checks to see that the diagnostic functions can set the bits they are supposed to.

Table 1 DGKBB Test Summary (Cont)

Test	Description
26	Checks to see if all the setable bits can be reset. NOTE The initialization for tests 27 and 28 loads the AC program common to both of the tests. The AC program for tests 27 and 28 sets up an address in AC17 and then reads from that address. Data is ignored; instead, an SBus diagnostic function 0 is done to read the address in the DMA20 address register. Bit 13 of the address is on to force NO-AC-REF.
27	This is the address path test to and from the DMA20. It sends a pattern of all 1s, all 0s, sliding 1s, and sliding 0s through the DMA20. If there is a stuck bit or short in this path it will show up here. If there is a fault it is probably on the M8560 board.
28	Checks DMA20 address parity for 1-word requests. The test first runs through patterns while checking for address parity errors, then ensures that incorrect address parity can be detected on a read. To make sure an error can be reset via SBus diagnostic function 0, the test forces an error for write and ensures that the error actually occurs.
29	Checks each data path through the KBus buffers, runs all patterns through each buffer, then goes on to the next buffer. The patterns are 1s, 0s, and sliding 1s and 0s.
External Memory Tests	
30	Checks data parity on all the KBuses via loopback.
31	Checks to see if each KBus can detect and report bad data parity. NOTE Tests 32 and 33 are the external memory data path tests. These tests are designed to show up an error in the data path to and from memory. Sufficient addresses are tested to make sure that all data paths are checked, regardless of memory type, configuration, and interleaving. Both tests leave a fast scope loop running upon detection of an error.
32	This is the 1s and 0s data path check. The initialization loads an AC program which performs the test.
33	This is a sliding pattern test. The initial pattern is 757020,,020757. This pattern is rotated 36 times, each time being sent to memory and checked. The initialization loads the AC program.
34	This is the external memory address/tag test. The basic function of this test is to ensure that all the address selection logic is working properly. There are many possible failure modes, and they may be difficult to distinguish. There is no initialization for this test.
35	The external memory read-pause-write test checks to ensure that the DMA20 correctly completes the read-pause-write cycle without modifing the address. Test initialization loads the AC program.

Table 1 DGKBB Test Summary (Cont)

Test	Description
36	This is the external memory (DMA20) single-step test. It starts the AC program in a single-step mode and runs through it to ensure that events occur in the correct order. The initialization for test 36 loads the AC program.
37	DMA20 (external memory) 4-word read test. See subroutine FWDRDT for explanations. The initialization for the DMA20 4-word read test consists of setting up the ACs and some constants for the test.

ADRTAG

ADRTAG is a subroutine designed to do an address-is-data test, otherwise known as an address-tag test. The basic procedure is this: write the address of each word into the word and read all the words back. If there was any address interference on the writes it will show up on the readback. The addresses are also written in a reverse order to catch address interference problems from high to low memory. Finally, both of the above procedures are repeated with complementary data to make sure that each bit of the memory can hold a 1 and a 0, and also to catch data-related address interference problems.

This address-tag test features patterns in both halves of the data word. This is necessary because the memories are built as two 18-bit modules in parallel. The problem is that the maximum address is 22 bits and two 22-bit numbers do not fit into 36 bits. In this test, the 22-bit address is kept in the right half of the word and a special "1 bit insensitive" pattern is kept in the high-order 14 bits of the word. The nature of this pattern is such that any 1-bit address error will go to a word whose high-order pattern is different. Thus, it can be determined whether the address interference occurred in the left, right, or both halves of the word.

To preserve PDP-11 memory space, AC programs are not given for all nine subtests. Instead, the AC programs are constructed by the PDP-11 in the ACs at run time. Much of the PDP-11 code is used to this end and is not actually part of the test. Note the capability to change the order of the subtests.

Subtests 1, 4, 6, and 8 should not find any errors except perhaps for erroneous NXMS which indicate faulty address-acknowledge logic.

Subtests 2, 5, 7, and 9 will catch most of the errors in this test. These include dropouts, picked and stuck bits, and address interference. If either the pattern in bits 00-13 or the address in 14-35 is correct, then the error was probably a dropout, picked or stuck bit. If both halves have strange but incorrect data, then the problem is probably address interference.

An error in subtest 3 indicates read-restore problems if the data coming back has massive dropouts; otherwise, the problem is probably an intermittent read (sense) error.

Contents of Accumulators for Subtests 1 to 4.

```
Subtest 1            Subtest 2*           Subtest 4

Write Forward        Read Forward         Write Forward
Address Comparison   Address Comparison   Address Pattern
Pattern              Pattern

TRNN,17,,37777       TRNN,17,,37777       TRNN,17,,37777
HRRI,2,,-320,2       HRRI,2,,-320,2       HRRI,2,,320,2
HRLOI,16,,-1         HRLOI,16,,-01        HRLZI,16,,0
XOR,16,,17           XOR,16,,17           XOR,16,,17
HRRI,2,,-460,2       HRRI,2,,-460,2       HRRI,2,460,2
SXCT,,,13            SXCT,,,13            SXCT,,,13
JRST,,,10            CAME,15,,16          JRST,,,10
                     HALT,,,1
CAME,17,,14          CAME,17,,14          CAME,17,,14
AOJA,17,,0           AOJA,17,,0           AOJA,17,,0
HALT,,,0             HALT,,,0             HALT,,,0
MOVEM,16,,0,17       MOVE,15,,0,17        MOVEM,16,,0,17
LAST ADR             LAST ADR             LAST ADR
ECHO                 ECHO                 ECHO
PATTERN              PATTERN              PATTERN
ADDRESS              ADDRESS              ADDRESS
```

*Subtest 3 is the same as subtest 2. This is specifically to check for the possibility of a faulty read-restore operation in the memory.

Contents of Accumulators for Subtests 5 to 7.

```
Subtest 5                            Subtest 6         Subtest 7
Read Forward                         Write Backward    Read Backward
Address                              Address           Address
Comparison                           Comparison        Comparison
Pattern          AC                  Pattern           Pattern

TRNN,17,,37777   ! 00                TRNN,17,,37777    TERNN,17,,37777
HRRI,2,,320,2    ! 01                HRRI,2,,-320,2    HRRI,2,,-320,2
HRLZI,16,,0      ! 02 <START>        HRLOI,16,,-1      HRLOI,16,,-1
XOR,16,,17       ! 03                XOR,16,,17        XOR,16,,17
HRRI,2,,460,2    ! 04                HRRI,2,,-460,2    HRRI,2,,=460,2
SXCT,,,13        ! 05                SXCT,,,13         SXCT,,,13
CAME,15,,16      ! 06                JRST,,,10         CAME,16,,16
HALT,,,1         ! 07                                  HALT,,,1
CAME,17,,14      ! 10                CAME,17,,14       CAME,17,,14
AOJA,17,,0       ! 11                SOJA,17,,0        SOJA,17,00
HALT,,,0         ! 12                HALT,,,0          HALT,,,0
MOVE,15,,0,17    ! 13                MOVEM,16,,0,17    MOVE,15,,0,17
LAST ADR         ! 14                LAST ADR          LAST ADR
ECHO             ! 15                ECHO              ECHO
PATTERN          ! 16                PATTERN           PATTERN
ADDRESS          ! 17                ADDRESS           ADDRESS
```

Contents of Accumulators for Subtests 8 and 9.

```
Subtest 8          Subtest 9
Write Backward     Read Backward

Address Pattern    Address Pattern    AC

TRNN,17,,37777     TRNN,17,,37777     ! 00
HRRI,2,,320,2      HRRI,2,,320,2      ! 01
HRLZI,16,,0        HRLZI,16,,0        ! 02 <START>
XOR,16,,17         XOR,16,,17         ! 03
HRRI,2,,460,2      HRRI,2,,460,2      ! 04
SXCT,,,13          SXCT,,,13          ! 05
JRST,,,10          CAME,15,,16        ! 06
                   HALT,,,1           ! 07
CAME,17,,14        CAME,17,,14        ! 10
SOJA,17,,0         SOJA,17,,0         ! 11
HALT,,,0           HALT,,,0           ! 12
MOVEM,16,,0,17     MOVE,15,,0,17      ! 13
LAST ADR           LAST ADR           ! 14
ECHO               ECHO               ! 15
PATTERN            PATTERN            ! 16
ADDRESS            ADDRESS            ! 17
```

The 4-word read test checks the ability of memory controllers to properly do a 4-word read. The test uses the only piece of hardware guaranteed to be on a KL10 system and capable of doing a 4-word read: the pager. The pager is stimulated by using a MAP instruction. After the pager is cleared by a CONO PAG, eight MAP instructions are done, the first of which causes a page refill: a 4-word read, word 0 first.

The pattern in the executive page table (EPT+600) consists of three 0 words and a -1 word. Since the information returned by the MAP instruction reflects the contents of a halfword, 8 MAPS are necessary to check 4 words. Also, since it is known only that the pager can hold 0s (at this point in the test sequence), the occurrence of a 1 bit is taken to mean that the corresponding halfword was all 1s. In this manner a halfword response pattern is built up, which is then checked against the expected response pattern.

After this is done, the page table is checked to ensure that it is unchanged. A failure here indicates that the read-restores of the 4-word read were unsuccessful. If the page table is correct, the procedure is repeated with the next 4-word pattern.

If an error is detected, the PDP-11 starts a fast loop running in the KL10. It is a 3-instruction loop which does 4-word reads. Sync on MEM START L which is CPU pin DV2, slot 22.

Memory test fault data dumping subroutines depend on the following.

AC15	Data pattern back from memory
AC16	Data pattern sent to memory
AC17	Address under test

GENERAL INFORMATION

Code	DGKBC.A11
Title	KL10 Paging Logic Diagnostic
Abstract	This diagnostic program is designed to detect and isolate all faults in the MBox and EBox logic associated uniquely with paging operations. This logic is on the PAG, PMA, and CSH boards in the MBox, and the SCD and MCL boards in the EBox.
Hardware Required	KL10-PA mainframe/at least 16K of KL10 main memory.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	The 16K of KL10 memory must be configured.
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKBC TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGKBC Test Summary

Test	Description
1	Cache to IR Data Line Test This test checks the cache data line to IR register data path. It executes two diagnostic functions, of which the first disables the cache and the second selects the AR register to be read through the memory to cache multiplexer on the MB board. Next, a test pattern is loaded into the AR. An MBox cycle is started by the microcode and at the end of the MBox cycle (MBox response is true), the data from the AR, through the memory to cache multiplexer, through the cache data buffers, to the IR register, is actually loaded into the IR. The PDP-11 then reads the IR to verify that the data transfer has occurred correctly.
2	Page Table RAM Test - Addresses 0-511, Data Bits and Addressing This is the test for the page table RAM (M8520-PAG1, PAG2, PAG5) and serves as both an address test and a data test. The test is intended to detect any data bits stuck either high or low, and to detect any addressing problems. This includes the address path from the VMA, to the address bit buffers (PAG5), to the page table RAMs themselves.

Table 1   DGKBC Test Summary (Cont)

Test	Description
	The testing algorithm is commonly referred to as galloping write-recovery and the algorithm is described as follows. Write 0s to location 0 (cell A) Loop Write 1s to location 1 (cell B) Verify 0s in location 0 (cell A) Increment B, go to loop Continue repeating above process until B equals last page table address. When B is equal to the last page table address, increment A, set B=A and start the whole process over again. When A is equal to the last page table address, start the whole process over again, writing 1s instead of 0s, and 0s instead of 1s. The test will be run in two major passes. The first pass with 0s disturbs a cell under test (C.U.T) containing 1s. This will run in the ACs, for each cell with no interference from the PDP-11. When it completes, the ACs will be reloaded to run the same test, only using 1s to disturb a C.U.T holding 0s.
3	Page Table Directory RAM Test - Addresses 0-128, Data Bits and Addressing This is the test for the page table directory RAM (M8520-PAG3) and serves as both an address test and a data test. The test is intended to detect any data bits stuck either high or low, and to detect any addressing problems. This includes the address path from the FMA, to the address bit buffers (PAG5), to the page table RAMs themselves. The testing alogrithm is commonly referred to as galloping write-recovery and the algorithm is described as follows. Write 0s to location 0 (cell A) Loop Write 1s to location 1 (cell B) Verify 0s in location 0 (cell A) Increment B, go to loop Continue repeating above process until B equals last page table address. When B is equal to the last page table address, increment A, set B=A and start the whole process over again. When A is equal to the last page table address, start the whole process over again, writing 1s instead of 0s, and 0s instead of 1s. The test will be run in two major passes. The first pass with 0s disturbs a cell under test (C.U.T) containing 1s. This will run in the ACs, for each cell with no interference from the PDP-11. When it completes, the ACs will be reloaded to run the same test, only using 1s to disturb a C.U.T holding 0s.
4	Page Table Public Bits to SCD Flags This test checks two gates on the SCD board, as they affect the two SCAD flags, SCD PUBLIC and SCD PRIVATE INSTR. The gates that are checked are gates which include SCD PRIVATE INSTR or SCD PUBLIC PAGE as inputs. See microcode listing for more precise test explanation.
5	Paged Read (Page O.K.) Sequence This test checks the sequencing of a paged read from memory by stepping through a MOVE 17,410123. The exec page table entry has the access, public writable, software, and cache bits all "on." A CONO PAG at the beginning of the test sets the EPT to 5 and clears the paging RAM valid bits so that a refill is needed.

Table 1 DGKBC Test Summary (Cont)

Test	Description
	Subtest 1 checks the MBox states at refill T4, the beginning of the page refill cycle. Subtest 2 checks the MBox states at the EBox following the refill cycle, when the MOVE should begin to be executed. Subtest 3 checks the MBox states at the start of the core read cycle of the MOVE.
6	Refill and Paged ADDR Test This test checks the generation of refill addresses and the resulting paged address by using a MOVE 17XXX123 instruction. XXX is the page number and is picked up from an index register. References are done in either kernel (exec) or concealed (user) mode depending on the subtest. The addresses are checked by enabling the EBus register to be loaded from the PMA lines on every clock, and reading that register with a diagnostic function when desired. The exec and user process tables have the access, public, writable, and software bits on, and the exec table also has the cache bit on, for no particular reason. If an error occurs, the test setup is fully described in the error report. There are 18 patterns used in this test. Subtests 1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52 check the address to be sent to memory for the refill cycle. The address is checked at CSH T2 time and is generated by the PMA board. Subtests 2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, 53 check for refill error or page fail at CSH EBox T3 after the refill cycle is complete. The page fail code is reported if an error occurs. The PAG board is the suspect. Subtests 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54 check at CSH EBox T3 for the correct paged address generated by the PAG and PMA boards together.
7	Page O.K. Logic Test This test tries 11 paged operations which should not cause a page fail. It tests the page o.k. - page fail logic on the PAG board and the EBox signals which feed to it. Each test is preceded by a CONO PAG so a refill cycle will occur. In the event of an error the test conditions are described in the typeout. All the pages used are referenced for reads in test 6. Subtests 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 check at refill T4 to make sure a refill was started. Subtests 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22 check at CSH EBox T3 after the refill to make sure no page fail has occurred.

GENERAL INFORMATION

Code	DGKBD.A11
Title	KL10-PA Internal Channel Control Test
Abstract	This diagnostic routine is designed to detect and isolate faults in the operation of KL10-PA internal channel control. It is not a complete test of the internal channel logic. To complete testing, run DGKBE (Internal Channel Loopback Test).
Hardware Required	KL10-PA mainframe/internal channels
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restriction	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKBD TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. (Refer to the 11/10 STD module.)

Table 1 DHKBD Test Summary

Test	Description
1	Checks the channel logic timing. Subtests 1-5 check CH MR reset and CH T0-T3. Subtests 6 and 7 check CH timing block and MB REQ INH. Subtests 8-37 check CBus select logic. Even subtests check the CH board CBus selects, and odd subtests check the CRC CBus SEL D signals for the proper CBus selects at D-TIME.
2	Checks the state of the channel logic after a master reset + 1 tick.
3	Checks the channel reset function. See subroutine CHRCHK for additional information. It also checks that no RH20 is interfering with the CBus during a channel scan.
4	Checks the ability to generate channel EPT references for fetching initial command words for each channel. The base page used is determined at assembly time.
5	Checks the ability to fetch a halt with an address field of 17777777.
6	Checks the CCW ADDRESS, CCW MEM ADR = 0 and CCW ODD ADR PAR for a CCW of 0.
7	Slides a 1 and a 0 through the initial CCW fetch address. The sliding 1 is used to test CCW MEM ADR = 0 and CCW ODD ADR PAR generation as well as the CCW address.
8	Checks CCL WC = 0, 1, 2, 3, GE4 and ODD WC PAR by counting up a halt with word counts of 0-4 and then a sliding 1 for word counts greater than 4.

Table 1 DHKBD Test Summary (Cont)

Test	Description
9	Checks sequencing and addressing for store status. Proper status for an MB PAR ERR with a CLP - 17777777 is also tested for all channels.
10	Checks storing of status word 2 with data of 400000 000000. All channels are tested.
11	Checks storing of status word 2 with data of all binary 1s. All channels are tested.
12	Checks status word 1 for address parity error. All channels are tested.
13	Checks status word 1 for nonexistent memory error and a CLP = 0 for all channels. The CBus start is also tested.
14	Checks the jump.
15	Checks that no action flag request occurs for CTOM transfers with a word count of 0 for all channels. Incremental CCW fetches are also tested.
16	Checks channel priority, and it address-tests the CCW buffer. It does this by causing one channel to hold a command list pointer of 17 777676 and a channel command word of 400000 000101. This channel is started first. All other channels are then started. These channels generate a complement CLP and CCW. After all channels have logged out with parity errors, the first started channel status is tested to ensure that no other channel has disturbed its data. The test continues until each channel has been first.
17	Checks proper time states for a 1-word CTOM 0-fill with a word count of 1 and the done interrupt for all channels. Decrementing the word count from 1 to 0 tests the CCL ALU for the all-carry case.
18	Checks the action counter for initial word counts of 1-4 and address = 0 (CTOM 0-fills all channels)
19	Checks the CCL ALU and REQ CTR. It performs this by down-counting word counts of 1-7. In order to test that the CCL ALU carry propagation stops at the correct point, the maximum word count is then down-counted.
20	Checks data fetch addressing of one word for all channels.
21	Checks data fetch addressing of two words for all channels.
22	Checks data fetch addressing of three words for all channels. Long WC error status is also tested.
23	Checks data fetch addressing of four words for all channels. The ability to unload the buffer and normal status are also tested.
24	Checks reverse read addressing of one word for all channels.
25	Checks reverse read addressing of two words for all channels.
26	Checks reverse read addressing of three words for all channels.
27	Checks reverse read addressing of four words for all channels.
28	Checks the CCW ALU for all carry cases not checked in preceding tests. This is done by performing a 1-word CTOM data transfer from a memory address chosen to carry to the next high-order address bit. A store status is used to load the CCW CHA register with the incremented address portion of the data transfer word. The test terminates when a full 22 bits have been tested or a nonexistent memory has been detected.

Table 1 DHKBD Test Summary (Cont)

Test	Description
29	Checks 0 fill data fetch addressing of four words.
30	Checks RH20 error status for all channels.
31	Checks overrun error status for channels 0-3.
32	Checks overrun status for channels 4-7. It also tests RAM addressing for the overrun bit by checking for interference between the RAM address and its complement.
33	Checks status for a last-transfer error followed by a no error transfer for each channel.
34	Checks status for a last-transfer error followed by another last transfer error to test CMD toggled and CMD stored.

GENERAL INFORMATION

Code	DGKBE.A11
Title	KL10-PA and PV Internal Channel Loopback Test
Abstract	This diagnostic routine is designed to detect and isolate faults in the operation of both the KL10-PA and KL10-PV internal channel data paths. It is not a complete check of the internal channel logic.
Hardware Required	KL10-PA or -PV mainframe/internal channels
Preliminary and Associated Procedure	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None

Notes

The following notes apply to the operation of DGKBE.

1. If the APR ID does not have the internal channel option bit set, the test will ordinarily be aborted. This is to permit proper chain-mode operation of KLDIAG when no internal logic is installed in the KL10 being tested. To force running of DGKBE, do a DIACON TS command and start at test 1.

2. The DIACON test start has been extended to permit the testing of some channels to be bypassed. When performing a DIACON TS command, a starting channel will be solicited. The channel number entered will be used as the CTOM channel of a channel loopback pair. Channel pairs are selected as follows.

CTOM Channel	CTOM Channel
0	1
1	2
2	3
3	4
4	5
5	0
6	7
7	0

Fast mode loopback tests are constrained by hardware to operate only on channels 2 and 3.

To test all channels when performing a test start, type 0.

3. The DIACON test loop has been changed to freeze test loops to a specific channel pair if possible. When performing a DIACON TL command, a channel will be solicited. The channel entered defines a channel pair for the range of tests specified.

Loading and Starting Procedures	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.)
Exceptions	The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 4 (INHPAG), and 3 (MODDVC).

Switch 6 INHIBIT PROGRESS REPORTING inhibits normal pass typeouts for progress reporting except for end of pass.

Switch 2 INHIBIT CACHE inhibits the use of cache when performing fast-mode loopback tests.

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON. (Also see notes.)

DGKBE TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGKBE Test Summary

Test	Description
1	Checks data paths and the channel buffer for 777777 000000.
2	Checks data paths and the channel buffer for 000000 7777777
3	Checks channel buffer addressing for channel pairs by looping back 16 words with one word of 1s in a field of 0s. The word of 1s is tried in each channel buffer location.
4	Checks channel parity addressing for channel pairs by looping back 16 words with even upper/lower parity in a field of 0s. The word of differing parity is tried in each channel buffer location.
5	Address-tests the channel buffer by performing a channel buffer load in such a way as to ensure that a specific channel buffer location is loaded with all 7s. All preceding locations for that channel are cleared. Enough CBus requests are then performed to ensure that the location in question will unload at the next CBus request for that channel. All other channels' buffer locations are then cleared. CBus requests are then performed on the loopback channel to ensure that it terminates normally. This channel is then started with a 1-word CTOM command list. Loopback of one word from the original channel is used to ensure that no channel buffer load has disturbed the original location's data. The test continues until each buffer location on the test channel has been the first to hold 7s and all channels have been first.
6	Address-tests the channel buffer parity bit by performing a channel buffer load in such a way as to ensure that a specific channel buffer parity location is loaded with 0 upper/lower parity. Preceding locations for that channel are cleared. Enough CBus requests are then performed to ensure that the location in question will unload at the next CBus request for that channel. All other channels' buffer locations are then cleared. CBus requests are then performed on the loopback channel to ensure that it terminates normally. This channel is then started with a 1-word CTOM command list. Loopback of one word from the original channel is used to ensure that no channel buffer load has disturbed the original location's parity. The test continues until each buffer location on the test channel has been the first to have zeroed parity bits and all channels have been first.
7	Checks upper/lower half swap for read reverse. Slow data path switching may also be caught.
8	Checks fast-mode data transfers to memory. Data is taken from cache if cache exists and cache has not been disabled by setting console switch 2. The loopback data is a sliding bit pattern. First it is determined whether or not fast-mode loopback can be run within the memory bandwidth restrictions. If the memory is one-way interleaved and no cache is installed or selected, the test may not be run due to overruns.
9	Checks fast-mode reverse loopback of a sliding bit pattern.

GENERAL INFORMATION

Code	DGKCA.A11
Title	KL10-PA Meter Board (M8538) Diagnostic
Abstract	This diagnostic is designed to detect and isolate all faults related to the operation of the KL10-PV meter board.
Hardware Required	KL10-PA mainframe/Meter Board/ MCA20 (optional)
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), and 3 (MODDVC).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGKCA TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This program uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGKCA Test Summary

Test	Description
Quick MR Test for Meter Board	
1	A quick master reset test used to verify that everything on the board is clear so that the rest of the tests may run.
Period Register Data Path Test	
2	Checks the data path: EBus, EBus latches, period register, and 1/8 multiplexer to EBus.
Meter Board Time Base Clock Tests	
3	Checks the on/off function of time base. NOTE Meter board Tests 4 and 5 check clearing functions for the time base counter.
4	Checks master reset.
5	Checks the clear-time-base function.
6	Checks the function of the time base cntr.

Table 1 DGKCA Test Summary (Cont)

Test	Description
Meter Board Interval Timer Tests	
7	Checks the on/off function of the interval timer. It also checks that the timer does not run when off.
8	Checks the master reset with respect to the interval timer.
9	Checks the function of the clear-interval-timer bit of the CONO TIM instruction.
10	Checks the data path for the interval timer priority interrupt assignment.
11	Checks the interval timer with respect to the done and overflow flip-flops, and the ability to clear them. It also checks the ability of the timer to cause a vector interrupt. Initialization consists of setting up the EPT, and loading a program which will allow the vector interrupt to occur.
12	The count read test for the interval timer. Initialization just clears the PI system from the last test.
Meter Board Accounting Logic Tests	
13	Checks the data path through the accounting flip-flops.
14	Tests the accounting flip-flop logic. Initialization loads the PI in progress AC program (not on loops), and does a SM. The test itself is table-driven (see below). The table bytes direct the AC program to set up the desired machine state to test the desired combination of the accounting control bits. Driver byte format is XABCUPPP Where: X=1 means counters should be changing A,B,C are PIACTE, EXACTE, and ACCTON, respectively U=1 (bit 3) means user mode test PPP is the PI level desired (000 means NO-PI). (See listing for driver byte contents.)
Meter Board EBox use Counter Tests	
15	Tests the master reset function of the EBox counter, and that it is not counting MBox waits.
16	The count read test for the EBox accounting counter. AC program-loaded by I.15.
Meter Board MBox (Cache) Counter Tests	
17	Checks the master reset function of the MBox (cache) accounting meter of the M8538 board.
18	Checks the ability of the MBox (cache) counter to function properly. Also checks the multiplexer inputs.
Meter Board Performance Analysis Counter Tests	
19	Checks out the event duration bit of the PERF counter enables, and makes sure that other ignores can all be set and are functional. NOTE Meter board tests 20 and 21 check the clearing function of the performance analysis counter.
20	Checks the master reset function of the PA counter.
21	Checks the function of the CLR-PA-CNTR bit of the BLKO TIM instruction.

Table 1 DGKCA Test Summary (Cont)

Test	Description
22	Count read test for the performance analysis counter.
23	Checks the PI-level inputs and enables for the performance analysis counter. The outer loop cycles through PI levels 7-1, 0 (via an examine), and NO-PI. During each of these levels all PI-level enables are set, one at a time, to check out the select circuitry. Initialization is the same as for I.14.
24	Performance analysis counter user/exec mode test.
25	Checks the performance analysis counter input and enables for the probe.
26	Performance analysis counter cache inputs test.
Meter Board Miscellaneous Tests	
27	Meter board test 27 is the functional test of the instructions which are used to access the four 16-bit counters; i.e., the combinations of (DATAI, BLKI)*(TIM,MTR). The order of the test is as follows.

Subtest	Instruction	Counter	DFRD No.
1	DATAI TIM	Time base counter	110
2	BLKI TIM	Performance analysis	111
3	DATAI MTR	EBox accounting counter	112
4	BLKI MTR	MBox accounting counter	113

GENERAL INFORMATION

Code	DGMCA.A11
Title	KL10-PA MCA20 Cache Option Diagnostic
Abstract	This diagnostic routine is designed to detect and isolate faults relating to the operation of the MCA20 cache option.
Hardware Required	KL10-PA mainframe/MCA20
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 11 (NOT USED), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGMCA TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic used the Standard error message format. Refer to the 11/10 STD module. Also refer to the individual test descriptions in this module.

Table 1 DGMCA Test Summary

Test	Description
1	Refill Ram Test This test checks addressing and data paths of the cache refill algorithm RAM on the CHX (M8515) board. The refill RAM consists of the three 10147 RAMS on CHX3 whose outputs are called CSH USE IN 2, CSH USE IN 3, and CSH USE IN 4 H. When writing into this RAM, the data and address come from the VMA (M8523). A PDP-10 instruction WRFIL 700100,,Y writes one address of this 3-bit RAM. The diagnostic can read the RAM address inputs and data outputs by doing diagnostic reads from the PDP-11. First, a refill RAM writing program is run in the KL10 ACs to try to clear the RAM to all 0s. Then a program to write 1s is loaded into the ACs but not run. Instead it is single-pulsed by the PDP-11 through 128 writes of the RAM. The program checks the data and the address at the RAM twice during each of the 128 writes - once just before the write pulse to check for 0s, and again just after the write pulse to check for 1s just written. In the error message, 16 bits are typed out. The left half is the address lines, the right is the data: 0 AAA AAA A00 000 DDD where A is address bit, D is data bit. Subtest 1 checks for correct address count and for 0 data. If the address is 0 and the data is not, then the original write of all 0s may not have worked because of failing write pulse generation. If the address is not 0, but the data is, then there is an address path failure somewhere. A failure because of non-0 data is probably a failure of the 10147 RAM.

Table 1 DGMCA Test Summary (Cont)

Test	Description
	Subtest 2 rechecks addressing and checks the data for 1s. An error in an address bit is probably due to a data bit being tied to the address bit. Zeros in the data can be an open path in the data input or a failed 10147. An error report with no subtest number and no data typeout means that the write refill RAM instruction control logic is not working and the MBox never got to CSH EBox T1 or CSH EBox T3. Loop for sequencing problems, microcode hangs, execute the following console commands: R SM,EX 700100 0 — repeat start microcode, execute a write refill RAM instruction. Loop for data/address problems (console commands): SM — reset and start microcode. DM 0/700100 000AAA — write 0 to RAM address AAA. DN 700100 700AAA — write 1s to RAM address AAA. DN 254000 0 — loop ST0 — start at AC0
2	Refill and Use RAM Test This test checks the refill RAM and the use bit RAM together by using a cache sweep. The use bit RAM is the five 10147s on CHX3 whose inputs are called CSH USE IN 0 (1,2,3,4) H. During an all-pages sweep, this RAM is written into four times for each address. Its outputs are used as part of the address for reading the refill RAM. The refill RAM's outputs are part of the data written into the use bit RAM. The test writes all 0s to the refill RAM and does a sweep at full clock speed. This should cause all 0s to be written into the use bit RAM by the sweep. Next, a special repeating test pattern is written into the refill RAM. The test then burst-pulses a sweep of all pages, invalidating cache, and checks at each CCA INVAL T4 that: 1. The CCA address counter has counted down correctly (it goes from 777 to 000) 2. The use RAM outputs indicate that the RAM is being written correctly. Subtest 1 checks all the MBox diagnostic functions to see if the sweep correctly got to CCA INVAL T4 for the first time. Subtest 2 errors will only occur during program development. Subtest 3 checks the CCA address by clocking the PMA address lines into the EBus register which is read by diagnostic function. Subtest 4 reads the data and address signals from the M8515 board and checks the use RAM and refill RAM. The error printout shows 16 bits as in test 1. Errors in the left 8 bits may be caused by bad 10147s, an unterminated address line, faulty use bit write pulses, or failure of CSH USE HOLD (M8513) to hold the refill address during a sweep. If there is a failure only in the right-hand three bits of the typeout, there are probably two refill RAM outputs tied together. Loop for sequencing problems: EX 701440 0,TD 20 ,or R SM,EX701440 0,TD 20 The TD time delay may be either adjusted or omitted as preferred.

Table 1   DGMCA Test Summary (Cont)

Test	Description

	AC loop which waits for sweep done and loads the EBus register with the sweeper address as in subtest 3:
	
	`SM                  CONSO APR, clear sweep done`
	`DM 0/700200 20020   start invalidating sweep`
	`DN 701440 0         CONSO APR, sweep done?`
	`DN 700340 20        wait for done`
	`DN 254000 2         loop`
	`DN 264000 0         set PMA to EBus reg enable`
	`STO                 run it`
	
	Sync: CSH CCA CYC, CSH USE HOLD

3	Cache 0 - Write to Cache Test
	
	This test checks the control logic sequencing of a write that should go to cache. For the sake of symmetry this test is performed on each cache separately although almost all of the control logic involved is the same for each cache. The one difference is the directory for the cache being tested, and the gate that detects any written bits. The cache under test should have had all written bits cleared out of its directory by the cache sweep. If the control logic detects any written bits, it will try to do a writeback to core before allowing the write to cache.
	
	Subtest 1 checks the state of MBox signals at clear WR T0. This check is performed at a fixed number of clocks after the start of the write instruction, so any failure in the sequence of time states - including a branch into the writeback logic - will cause an error in this subtest. Also failure of CON CACHE LOOK and CON CACHE LOAD EN will cause a reference to core instad of cache.
	
	Subtest 2 checks whether the CSH logic sequences to CSH DATA CLR T2. It checks the state of all MBox signals at this time.
	
	Subtest 3 advances to CSH EBox WR T4 and checks the MBox diagnostic signals at this state.
	
	Subtest 4 checks the address of the write instruction by clocking the PMA lines into the EBus register and checking the EBus register contents using a diagnostic read.
	
	Subtest 5 runs the clock and checks that the MBox gives MB response and that the EBox microcode then sequences back to the halt loop.
	
	Loop for sequencing problems:
	
	`R SM,EX 701200 600000,EX 202000 1777`
	
	For looping in the ACs:
	
	`SM`
	`DM 0/701200 600000    set CON CACHE LOOK,LOAD`
	`DN 202400 1777        move to cache`
	`DN254000 1            loop`
	
	`DM 10/ "data"`
	`STO                   start it going`

4	Cache 0 - Sweep, Valid, Written Test This test uses and checks an invalidating cache sweep of one page. At the same time that the sweep is being checked, the test also checks that a write like the one used in the previous test actually succeeded in writing a valid and a written bit in cache loction 1777. Since the 1-page sweep and the all-pages sweep (tested earlier) are very similar, only the differences are tested. The main difference is that the CCA counter on the PMA board counts down by 4 instead of by 1, and that the low two address bits are determined by which cache's directory (if any) matches the physical page in VMA 14-26.

Table 1 DGMCA Test Summary (Cont)

Test | **Description**

Subtest 1 single-pulses the sweep to the first CSH T3 (CCA T3) and checks the state of all MBox diagnostic signals.

Subtest 2 checks the CCA and page address by clocking the PMA outputs to the EBus register and reading the register with a diagnostic function.

Subtest 3 checks the CSHn ANY VALID and CSHn ANY WRITTEN signals, where n is the cache number currently being tested. Although the previous test (3,8,13,18) checked the control logic used for writing the valid a bits, it is not known that M8514(CHA) or M8515(CHX) did store those bits. A failure in this subtest means either that a move to cache is failing or that the sweep is unable to read out these bits.

Subtest 4 burst-pulses the sweep to the second CCA T3 and again checks the sweep address to be sure it was decremented by 4 and not 1. (Actually CCA 34 and CCA 35 keep on decrementing by 1 but the carry or borrow between bit 34 and bit 33 is disabled.)

Subtest 5 runs the clock to finish the sweep and starts a new sweep by burst-stepping it to CCA T3.

Subtest 6 checks the valid and written bits to see if they were cleared by the previous sweep as they should have been.

Loop - To loop on a write followed by a 1-page invalidating sweep:

```
SM
DM 0/701200 600000    set CON CACHE LOOK,LOAD
DN 700200 20020       clean sweep done
DN 202400 1777        MOVEM 10,1777
DN 701640 1           sweep invalidating page 1
DN 700340 20          check sweep done
DN 254000 4           wait for it
DN 254000 1           loop
```

5 Cache 0 - Directory RAM Test

This is a test of the cache directory address, valid, and written bits. The cache is tested one (out of every four) word at a time; that is, all 128 "word 3s" are tested. Then word 2s, and so on until all 512 words of the cache are tested.

Each time that 128 words are written, a sweep is pulsed by the PDP-11 program and stopped at each of the 128 CCA INVAL T4s that occur. At this time state, the program checks CSHn ANY VALID and CSHn ANY WRITTEN, as well as the cache directory parity bit for errors.

After the entire cache has been checked and swept, a second sweep of the same type is single-stepped through completely to make sure that no valid and written bits remain stuck true.

Subtests 0 - 511 are the decimal address of the failing cache line. Some directory bits (e.g., "valid") are stored by word number, which is the low two bits of the address.

If the low address digit is:	The word number is:
0,4,8	0
1,5,9	1
2,6	2
3,7	3

If it is not convenient to convert addresses from decimal to octal, the console ERG command can be used to type the EBus register after the test has halted. The right three octal digits are address bits 27-35.

Table 1 DGMCA Test Summary (Cont)

Test	Description
	If a cache DIR PAR odd error occurs, the bit can be located by the following procedure: Set the halt on error switch and run the test up to this error. Locate the backplane pins on the M8514 drawing for the CSH DIR bits 14 through 26. Make very sure you have the bits for the failing cache. There are four sets of DIR bits. Then use a scope on the backplane to check these bits. Only bit 26 should be high at this time; all others (14-25) should be low. Replace the bad 10144. Subtest 512 checks the sweep control bits too ensure that they are still the same as in the beginning. Subtest 513 checks the cache after the first sweep and reports the first address it found that had either a valid or a written bit still true. The failing address is typed out in binary. Loop - See looping instructions for test 4.
6	Cache 0 - Directory Disturb Test A This is a test of cache directory addressing. Together with the next test in sequence, it determines whether the internal address decoders of the cache directory and extension RAMs are working properly. For initial conditions, it depends on the fact that the intialization has written page 1 with correct directory parity in all locations, and that the valid and written bits have all been swept off. Passing the previous test makes this assumption valid. The test then writes four words to each of seven quadwords in cache. The quadword addresses are determined by floating a 1 through address bits 27 to 33. The page written is the complement of page 1; that is; bits 14-26 = 17776. The theory is that if a chip's address decoding is wrong, writing the 28 disturber words will cause a detectable error in one of the locations containing page 1, or a cleared valid or written bit. The test steps through a sweep checking cache directory parity and looking for valid or written bits that were set. Subtests 0-511 - The subtest number is the decimal address of the cache line having the failure and the error report indicates whether a valid, written, or parity failure occurred. In the case of a parity error, the diagnostic cannot indicate which directory bit was picked. With an oscilloscope, however, you can check on the backplane the signals called CSH DIR 14 n through CSH DIR 26 n H. (n is the cache number printed in the error message). Only bit 26 should be high. Any other high bit is an error. The MBox is at CCA T3 (CSH T3) of a sweep.
7	Cache 0 - Directory Disturb Test B This test is a companion of the previous test and completes the disturb testing of the cache directory by using a different pair of pages (5252 and 12521) than before. The test procedure is the same as for test 6. Subtests 0-511 - As in test 6, the subtest number is the decimal address of the failing cache line. In the case of a directory parity error, the expected data on CSH DIR 14 n through CSH DIR 26 n is 5252. That is, bits 15, 17, 19, 21, 23, and 25 should be high and all others low.
8	Cache 1 - Write to Cache Test (Refer to test 3)
9	Cache 1 - Sweep, Valid, Written Test (Refer to test 4)
10	Cache 1 - Directory RAM Test (Refer to test 5)
11	Cache 1 - Directory Disturb Test A (Refer to test 6)

Table 1 DGMCA Test Summary (Cont)

Test	Description
12	Cache 1 - Directory Disturb Test B (Refer to test 7)
13	Cache 2 - Write to Cache Test (Refer to test 3)
14	Cache 2 - Sweep, Valid Written Test (Refer to test 4)
15	Cache 2 - Directory RAM Test (Refer to test 5)
16	Cache 2 - Directory Disturb Test A (Refer to test 6)
17	Cache 2 - Directory Disturb Test B (Refer to test 7)
18	Cache 3 - Write to Cache Test (Refer to test 3)
19	Cache 3 - Sweep, Valid, Written Test (Refer to test 4)
20	Cache 3 - Directory RAM Test (Refer to test 5)
21	Cache 3 - Directory Disturb Test A (Refer to test 6)
22	Cache 3 - Directory Disturb Test B (Refer to test 7)
23	Cache 0 - Cache Read and Data Disturb This test checks the first read to cache and then performs a 1s and 0s data test on the cache. All of page 1 is first written with 0s. A SKIPE 1770 is burst-stepped and the state of the MBox diagnostic signals checked at two places in the execution. Then a subroutine running in the KL10 ACs is used to perform a full-speed data test of the whole page. As the addresses are counted from 1000 to 1777, the cache data is checked for 0s and replaced with 1s. The process is repeated, checking for 1s and replacing with 0s. The first error detected is reported. Subtest 1 burst-steps the SKIPE 1770 to CSH EBox T2 and checks all the MBox diagnostic signals. Subtest 2 burst-steps the instruction to the first EBox clock after MB RESP and checks all MBox signals. Subtest 3 only fails if the AC program hangs up and fails to reach a halt instruction. Subtest 4 reports the address and failing test pattern of a data error. If the expected data is all 0s, possible problems are as follows. a. If the actual data has one or two bits at 1, the problem is probably a bad RAM chip. It could also be the write pulse CSH WRITE A, B, C, etc., on M8521. b. If the actual data has 1s in only one of the four 9-bit groups, the problem may be the address lines on the M8521 containing those nine bits. Also, the write pulse cache WR 00,09, etc., may not be reaching the M8521. c. If the actual data has 1s in more than one 9-bit group, the problem may be the cache address lines coming from M8531.

Table 1 DGMCA Test Summary (Cont)

Test	Description

	If the expected data is all 1s and the actual data contains one or two 0s, the failure may be a RAM chip or open data path etch on the M8521. If the program got this far, the all 0s case passed the test and the problem is not likely to be addressing or write pulses.
	
	Subtest 5 should only fail in program development.
	
	Loop for sequencing problems:
	
	`R SM,EX701200 600000, EX332000 1770 repeat start ucode.`
	`                       set look and load, and SKIPE 1770`
	
	To loop in ACs, using a MOVE instead of SKIPE:
	
	`SM`
	`DM 0/701200 600000    set look and load`
	`DN 202400 1770        write data from 10`
	`DN 200440 1770        read data to 11`
	`DN 154000 11          loop`

24	Cache 0 - Tied Data Bit Test
	
	This test completes the cache data test by checking for cache data inputs or outputs tied together. It uses six patterns having 1s and 0s grouped in various combinations.
	
	Location 1000 of the cache is written with the pattern and then read and checked.
	
	Subtests 1-6 are the pattern numbers. The expected and actual data are shown in the error typeout.
	
	Loop - To loop in ACs, do the following.
	
	NOTE
	ACs 0 to 5 contain the six patterns. Subtract 1 from the failing subtest number to get the AC number.
	
	`SM`
	`DM10/701200 600000    set cache look, load`
	`DN 200740 1000        read from 1000 into 17`
	`DN 254000 10          loop on it`
	
	To loop on the write, change AC11 to MOVEM with an AC field 0 to 5 to select the desired pattern; e.g., DM11/202140 1000 to write AC3 pattern.

25	Cache 0 - Directory Parity Test This test checks the cache directory parity network and error detection logic by using eight test patterns. Each test pattern is a physical page number and either even or odd parity bit. Bit 20 is 1 for writing even (bad) directory parity and bit 20 equals 0 for writing odd (correct) parity. A routine runs in the KL10 ACs which writes, reads, and checks one pattern each time it is started from the PDP-11. The routine does a CONI APR, and the PDP-11 program checks the cache directory parity error bit (28) for the proper result. Subtests 1-4 are writing patterns with odd parity and expect no directory parity errors. Subtests 5-8 are writing patterns with even parity and expect a directory parity error. Loop - If the program stops on an error, the test code is in the KL10 ACs and may be looped by changing the contents of location 10 to 254000 0 (it was a 254200 0).

Table 1 DGMCA Test Summary (Cont)

Test	Description
26	Cache 0 - Writeback 1-Word Test This is the first test of the writeback logic. It uses a sweep and update core instruction to request one word of a quadword to be written back to core and to invalidate the word in cache. The test is repeated for each word of a quadword to ensure that the correct word request to memory is brought up in each case. All core memories are disabled, so a nonexistent memory timeout will occur for each memory reference. Subtest 1 writes word 0 of a quadword in cache, then burst-steps a sweep, 1-page update and invalidate instruction to write back T1 time state. The state of all MBox signals is verified. Subtest 2 burst-steps the sweep to a point where the memory request is active and checks the state of the MBox. Subtest 3 checks after the NXM timeout has completed the memory request. It ensures that the sweep correctly reaches CCA INVAL T4 to invalidate the word which has been written back. Subtests 4, 5, 6 are the same as 1, 2, 3, but for word 1. Subtests 7, 8, 9 are the same as 1, 2, 3, but for word 2. Subtests 10, 11, 12 are the same as 1, 2, 3, but for word 3.
27	Cache 1 - Cache Read and Data Disturb (Refer to test 23)
28	Cache 1 - Tied Data Bit Test (Refer to test 24)
29	Cache 1 - Directory Parity Test (Refer to test 25)
30	Cache 1 - Writeback 1-Word Test (Refer to test 26)
31	Cache 2 - Cache Read and Data Disturb (Refer to test 23)
32	Cache 2 - Tied Data Bit Test (Refer to test 24)
33	Cache 2 - Directory Parity Test (Refer to test 25)
34	Cache 2 - Writeback 1-Word Test (Refer to test 26)
35	Cache 3 - Cache Read and Data Disturb (Refer to test 23)
36	Cache 3 - Tied Data Bit Test (Refer to test 24)
37	Cache 3 - Directory Parity Test (Refer to test 25)
38	Cache 3 - Writeback 1-Word Test (Refer to test 26)
39	Two-word Writeback Test This test tries three 2-word writebacks and counts ACKN pulses. It is primarily concerned with checking the request counting logic on the MBC board. In each test case, RQ0 is paired with one of the other three requests - RQ1, RQ2, or RQ3. The test writes a pair of words to cache and then steps a sweep-and-update-core in order to cause a writeback request. The pairs of locations used are:

Table 1 DGMCA Test Summary (Cont)

Test	Description
	Subtest / Address / Requests: 1 and 2 — 1774,1775 — RQ0,RQ1 3 and 4 — 1774,1776 — RQ0,RQ2 5 and 6 — 1774,1777 — RQ0,RQ3
40	Writeback Address Path Test This test checks the address paths from the cache directory through the CAM 14-26 mixers on the CHX board, through the PMA address mixers to the EBus register. The EBus register is forced to load from the PMA so that it can be checked for the proper address at memory start time. Sixteen entries are written in the cache directory and forced out one at a time by single-stepping a sweep. The page numbers of the 16 entries are chosen to detect any multiple selection problems in the CAM mixers. Subtests 1-8 identify the pattern being used. The cache being selected by each subtest is as follows. Subtests / Cache: 1,5,9,13 — 3 2,6,10,14 — 2 3,7,11,15 — 1 4,8,12,16 — 0
41	Write Forcing Writeback This test causes a writeback by writing to a quadword of cache that is occupied by a word written to a different page. It checks the MBox state when the writeback should happen and when the new word should be getting written into the cache. Cache 1 is used for this test. Subtest 1 writes a word to location 357 in cache, then starts execution of a write to location 1357, burst-stepping to where writeback T1 should be true. The state of the MBox is checked at this point. Subtest 2 burst-steps the instruction to memory start and checks the state of the MBox memory requests. Subtest 3 bursts to what should be CSH EBox WR T4 and checks the MBox state as it should be writing into cache. Subtest 4 checks states at the end of the write to cache.
42	Read Forces Writeback, 4-Word, Read This test uses a read to cause a writeback. Cache 2 is configured and one word written to cache. A read of the same quadword of a different page is used to cause a writeback and also to check writing into cache of the four words fetched from core. Memory is not configured for this test, so the NXM timeout logic is relied on to simulate memory responses. Subtest 1 burst-steps the read to CSH EBox T3 and checks MBox states. Subtest 2 checks the MBox states just after the writeback to memory has been started. Subtest 3 checks MBox states after the NXM logic has finished the writeback. Subtest 4 checks the MBox state at the time when core read in progress goes true for the 4-word read. Subtest 5 checks states when MBox response should be true to deliver the read word (word 3) to the EBox. Subtest 6 checks the MBox as it should be writing word 3 to cache. Subtest 7 checks the MBox as it should be writing word 0 to cache.

Table 1 DGMCA Test Summary (Cont)

Test	Description
	Subtest 8 checks the MBox as it should be writing word 1 to cache.
	Subtest 9 checks the MBox as it should be writing word 2 to cache.
43	Read With Look Only
	This test checks cache operation with CACHE LOOK EN true but CACHE LOAD EN false. A cache word is written and then a read is performed and checked to see that it does not cause a writeback of the word in cache (a different page, of course) and that the word read from core does not get written into cache. Cache 2 is used.
	Subtest 1 starts executing the read and checks MBox states at CACHE EBOX T2.
	Subtest 2 checks the start of the memory reference at CSH DATA CLR DONE.
	Subtest 3 checks MBox states at the end of the read sequence.
	Subtest 4 steps a read of the original word in cache and checks states at CSH EBOX T3 to be sure the cache directory is still intact.
44	Read-Pause-Write Data in Cache
	This test does a read-pause-write with cache turned on and checks to see that it gets turned into a read reference followed by a write reference when the data is already in cache. Location 357 is written in cache and an AOS 357 is used as the read-pause-write reference instruction. The cache is configured to operate in all four caches. This test should use cache 3.
	Subtest 1 starts the AOS instruction, burst-steps to where MB RESP should be true, and checks MBox states.
	Subtest 2 burst-steps to the MB RESP of the write reference and checks MBox states.
45	Read-Pause-Write Data Not in Cache
	This test checks a read-pause-write operation with the cache containing the right quadword but not the desired word. The read reference should cause a 3-word fill read to be performed and the write reference should write to cache.
	Subtest 1 checks at CSH EBOX T3 of the read request.
	Subtest 2 checks after the NXM timeout when MB RESP for the read should be true.
	Subtest 3 checks MBox states during the writing of word 1 from memory to cache.
	Subtest 4 checks MBox states during the writing of word 2 from memory to cache.
	Subtest 5 checks MBox states during the writing of word 3 from memory to cache.
	Subtest 6 checks MBox states when cache idle should be true at the end of the write reference of the read-pause-write.
46	Error Address Register Check
	This test forces errors during cache operations and checks that the error address register gets the right information. Cache 3 is used for the test.
	Subtest 1 writes a word of bad data parity into cache and then uses a sweep to write it back to memory. The writeback operation should detect the parity error.

Table 1 DGMCA Test Summary (Cont)

Test	Description
	Subtest 2 writes a word in cache with bad data parity. A write to the same line of a different page is used to cause a writeback during which the data parity error should be caught. Subtest 3 writes a word in memory with bad parity and does a read-pause-write reference to it with the cache turned on.
47	CSH DIR PAR ERR Inhibits Cache This test causes a cache directory parity error and checks that cache bit is held false and force-no-match is held true from then until the APR CSH DIR PAR ERR flag is cleared. It checks that clearing the APR flag restores the cache to operation. Subtest 1 writes a word with incorrect directory parity to cache location 777. It then steps a read which will cause a writeback (MOVE 17,1777). The CSH DIR PAR ERR will be detected during the writeback. A full scan of MBox states is taken at the time when APR CSH DIR PAR ERR is true. Subtest 2 bursts the clock to finish the MOVE and then checks that CSH DIR PA ERR is holding the cache off. Subtest 3 clears the CSH DIR PAR ERR flag and checks that cache bit goes true and force no match goes false.
48	Cache to Memory Data Exercise This test runs a read/write program in the ACs and monitors the results. The AC program compares the data read with that written and also checks APR flags for SBus error, NXM, MB PAR ERR, and CSH DIR PAR ERR. If the 11 program finds that the AC program made an error halt, the information pertaining to the type of error is collected and printed (CONI APR data, error address register, data address, expected and actual data). The AC program writes eight pages (4096 words) of data through the cache into memory and then reads and checks eight pages. Addresses are down-counted from X17777 to X10000. (X is normally 0, but it depends on where the lowest 16K memory module was configured in the address space.) The data word is down-counted in each of three 12-bit groups, changing from 777777 777777 to 777677 767776, for example. This pattern gives every bit a chance to change while causing parity to change on every word.
49	Cache/Paged References This test checks the control and data paths used to fetch page refill entries from the cache. It also checks that the cache bit signal is controlled by PT cache when paging is enabled.
50	DMA20 3-Word OPS This test performs three word reads and writes to the DMA20 to check its multiword request and acknowledge counting logic. This test is in DGMCA because using the cache is the only way to cause multiword writes to the DMA20 on a KL10 without internal channels. Subtest 1 performs 512 3-word writes. One cache (cache 1) is configured so that writebacks will occur on the first of three writes to a new page. A program running in the ACs writes to the cache in this sequence: (Addresses) XXX703,XXX702,XXX701 XX1700,XX1703,XX1702 XXX701,XXXX700,XXX703 XX1702,XX1701,XX1700 and then it repeats.

Table 1 DGMCA Test Summary (Cont)

Test	Description
	The XXs depend on the address of the lowest memory configured on the DMA20.
	The write to cache address: — Causes DMA 3-word requests:
	XX1700 — 3,2,1
	XXX701 — 0,3,2
	XX1702 — 1,0,3
	XXX703 — 2,1,0
	Subtest 2 causes 3-word reads (and a 1-word write) by using a modification of the same AC program as above. It writes one word of a page, then reads a different word, causing the cache to read the remaining (unwritten) words from the DMA20. The write will cause a 1-word writeback of the word written to the previous page.
	Program Sequence — Cache and Memory Actions
	WRITEXXX703 — Writeback 1702
	READXXX702 — 3-word read 703,700,71
	READXXX701 — Read cache 701
	WRITEXX1700 — Writeback 703
	READXX1703 — 3-word read 1703,1701,1702
	READXX1702 — Read cache 1702
	WRITEXXX701 — Writeback 1700
	READXXX700 — 3-Word read 700,702,703
	READXXX703 — Read cache 703
	WRITEXX1702 — Writeback 701
	READXX1701 — 3-Word read 1701,1703,1700
	READXX1700 — Read cache 1700

GENERAL INFORMATION

Code	DGMCB.A11
Title	KL10-PA Cache RAM Banger Diagnostic
Abstract	DGMCB is a diagnostic aimed specifically at the ECL RAMs associated with the KL10 cache subsystem. It has tests for the cache directory, written bits, valid bits, data words, and the use bits. All these items are stored in ECL RAMs. The tests in this diagnostic force the KL10 to run worst-case patterns through these RAMs, and then, if an error is detected, isolate the board and the RAM chip associated with the error. The diagnostic is meant specifically to find intermittent faults in the cache RAM ICs. If it finds one it will call out the board and the chip. The callout for cache logic errors is wrong but the RAM chips called out may lead to the failing control logic. Scope loops to look for intermittent RAM errors are almost useless; however, the diagnostic does do long strings of operations with the cache which could be valuable in tracking down other ,problems while using a scope.
Hardware Required	KL10-PA mainframe/MCA20
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	This diagnostic should be run after DGMCA (the cache diagnostic) since it assumes that the cache control logic is working. If a given error is a result of control logic failure, then the board and DIP numbers are meaningless. There are no isolation routines associated with this diagnostic because the test error routines contain the isolation algorithms. Due to the highly interdependent nature of the cache logic, it is impossible to guarantee that an error in a given test is due to the part being tested. Do not assume that because a chip is called out it must be at fault. There are some wire ORs in existence, and, furthermore, the inputs and outputs of the chips go through other gates which could fail. In other words, think before replacing anything.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), and 3 (MODDVC).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DGMCB TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGMCB Test Summary

Test	Description
1	Cache Refill RAM Test This test is designed to find bit interference within the refill RAMs (E15, E20, and E25 on the CHX board). Hard failure of these RAMs should be detected by DGMCA.
2, 3, 4, 5	Cache Directory RAM Verify Each line (4 words) of the cache is associated with a page number. This number is held in the cache directory RAM on the CHA (M8514) board and the RAMs which hold it are the subject of this test. Note that there is only one page number per four words; therefore, only the first word of each line need be referenced to exercise the directory word. There are three parts to these tests: initialization, fault detection, and fault isolation. The initializations merely select which cache bank is to be used by the test. In addition, test 2 initialization loads the AC program to be used by all four of the tests, does XOR initialization, and shuts off memory. The test itself is done almost entirely from the KL10 AC blocks. The PDP-11 part of the test only loads parameters into the ACs for the AC program to use. Note that the AC program fills multiple AC blocks. There is a great deal of setup involved with this. In particular, pointers must be maintained separately in each block, and control must be passed from one block to the next. AC15-17 perform this latter function and are not part of the test. The errors that this test looks for are cache directory parity and nonexistent memory (NXM). If a directory bit is altered by the writing of another bit in the directory RAM, then a CSH DIR PAR error will occur. If it was the high page address which was modified, an NXM will also occur, and this will lock up the error address register. The NXM occurs because the words in cache were written. Changing the address will cause no match, which causes a writeback to the address specified in the directory. The same can occur with the low page words and a CSH DIR PAR error will result also. Once the fault isolation routine has the error address, it is a simple matter to translate this to the chip number.
6, 7 8, 9	Cache Valid Bit RAM Test Each word of the cache has a bit associated with it which indicates whether the word contains valid or indeterminate data. The RAMs which hold these bits are the subject of this test. The RAMs are located on the CHX board. As in the other tests in this diagnostic, most of the testing is done by the multiblock AC program, with the PDP-11 just providing support. The test works as follows. All the valid bits are cleared by an invalidating sweep. A test bit is chosen. All other valid bits in the RAM are set by writing to the other cache locations. The test location is then read. If a core read was done, everything functions properly. The test location valid bit is now set. All other locations are now invalidated and another read is done to the test location. If the test valid bit is still set, no memory read is done and all is well. The next test location is now selected and the process repeated until all bits of the selected cache have been tested.
10, 11 12, 13	Cache Written Bit RAM Test One of these bits is associated with each word in the cache. If the written bit is set, it means that the word in the cache was created in the CPU and sooner or later has to go to memory. This test is designed to ensure that the written bit RAMs work. The test occurs in the following three parts.

Table 1 DGMCB Test Summary (Cont)

Test	Description
	a. The initializations select which cache bank is being tested. In addition, the first unit of this series of tests loads the AC program. Note that it is a multiple AC block (1-6) program. b. The test itself runs entirely from the KL10 ACs. The PDP-11 does nothing more than start the KL10. See comments with each AC block for description of operation. c. Once a fault has been found the error isolator looks at the ERA to get the error word number. This plus the cache bank number leads directly to the chip number.
14	Test 14 initialization is used only to clean up the incorrect parity words left by the written bit tests. The test itself is a no-op.
15, 16 17, 18	Cache Data Word RAM Test Tests 15-18 are the cache data word tests. Test 15 checks cache bank 0, etc. All testing is done by a multiple AC block program. All the PDP-11 code is used to either load, run, or analyze the results of this AC program. The cache data resides in the M8521 (CHD) boards. Each word in the cache is split across all four of the CHD boards as follows. Bits Slot 00-08 25 09-17 24 18-26 19 27-35 17 These tests are designed primarily to exercise the cache data RAMs (type 10144). If an error is found, they call out the RAM(s) which correspond to the error data. If the error was caused by non-RAM chips, then the E numbers called out will be erroneous, though they may help lead to the failing chip. In other words, think before you grab the soldering iron. Note that the 10144s are 256X1 RAMs. Because of this, the nth bits of the four words of a cache "line" reside in two (not four) RAM chips; the even bits go to one, the odd bits to the other. Hence, the test and associated code refer to even/odd pairs rather than lines. See the individual AC program blocks for description of operation. NOTE Since the RAM outputs are wire-ORed and the inputs are driven as a group, a stuck-at-1 condition may result in the wrong chip being called out. The even word chip will be called out even if an odd word bit is stuck. There is a lot of setup involved in switching between the AC blocks and maintaining separate parameters in each AC block. Don't let this (necessary) setup confuse you. Initializations for the cache data RAM tests select which cache bank will be tested: test 15 checks bank 0, etc. In addition, test 15 initialization loads the AC program for all 4 tests. In the cache data RAM tests, all testing is done by the AC program. If the program finds no error, the PDP-11 just moves on to the next test. Most of the code here is used for generating error typeouts.
19, 20	Cache Data Parity RAM Test

Table 1 DGMCB Test Summary (Cont)

Test	Description
21, 22	These tests are similar to the cache data tests (QV), with the following differences. a. The arrangement of the parity bits with respect to slots is as follows. Cache bank 0 parity - slot 25 Cache bank 1 parity - slot 24 Cache bank 2 parity - slot 19 Cache bank 3 parity - slot 17 b. Because there are only two parity RAMs per board (even and odd words), calling out which RAM is very easy. c. Since complementary patterns in a data path with an even number of bits generate the same parity bit value, different patterns are used: 0 for the nontest pairs, 1 (not -1) for the test pair. This is the reason for having separate data and parity bit tests. d. Since the parity bit is being tested, the test never looks at the data patterns, but only does sweeps followed by a check of the APR flags. If a parity bit was incorrect, the MB parity error flag will be set, with the ERA indicating which word was in error. Initializations for the cache data parity tests select which cache bank will be tested: test 19 checks bank 0, etc. In addition, test 19 initialization loads the AC program for all four tests. In the cache data parity RAM tests, all testing is done by the AC program. If the program finds no error, the PDP-11 just moves on to the next test. Most of the code here is used for generating error typeouts.

under KLDCP

GENERAL INFORMATION

Code	DGMMA.All
Title	KL10 Core Memory Reliability and Margining Test
Abstract	This diagnostic program is designed to detect and isolate faults in the operation of the KL10 core memory. It is not a complete test of the memory system. DGKBB is assumed to have been run successfully. Internal memory controllers are tested one at a time in sequence for each test. This memory is configured at address 200000 while being tested in order to avoid shadow core. External memory is of course tested wherever it appears in the address space. In this case, the program will test shadow core directly.
Hardware Required	KL10-PA or -PV mainframe/internal memory/ME10s/ MF10s/MG10s/MH10s/ MA20s/MB20s.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	This diagnostic may not be run with KLDCP versions before version 12.
Notes	1. Certain external memory bus configurations will not function correctly unless the CPU is operating at clock rate 0. Single-pulsing through external memory references is permitted only in 1-bus mode. 2. External memory tests rely on correct setting of the memory switches in order to determine how the memory is interleaved. It is assumed that the memory is noninterleaved, and it is recommended that reliability runs be performed in 1-bus mode. 3. DGMMA supplies its own microcode. 4. Quick verification of all core memory which responds to SBus diagnostics may be performed by setting the console switches to 0 and executing a SED 1 command to KLDCP following the load. A reliability run may be initiated by setting console switch 6 and executing an SED. All memory which has not failed the test will be properly configured only if the program has completed its last pass. Console switch 15 will cause the program to abort at end of pass.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not applicable: 11 (NOT USED), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL

This diagnostic is controlled via DIACON. In addition, software switches may be appended to the loading and starting procedure.

Format:

```
SED#/S/S/S
```

Refer to Table 1 for a description of the available software switches.

DGMMA TEST SUMMARY

The individual tests performed by this diagnostic are summarized in Table 2.

ERROR SUMMARY

This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DGMMA Software Switches

Switch	Description
/H	Print help message.
/FREEZE:Cx,My	Test only the controller specified by x. If My is specified, test only the module specified by y.
/FREEZE:NONE	This resets to the default of testing all memory.
/MARGINS:x	Invokes margin testing of internal memory where x is one of the following: NONE (default) ALL CURRENT H,L (or H or L) STROBE H,L (or H or L) THRESHOLD H,L (or H or L) Margins are run in sequence with the memory being cleared at normal margins before using the next margin.
/PATTERN:x	X is a 36-bit value to be used for pattern tests. Typing NONE for X will restore the default patterns.
/BUS:x	Forces testing with a specific DMA20 bus mode, where x is NORMAL (default), 0 (off), 1, 2, or 4. Note that setting a bus mode greater than that selected on any memory will lose, and that setting a lower bus mode is guaranteed only for 1-bus mode at clock rates other than clock-rate 0.
/TYPE:x	Identifies the external memory type. No entry is required to verify proper operation unless it is desired to test an ME-10. The use of this switch may, however, improve fault resolution of some addressing faults.
/SEGMENT:sa,n	Tests only a contiguous block of external memory, where sa is the starting address and n is the number of 16K segments. All the memory which is interleaved must be included if it is desired to perform the correct worst-case test for the entire segment.
/SEGMENT:NONE	Restores the default to testing all of memory.

Table 2 DGMMA Test Summary

Test	Description
1	Memory system configuration validity check.
2	Checks loopback to each memory module for 1s and 0s.
3	Checks loopback to each memory module for a rotating pattern unless a specific pattern has been selected, in which case only that pattern is used.
4	Address-tests MA20/MB20 memory modules. The test is performed by executing the following sequence. a. Write 0s to a memory location. b. Read the location just written. c. Read the same location again (to eliminate data paths from consideration). d. Write all 1s to a different location. e. Test that the original location did not change.

Table 2 DGMMA Test Summary (Cont)

Test	Description
	Solid faults in this test should be generated only by addressing problems. Patterns used correspond to the input combinations for module X-Y drive. A loop is left running in the ACs if a fault is detected. The addresses are left in AC15 and AC16.
5	Address-tests external memory modules. The test is not necessary to verify memory operation, but it serves to improve fault isolation over the address tag test for certain malfunctions. On the KL10-PA, only addresses less than 777777 are tested, so memory switches must be set accordingly. The test is performed only if the module type has been specified. Since this test is performed in 1-bus mode, it is run only at clock rate 0. It executes the following sequence. a. Write 0s to a memory location. b. Read the location just written. c. Read the same location again (to eliminate data paths from consideration) d. Write all 1s to a different location. e. Test that the original location did not change. Solid faults in this test should be generated only by addressing problems. Patterns used correspond to the input combinations for module X -Y drive. A loop is left running in the ACs if a fault is detected. The addresses are left in AC15 and AC16.
6	Checks each memory core for its ability to hold a 1. Read restore of 1s is also tested.
7	Checks each memory core for its ability to hold a 0. Inhibiting of 1s on read restore is also tested.
8	Checks the parity bit for its ability to hold a 0.
9	Pattern-tests the memory.
10	This is the controller level address tag test for internal memories. The only faults detected here should be stack select errors. Setting the reliability switch causes this test to be run three times. The pattern order and address direction are reversed for the second and third passes respectively. For external memory, almost any addressing error could be detected by this test. See subroutine ADRTAG at the end of this table.
11,12	Exercise the memory with the worst-case pattern loaded. Test 11 checks all bit planes except the parity bit. Test 12 checks all bit planes except bit 35. Exercising of the memory is performed using a program in AC Block 0 which read-restores or double complements the memory while changing an address bit at a fast rate. If the reliability switch is set, the worst-case pattern is verified for each fast-rate bit; otherwise, the memory is exercised for all bits and then checked. The reliability switch also enables 1s/0s checkerboard pattern testing for all data bits.

ADRTAG

ADRTAG is a subroutine designed to do an address-is-data test, otherwise known as an address-tag test. The procedure is as follows. Write the addresses of each word into the word and read them all back. If there was any address interference on the writes it will show up on the readback. The addresses are also written in a reverse order to catch address interference problems from high to low memory. Finally, both of the above are repeated with complementary data to make sure that each bit of the memory can hold a 1 and a 0, and also to catch data-related address interference problems.

This address-tag test features patterns in both halves of the data word. This is necessary because the memories are built as two 18-bit modules in parallel. The problem is that the maximum address is 22 bits and two 22-bit numbers will not fit into 36 bits. In this test, the 22-bit address is kept in the right half of the word and a special "1 bit insensitive" pattern is kept in the high-order 14 bits of the word. The nature of this pattern is such that any 1-bit address error will go to a word whose high-order pattern is different. Thus it can be determined whether the address interference occurred in the left, right, or both halves of the word, which gives some indication of where the error is occurring.

To preserve PDP-11 memory space, the AC programs are not given for all nine subtests. Instead, the AC programs are constructed by the PDP-11 in the ACs at run time. A lot of the 11 code is used to this end and is not actually part of the test. Note the capability to change the order of the subtests.

Subtests 1, 4, 6, and 8 should not get any errors except perhaps for erroneous NXMS which indicate faulty address-acknowledge logic.

Subtests 2, 5, 7, and 9 will catch most of the erros in this test. These include dropouts, picked and stuck bits, and address interference. If either the pattern in bits 00-13 or the address in 14-35 is correct, then the error was probably a dropout, picked or stuck bit. If both halves have strange but incorrect data then the chances are that the problem is address interference.

An error in subtest 3 indicates read-restore problems if the data coming back has massive dropouts; otherwise, the problem is probably an intermittent read (sense) error.

Contents of Accumulators for Subtests 1 to 4.

	AC	Subtest 1 Write Forward Address Comparison Pattern	Subtest 2 (see note) Read Forward Address Comparison Pattern	Subtest 4 Write Forward Address Pattern
	00	TRNN,17,,37777	TRNN,17,,37777	TRNN,17,,37777
	01	HRRI,2,,-320,2	HRRI,2,,-320,2	HRRI,2,,320,2
Start	02	HRLOI,16,,-1	HRLOI,16,,-1	HRLZI,16,,0
	03	XOR,16,,17	XOR,16,,17	XOR,16,,17
	04	HRRI,2,,-460,2	HRRI,2,,-460,2	HRRI,2,,460,2
	05	SXCT,,,13	SXCT,,,13	SXCT,,,13
	06	JRST,,,10	CAME,15,,16	JRST,,,10
	07		HALT,,,1	
	10	CAME,17,,14	CAME,17,,14	CAME,17,,14
	11	AOJA,17,,0	ADJA,17,,0	ADJA,17,,0
	12	HALT,,,0	HALT,,,0	HALT,,,0
	13	MOVEM,16,,0,17	MOVE,15,,0,17	MOVEM,16,,0,17
	14	LAST ADR	LAST ADR	LAST ADR
	15	ECHO	ECHO	ECHO
	16	PATTERN	PATTERN	PATTERN
	17	ADDRESS	ADDRESS	ADDRESS

NOTE

Subtest 3 is the same as Subtest 2. This is specifically to check for the possibility of a faulty read-restore operation in the memory.

Contents of Accumulators for Subtests 5 to 7

	AC	Subtest 5 Read Forward Address Pattern	Subtest 6 Write Backward Address Comparison Pattern	Subtest 7 Read Backward Address Comparison Pattern
	00	TRNN,17,,37777	TRNN,17,,37777	TRNN,17,,37777
	01	HRRI,2,,320,2	HRRI,2,,-320,2	HRRI,2,,-320,2
Start	02	HRLZI,16,,0	HRLOI,16,,-1	HRLOI,16,,-1
	03	XOR,16,,17	XOR,16,,17	XOR,16,,17
	04	HRRI,2,,460,2	HRRI,2,,-460,2	HRRI,2,,-460,2
	05	SXCT,,,13	SXCT,,,13	SXCT,,,13
	06	CAME,15,,16	JRST,,,10	CAME,15,,16
	07	HALT,,,1		HALT,,,1
	10	CAME,17,,14	CAME,17,,14	CAME,17,,14
	11	ADJA,17,00	SOJA,17,,0	SOJA,17,,0
	12	HALT,,,0	HALT,,,0	HALT,,,0
	13	MOVE,15,,0,17	MOVEM,16,,0,17	MOVE,15,,0,17
	14	LAST ADR	LAST ADR	LAST ADR
	15	ECHO	ECHO	ECHO
	16	PATTERN	PATTERN	PATTERN
	17	ADDRESS	ADDRESS	ADDRESS

Contents of Accumulators for Subtests 8 and 9

	AC	Subtest 8 Write Backward Address Pattern	Subtest 9 Read Backward ADDRESS Pattern
	00	TRNN,17,,37777	TRNN,17,,37777
	01	HRRI,2,,320,2	HRRI,2,,320,2
Start	02	HRLZI,16,,0	HRLZI,16,,0
	03	XOR,16,,17	XOR,16,,17
	04	HRRI,2,,460,2	HRRI,2,,460,2
	05	SXCT,,,13	SXCT,,,13
	06	JRST,,,10	CAME,15,,16
	07		HALT,,,1
	10	CAME,17,,14	CAME,17,,14
	11	SOJA,17,,0	SOJA,17,,0
	12	HALT,,,0	HALT,,,0
	13	MOVEM,16,,0,17	MOVE,15,,0,17
	14	LAST ADR	LAST ADR
	15	ECHO	ECHO
	16	PATTERN	PATTERN
	17	ADDRESS	ADDRESS

GENERAL INFORMATION

Code	DGQDA.BIN
Title	DECSYSTEM Diagnostic Console Program
Note	DGQDA is the diagnostic code name for KLDCP. Refer to the KLDCP module for further information.

GENERAL INFORMATION

Code	DGQDD.A11
Title	DN87S Loader Utility Program
Abstract	DGQDD works in conjunction with DGQDE, the DL11E monitor program to down-line load the DN87S communications front-end subsystem from the console front-end. These programs and their relationships are described in the summary for DGQDE.
Note	Refer to the DGQDE summary for further information.

GENERAL INFORMATION

Code	DGQDE.A11
Title	DLDP-DL11E Monitor
Abstract	Normally, DFSXA (a 10-based 10 maintenance program) is used to down-line load programs from the KL10 through the DTE20 and into a DN87S communications front-end for execution. Should this method fail, DGQDD and DGQDE provide an alternate method for loading programs into the DN87S for execution. DGQDD is a primary down-line loader that runs under KLDCP in the console front-end. DGQDE is a secondary down-line loader that runs in the DN87S. Before this alternate path can be used, however, the KLINIK DL11-E in the console front-end must be connected to the diagnostic DL11-E in the DN87S. Also a terminal must be connected to the DL11-C in the DN87S. Once this path has been established, it may be used to down-line load and execute 11-based 11 maintenance programs in the DN87S. You may also run 11-based 10 maintenance programs in the DN87S by first down-line loading KLDCP.
Hardware Required	KL10 console front-end subsystem/DN87S communications front-end subsystem with an H312 null modem and a terminal.
Preliminary and Associated Programs	KLDCP, DGQDD, appropriate 11/11 and 11/10 diagnostic programs. This program assumes that the console front-end and both DL11-Es are fully operational.

Restrictions

1. The KLINIK DL11-E in the console front-end must be connected to the diagnostic DL11-E in the DN87S via the H312 null modem.
2. Both the KLINIK DL11-E and the diagnostic DL11-E must be set to 9600 baud rate.
3. A terminal must be connected to the DL11-C in the DN87S.
4. All programs to be run, including DGQDD and DGQDE must be stored on the selected KLDCP load medium.
5. If KLDCP is used to run the 11/10 diagnostic in the DN87S, the DTE20 for the DN87S should be set to privileged mode.
6. The copy of KLDCP running in the console front-end must be version 11 or later.

Notes

1. The DL11-E for the KLINIK link is normally set at 300 baud rate.
2. Remember to return the system to its exact original configuration. Then verify that it is fully operational, including the KLINIK link.
3. Functionally, the commands supported by DGQDE are the same as those supported by the 11/11 XXDP diagnostic monitor. If you need more information than is supplied by this command summary and description, refer to the XXDP summary in the 11/11 Maintenance Library section.

Loading, Starting and Stopping Procedures

Procedure 1 describes how to configure the system so that KLDCP, DGQDD and DGQDE can be used to down-line load and execute programs in the DN87S.

Procedure 2 describes how to return the system to its original configuration upon completion.

Procedure 3 describes how to load and run 11/10 programs in the DN87S.

OPERATIONAL CONTROL

DGQDE is controlled via commands entered on the terminal connected to the DN87S.

Table 1 summarizes DGQDE command conventions and control characters.

Table 2 summarizes the DGQDE command set.

Procedure 1 Preparing the System to Run DGQDD and DGQDE

Step	Procedure
1	Power down both the console and communications front-end subsystems.
2	Set the DL11-E for the KLINIK link in the console front-end to 9600 baud.
3	Check to assure that the DL11-E for the diagnostic link in the DN87S is at 9600 baud.
4	Remove the cable from the KLINIK link modem and plug it into the H312 null modem attached to the DN87S.
5	Connect a terminal to the DL11-C in the DN87S.
6	Power up both front-end subsystems.
7	Load KLDCP into the console front-end.
8	Mount and select (via KLDCP) the storage medium containing DGQDD, DGQDE and the programs to be run on the DN87S.
9	Load and start DGQDD via KLDCP. (e.g., >. P DGQDD$)
10	At the DN87S console, press the UNLOCK and LOAD SELECT 1 switches simultaneously. DGQDE will automatically load into the DN87S and enter command mode. Refer to the DGQDE Command Summary, Table 2.

Procedure 2 Restoring the System After Running DGQDD and DGQDE

Step	Procedure
1	Power down the console and communications front-end subsystems.
2	Disconnect the terminal from the DN87S.
3	Disconnect the KLINIK cable from the H312 null modem and reconnect it to the KLINIK modem.
4	Return the DL11-E for the KLINIK link to 300 baud.
5	Power up both front-end subsystems.
6	Verify that the system (including the KLINIK link) is fully operational.

Procedure 3 Running 11/10 Diagnostics in the DN87S

Step	Description
1	Perform Procedure 1
2	R KLDCP<CR> FE#>. At the DN87S terminal, direct DGQDE to load and start KLDCP. KLDCP responds with its prompt (FE#>.).
3	FE#>.DL<CR> Direct KLDCP to select the DL11-E in the DN87S as the load medium. Now, when the copy of KLDCP that is running in the DN87S front-end is directed to load and run an 11/10 program, it will pass the load request (via the DL11-E) to DGQDD running in the console front-end. DGQDD will then direct the console copy of KLDCP to read the program from its selected load medium and transfer it (via the DL11-E) to the DN87S. The copy of KLDCP running in the DN87S will automatically start the program. From this point on the DN87S front-end subsystem will act as if it were the console front-end (i.e., the DN87S console switches will be used for program control etc.)

Table 1 DGQDE Command Conventions and Control Characters

Convention	Description
<CR>	Carriage return in the standard DGQDE command line terminator.
^C	Control C - Unless DGQDE is executing an R or S command, a control C typed on the DN87S terminal will abort the current operation and return the terminal to DGQDE command mode (monitor state). If an R or S command is in effect (i.e., an 11/11 program is running in the DN87S), then the DN87S must be manually halted and restarted at the DGQDE restart address.
^C	Control C typed on the console terminal will disconnect the console from DGQDE and return the terminal to KLDCP command mode. Control C out of DGQDE first; otherwise, the results may be unpredictable.

Table 2 DGQDE Command Summary

Command	Description	Cross Ref.
C	C CPU34<CR> Execute the chain or command file specified (CPU34).	1
C/QV	C CPU34/QV<CR> Execute the chain or command file specified (CPU34) in quick-verify mode.	1
F	F<CR> Display the DN87S terminal fill count and leave it open for modification.	2
L	L DFKAA<CR> Load but do not start the program specified (DFKAA).	
R	R DFKAA<CR> Load and run the program specified (DFKAA).	3
S	S 204<CR> Start the DN87S at the PDP-11 address specified (204).	3
	S<CR> Restart the DN87S at the last address specified. If a program has been loaded (via the L or R command) since the last address was specified, then start that program at its normal starting address. See the L command.	3

COMMAND DESCRIPTIONS
This section describes in more detail those commands with a cross reference number in Table 2.

1. C CPU34<CR> or C CPU34/QV<CR> - The C command causes the command (chain) file specified to be read into core and executed. The /QV switch will cause each program listed in the command file to be executed only once, regardless of the pass count specified in the command file. See Note 3 under GENERAL INFORMATION.

2. F<CR> - The F (fill) command allows the number of fill characters following a carriage return to be changed. DGQDE will respond to the F command by displaying the current fill value. To leave the value unchanged, type a carriage return. To change the value, type the new value followed by a carriage return. See Note 3 under GENERAL INFORMATION.

3. R DFKAA<CR> or S addr<CR> or S<CR> - These command will cause a program other than DGQDE to run. In all cases, manual intervention at the DN87S is required to stop the program. The DN87S should then be restarted at the restart address specified by the DGQDE header message.

ERROR SUMMARY
The following is a list of standard error messages and their meanings.

INVADR	Invalid address on S command
INVCMD/SW	Unrecognizable command or switch
INVNAM	Invalid format on R, L, or C command
NON-EX FILE	File specified on R, L or C command could not be found on the console front-end load device

The following errors indicate that there is a transmission problem between the console front-end and the DN87S. Try reloading the system. If that does not correct the problem, check the version of KLDCP; it must be 0.11 or later.

?UNK	Unknown response from console front end
CAN'T LOAD	4 successive ASCII line error commands
CKSUM	ASCII line checksum error
EOR?	No end of file or premature end of file
FORMAT	ASCII line format error
LONG LINE	3rd consecutive transmission error was a message with too many characters
MSG CKSUM	3rd consecutive transmission error was a bad message checksum
MSG NBR	3rd consecutive transmission error was a bad message number
NAK	3 consecutive NAKS were received
NO CR	3rd consecutive transmission error had no <CR> in a message
POFLO	Program being loaded overflows into monitor

GENERAL INFORMATION

Code	DGQDF.All
Title	DN2X Front-End Loader Utility
Abstract	DGQDF works in conjunction with DGQEA, the DN2X bootstrap loader program and DGQDG, the DN2X secondary front-end monitor to down-line load the DN2X from the console front-end subsystem. These programs and their relationships are described in the DGQDG summary.
Note	Refer to the DGQDG summary for further information.

GENERAL INFORMATION

Code	DGQDG.A11
Title	DN2X Secondary Front-End Monitor

Abstract

Normally, DFSXA (a 10-based 10 maintenance program) is used to down-line load programs from thee KL10 to a DN2X communications front-end for executions. Should this method fail, three programs (DGQDF, DGQEA and DGQDG) provide an alternate method of down-line loading DN2X front ends.

Basically, this is what happens. You direct the copy of KLDCP that is running in the console front end to load and start DGQDF. DGQDF will ask if you want to boot the DN2X front end. If you answer yes, it will ask which front end you want to boot (1, 2 or 3). It will then tell you to start that front end at a specific address. Note that DGQDF will remain co-resident with KLDCP in the console front end.

When you start the DN2X at the address specified by DGQDF, DGQDF will automatically load the DN2X Bootstrap Loader program (DGQEA) into the DN2X. DGQEA will in turn cause the DN2X Secondary Front-End Monitor (DGQDG) to automatically load into the DN2X and start. When DGQDG starts, the console terminal normally connected to KLDCP will be connected to DGQDG.

Now, via the CTY, you may direct DGQDG to load and run 11-based 11 maintenance programs. If you want to run 11-based 10 programs in the DN2X. First switch the DTE20 for that front end to privileged mode, then direct DGQDG to load and start KLDCP. Note that once you have a copy of KLDCP in the DN2X, you must unload the DN2X to get back to DGQDG.

Hardware Required

KL10 console front end subsystem/DN2X communications front end subsystem/H312 null modem.

Preliminary and Associated Programs

KLDCP, DGQEA, DGQDF and appropriate 11/11 and 11/10 maintenance programs. This program assumes that the console front end and the DL11-E and DL11-W are fully operational.

Restrictions

1. A terminal should not be directly connected to the DN2X subsystem.
2. The DL11-E in the console front end must be connected the DL11-1N in the DN2X via a H312 null modem.
3. All programs to be run, including DGQDF, DGQEA, DGQDG (this program) must be stored on the selected KLDCP load device.
4. The copy of KLDCP running in the console front end must be version 13 or later.
5. If KLDCP is used to run 11/10 diagnostics in the DN2X, the DTE20 for that DN2X should be set to privileged mode.

Notes

Functionally, the commands supported by DGQDG are the same as those supported by the 11/11 XXDP diagnostic monitor. If you need more information than this document supplies, refer to the XXDP summary in the 11/11 Maintenance Library section.

Loading and Starting Procedures

1. Assume that the front ends are properly connected.
2. Set the DTE20 for the DN2X to privileged mode if 11/10 programs are going to be run in the DN2X.
3. Direct KLDCP (console copy) to load and start DGQDF.
4. Make proper responses to DGQDF operator dialogue.
5. DGQDG prompt is a period (.).
6. To run 11/10 programs in the DN2X, direct DGQDG to load and start KLDCP in the DN2X. Type R KLDCP. A copy of KLDCP will load into the DN2X and display the prompt, FE#>.
7. Direct KLDCP to use the front end load device (FE#>.FE<CR>).

OPERATIONAL CONTROL
DGQDG is controlled by commands entered on the terminal connected to the console front end.

Table 1 summarizes DGQDG command conventions and control characters.

Table 2 summarizes the DGQDG command set.

Table 1 DGQDG Command Conventions and Control Characters

Conventions	Description
<CR>	A carriage return is the standard DGQDG command line terminator.
^C	Control C - If DGQDG was used to load a copy of KLDCP into the DN2X, then a control C will return control to that copy of KLDCP.
^C	Control C - If DGQDG is running in the DN2X, then a control C will return control to it unless an R or S command is being processed. If an R or S command is in process, then the DN2X must be manually halted and restarted at the DGQDG restart address.
^\	Control backslash - Regardless of what the DN2X is doing, a control backslash will return control to the copy of KLDCP running in the console front end. Control C the program running in the DN2X first; otherwise the results may be unpredictable.

Table 2 DGQDG Command Summary

Command	Description	Cross Ref.
C	C CPU34<CR> Execute the chain or command file specified (CPU34).	1
C/QV	C CPU34/QV<CR> Execute the chain or command file specified (CPU34) in quick-verify mode.	1
F	F<CR> Display the DN87S terminal fill count and leave it open for modification.	2
L	L DFKAA<CR> Load but do not start the program specified (DFKAA).	
R	R DFKAA<CR> Load and run the program specified (DFKAA).	3
S	S 204<CR> Start the DN2X at the PDP-11 address specified (204).	3
	S<CR> Restart the DN2X at the last address specified. Or, if a program has been loaded (via the L or R command) since the last address was specified, then start that program at its normal starting address. See the L command.	3

COMMAND DESCRIPTIONS

This section describes in more detail those commands with a cross reference number in Table 2.

1. C CPU34<CR> or C CPU34/QV<CR> - The C command causes the command (chain) file specified to be read into core and executed. The /QV switch will cause each program listed in the command file to be executed only once, regardless of the pass count specified in the command file. See Note 6 under GENERAL INFORMATION.

2. F<CR> - The F (fill) command allows the number of fill characters following a carriage return to be changed. DGQDG will respond to the F command by displaying the current fill value. To leave the value unchanged, type a carriage return. To change the value, type the new value followed by a carriage return. See Note 6 under GENERAL INFORMATION.

3. R DFKAA<CR> or S addr<CR> or S<CR> - These commands will cause a program other than DGQDG to run. In all cases, manual intervention at the DN2X is required to stop the program. The DN2X should then be restarted at the restart address specified by the DGQDG header message.

ERROR SUMMARY

The following is a list of standard error messages and their meanings.

INVADR	Invalid address on S command
INVCMD/SW	Unrecognizable command or switch
INVNAM	Invalid format on R, L, or C command
NON-EX FILE	File specified on R, L or C command could not be found on the device

The following errors indicate that there is a transmission problem between the console front end and the DN2X. Try reloading the system. If that does not correct the problem, check the version of KLDCP; it must be 0.13 or later.

?UNK	Unknown response from console front end
CAN'T LOAD	4 successive ASCII line error commands
CKSUM	ASCII line checksum error
EOR?	No end of file or premature end of file
FORMAT	ASCII line format error
LONG LINE	3rd consecutive transmission error was a message with too many characters
MSG CKSUM	3rd consecutive transmission error was a bad message checksum
MSG NBR	3rd consecutive transmission error was a bad message number
NAK	3 consecutive NAKS were received
NO CR	3rd consecutive transmission error had no <CR> in a message
POFLO	Program being loaded overflows into monitor

GENERAL INFORMATION

Code	DGQEA. BIN
Title	DN2X Bootstrap Loader Program
Abstract	DGQEA works in conjunction with DGQDF, the DN2X front-end loader utility and DGQDG, the DN2X secondary front-end monitor to down-line load the DN2X from the console front-end subsystem. These programs and their relationships are described in the DGQDG summary.
Note	Refer to the DGQDG summary for further information.

GENERAL INFORMATION

Code	DGQFB.All
Title	KL10 Memory Configuration
Note	DGQFB is the diagnostic code name for MEMCON. Refer to the MEMCON module for further information.

GENERAL INFORMATION

Code	DHKAA.A11
Title	KL10-PV CPU EBox Diagnostic Part 1
Abstract	This diagnostic is designed to detect and isolate faults in the EBox logic
Hardware Required	KL10-PV mainframe
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKAA TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DHKAA Test Summary

Test	Description
1	(EBUS0): Basic EBus and Diagnostic Function Bus This test attempts to clear the machine and read all 0s, then all 1s across the EBus.
2	(ECLK0): Clock Board Initialization This tests that the diagnostically readable signals on the clock board are properly initialized by a master reset. This test (and those concerned with clock board signals which follow) uses subroutine DCOMP. A table of initially expected data for the clock board appears in the initialization file, EINIT1.P11.
3	(ECRAM0): Command Register Initialization This tests that the microcode data register (command register) is cleared after a reset.
4	(ECTL0): Control Boards Initialization This tests whether the APR, CON, CTL and MCL board logic assumes its expected state following a clear. This test (and the control logic tests which follow) uses a master comparison subroutine, CTLCMP, which in turn uses subroutine DCOMP. A table showing initial expected data and masking may be found in the initialization file, EINIT1.P11.
5	(EFLAG0): Flag Logic Initialization This tests that the flag logic on the SCD board assumes its expected state following a clear. This test (and the flag logic tests which follow) uses a master comparison subroutine, FLGCMP, which in turn uses subroutine DCOMP. A table showing initial expected data and masking may be found in the initialization file, EINIT1.P11.

Table 1 DHKAA Test Summary (Cont)

Test	Description
6	(ESCD0): 10-Bit Arithmetic Initialization This tests that the 10-bit arithmetic on the SCD starts off in its expected state following a clear. Tables of these values may be found in the initialization file, EINIT1.P11.
7	(PIZZA0): PI Board Reset Test This test simply performs a master reset of the KL10 and then issues enough clock which would normally cycle the PI board. Since no requests should be pending, the PI board should not cycle. If the PI board does appear to cycle it could be the PIR0 flip-flop stuck, or the EBUS PI00 line from the DTE20 stuck, or the PI4, PI2, PI1 lines stuck. Either way the problem is isolated to the PI board, DTE20, or translator board.
8	(ECLK1): Basic Clock Control Registers This tests the ability of the clock control registers on CLK5 to load and hold data via the E and diagnostic buses. The test floats a 1 through the various control registers at each possible source/rate selection. Thus the production of clocks at each source/rate combination (as well as the appearance of the correct register bits) is checked.
9	(ECLK2): Single-Step Clock Modes This tests the ability of the clock board to generate single EBox and MBox clocks and conditions EBox clocks at each source/rate combination. Also tested are the EBus and SBus clocks.
10	(ECLK3): Burst Counter This test verifies that the burst counter down-counts to 0 at each source/rate. It does not verify that a corresponding number of clocks are produced, as this would require much of the untested EBox logic be working. This test uses a table and subroutine from ECLK1.
11	(ECRAM1): Diagnostic CRAM Address Register This checks the path from the EBus to the diagnostic CRAM address register and back. It uses a set of eight test patterns, generated by subroutine PATTY, which are sufficient to prove that each bit can independently be 1 and 0.
12	(ECRAM2): CRAM Data Paths This test checks that each of five CRAM locations (0, 1, 2, 3, 1024) can be loaded and read back correctly. Fourteen 11-byte test patterns from the pattern generation subroutine are used with each CRAM address to verify that each RAM chip can store a 1 and a 0 and that no two chips interact. This test also checks that the EBus was correctly received and transmitted on each CRAM (and the CRA) board, the diagnostic read and write CRAM functions work, and the data holding register (command register) works.
13	(ECRAM3): CRAM Addressing This test checks the addressing of all CRAM chips and the ability of each individual cell to hold a 0 or 1. To begin with, the entire CRAM has been set to all 0 data by the initialization routine, ICRAM1. Subtest 1 reads location 0 to see that it is indeed all 0s, and then writes all 1s back into location 0. It then steps to location 1, reads 0s and writes 1s and so on until all locations in the CRAM have been tested. This demonstrates that each cell in the RAM can hold a 0 and that there are no address line faults of the sort where writing a location with 1s causes some higher location to be modified.

Table 1 DHKAA Test Summary (Cont)

Test	Description
	Subtest 2 reads 1s from the top of the CRAM and writes 0s down to the bottom, checking that each cell holds a 1 and that no address faults propagate 0 data downwards. Subtest 3 begins at the top, reading 0s and writing 1s; and subtest 4 starts low and reads 1s and writes 0s (leaving the entire CRAM once again cleared).
	Problems can usually be localized to the chip by noting the bit position of the failing data. Since the test uses KLDCP RAM reads and writes, EC and DC commands from the console can be used to verify/track down detected addressing problems.
14	(ECRAM4): CRAM Parity Network
	This test checks the operation of the CRAM parity network by loading four test microwords. The operation of the logic on the clock board to stop the clock on a CRAM parity error is also checked.
15	(ECLK5): Clock Delay (Microcode T Field).
	This tests the 31, 62, 93 and 520 nanosecond delay logic on the clock board. The T field microcode bits and the CON DELAY REQ signal are also checked. The test issues single MBox clocks and counts the number between EBox clocks. Both too early and too late EBox clocks cause an error.
16	(EDRAM1): IR Register/DRAM Address (I/O, JRST OFF)
	This test checks that the 13 bits of the instruction register can each store 1s and 0s and that no two bits interact. This register is read by reading the DRAM address with 7XX addressing turned off (bits 0-8) and the AC field with AC decoding turned on (bits 9-12). Consequently these features are also checked by this test. Eight patterns generated by subroutine PATTY are used.
17	(EDRAM2): DRAM Address - I/O and JRST Logic
	This test checks the logic which looks for JRST (OP CODE 254), JRST 0, and 7XX instructions and alters the DRAM address accordingly. The effect of JRST instructions on the DRAM J field and the AC field is also tested.
18	(EDRAM3): DRAM Data Paths
	This test checks that each DRAM data cell in a pair of adjacent locations (locations 0 and 1 chosen for convenience) can independently store a 1 and a 0, and that no two cells interact. The test uses 12 5-byte patterns generated by subroutine PATTY.
19	(EDRAM4): DRAM Addressing
	This test checks the address lines to all the DRAM chips by filling the entire RAM with 0s, stepping through the addresses one at a time reading 0s and changing to 1s; and then stepping through the addresses in the reverse order, reading 1s and restoring 0s. This leaves the RAM cleared and completes the verification that each cell is uniquely addressable and capable of storing both a 1 and a 0. Address 254 reads IR bits 9-12 in place of J07-J10 (hardware does also).
20	(EDRAM5): DRAM Parity Network
	This test checks the DRAM parity network with three test patterns. The logic on the clock board to stop the EBox clock on a DRAM parity error is also checked.
21	(ECTL1): DISP Field Decoding and AR, ARX and MQ Control Logic
	This tests the decoding of the DISP field on CTL1 and all of the logic on CTL2 for controlling the AR, ARX and MQ multiplexers.

Table 1 DHKAA Test Summary (Cont)

Test	Description
22	(ECTLA2): ADXCRY Logic This test checks the ADXCRY gates on CTL1 and PC + 1 INH on CON4.
23	(ECON1): COND Field Decoders This tests the decoders on CON1 and various gates using the decoded signals on the control logic boards. The following decoder signals are not verified, as they will be checked with the logic they control: COND/AD FLAGS, COND/PCF-#., COND/FE SHRT, COND/EBOX STATE, COND/EBUS CTL, CON SKIP EN 60-67, and CON SKIP EN 70-77. The four signals COND/024-COND/027 are not tested because they are not used.
24	(ECON2)/EAPR1: CONO APR, PI, PAG and DATAO Logic This test exercises the flip-flops which are controlled by the CON number FUNC 01X decoding of CON COND/DIAG FUNC and the magic # field. This includes the decoders on CON3, the registers controlled by CONO PI and CONO PAG on CON3, the DATAO APR register on APR3 and the APR error interrupt logic on APR1 and APR2.. Note that only the internal control of the error flip-flops is tested here; the response to actual error conditions comes much later.
25	(ECON3): Ucode and Processor State Registers This tests the microcode and processor state registers on the CON board.
26	(EAPR2): EBus CTL, MBox CTL and REG FUNC with # Field Decoding This tests the EBus control register on APR3, the MBox control logic on APR5, and the register function decoding on APR6. All three involve the decoding of a microcode function with the magic # field.
27	(EAPR3): Previous Context and AC Block Registers This test checks the CON LOAD PREV CONTEXT and CON LOAD AC BLOCKS logic on CON3, the previous section register on APR3, and the AC block registers on APR5. The FM block mixer logic is covered in the next test, EAPR4.
28	(EAPR4): Fast Memory Address Mixer and AC + 1, 2 and 3 Logic. This test checks the logic which adds 1, 2 or 3 to the AC number and the fast memory address mixer on APR4. The mixer inputs from ARX 14-17 and VMA 32-35 are tested later after the data paths and VMA have been checked.
29	(EMCL1)/EFLAG1: AD Function Logic and VMA Held Flip-Flops This test uses MEM/AD FUNC and adds bits 0-12 to independently set each of the principal flip-flops on the MCL board. These flip-flops are used, in turn, to test VMA held register and the VMA held/PC flags multiplexer. Also, the PC flags are set and checked using SCD LOAD FLAGS and AR bits 0-12 to provide interference patterns for testing the VMA held/PC flags multiplexer.
30	(EMCL2): Memory Request Address Mode Control Logic This test checks the decoding of the microcode MEM field (MCL1) and the memory request generation logic (MCL1 and MCL5, CON5). The request-type memory (flip-flops clocked by REQ EN on MCL2 and 6), the SXCT/PXCT/VMAX extension logic on MCL4, the DRAM A field decoding on MCL5 and the PREV SEC TO ARMM gate also on MCL5. It uses 24 patterns and depends on many previously tested machine features (for example: CWSX, AC# and AC reference).

Table 1 DHKAA Test Summary (Cont)

Test	Description
31	(EMCL3): VMA Context Storage Logic This tests the flip-flops set with LOAD VMA CONTEXT on MCL2, 3 and 6 and the USER EN and PUBLIC EN logic on MCL2.
32	(EAPR5): FM Block Selection This test checks the FM block, VMA block and XR block mixers and the VMA block register on APR5. It uses some of the PXCT and previous enable logic on the MCL board. It also checks the AC + # logic on APR4 and SH AR extended.
33	(EFLAG2): Processor Flags This checks all of the processor flag logic on SCD4 and 5 not already tested in EMCL1 except some gating involving MBox, signals for the private instruction flag and the arithmetic overflow flags (checked in EFLAG2). The trap mixers.
34	(ECRA01): DISP RAM to Control REG, DISP Enables, DISP Parity This test is designed to check the DISP field of the CRAM to the DISPATCH/SPEC field of the control register. It tests all bits of this section of control register and also tests the DISP FIELD ENABLE gates. The test begins by loading a 5-bit pattern into the DISP field of the CRAM. This pattern (pattern list is in Table DCRA01:) is then clocked to the control register. The control register and DISP enable gates are then read back and verified to be correct.
35	(ECRA01): Dispatch Codes 1, 2, 3 and 6 and also AREAD Logic This test checks DISP field dispatch codes 1, 2, 3, and 6. It assures that when these codes are selected, the appropriate data is multiplexed into the CRAM address. For dispatch code 1, we get DRAM J. For code 2, we get AREAD. For code 3 we get SBR RET, and for code 6 we get CTL NICOND. When checking code 2 and AREAD, the test also runs a selection of patterns through the AREAD network to assure that it is in good condition. The basic test procedure is to load the DISP field with the dispatch code under test. Next the DRAM J field is loaded with a test pattern (if checking code 1) or the complement of the expected data. The diagnostic address register is also loaded with the complement of the expected data. Finally the CRAM address is loaded to be sure the dispatch code is selecting what is expected.
36	(ECRA03): Dispatch Codes 30, 32, 33 and 35. Also NORM Logic This test is designed to check the control RAM address board dispatch codes 30 (MQ), 32 (AR, BR, AD signs), 33 (DRAM B), and 35 (NORM logic). It also tests the NORM logic priority encoder found on the IR/DRAM board. The basic test procedure is to load the dispatch code into the CRAM. If the test is checking dispatch code 33 which requires DRAM data, it will clock the microcode word, just loaded into the CRAM, into the control register, then load the appropriate DRAM data and test the CRAM address to verify that it is correct. If the subtest does not require DRAM data, then it loads the AR with test data and at the same time loads the CRAM dispatch data to the control register. The CRAM address is then tested for correctness.

Table 1 DHKAA Test Summary (Cont)

Test	Description
37	(ECRA04): J-Field and CRA LOC Register Test. Executed at Burst Speed. This test is designed to check the control RAM address board J-FIELD to CR ADR lines, the CR ADR to CRA LOC register lines, and the CRA LOC register itself. All tests take place at burst speed (full speed at the currently selected clock rate using the burst counter). The basic test procedure is to load the J-field test pattern into the J-field of CRAM location 0, and leave all other bits at location 0 at 0 (except for a dispatch code = 10). The CRAM location which would be addressed by the current test pattern is then loaded with all 1s. Next the current CRAM address is set to 0. Finally a burst of clock ticks is given to cause two EBox clocks. This should force the J-field test pattern into the control register (on the first EBox clock), and on the second EBox clock the all 1s RAM word should be addressed and loaded into the control register. Also, the CRA LOC register should be loaded with the J-field test pattern. Now, if the control register is not all 1s, a J-field hardware error has occurred. If the CRA LOC register is wrong, it has a hardware failure.
38	(ECALL1): Microcode Subroutine Stack Forward Sequencing Test This test verifies that the dual shift register with its associated input gating can be reset to a known state, and then sequenced through its 15 decimal preset states, by performing a series of calls (see print CRA4). For each call, the EBox clock is pulsed high and the result checked; then it is pulsed low and the result checked. This assures that the 4 X 2 mixer (on print CRA40) does not mis-select an address.
39	(ECALL2): Microcode Subroutine Stack Return Sequencing Test This test verifies that the same logic can be sequenced back from its last preset address to its state after the first call was performed in test (ECALL1); e.g., from BCDE=1110,DEFG=1000, to BCDE=1110,DEFG=1100 (see print CRA4). The test performs a series of subroutine returns to verify that following a master reset, the previously set-up stack (ECALL1) can be popped one location at a time. The CRA stack address generator is sequenced backwards from the reset state. It is then verified that bringing the EBox clock high will not switch the select input to the 4 X 2 mixer feeding the RAM illustrated on print CRA4. Similarly, it is verified that bringing the EBox clock low will switch the select input to the 4 X 2 mixer.
40	(ECALL3): Microcode Subroutine Stack Output Interference Test In this test a sequence of eight test patterns generated by PATTY are pushed one at a time onto the stack, and each is verified. These patterns are sufficient to verify that each output of the SBR RET buffer (CRA SBR RET 00-10) can be independently a logic 1 and a logic 0, and that no outputs are tied together or otherwise interfering.
41	(ECALL4): Microcode Stack Addressing Test This test is an addressing test performed at burst speed. The test is performed in two consecutive parts. In the first part, each of the 15 valid stack addresses is written into in the corresponding numerical address, one word at a time, but excluding address 0. All other locations are filled with the address complemented. Next, all locations are checked together with their corresponding CRA stack addresses.

Table 1 DHKAA Test Summary (Cont)

Test	Description
	In the second part, each of the valid stack addresses is written with its address complement, while all other addresses are written with the uncomplemented address. Then all locations are checked together with their CRA stack address.
42	(ECALL5): Microcode Subroutine Stack Reliability Test This is a data reliability test which writes and reads 200 octal (random) sets of selected patterns in the stack at burst speed. The CRA stack address for each entry written into the stack is checked also.
43	(EAPR6): Fast-Memory Extended Testing This test checks that the 1 X 128 fast-memory extended RAM can be written with all 1s. It uses SH AR extended and COND FM write, previously tested.

GENERAL INFORMATION

Code	DHKAB.A11
Title	KL10-PV CPU EBox Diagnostic Part 2
Abstract	This diagnostic program is designed to detect and isolate faults in the EBox logic.
Hardware Required	KL10-PV mainframe
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	None
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKAB TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DHKAB Test Summary

Test No.	Description
1	EEDP01 Data Paths Basic Multiplexer Select Test This test verifies that all mixers on the data path board can independently select any of their individual inputs, that no multiplexer select lines are stuck either high or low, and that none of the mixer output lines are stuck low. This test also verifies that each register can hold all 1s and all 0s. The test begins by resetting the EBox and then uses the MQ as a source of 1s (MQ reset state = 1s) and passes these 1s from one register to another. Registers are cleared of their 1s to ensure that if a select line fails, the correct register will contain the wrong thing. The test begins at microinstruction 1, bursts one EBox clock, and tests the registers. Subtest 2 begins at microinstruction 1, bursts two EBox clocks, checks. Subtest 3 begins at 1, bursts 3, and so on, until the complete set of transfers has been executed at burst mode.
2	EEDP02 Data Paths Basic Interference Test This test verifies that the outputs of all mixers and registers on the data board have independent lines; i.e., that the AR has 36 independent lines, that the ADA has 36 independent lines, and so on for all registers and mixers. The test runs in burst mode and first loads the AR with interference data. Next, a burst of EBox clocks is given and the interference pattern is sent on the following journey.

Table 1 DHKAB Test Summary (Cont)

Test	Description
	from the AR to BR BR through ADB through AD to ARX ARX to BRX BRX through ADXB through ADX to AR AR through ADA through AD to MQ MQ to ARX ARX through ADXA through ADX to AR done
3	ESCD1 SC Register (SC from AR and SC Recirculation) This test verifies that each bit of the SC register can store a 1 and a 0 and that all bits except SC00 and SC01 are independent. It also verifies the path from AR bits 18 and 28 to 35 to the SC via the SCM and the recirculation path of SC to SC through the SCM. It uses the standard test patterns of subroutine PATTY and bursted microcode. It precedes the rest of the SC tests because the SC register must work for the remaining data path tests to work.
4	EEDP03 Shifter Board Test - Burst Speed This test runs 85 patterns through the shifter board and verifies that the board has no errors. It runs at burst speed. At single-step speed the BR and BRX, respectively, are loaded with test patterns destined for the AR and ARX. Next, an SC count is loaded into the AR. The burst is started and the microcode does the following. AR (SC COUNT) to SC BR to AR, and BRX to ARX SH (shift board) to the AR, ARX and MQ The MQ, AR and ARX are verified to have the correct data. Expect all failures in this test to be on either shifter board (M8510) or backplane from AR or ARX to shifter board.
5	EEDP04 Data Path Adder ALU and Carry Skip Network Test This test checks and verifies the correct operation of the KL10 72-bit adder, including the associated carry skip logic (found on the IR/DRAM board). The tests are divided into 3 groups. a. Tests where only the ARX and BRX require test patterns and only the ADX is tested b. Tests where only the AR and BR require test patterns and only the AD is tested c. Tests where the AR, BR, ARX, and BRX all require test patterns and both AD and ADX are tested, including carries from ADX to AD There is a separate microcode routine for each of these three groups. After the correct test patterns have been loaded to the AR, BR, ARX, and/or BRX at single-step speed, a burst mode clock is given to actually do the add test at full machine speed.
6	EDP4A -AD=0 Test Using Floating 1s Down the AD This test checks the -AD=0 logic on each of the EDP boards. The test simply floats a 1 down the AD, and does a burst of clocks and a skip on -AD=0, in order to ensure that each AD bit individually is capable of causing the condition -AD=0. Failures during this test should be easily traced to the EDP board which contains the bit under test.

Table 1 DHKAB Test Summary (Cont)

Test	Description
7	EEDP05 Fast Memory Basic Interference Test - Burst Speed This test detects any interference between the 36 data bits coming from the fast memory RAMs located on the data path board. The test first loads the AR with the current interference pattern, then loads that pattern into the currently addressed fast memory location. The output of the fast memory is then checked to see that there is no interference between any of its bits.
8	EEDP06 Fast Memory RAM Test This test detects any fast memory RAM addressing problems, fast memory cells stuck high or low. The test algorithm consists of the following three segments. a. Fill all RAM locations with 0s. b. Begin at block 0, address 0, read all 0s there, then write all 1s there, increment to the next address. It should still be 0s. Now read 0s there, then write 1s, and again increment to the next address. Continue to the last address and block. c. This segment is similar to segment b. Begin at address block 7, address 17. Read 1s there, then write 0s. Decrement to the next address. It should still be 1s. Now read the 1s, write 0s and decrement again. Continue to block 0, address 00. The test begins by loading three 36-bit words into several data path registers at single-step speed. Once the four registers are initialized (BRX-41,,0) (BR=777757,,0) (AR=770000,,0) (MG=770000,,0), the rest of the test is run in the microcode at full machine speed (current console selected clock rate). Next, the program waits a finite amount of time until the microcoded test should have completed, then stops the clock and examines what halt loop the test is in (i.e., the NORMAL TERMINATION halt loop or the ERROR HALT halt loop).
9	EVMA01 VMA and Address Break Registers Data and Interference Test This test verifies that the VMA and address break registers can both hold all 1s and all 0s, and that no register lines interfere with one another. The AR is first loaded with the current test data. Next, the starting address of the microcode for this test is loaded, and finally a burst of EBox clocks is given. The microcode that is executed by this burst should load the test pattern from the AR, through the AD, and into the VMA and ADR BRK registers. Single discrepancies imply bad register bits; multi-discrepancies imply failure of the registers to load properly. Dropped bits could indicate no terminator.
10	EVMA02 VMA and Address Break Match Logic, Bits 18-35 This test verifies correct operation of the VMA/address break match logic on print VMA3, and of the VMA 18-31=0 logic. Both registers are loaded with all 1s and all 0s and verified to match. This checks the lines for stuck highs and lows. Next, the address break register is kept loaded with 0s and a single 1 is floated down the VMA register to verify that the XOR gates operate correctly (also checking 18-31=0).

Table 1 DHKAB Test Summary (Cont)

Test	Description
11	EVMA07 PC and VMA HELD Register Tests - Static This test checks the diagnostic multiplexers that read out the contents of the PC and VMA held registers. The test simply loads interference patterns into the PC, then reads the PC, transfers the pattern to the VMA held and then repeats this for all the interference patterns.
12	EVMA03 PC and VMA Held Register Tests - Burst Speed This test verifies that neither the PC nor VMA held register has any bits stuck at 1 or stuck at 0. It also tests that each register's bits do not interfere with any other bits in that register. The backplane lines are tested between the PC or VMA held 2 mixer and the ADA mixer on the data path board. The microcode for this test begins with the test data in the AR. At burst mode the VMA is loaded then immediately the VMA held register is loaded. The microcode then steers the PC through the ADA into the ARX. Next the VMA held register is steered through the ADA into the AR. Finally the PC is loaded from the VMA. Then the results are tested. Because of the loading sequence, the AR should contain the current test pattern as transferred from the VMA held register. The ARX should contain not the current, but the previous test pattern, which was just transferred from the PC. This way the PC and VMA held registers always contain different patterns, which also enables verification of the PC or VMA held mixer.
13	EVMA04 VMA Register Binary Counter and VMA Adder Tests This test has two phases. The first checks the VMA register binary counter and its ability to increment by 1 and decrement by 1. It ensures that each bit of the VMA register can carry into the next bit (VMA INC) and also can borrow from the next higher order bit (VMA DEC). It also ensures that the carry from each 4-bit chip is connected and operating correctly. It extends an all 1s pattern across the VMA to check that a carry into a next higher bit only carries 1 bit's worth (i.e., that the carry lines internal to the chip are not shorted). The second phase of the test uses patterns out of the PC and the CRAM number field to ensure that the VMA adder ALUs have no fault either internal or external to the chip.
14	EVMA05 VMA AC REF and VMA SECTION 0 This test verifies correct operation of the VMA AC REF logic and VMA SECTION 0 gate. It does this by setting up 10 patterns which drive the VMA AC REF logic. The pattern set up is completely microcoded and is merely driven by the PDP-11 which selects the correct microstarting address and data pattern for the VMA section.

VMA 18-31=0	PAGE UEBR REF	VMA FETCH	VMA RD-WRT	VMA EXTENDED	VMA 13-17	EXPECT VMA AC REF
1	0	0	1	1	0	T ST 1
0	0	0	1	1	0	F ST 2
1	1	0	1	1	0	F ST 3
1	0	0	0	1	0	F ST 4
1	0	0	1	0	37	T ST 5
1	0	1	1	1	37	T ST 6
1	0	0	1	1	20	F ST 7
1	0	0	1	1	10	F ST 8
1	0	0	1	1	4	F ST 9
1	0	0	1	1	2	F ST 10
1	0	0	1	1	1	T ST 11

Table 1 DHKAB Test Summary (Cont)

Test	Description
15	EVMA06 Previous Section REG, ARMM MUX, and VMA in MUX This test checks the interference through and the selectability of the ARMM multiplexer. It also completely tests the previous section register and the selectability of the VMA IN multiplexer. The basic test sequence is as follows. a. Load test pattern to previous section register. b. Previous section register through ARMM to AR. c. Previous section through VMA IN to the VMA. Stop and verify AR and VMA. d. PC through VMA IN to the VMA. This should clear the VMA. Again stop and verify the VMA contents. Continue this loop to float a 1 down the previous section register. Finally, load the previous section with all 1s, then read the PC of all 0s through the ARMM into the AR, to verify the ARMM selectability.
16	EVMA08 Random Signals From VMA LOCAL, LOCAL AC ADDR, and AC REF Five subtests designed to pick up signals and gates left untested by previous tests. Basic test algorithm is as follows. Load the AR with a 36-bit stimulus pattern. Load that 36-bit data to the VMA. Increment the VMA (for the purpose of setting VMA 12 where necessary). Set VMA READ. Set MCL VMA EXTENDED as desired. Read the State of the VMA board. Each 36-bit stimulus datum is followed by a short description of which gate in particular is being tested, and which signal is in a state previously untested.
17	EMCL4 Page Illegal Entry Logic This tests the PAGE ADDRESS COND/PAGE ILL ENTRY logic on MCL3. It uses the VMA and ADR BRK registers on the VMA board, the COMP register on APR3, and the logic used to load them.
18	EFLAG3 Arithmetic Overflow Flags This test checks the carry and overflow flags on SCD4 with the associated gates on the data path boards. Note that it depends on the following functions working: AD/A, AD/A+B, ADB/AR*4, ADB/BR, SC/A, SC/A+B&SCADA&B/#.
19	ETSAT1 Test Satisfied Logic This test checks the test satisfied logic on IR3 and the skip satisfied logic on MCL 4 and 5. It depends on the IR/DRAM working and the AD and AD carry logic working.
20	ECON4 GO/START/RUN, I/O LEGAL, and COND ADR 10 Logic This test checks most of the logic on CON2. It requires some flags to be working, MCL VMA FETCH and VMA AC REF. Note that the diagnostic functions: SET RUN, CLR RUN and CONTINUE (decoded on CON2) are also tested here. The DTE20 status bits for EBOX HALTED and RUN are also checked.
21	ECLK6 Clock Board Page Fail Logic This test forces a page fail using APR SET PAGE FAIL and then checks the sequencing of the logic on CLK4 and CLK3 (and the addressing on the CRA board). It runs at burst speed. Each subtest begins with the logic reset and issues a burst of MBox clocks which is one greater than the previous subtest's burst. Thus the page fail sequence is checked clock tick by clock tick, but is always stepped at machine speed rather than single-step.

Table 1 DHKAB Test Summary (Cont)

Test	Description
22	ECLK7 Simulated MBox Response This test checks the MB wait and MB XFER logic on the CON and CLK boards. The test makes two simulated MBox cycles: one an AC reference and one not.
23	ESCD2 Basic 10-Bit Data Paths This test uses the AR to SC input path tested in ESCD1 to send patterns over the following 10-bit arithmetic data paths: SC from SC via the SCAD (checks first position on SCADB multiplexer. SCAD passes B data on A+B and third position on SCM multiplexer), FE from SCAD, FE recirculation and shift right, SC from FE (second position on SCM multiplexer) and AR (upper bits) from SCAD via ARMM (ARMM positions 3 and 4). It uses standard test patterns and bursted microcode.
24	ESCD3 SCAD and ARMM Multiplexers This test checks the SCADA, SCADB and ARMM mixers. The test uses two bursted microwords to steer AR and # field patterns through the mixers into the AR, SC and FE registers. The first microword is loaded separately by each subtest based on the stimulus table. The second, which is the same for all subtests simply recirculates the register data and is there to switch the mixers to catch any slow-propagating signals. The test assumes that the SCADDER can pass A data (SCAD/A) or B data (SCAD/A+B, SCADA disabled). The adder is checked in test ESCD5.
25	ESCD4 SC .GE. 36 Logic This test checks the SC .GE. 36 gates on SCD2. It uses 1 microword which loads the magic # field through the SCADB, SCAD and SCM into the SC. Nine magic # patterns are used.
26	ESCD5 SCADDER This test checks the 10-bit adder on the SCD board. It loads the SC and FE registers from the magic # field, performs an arithmetic adder function using SC and FE as inputs and stores the result back in the SC and FE. An extra microword is provided which switches the adder multiplexers to cut off any slow-propagating signals. The first three microwords, found in the EBox diagnostic listing at CRAM location START5 = SCM10 are modified according to the stimulus table data. The test runs at burst speed.
27	PIZZA1 PI ON LEVEL Set and Clear, GEN LEVEL Set and Clear, ON, OFF, SYS This test checks the ability of the PI ON LEVEL flip-flops and the PI GEN LEVEL flip-flops to load, hold, and clear. Also the PI ACTIVE flip-flop and the ability of PI SYS CLR to clear the PI system. The basic test sequence is as follows. a. Set all PI ON LEVEL. b. Individually clear each PI ON LEVEL. c. Set all PI GEN LEVEL. d. Individually clear each PI GEN LEVEL. e. Set the PI SYSTEM ACTIVE, clear ACTIVE, set all ON and GEN level flip-flops, and clear all with a PI SYS CLR.

Table 1 DHKAB Test Summary (Cont)

Test	Description
28	PIZZA2 CS Lines, Load/Test Ring Counter, PI EBus REG and EBUS PI This test is divided into two sections. The first section tests the CS lines both for stuck condition and for interference between the CS lines. It does this by loading the IR with interference patterns, dumping the IR onto the CS lines, and then reading the CS lines. The second section of the test checks the EBus request logic, the EBUS PI GRANT flip-flop and partially checks the PI load/test ring counter. It does this by issuing PI GEN requests to the PI board, then bursting the PI clock part way into the request cycle. The PI board is then verified to be in the correct state. See the expected data table for the test patterns used in this test.
29	PIZZA3 PIR Flip-Flops, PIR EN, PI REQ SET, PIH Flip-Flops, PI CLRS, PIR/PIH PRI This test exercises and checks the following PI board logic: the PIR1-PIR7 flip-flops, the PIR/PIH priority encoder pair, PI REQ SET decoder, PI CLR decoder, the PIH1-PIH7 flip-flops, and the PIR EN gates. The basic test procedure is as follows. a. Set PI cycle to hold the load/test ring counter. b. Do a CONO PI to set one or several GENs. c. Drop PI cycle to enable the load/test ring counter to advance. This should load the PIR flip-flops. d. Stop and read the state of the PI board to see if the PIRs set. e. Set PI cycle then do a SPEC/SAVE FLAGS which loads the PIH flip-flops via the PIR EN gates. f. Again read the state of the PI board to check that the PIH flip-flops did load. g. Do a PI DISMISS, then verify that the highest level PIH was set.
30	PIZZA4 PI TIM1-TIM7-PI COMP Ring Counter and Timer Done Counter The following two PI board tests are confusing and difficult tests to understand. The connection between the error symptom printed on the terminal and the actual hardware fault causing the symptom is very difficult to explain even with an excellent understanding of how this part of the PI board is intended to work. This test is intended to check two counters on the PI board. The PI time state counter is a pair of shift registers on print PI2 whose eight outputs are labeled: TIM1, TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, COMP. The timer done counter consists of two binary counters connected serially whose only output is labeled TIMER DONE. These two counters operate independently, but must work together. When a cycle is started the time state counter goes to time state TIM1. It remains static, in TIM1 while the timer done counter begins counting. After timer done counter has counted a specific number of MBox ticks, it sets the flip-flop timer done. This act turns the time state counter on and it advances from TIM1 to TIM2. Again it waits for timer done to count before it can advance.

Table 1 DHKAB Test Summary (Cont)

Test	Description
	This test will operate as follows: It will issue one too few ticks to set TIM1. It will verify that TIM1 does not come up. It will reset the PI board and issue exactly enough ticks to set TIM1, then verify that TIM1 does occur. Next, it resets the PI board and issues one too few ticks to set TIM2. It verifies that TIM2 does not come up, resets, issues exactly enough ticks to set TIM2, verifies that TIM2 comes up, and resets. The same procedure is followed for TIM3, TIM4, TIM5, TIM6, and TIM7.
	Faults in these two counters can be isolated to the PI board, but the logic failure on the board itself is difficult or impossible to call out with software. Only by putting the board on extender can the problem be found.
31	PIZZA5 STATE HOLD Logic, EBUS DEMAND Logic, OK ON HALT Decoder
	This test checks the combinational logic that produces STATE HOLD, the combinational logic that produces EBUS DEMAND, and the EBus data lines and other PI board signals that are used with the decoder that causes PI14 OK ON HALT. The basic test procedure is to set up microcode which would normally take the PI board through all seven time states. The microcode also includes the special conditions under test and sets the appropriate flip-flops needed for the specific subtests (examples include: APR EBUS RETURN, APR EBUS DEMAND, CON EBOX HALT, CON EBUS REL, and APR EBUS REQ). Finally, the PDP-11 controls the number of clocks given to the PI board and stops the PI board in the time state desired, then reads the PI board and verifies that the combinational logic under test is working.
32	PIZZA6 Physical Number Flip-Flops, from EBus 00-15 and Physical Number Priority Encoders
	This test checks the 16 physical number flip-flops on print PI2 and the dual priority encoders which take the physical numbers and produce the signals SEL PHY8, SEL PHY4, SEL PHY2, and SEL PHY1. The test also checks the SEL PHY4X to EBus bit 7, 8, 9, 10 mixer on print PI5.
	The test uses microcode which runs through all seven timing states of the PI board. The PDP-11 stops the clock during PI TIM3 and examines the state of the PI board to ensure that the correct physical numbers have come up. Next, the test is continued through TIM6, with the AR function AR/EBUS. After TIM6, the PDP-11 again stops the clock and examines the PI board and the AR to verify that the correct signals are being put onto the EBus by the PI to EBus mixer.
33	PIZZA7 APR PIA 04,02,01, APR PHY NO. and PIR EN
	This test checks the APR PIA 04,02,01 flip-flops which are set by the CONO APR, checks the APR PIA decoders, and checks the output of the APR PIA decoder to the PI request flip-flops. The test does this by setting all PI levels ON and PI ACTIVE, then setting the 3-bit APR PIA register with 0 (1-7 on successive subtests), cycling the PI board to TIM3, and checking the state of the PI board for correct results. This test also checks the PI2 APR requesting flip-flop.

Table 1 DHKAB Test Summary (Cont)

Test	Description
34	PIZZA8 MTR PIA 04,02,01, MTR PHY NO. and PIR0 This test checks the MTR PIA 04,02,01 flip-flops which are set by the CONO MTR generated on the MTR board. The purpose is to check the MTR PIA decoders and the output of the MTR PIA decoders to the PI request flip-flops. The test also checks the DK20 requesting flip-flop on the PI board. The test sets all PI levels ON and PI ACTIVE. Then it sets the MTR PIA register with (1-7), and cycles the PI board to TIM3, which should then set the DK20 requesting flip-flop. The cycling to TIM3 should also set a PI request. The second phase of the test checks the DTE20's PI REQ 0 line to the PI board. It tests that the DTE20 can cause a PI request on level 0.
35	EPAR1 Parity Test - APR FM-36 RAM Chip Address and Data Test This test is designed to test the APR FM-36 128 X 1 RAM chip, both data and addressing operation. First the test attempts to fill every RAM location with a 0. This pass of the test will find any stuck-at-1 bits. Next, an addressing test is begun. The test reads the 0 in location 0 to ensure it has not changed, writes a 1, then goes to the next address. It repeats the sequence to the last RAM address. The RAM should now be full of 1s. Now the test starts at the last RAM address, verifies that the 1 is there and then writes a 0. It decrements the location, verifies the 1, writes a 0; and does this until the whole RAM has been refilled with 0s. This last test phase may fail because of either addressing problems, or bits stuck at 0.
36	EPAR2 AR and ARX Parity Chain, CON AR and ARX Parity Bit Generator and CLK This test checks the AR/ARX parity chain located on the shifter board, the CON AR 36 logic on the CON board used to generate the odd parity bit, and the parity checking logic on the CLK board which stops the clock or causes a page fail on FM, AR, or ARX bad parity. The test begins by selecting one fast-memory (FM) location and using it throughout the entire test. It attempts to set the FM-36 bit to a 1 or a 0 as the test requires and on each subtest checks that the FM-36 bit has indeed gone to a 1 or 0. If it has not, the failure could be in that RAM chip. Subtests 1-10 load test patterns to check the CON AR 36 logic and the CON ARX 36 logic. In conjunction with the FM-36 bit the subtests also set and clear CON MBOX DATA?, -CON FM DATA, CON FM BIT 36, CON AR LOADED, CON AR FROM MEM, and CON ARX LOADED, all necessary in the CON ARE 36 and CON ARX 36 test patterns. It is possible that CSH PAR BIT A and CSH PAR BIT B could cause these subtests to fail because the EBox has no control over these bits. They are expected to be in their reset state (low). While subtests 1-10 are occurring, the AR and ARX registers are loaded with data that, when combined with CON AR 36 and CON ARX 36, tests the AR and ARX parity chain logic found on the shifter board. Finally, CON AR LOADED and CON ARX LOADED are used with SH AR PAR ODD and SH ARX PAR ODD to test the CLK PAGE FAIL logic.

Table 1 DHKAB Test Summary (Cont)

Test	Description
37	EPAR3 AR (ARX) Parity Page Fail, FM Parity Chain, FM Parity Error Stop This test checks that incorrect AR or ARX parity causes CLK PAGE FAILS and that the right page fail address goes to the control RAM address board. It also verifies that the fast-memory (FM) parity chain works and that bad fast-memory parity causes a CLK ERROR STOP.
38	ECRA06 Microcode Skip Conditions (COND 40-57) and Dispatches This test checks the control RAM address board SKIP conditions and COND FUNCTIONS 40-57. It also picks up several microcode dispatches - the individual inputs to the multiplexers that do the actual microcode dispatching. Each subtest has its own microcode which sets up the desired skip condition or dispatch condition and ends with the skip or dispatch being multiplexed onto the CRAM address line. The actual skip or dispatch is taken, and the CRAM bits which set up the skip or dispatch are left behind in the control RAM while the PDP-11 examines the CRAM location register to verify that the correct dispatch or skip was taken.
39	PIDTE PI Board to DTE20 Interface Test This test checks the basic lines that connect the DTE20 to the EBus and to the EBox PI system. It verifies that PI interrupt levels can be assigned to the DTE20 and that the DTE20 can issue an interrupt to the PI board at that assigned level. It ensures that a CONI DTE reads what it should, that a CONO DTE sets what it should and that the bits a CONO DTE sets show up in the DTE20's status register. It checks that KL10 HALT LOOP, KL10 RUN FLOP, and EBOX CLK ERR STOP are all readable in the DTE20 register DIAG1. It also checks that when the DTE20 issues an interrupt, the correct API function type is sent to the EBox (i.e., correct IOP function type is sent on EBus bits 3-5, correct address space specification is sent on EBus bits 0-2, correct qualifier is sent on bit 6 and the correct physical number is decoded and put onto EBus bits 7-10). It also checks the DTE20 decoding of the CS lines CS00-CS06 and ensures that the privileged DTE20 responds only to its own device code.

GENERAL INFORMATION

Code	DHKBA.All
Title	KL10-PV Basic MBox Diagnostic
Abstract	This diagnostic is designed to detect and isol faults in the KL10-PV MBox logic.
Hardware Required	KL10-PV mainframe/internal channels (optional)/ MCA20 (optional)
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	The individual test descriptions for this diagnostic are the same as those for DGKBA.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKBA TEST SUMMARY
(Refer to DGKBA Test Summary.)

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer the 11/10 STD module.

GENERAL INFORMATION

Code	DHKBB.All
Title	KL10-PV Memory Systems Diagnostic Test
Abstract	This diagnostic program is designed to detect a.._ isolate faults related to the operation of the KL10-PV memory subsystem. It checks all internal and external memory controllers which respond to an SBus diagnostic. If run in a normal fashion, all will be tested. All internal memory is tested for addressing, all 1s and 0s. All external memory is tested.
Hardware Required	KL10-PV mainframe/MA20s/MB20s/DMA20s
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	The individual test descriptions for this diagnostic are the same as those for DGKBB.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKBB TEST SUMMARY
(Refer to DGKBB Test Summary.)

ERROR MESSAGE FORMAT
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

GENERAL INFORMATION

Code	DHKBC.All
Title	KL10 Paging Logic Diagnostic
Abstract	This diagnostic is designed to detect and isolat all faults in the MBox and EBox logic associated uniquely with paging operations. This logic is on the PAG, PMA, and CSH boards in the MBox, and the SCD and MCL boards in the EBox.
Hardware Required	KL10-PV mainframe/at least 16K of KL10 main memory.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	The 16K of KL10 memory must be configured.
Note	The individual test descriptions for this diagnostic are the same as those for DGKBC.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKBC TEST SUMMARY
(Refer to DGKBC Test Summary.)

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

GENERAL INFORMATION

Code	DHKBD.A11
Title	KL10-PV Internal Channel Control Test
Abstract	This diagnostic program is designed to detect and isolate faults in the operation of the KL10-PV internal channel control. It is not a complete test of the internal channel logic. To complete testing, run DGKBE (Internal Channel Loopback Test).
Hardware Required	KL10-PV mainframe/internal channels
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	The individual test descriptions for this diagnostic are the same as those for DGKBD.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKBD TEST SUMMARY
(Refer to DGKBD Test Summary.)

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

~~GENERAL INFORMATION~~

Code	DHKBF.All
Title	MF20 Diagnostic Part 1 of 2
Abstract	This diagnostic, together with Part 2 (DHKBG), performs a basic functional check of the MF20 control logic and MOS RAM array boards. This diagnostic concentrates on the control logic; no array boards are tested. DHKBG tests the remainder of the control logic and the array boards. Together, these diagnostics are used to get MF20s running to the point where the 10/10 memory diagnostics can be used for further fault diagnoses and reliability testing.
Hardware Required	KL10-PV mainframe/MF20 memories.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	The master oscillator option bit must be wired into the APRID.
Notes	1. When an array board is returned for repair, make sure that the error printout showing the serial number is included. 2. KLDCP commands may be executed from this diagnostic by preceding the command with a period(.). 3. No response, undefined signals, or unexpected SBus diagnostic function errors indicate a fault in the diagnostic logic.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	The following standard switches are implemented: 15 ABORT, 12 NOPNT, 10 DING, 9 LOOPER, 8 ERSTOP, 7 PALERS, 5 TXTINH, 2 INHCSH, 1 OPRSEL. NOTE Tests which use the cache will not be run if switch 2 is set (1).

OPERATIONAL CONTROL
In addition to the test selection control provided by DIACON (switch 1 set), DHKBF also supports its own set of operating commands. Refer to Table 1.

DHKBF TEST SUMMARY
The individual tests performed by this program are summarized in Table 2.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DHKBF Command Summary

Command	Description
Altmode	$ Interrupt program execution for one KLDCP command.
/DA	/DA<CR> Print the PC, VMA, previous and current AC block numbers, and the contents of AC blocks 0 through 6. Very useful data to accompany a diagnostic error report.
/DR x #	/DR B 14<CR> or /DR B<CR> Print the contents of the RAM specified by x. # specifies the MF20 controller number in the range of 10 through 17. If no controller number is specified, then the currently selected controller is selected. (See the /TC command.) x = A for address response RAM. x = B for bit substitution RAM. x = F for fixed value logic RAM. x = T for timing RAM. If a refresh is in progress it may interfere with a timing RAM dump.
/ER	/ER<CR> Erase the recorded single-step diagnostic function data. (See the /TR switch.)
/IC #	/IC<CR> or IC 13<CR> Perform the minimum initialization of memory necessary to communicate with the MF20 specified by # (10-17). The address response RAM is set up. Bits 18-21 determine which block is used. /IC<CR> reinitializes the currently selected controller (see the /TC switch).

/MO x	/MO 3<CR>
	Select master oscillator frequencies source.
	
	x = 3 for normal operation 30 MHz.
	x = 2 for slow 25 MHz.
	x = 1 for fast 31 MHz.
	x = 0 for external oscillator. An external oscillator must be connected.

/PD	/PD<CR> Enter diagnostic patch dialogue. The command prints the address and content of the first free patch location. Type a <CR> to leave the content unchanged. Type new data<CR> to change the content. Type <ESC> (escape) to cause the patches to reguest a new address. Type <ESC> to the address inquiry to exit. The address pointer is automatically updated each time a <CR> is typed.
/QV	/QV<CR> Toggle the QV switch. The QV switch executes a set of tests that require manual intervention. These tests are required by manufacturing and are not normally done in the field. (See Test 7.)
/RA	/RA<CR> Perform a START MICROCODE and restore AC block 0 to the contents saved by the last /SA command.
/RI	/RI<CR> Reinitialize diagnostic. Set all software switches and control flags to their initial state.
/SA	/SA<CR> Save the contents of the current AC block. (See the /RA command.)
/SD x	/SD 150000 000000 SBus Diagnostic. Perform a BLKO PI using the 36-bit word specified. The memory's response will be printed. Specify controller number 00 (bits <00:04> to select the default controller. (See the /TC switch.) No printout indicates that the memory failed to respond. Fix it.

Table 1 DHKBF Command Summary (Cont)

Command	Description
/SM	/SM<CR> Automatically voltage margin (SHMO) the memory system. Standby, this requires 81 passes of the diagnostic. Depending on the size of the memory system it may require several hours for complete testing; i.e., both DHKBF part 1 and DHKBG part 2.
/SR	/SR<CR> Perform as SBus reset.
/TC #	/TC<CR> or TC 0<CR> or TC 13<CR> Test only the controller specified by # (range 10 to 17). TC 0<CR> specifies all controllers. /TC<CR> will cause the number of the currently selected controller to be printed. The controller specified will become the default for the /DR, IC, and SD switches.
/TI	/TI<CR> Allow the user to input a test loop in PDP-11 machine language. The command will indicate how many times the test loop can be executed in a 5 second period.
/TN	/TN<CR> Toggle the test-number type-out switch. When on, this switch causes the test numbers to be printed before the tests are executed.
/TR	/TR<CR> Allow the user to define and execute a single-step trace of an MF20 memory cycle. The /TR dialogue will ask all the right questions as it goes along. This command is the best way to isolate a fault in the MF20 random control logic. If the message, OUT OF MEM, is printed erase the single-step record (see the /ER switch) and try again. Reload the diagnostics after repairing the fault.

/VM x	/VM<CR> or /VM 0<CR> or /VM 11<CR>
	Set voltage margins equal to x.
	
	x = 12 sets the 12 volt supply to 12.6 V.
	x = 11 sets the 12 volt supply to 11.40 V.
	x = 5 sets the 5 volt supply to 5.25 V.
	x = 4 sets the 5 volt supply to 4.75 V.
	
	x = -2 sets the -2 volt supply to -2.10 V.
	x = -1 sets the -2 volt supply to -1.90 V.
	x = -5 sets the -5 volt supply to -5.46 V.
	x = -4 sets the -5 volt supply to -4.94 V.
	
	x = 0 clears all voltage margins.
	
	/VM<CR>
	Print the state of all voltage margins.

Table 2 DHKBF Test Summary

Test	Description
0	ONE-TIME INIT COMMON TO DHKBF AND DHKBG A one-time pretest initialization routine.
1	MASTER OSC TEST FIRST, COMMON TO DHKBF AND DHKBG Check the master oscillator bit in the APRID. If the bit is clear (0) the program halts. If the bit is set (1) a START MICROCODE is performed to determine if the master oscillator is running. If the oscillator is not running then the memory subsystem must be powered down and up again to reselect the internal clock.
2	CONTROLLER CONFIGURATION TEST, COMMON TO DHKBF AND DFKBG Determine the controller types and numbers. The results are printed once. Error codes indicate the following. 1 = No controllers responded. 2 = Unrecognized controller type. 3 = A controller is responding to the wrong number. 4 = A mismatch between the master oscillator and the MF20/MX20.

Table 2 DHKBF Test Summary (Cont)

Test	Description
3	PORT LOOPBACK TEST Set the PORT LPBK bit (diagnostic function 4, bit 05) for each controller and check for proper response. Next the PORT LPBK bit is cleared via a memory reset to determine if the memory reset function works properly.
4	SBDIAG EXPECTED 0 BITS FORMAT TEST Check all unused SBus diagnostic function bits. Zeros are output and zeros are expected to return.
5	CORDIS, ECC/SYN REG, BIT FIXER, AND READ PATH 1S TEST Check the remainder of the internal read path logic. 98.6% of the failures detected by this test will be on the SYN module.
6	SBUS DIAG DATA PATH AND INTERFACE TEST Check the majority of the SBus diagnostic data paths that normally do not interfere with each other.
7	VOLTAGE MARGIN CONTROL TEST Check the ability of the controller to send the margining signals to the power supply and the ability of the power supply to margin voltages properly. The backplane LEDs and dc bad bit are also checked. NOTE The /QV switch must be used to envoke this test. It requires manual intervention.
8	CONTROL RAM GALPAT TEST Check for inteference between the bit substitution RAMs, the timing RAMs, the address response RAMs, and the fixed value RAMs.
9	REFRESH INTERVAL AND ADDRESS COUNTERS TEST Check the refresh interval and address counter.
10	MF20 SINGLE STEP TEST Perform a single-step trace of MF20 memory cycles. This routine supports the /TR command.
11	There is no Test 11.
12	ADDRESS SEND/RECEIVE, ADR PAR, AND FCN 0 LOCK TEST This is the first test that performs reads and writes at full speed. It checks that: 1. The CPU can send an address. 2. The SBus can carry it. 3. The MF20 can receive it. 4. The MF20 can check for correct address parity. 5. The MF20 can indicate an address parity error. 6. The CPU can detect the address parity error. 7. Function 0 locks on error and not on no error. 8. Function 0 flags can be cleared after an error. Refer to the Document section of the listing on microfiche for troubleshooting scope loops.
13	ADDRESS RESPONSE/NO RESPONSE (NXM) TEST This is a two-part test. First all blocks in a controller are deselected, then one block at a time is selected and a read operation is performed. Part 2 tests for nonexistent memory. Here all blocks are selected, then one block at a time is deselected and a read operation is performed. A NXM should result.
14	WRP BOARD ECC GENERATOR TEST Verify the design and construction of the ECC network on the M8574 board. This is not a reliability test. That is, it does not test all possible failure modes of the network.
15	FCC COMPLEMENT REGISTER FUNCTION TEST Test the ability to read and write the ECC complement register and test the ability to latch and read the data bits 36-43 mixer.

Table 2 DHKBF Test Summary (Cont)

Test	Description
16	WRITE BAD PARITY DETECT AND CLEAR 0-5 Verify that the MF20s can detect bad data parity on a write, that function 0 error flag can be cleared and that the APR SBus error flag is set after the controller detects the error.
17	CONTROL RAM PARITY ERROR DETECT AND FLAG TEST Verify that each MF20 can detect, report and clear control RAM parity errors. The RAMs tested are the bit substitution RAM (FCN 7), the timing RAM (FCN 11) and the address response RAM (FCN 12).
18	WRITE PATH SPARE-BIT-OUT TEST Check the ability to select and set the spare bit from any position (00-42) on write. All positions for each spare-bit address bit are checked.

GENERAL INFORMATION

Code	DHKBG.A11
Title	MF20 Diagnostic Part 2 of 2
Abstract	This diagnostic, together with Part 1 (DHKBF), performs a basic functional check of the MF20 control logic and MOS RAM array boards. DHKBF concentrates on the control logic; no array boards are tested. This diagnostic tests the remainder of the control logic and the array boards. Together, these diagnostics are used to get MF20s running to the point where the 10/10 memory diagnostics can be used for further fault diagnoses and reliability testing.
Hardware Required	KL10-PV mainframe/MF20 memories.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	The master oscillator option bit must be wired into the APRID.
Notes	1. When an array board is returned for repair, make sure the error printout showing the serial number is included. 2. KLDCP commands may be executed from this diagnostic by preceding the command with a period(.). 3. No response, undefined signals, or unexpected SBus diagnostic function errors indicate a fault in the diagnostic logic.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	The following standard switches are implemented: 15 ABORT, 12 NOPNT, 10 DING, 9 LOOPER, 8 ERSTOP, 7 PALERS, 5 TXTINH, 2 INHCSH, 1 OPRSEL. NOTE Tests which use the cache will not be run if switch 2 is set (1).

OPERATIONAL CONTROL
In addition to the test selection control provided by DIACON (switch 1 set), DHKBG also supports its own set of operating commands. Refer to Table 1.

DHKBG TEST SUMMARY
The individual tests performed by this program are summarized in Table 2.

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

Table 1 DHKBG Command Summary

Command	Description
Altmode	S Interrupt program execution for one KLDCP command.
/CU	/CU<CR> Toggle the cache-use switch. Normally the tests that use the cache are skipped because, at the point in the KLDCP "B" string (command file) where this diagnostic is run the operational status of the cache is unknown. If the system has cache and you know it works use this command to turn it on. (See /QV.)
/DA	/DA<CR> Print the PC, VMA, previous and current AC block numbers, and the contents of AC blocks 0 through 6. Very useful data to accompany a diagnostic error report.
/DR x #	/DR B 14<CR> or /DR B<CR> Print the contents of the RAM specified by x. # specifies the MF20 controller number in the range of 10 through 17. If no controller number is specified, then the currently selected controller is selected. (See the /TC command.) x = A for address response RAM. x = B for bit substitution RAM. x = F for fixed value logic RAM. x = T for timing RAM. If a refresh is in progress it may interfere with a timing RAM dump.
/IC #	/IC<CR> or IC 13<CR> Perform the minimum initialization of memory necessary to communicate with the MF20 specified by # (10-17). The address response RAM is set up. Bits 18-21 determine which block is used. /IC<CR> reinitializes the currently selected controller (see the /TC switch).

/MO x	/MO 3<CR>
	Select master oscillator frequencies source.
	
	x = 3 for normal operation 30 MHz.
	x = 2 for slow 25 MHz.
	x = 1 for fast 31 MHz.
	x = 0 for external oscillator. An external oscillator must be connected.

/PD	/PD<CR> Enter diagnostic patch dialogue. The command prints the address and content of the first free patch location. Type a <CR> to leave the content unchanged. Type new data<CR> to change the content. Type <ESC> (escape) to cause the patches to request a new address. Type <ESC> to the address inquiry to exit. The address pointer is automatically updated each time a <CR> is typed.
/QV	/QV<CR> Toggle the QV switch. The QV switch executes a set of tests that use the cache and require manual intervention. These tests are required by manufacturing and are not normally done in the field. (See Test 7.) (See the /CU command.)
/RA	/RA<CR> Perform a START MICROCODE and restore AC block 0 to the contents saved by the last /SA command.
/RI	/RI<CR> Reinitialize diagnostic. Set all software switches and control flags to their initial state.
/SA	/SA<CR> Save the contents of the current AC block. (See the /RA command.)

Table 1 DHKBG Command Summary (Cont)

Command	Description
/SD x	/SD 150000 000000 SBus Diagnostic. Perform a BLKO PI using the 36-bit word specified. The memory's response will be printed. Specify controller number 00 (bits <00:04> to select the default controller. (See the /TC switch.) No printout indicates that the memory failed to respond. Fix it.
/SM	/SM<CR> Automatically voltage margin (SHMO) the memory system. Standby, this requires 81 passes of the diagnostic. Depending on the size of the memory system it may require several hours for complete testing; i.e., both DHKBF part 1 and DHKBG part 2.
/SR	/SR<CR> Perform as SBus reset.
/TC #	/TC<CR> or TC 0<CR> or TC 13<CR> Test only the controller specified by # (range 10 to 17). TC 0<CR> specifies all controllers. /TC<CR> will cause the number of the currently selected controller to be printed. The controller specified will become the default for the /DR, IC, and SD switches.
/TI	/TI<CR> Allow the user to input a test loop in PDP-11 machine language. The command will indicate how many times the test loop can be executed in a 5 second period.
/TN	/TN<CR> Toggle the test-number type-out switch. When on, this switch causes the test numbers to be printed before the tests are executed.

/VM x	/VM<CR> or /VM 0<CR> or /VM 11<CR>
	Set voltage margins equal to x.
	
	x = 12 sets the 12 volt supply to 12.6 V.
	x = 11 sets the 12 volt supply to 11.40 V.
	x = 5 sets the 5 volt supply to 5.25 V.
	x = 4 sets the 5 volt supply to 4.75 V.
	
	x = -2 sets the -2 volt supply to -2.10 V.
	x = -1 sets the -2 volt supply to -1.90 V.
	x = -5 sets the -5 volt supply to -5.46 V.
	x = -4 sets the -5 volt supply to -4.94 V.
	
	x = 0 clears all volt margins.
	
	/VM<CR>
	Print the state of all voltage margins.

Table 2 DHKBG Test Summary

Test	Description
0	ONE-TIME INIT COMMON TO DHKBF AND DHKBG A one-time pretest initialization routine.
1	MASTER OSC TEST FIRST, COMMON TO DHKBF AND DHKBG Check the master oscillator bit in the APRID. If the bit is clear (0) the program halts. If the bit is set (1) a START MICROCODE is performed to determine if the master oscillator is running. If the oscillator is not running, then the memory subsystem must be powered down and up again to reselect the internal clock.
2	CONTROLLER CONFIGURATION TEST, COMMON TO DHKBF AND DFKBG Determine the controller types and numbers. The results are printed once. Error codes indicate the following. 1 = No controllers responded. 2 = Unrecognized controller type. 3 = A controller is responding to the wrong number. 4 = A mismatch between the master oscillator and the MF20/MX20.

Table 2 DHKBG Test Summary (Cont)

Test	Description
3	STORAGE ARRAY BOARD PROM DATA TEST Verify the PROM data. Four levels of testing are performed: storage module (field), group, controller and system. Field - The year ranges between 0 and 9, the week number is less than 55, PROM bits 0-2 have even parity and byte 3 has odd parity. Group - All four fields either do exist or do not exist, and the PP, NN, and SS bits are the same within the group. Controller - Group 0 must exist. There are 1 or 2 RAM sizes, and there is only 1 timing number per RAM size. System - All serial numbers must be unique.
4	GROUP LOOPBACK GROUP SELECT TEST Test the ability to select and deselect each group in each MF20.
5	GROUP LOOPBACK DATA PATH TEST Check the 44-bit data path to the M8579 storage array boards. This is primarily a control board test. However, it may also detect multiplexer faults.
6	READ ERROR CORRRECTION AND FLAG TEST USING GROUP LOOPBACK Test the ability of the syndrome network to detect, correct and indicate read errors. Four cycles are checked: 1) no error 2) correctable error 3) correctable error ignored and 4) double-bit error.
7	FULL SPEED REFRESH VERIFICATION TEST The refresh was verified in SINGLE-STEP MODE (DHKBF - Test 10). Here it is tested at full speed. This test <u>must</u> be run at clock rate 0 (see the /MO switch).
8	NON-ADDRESS CONTROL SIGNALS GENERATION TEST Verify that the control signals on the control board are reaching the backplane.
9	NON-ADDRESS CONTROL SIGNALS PROPAGATION TEST Verify that the control signals are propagated to the array boards.
10	GROUP, BLOCK, SUBBLOCK UNIQUENESS TEST Verify that each group, block and subblock (SB) is unique unto itself (i.e., that they do not interfere with each other).
11	MOS ADDRESS GENERATION TEST Test the MOS address generation capability of the ADT (M8577) board. It verifies that the row and column address bits do not interfere with each other and that they are not stuck either high or low.
12	MOS ADDRESS PROPAGATION TEST Verify that the MOS address bits propagated to the array boards.
13	DATA PATH TO RAM TEST A 2-part test of the 44-bit data path to the memories. Part 1 checks the 36 real data bits and Part 2 checks the 8-bit ECC (and spare).
14	SYN BOARD SPARE-BIT-IN TEST USING RAM Check the ability of the syndrome board to substitite the spare bit (coming from memory) for another bit as determined by the spare bit number (SBN) in the bit substitution RAM.
15	READ ERROR CORRECTION AND FLAG TEST USING MEMORY Test the ability of the syndrome network to detect, correct and indicate read errors. This test is similar to Test 6; however, the MOS RAMs provide the data path instead of loopback.

Table 2 DHKBG Test Summary (Cont)

Test	Description
16	FULL SPEED READ-PAUSE-WRITE TEST Verify that each controller can perform a read-pause-write with refresh off. Then refresh is turned on and the test is repeated. This verifies that there is no interference between the read-pause-write and the refresh cycles.
17	There is no Test 17
18	PAGE REFILL CYCLE AND DATA TEST This test performs a full speed 4-word read operation without using cache.
19	DOUBLE-BIT ERROR SCAN AND FIX TEST Scan memory for double-bit errors, fix them where possible and then test again. If a block can be used, it is left patched. Otherwise, the appropriate bit is set in the bad block map located in the bit substitution RAM. This test will only fail if it finds less than 128K (total) usable MOS memory.

GENERAL INFORMATION

Code	DHKCA.All
Title	KL10-PV Meter Board (M8538) Diagnostic
Abstract	This diagnostic is designed to detect and isolat all faults related to the operation of the KL10-PV meter board.
Hardware Required	KL10-PV mainframe/meter board/MCA20 (optional).
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	The individual test descriptions for this diagnostic are the same as those for DGKCA.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), and 3 (MODDVC).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHKCA TEST SUMMARY
(Refer to DGKCA Test Summary.)

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

GENERAL INFORMATION

Code	DHMCA.A11
Title	KL10-PV MCA20 Cache Option Diagnostic
Abstract	This diagnostic routine is designed to detect a.. isolate faults relating to the operation of the MCA20 cache option.
Hardware Required	KL10-PV mainframe/MCA20
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	The individual test descriptions for this diagnostic are the same as those for DGMCA.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 11 (NOT USED), 4 (INHPAG), 3 (MODDVC), and 2 (INHCSH).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHMCA TEST SUMMARY
(Refer to DGMCA Test Summary.)

ERROR MESSAGE SUMMARY
This diagnostic uses the standard error message format. Refer t.. the 11/10 STD module.

GENERAL INFORMATION

Code	DHMCB.A11
Title	KL10-PV Cache RAM Banger Diagnostic
Abstract	DHMCB is a diagnostic aimed specifically at th ECL RAMs associated with the KL10 cache subsystem. It has tests for the cache directory, written bits, valid bits, data words, and the use bits. All of these items are stored in ECL RAMs. The tests in this diagnostic force the KL10 to run worst-case patterns through these RAMs, and then, if an error is detected, isolate the board and the RAM chip associated with the error. The diagnostic is meant specifically to find intermittent faults in the cache RAM ICs. If it finds one it will call out the board and the chip. The callout for cache "logic errors" is wrong but the RAM chips called out may lead to the failing control logic. Scope loops to look for intermittent RAM errors are almost useless; however, the diagnostic does do long strings of operations with the cache which could be valuable in tracking down other problems while using a scope.
Hardware Required	KL10-PV mainframe/MCA20
Preliminary and Associated Programs	Refer to diagnostic hierarchy (11/10 STD module).
Restrictions	None
Notes	1. The individual test descriptions for th diagnostic are the same as those for DGMCB. 2. This diagnostic should be run after DHMCA (the cache diagnostic) as it assumes that the cache control logic is working. If a given error is a result of control logic failure, then the board and DIP numbers are meaningless. There are no isolation routines associated with this diagnostic because the test error routines contain the isolation algorithms. 3. Due to the highly interdependent nature of the cache logic, it is impossible to guarantee that an error in a given test is due to the part being tested. Do not assume that because a chip is called out that it must be at fault. There are some wire ORs in existence, and the inputs and outputs of the chips also go through other gates which could fail. In other words, think before you replace anything.
Loading and Starting Procedure	Standard (Refer to the 11/10 STD module.)
Control Switches	Standard (Refer to the 11/10 STD module.) The following switches are not implemented: 14 (RSTART), 13 (TOTALS), 11 (NOT USED), 6 (RELIB), 4 (INHPAG), and 3 (MODDVC).

OPERATIONAL CONTROL
This diagnostic is controlled via DIACON.

DHMCB TEST SUMMARY
(Refer to DGMCB Test Summary.)

ERROR MESSAGE FORMAT
This diagnostic uses the standard error message format. Refer to the 11/10 STD module.

GENERAL INFORMATION

Code	DHQFA.All
Title	TRACON-KL10 Diagnostic Console Signal Tracer
Notes	DHQFA is the diagnostic code name for TRACON. Refer to the TRACON module for further information.