KS10 Maintenance Guide

Company Confidential

Volume II

digital

EK-OKS10-MG-001

KS10 Maintenance Guide

Volume II

digital equipment corporation • marlboro, massachusetts

First Edition, January 1980

Printed in U.S.A.

2/80-14

NOTE:

The following two pages contain two identical copies of an introduction to the KS10 Maintenance Guide (they begin with "To the Reader:").

Move one copy of this introduction into Volume I to replace the existing introduction on the second page of the volume. Discard the old introduction.

In Volume I, please remove the tab called SYSTEM SOFTWARE and all the pages following that tab. Discard this material. It has been revised and is included in Volume II.

Discard this page after you make the changes described.

To the Reader:

CONFIDENTIALITY - This guide contains Company Confidential information and is intended for use by DIGITAL field engineers only. Refer to the Field Service Methods and Procedures Manual for DIGITAL policy pertaining to handling confidential information.

OBJECTIVE - The objective of this guide is to organize and present that information which is most used during KS10 maintenance activities. This document is directed toward qualified KS10-trained technical personnel.

To maintain accuracy and improve this guide in subsequent revisions, we need feedback. Please forward any information or suggestions that would increase the usefulness of the guide to:

LCG Tools Supervisor
MR1-1 S35

RELATED KS10 DOCUMENTS

KS10-Based DECSYSTEM-2020 Technical Manual, EK-0KS10-TM

KS10-Based DECSYSTEM-2020 Installation Manual, EK-0KS10-IN

ORGANIZATION - The guide is divided into two volumes. Volume I contains KS10 hardware information. It is divided by thumb tabs into five subsections as follows.

1. GENERAL INFORMATION consists of miscellaneous maintenance information which cannot be classified and filed in any of the remaining four hardware sections.

2. SWITCHES/JUMPERS contains information pertaining to hardware switch positions and jumper connections.

3. TABLES/MAPS describes the process tables and bit maps associated with the KS10 mainframe and peripheral equipment.

4. CHECKS/ADJUSTMENTS consists of check and adjustment procedures performed during preventive and corrective maintenance.

5. DIAGRAMS/MULS contains power supply layouts and module utilization lists associated with KS10-based systems.

The information in each hardware section is indexed and arranged according to unit and subsystem (i.e., CPU, memory, disk, tape, and I/O).

Volume II contains software information. It is divided by thumb tabs into five subsections as follows.

1. CONSOLE SOFTWARE lists and describes the commands and messages associated with KNS 10, the KS10 console program.

2. TOPS-10 consists of:

   a. A module that describes the purpose and intended use of each program in the TOPS-10 program library

   b. A module that summarizes the command set and error messages associated with the TOPS-10 operating system

   c. A set of modules that summarize the command set and error messages associated with those system library programs most commonly used during maintenance. These modules are arranged in alphanumeric order according to program name.

The operating system and each system library program command set is summarized in a table similar to the one following. The commands are logically arranged in alphabetic order.

Table 2 TOPS-10 Command Summary

Command	Description	Cross Ref.
ALCFIL	R ALCFIL<CR. Allocates space for a new file or reallocates space for an existing file in one contiguous region on the disk.	4
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.	5

The first entry in the table is the base command. To the right of the base command is an example which illustrates the proper command format. Under the example is a brief description of the task performed by the command.

When more than a brief description is required to explain the task performed by the command, a numeric cross reference will appear in the right-hand column. The cross references are arranged in numeric order following the table.

The error message summaries are arranged in alphanumeric order at the end of each module.

3. TOPS-20 presents the same kind of information and uses the same organization described for TOPS-10.

4. MAINTENANCE SOFTWARE consists of:

   a. A standard module (KS10 STD) that contains the program identification scheme, standard program control switches and starting addresses, a set of tables listing the utility programs and diagnostic hierarchies, standard loading and starting procedures, and common error message formats.

   b. A set of modules that summarize the command sets and error messages for each utility program in the library. The utility summaries are printed on blue paper and are arranged in alphabetic order according to name.

   c. A set of modules that summarize each diagnostic program in the library. The diagnostic summaries are also arranged in alphabetic order according to name.

5. NOTES contains blank pages for your use. Notes written on these pages can easily be inserted into the appropriate sections.

To the Reader:

CONFIDENTIALITY - This guide contains Company Confidential information and is intended for use by DIGITAL field engineers only. Refer to the Field Service Methods and Procedures Manual for DIGITAL policy pertaining to handling confidential information.

OBJECTIVE - The objective of this guide is to organize and present that information which is most used during KS10 maintenance activities. This document is directed toward qualified KS10-trained technical personnel.

To maintain accuracy and improve this guide in subsequent revisions, we need feedback. Please forward any information or suggestions that would increase the usefulness of the guide to:

LCG Tools Supervisor
MR1-1 S35

RELATED KS10 DOCUMENTS

KS10-Based DECSYSTEM-2020 Technical Manual, EK-0KS10-TM

KS10-Based DECSYSTEM-2020 Installation Manual, EK-0KS10-IN

ORGANIZATION - The guide is divided into two volumes. Volume I contains KS10 hardware information. It is divided by thumb tabs into five subsections as follows.

1. GENERAL INFORMATION consists of miscellaneous maintenance information which cannot be classified and filed in any of the remaining four hardware sections.

2. SWITCHES/JUMPERS contains information pertaining to hardware switch positions and jumper connections.

3. TABLES/MAPS describes the process tables and bit maps associated with the KS10 mainframe and peripheral equipment.

4. CHECKS/ADJUSTMENTS consists of check and adjustment procedures performed during preventive and corrective maintenance.

5. DIAGRAMS/MULS contains power supply layouts and module utilization lists associated with KS10-based systems.

The information in each hardware section is indexed and arranged according to unit and subsystem (i.e., CPU, memory, disk, tape, and I/O).

Volume II contains software information. It is divided by thumb tabs into five subsections as follows.

1. CONSOLE SOFTWARE lists and describes the commands and messages associated with KNS 10, the KS10 console program.

2. TOPS-10 consists of:

   a. A module that describes the purpose and intended use of each program in the TOPS-10 program library

   b. A module that summarizes the command set and error messages associated with the TOPS-10 operating system

   c. A set of modules that summarize the command set and error messages associated with those system library programs most commonly used during maintenance. These modules are arranged in alphanumeric order according to program name.

The operating system and each system library program command set is summarized in a table similar to the one following. The commands are logically arranged in alphabetic order.

Table 2 TOPS-10 Command Summary

Command	Description	Cross Ref.
ALCFIL	R ALCFIL<CR. Allocates space for a new file or reallocates space for an existing file in one contiguous region on the disk.	4
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.	5

The first entry in the table is the base command. To the right of the base command is an example which illustrates the proper command format. Under the example is a brief description of the task performed by the command.

When more than a brief description is required to explain the task performed by the command, a numeric cross reference will appear in the right-hand column. The cross references are arranged in numeric order following the table.

The error message summaries are arranged in alphanumeric order at the end of each module.

3. TOPS-20 presents the same kind of information and uses the same organization described for TOPS-10.

4. MAINTENANCE SOFTWARE consists of:

   a. A standard module (KS10 STD) that contains the program identification scheme, standard program control switches and starting addresses, a set of tables listing the utility programs and diagnostic hierarchies, standard loading and starting procedures, and common error message formats.

   b. A set of modules that summarize the command sets and error messages for each utility program in the library. The utility summaries are printed on blue paper and are arranged in alphabetic order according to name.

   c. A set of modules that summarize each diagnostic program in the library. The diagnostic summaries are also arranged in alphabetic order according to name.

5. NOTES contains blank pages for your use. Notes written on these pages can easily be inserted into the appropriate sections.

Table of Contents

Summary	Version
KNS 10	N/A

GENERAL INFORMATION

Code: KNS 10

Title: Console Program

Abstract: Console program resides in the 8080 PROMs and supports the KS10-based system. The console operates in either of two modes.

1. User Mode - The CTY is a user terminal and commands are passed to and from the KS10 CPU under control of the console program.

2. Console Mode - Commands are directed to (and executed by) the 8080 console hardware. The console program initialized the CTY-to-console mode at power-up.

Notes: None

Loading and Starting Procedure: KS10 is automatically booted 30 seconds after power-up.

OPERATIONAL CONTROL
The KS10 is directed by the commands listed in Table 2.

Table 1 lists standard console messages.

ERROR MESSAGE SUMMARY
Table 3 lists the standard console error messages.

Table 1 Standard Console Messages

Message	Meaning
BT AUTO	Beginning automatic boot procedure after power-up.
BT SW	Beginning boot procedure as a result of BOOT switch being pressed (LOCK switch in UNLOCK position).
BUS 0-35	Message header for EB command.
CYC	Cycle type for DB command.
ENABLED	Entering CTY mode from user mode. (CTY mode is entered as a result of a control \ in user mode with LOCK switch in UNLOCK position.)
HLTD	Halt in KS10 processor program execution.
KS10>	Command prompt.
OFF	Current state is off. (Response to CE, TE, TP, and KL commands when current state of enable is requested and it is a 0.)
ON	Current state is on. (Response to CE, TE, TP, and KL commands when current state of enable is requested and it is a 1.)
RCVD	Data received from bus. (Indicates bus data received if failure occurred during EB command.)
SENT	Data sent to bus. (Indicates bus data transmitted if failure detected during DB command.)
USR MOD	Entering user mode. (User mode is entered as a result of a control Z or the successful completion of a CO, ST, BT, or MT command.)
>>UBA?	Query for UBA number.
>>UNIT?	Query for unit number.
>>TCU	Query for tape controller unit number.
>>RHBASE?	Query for RH11 base register address.
>>DENS?	Query for tape density.
>>SLV?	Query for tape slave number.

Table 2 Console Commands

Command	Description	Cross Ref.
Special Control Characters		
^\	Enter console mode.	NA
^\\	Used in conjunction with the KL1 command for KLINIK mode.	NA
^U	Rub out current line.	NA
^O	Switch: first one stops CTY output, second one resumes CTY output.	NA
^S	Stop TTY output and 8080 waits for control Q.	NA
^Q	Resumes TTY output.	NA
^C	Stop the 8080.	NA
^Z	Enter user mode.	NA
RUB-OUT	Rub out previous character typed.	NA
Load Commands		
LAxx	LAxx <CR> - Set KS10 memory address xx (0000000-1777777).	NA
LCxx	LCxx <CR> - Set CRAM address xx (0000-3777).	NA
LFxx	LFxx <CR> - Load diagnostic write function xx (0-7). The function specifies a 12-bit group within a CRAM address. **LF CRAM LF CRAM** 0 00-11 4 48-59 1 12-23 5 60-71 2 24-35 6 72-83 3 36-47 7 84-95	NA
LIxx	LIxx <CR> - Set I/O address xx	1
LKxx	LKxx <CR> - Set 8080 memory address xx (PROM address = 00000-17777; RAM address = 20000-21777).	NA
Deposit Commands		
DBxx	DBxx <CR> - Deposit xx (36 bits) onto KS10 bus.	NA
DCxx	DCxx <CR> - Deposit xx (96 bits) into CRAM. Address previously loaded by LC command.	NA
DFxx	DFxx <CR> - Deposit xx (12-bit group) into CRAM address and diagnostic function previously loaded by LC and LF commands.	NA
DIxx	DIxx <CR> - Deposit xx (16, 18 or 36 bits) into an I/O register. Address previously loaded by LI command.	NA
DKxx	DKxx <CR> - Deposit xx (8 bits) into 8080 memory. Address previously loaded by LK command (data cannot be deposited in PROM addresses; only in RAM addresses).	NA
DMxx	DMxx <CR> - Deposit xx (36 bits) into KS10 memory. Address previously loaded by LA command or EM.	NA
DNxx	DNxx <CR> - Deposit xx into next (KS10, 8080, I/O, CRAM) address.	NA

Table 2 Console Commands (Cont)

Command	Description	Cross Ref.
Examine Commands		
EB	EB <CR> - Examine KS10 bus. Prints contents of console registers 100-103 and 300-303.	NA
EC	EC <CR> - Examine contents of CRAM control register.	NA
FCxx	ECxx <CR> - Examine contents of CRAM address xx.	NA
EI	EI <CR> - Examine contents of I/O register. Address previously loaded by LI command.	NA
EIxx	EIxx <CR> - Examine contents of I/O address xx.	NA
EJ	EJ <CR> - Examine current CRAM address, next CRAM address, jump address, and subroutine return address.	NA
EK	EK <CR> - Examine contents of 8080 memory. Address previously loaded by LK command.	
FKxx	EKxx <CR> - Examine contents of 8080 memory address xx.	NA
EM	EM <CR> - Examine contents of KS10 memory. Address previously loaded by LA command.	NA
EMxx	EMxx <CR> - Examine contents of KS10 memory address xx.	
EN	EN <CR> - Examine contents of next (KS10, 8080, I/O, CRAM) address.	
Start/Stop Clock		
CH	CH <CR> - Halt CPU clock.	NA
CP	CP <CR> - Pulse CPU clock.	NA
CPxx	CPxx <CR> - Pulse CPU clock xx times.	NA
CS	CS <CR> - Start CPU clock.	NA
Start/Stop Microcode		
PM	PM <CR> - Pulse microcode. Performs a CP command to execute a microinstruction followed by an EJ command to print current CRAM address, next CRAM address, jump address, and subroutine return address.	NA
SM	SM <CR> - Reset and start microcode at CRAM address 0.	NA
SMxx	SMxx <CR> - Reset and start microcode at CRAM address xx.	NA
TR	TR <CR> - Trace. Repeats PM command until any CTY key is pressed.	NA
TRxx	TRxx <CR> - Trace. Repeats PM command until CRAM address xx is reached or until and CTY key is pressed.	NA
Start/Stop Program		
HA	HA <CR> - Halt KS10 program. Microcode enters halt loop.	NA
CO	CO <CR> - Continue KS10 program execution. Console program enters user mode.	NA
SH	SH <CR> - Shut down command. Deposits nonzero data into KS10 memory location 30 to allow orderly shut down of the monitor.	NA
SI	SI <CR> - Single instruct. Executes next KS10 instruction.	NA
STxx	STxx <CR> - Start KS10 program at address xx. Console program enters user mode.	NA

Table 2 Console Commands (Cont)

Command	Description	Cross Ref.
Select Device		
DS	DS <CR> - Select disk for bootstrap or microcode verification. Console program asks for UBA number (default = 1), RH11 base address (default = 776700), and disk unit number (default = 0).	2
MS	MS <CR> - Select tape for bootstrap or for microcode verification. Console program asks for UBA number (default = 3), RH11 base address (default = 772440), tape controller unit number (default = 0), tape density (default = 1600 bits/in), and slave number (default = 0).	3
Boot Commands		
BC	BC <CR> - Check the KS10 boot path.	NA
BT	BT <CR> - Bootstrap the KS10 from disk. Loads and starts microcode and monitor boot program from drive 0 on UBA 1 (default address) or drive selected by last DS command; starts KS10 at memory address 1000. The BT command is performed automatically 15 seconds after power-up. A control C aborts the automatic boot process.	NA
BT1	Same as BT command except that diagnostic boot program (not monitor boot program) is loaded and started.	
B2	B2 <CR> - Bootcheck 2. This loads in a separate PRE-BOOT which loads in the bootcheck 2.	NA
LB	LB <CR> - Load the monitor boot program from disk selected last. Does not load microcode. Program must be started at 1000.	NA
LB1	Same as LB command except that diagnostic boot program (not monitor boot program) is loaded. Program must be started at 1000.	
MB	MB <CR> - Load the monitor boot program from the tape selected last. Does not load microcode. Program must be started at 1000.	NA
MT	MT <CR> - Bootstrap the KS10 from tape. Loads and starts microcode and monitor boot program from tape unit 0, slave unit 0 on UBA 3 (default address) or drive selected by last MS command; starts KS10 at memory address 1000.	NA
Verify Microcode		
VD	VD <CR> - Verify CRAM against disk. Compares microcode in CRAM with microcode found on disk unit 0 on UBA 1 (default address) or disk selected by last DS command.	NA
VT	VT <CR> - Verify CRAM against tape. Compares microcode in CRAM with microcode found on tape unit 0, slave unit 0 on UBA 3 (default address) or tape selected by last MS command.	NA
Mark/Unmark Microcode		
MKxx	MKxx <CR> - Mark microcode word (set bit 95) at CRAM address xx.	NA
UMxx	UMxx <CR> - Unmark microcode word (clear bit 95) at CRAM address XX.	NA
Master Reset		
MR	MR <CR> - Master reset. Issue bus reset.	NA

Table 2 Console Commands (Cont)

Command	Description	Cross Ref.
Execute Command		
EXxx	EXxx <CR> - Execute the single KS10 systems - level instruction xx.	NA
Enable/Disable		
CExx	CExx <CR> - Enable (xx = 1) or disable (xx = 0) cache.	NA
PExx	PExx <CR> - Enable or disable parity detection as follows. xx 0 Disable all parity detection 4 Enable KS10 bus parity detection 5 Enable DPE/DPM parity detection 6 Enable CRA/CRM parity detection 7 Enable all parity detection	NA
TExx	TExx <CR> - Enable (xx = 1) or disable (xx = 0) CPU interval timer interrupts.	NA
TPxx	TPxx <CR> - Enable (xx = 1) or disable (xx = 0) CPU traps. Following an enable/disable command with a carriage return gives the current value.	NA
SCxx	SCxx <CR> - Enable (xx = 1) or disable (xx = 0) automatic recovery from soft CRAM parity errors.	NA
Read CRAM		
RC	RC <CR> - Read CRAM data. Performs diagnostic read functions 0-17 to read CRAM addresses and contents (of current address) as follows. 0 CRAM bits 00-11 1 Next CRAM address 2 CRAM subroutine return address 3 Current CRAM address 4 CRAM bits 12-23 5 CRAM bits 24-35 (Copy A) 6 CRAM bits 24-35 (Copy B) 7 0s 10 Parity bits A-F 11 KS10 bus bits 24-35 12 CRAM bits 36-47 (Copy A) 13 CRAM bits 36-47 (Copy B) 14 CRAM bits 48-59 15 CRAM bits 60-71 16 CRAM bits 72-3 17 CRAM bits 84-95	NA
Zero Memory		
ZM	ZM <CR> - Zero memory. Deposit 0s into all KS10 memory locations.	NA
Repeat Command		
RP	RP <CR> - Repeat last command, or last command string, until any CTY key is pressed.	NA
RPxx	RPxx <CR> - Repeat last command, or last command string, xx times.	NA
Lamp Test		
LT	LT <CR> - Lamp test. Momentarily lights (1-2 seconds) and turns off (1-2 seconds) STATE, FAULT, and REMOTE indicators. The indicators are then returned to their original state.	NA

Table 2 Console Commands (Cont)

Command	Description	Cross Ref.
Password Command		
PWxx	PWxx <CR> - Set password xx (xx = maximum of 6 alphanumeric characters). Following a PW command with a carriage return clears the password storage area.	NA
KLINIK Command		
KLxx	KLxx <CR> - Enable remote link with access to system to operate in mode 2 but not in mode 3 (xx = 0). Enable remote link with access to system to operate in mode 2 or in mode 3 (xx = 1). Following a KL command with a carriage return gives the current value.	NA
TT	TT <CR> - Force REMOTE DIAGNOSIS line from mode 3 to mode 2.	NA
8080 Register Commands		
LR	LR <CR> - Command to set into the 8080 RAM, the I/O register, to be either deposited or examined.	NA
DR	DR <CR> - Command to deposit a number into the last specified 8080 I/O register.	NA
ER	ER <CR> - Command to examine one of the 8080 internal registers and display the contents of that register.	NA

CROSS REFERENCES

1. LIxx - The address consists of a control number and a register address. If the console attempts to access its own register, no response occurs.

CTL	Register Address	Register
0	100000	Memory status register
1, 3	763000-77	UBA paging RAM
1, 3	763100	UBA status register
1, 3	763101	UBA maintenance register
1, 3	7XXXXX	Unibus device register

2. DS - The default value for the RH11 base address is currently the only value permitted. Also, a carriage return in response to any question retains the current value.

```
>>UBA? 1 <CR>
>>RHBASE?, 776700 <CR>
>>UNIT? 0 <CR>
```

3. MS - The default value for the RH11 base address is currently the only value permitted. Also, a carriage return in response to any question retains the current value.

```
>>UBA? 3 <CR>
>>RHBASE? 772440 <CR>
>>TCU? 0 <CR>
>>DENS? 1600 <CR>
>>SLV? 0 <CR>
```

Table 3 Console Error Messages

Message	Meaning
?A/B	A not equal to B. (A and B copies of a microcode field did not match.)
?BC xx	BC command failed. Bootcheck error messages are of the form: "?BC WWYYYY"
	WW=10 Indicates a failure during the CRAM check portion of bootcheck. YYYY will be failing CRAM address.
	WW=04 Indicates a failure during the memory check. Bootcheck is trying to verify page 1 of MOS memory, and tests address 1000-1777 for ability to hold all ones, all zeroes, and to sequence through that page of memory correctly. YYYY will be the failing memory address.
	WW=00 Indicates a failure during the KS10 bus check. Bootcheck is floating a one, then a zero across the KS10 bus.
	YYYY These are the failing bits in octal. Numbers will range from 0 to 43, which corresponds to decimal 0-35.
?BFO	Buffer overflow. (Too many characters typed; console's 80-character input buffer is full.)
?BN	Bad number. (Character typed is not an octal number.)
?BT xx	BT command failed. 8080 error messages of the type "?BT XXXYYY". These messages can occur anytime the 8080 is trying to access either a disk or tape.
	A message of the form "?BT XXXYYY" should be interpreted as follows:
	XXX=001 Means for disk that an error was encountered while trying to read the home blocks. Just about anything can cause this error, including no disk pack in the drive, wrong unit selected, incorrect RH base specified, wrong UBA selected, or bad disk drive. This message also occurs if both the home block and alternate home blocks can be read, but neither has the home block ID ("HOM" in six bit).
	Means for tape that an error was encountered while trying to read the first page of microcode from the magtape. Anything could be wrong in the CP11 to magtape path, including wrong unit selected, wrong RH base address, wrong UBA, wrong slave, wrong density, bad tape drive, bad TM02, bad magnetic tape, or tape in the wrong format.
	This error can occur on any 8080 command or process that accesses disk or tape.
	XXX-002 Means that a disk error was encountered while trying to read the page of pointers which makes up the 8080 file system. If you get this far, the home blocks may have been read successfully. The problem could be a pack that is not in the format required for 8080 loading, the home blocks are bombed, or bad disk drive or pack.
	This error can occur on any 8080 command or process that accesses disk or tape.
	XXX=003 Means that a disk error was encountered while trying to read a page of the microcode. If you get this far, the problem could be a pack not in 8080 format, or bad disk drive or pack.

Table 3 Console Error Messages (Cont)

Message	Meaning
	This error can occur on PWR FAIL recovery, SCE soft CRAM error recovery, VD, BT, or MT commands.
	XXX-004 Means that the microcode did not successfully start running after a BT, MT, MB, or LB command. This error will always occur when you do an LB, before the system microcode is loaded.
	XXX=010 Means that an error was encountered while trying to read in the pre-boot program. Problems could be the same as 003 above. If accessing the tape, failure could have occurred while doing a skip over the microcode file, or in the reading of the pre-boot program itself. Tape could be in the wrong format. This error can occur on LB, MB, or FORCED RELOAD.
	YYY Indicates the lower 8 bits of the 8080 address of the failing "Channel Command List" operation. Do not waste your time looking in the listing, unless you are positive that the RH11 or disk drive is bad. Instead, type EI on the CTY. It will print out the contents of the RH11 register that has the error bits set. That will give you more information than an 8080 listing. If you do find your way through the 8080 listing, you will do an EI anyway, so do the EI first.
?BUS	Bad KS10 bus. (All bus lines not zero after power-up or reset.)
?CCYC	Command/address cycle failed. (KS10 bus data failure detected during DB command; good and bad data printed.)
?CHK xx	PROM checksum error. (Bad checksum for PROM chip xx where xx = 1, 2, 3, or 4.)
?DCYC	Data cycle failed. (KS10 bus data failure detected during DB command; good and bad data printed.)
?DNC	Did not complete. (HA or SM command did not cause microcode to enter halt loop.)
?DNF	Did not finish. (ST, CO, or EX command did not complete.)
?FRC	Forced reload. (Monitor has requested reload; 8080 halts the KS10, reloads the pre-boot program, and starts in KS10 memory location 1000.)
?IA	Illegal address. (Address typed is out of range.)
?IL	Illegal command. (Command typed is not valid.)
?IL	Incorrect password. (Password entered via KLINIK line does not match password entered at CTY.)
?KA	Keep-alive error. (During timesharing, the monitor failed to update the keep-alive count for a period of approximately 15 seconds.)
?MRE	Memory refresh error. (Incomplete KS10 MOS memory cycle. Error occurs when memory must be refreshed in hung state.)
?NA	Not available. (Console not enabled to receive KLINIK line input.)
?NBR	No bus response. (Console did not receive GRANT after requesting KS10 bus.)
?NDA	No data acknowledge. (Console did not receive DATA CYCLE signal after a data request.)

Table 3 Console Error Messages (Cont)

Message	Meaning
?NR-SCE	Nonrecoverable soft CRAM error. This message is followed by the standard "?PAR ERR" message.
?NXM	Nonexistent memory. (Deposit or examine command referenced nonexistent KS10 MOS memory location.)
?PAR ERR xx	System parity error. (CPU clock stopped due to system parity; xx = contents of the following console status registers in the order indicated below. 100, 303, 103
?PWL	Password length error. (Password longer than six alphanumeric characters.)
?RA	Requires argument. (Command typed requires an argument.)
?RUNNING	Clock running. (Command typed requires CPU clock to be stopped.)
?UI	Unknown interrupt. (Console received interrupt but CTY or KLINIK line has no character.)
%SCE XXXXXX	Soft CRAM error. XXXXXX represents the error address. 8080 is attempting to recover by reloading the CRAM and continuing the instruction that got the parity error.

Table of Contents

SYSLIB-10

TOPS-10

PIP

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.
CREATE	CREATE file.ext<CR> Opens a new file on disk for creation with LINED.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
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.
DUMP	DUMP/S ...<CR> Writes a core image file, analyzes the file written, and provides printed output.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
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.
GRIPE	R GRIPE<CR> Accepts text from a user and records it in a disk file for the operations staff.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
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> Places 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, logged-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.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
SCHED	SCHED<CR> Outputs the schedule bits set by the last SET SCHED command.
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.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
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.
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.

Table 2 TOPS-10 Command Summary (Cont)

Command	Description
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.
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	AR1	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

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
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
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	CL0	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	DEBUG	DELRIB nonskip return
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

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
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
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

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
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
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

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
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
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

Table 3 TOPS-10 STOP CODE Summary (Cont)

Monitor Module	STOPCD Name	STOPCD Type	Comment
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
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

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.

Table 1 PIP Command Conventions
Table 2 PIP Command String Delimiters
Table 3 PIP Acceptable Device Mnemonics
Table 4 File Protection Codes
Table 5 UFD and SFD Protection Codes
Table 6 PIP Control Switch Summary
Table 7 PIP Magtape Switch Summary

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.
=	The equal sign must be used to separate the destination and source portions of a PIP command.

Table 2 PIP Command String Delimiters (Cont)

Delimiter	Use and Description
()	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(swlsw2..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>
```

Table of Contents

SYSLIB-20

TOPS-20

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 into 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.

Table 1 TOPS-20 System Program Library (Cont)

Program	Description
LINK	LINK is the linking loader for the DECSYSTEM-20. OVRLAY is the overlay handler for the DECSYSTEM-20. 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 DELETEDELETE	The RUBOUT or DELETE key will cause the last 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 specified 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 it may be used by giving a RUN command. The program 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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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:
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:

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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	UPDOF0 - 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
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
PHYICE	INF	PHYINI - Failed to assign resident STG
PHYLTF	HLT	PHYSIO - SCHLTM - Unexpected LATOPT failure
PHYNIR	CHK	PHYSIO - Null interrupt routine at operation done
PHYPOE	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:
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS (Cont)

Name	Type	Description
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
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

Table 3 TOPS-20 BUGCHKS and BUGHLTS

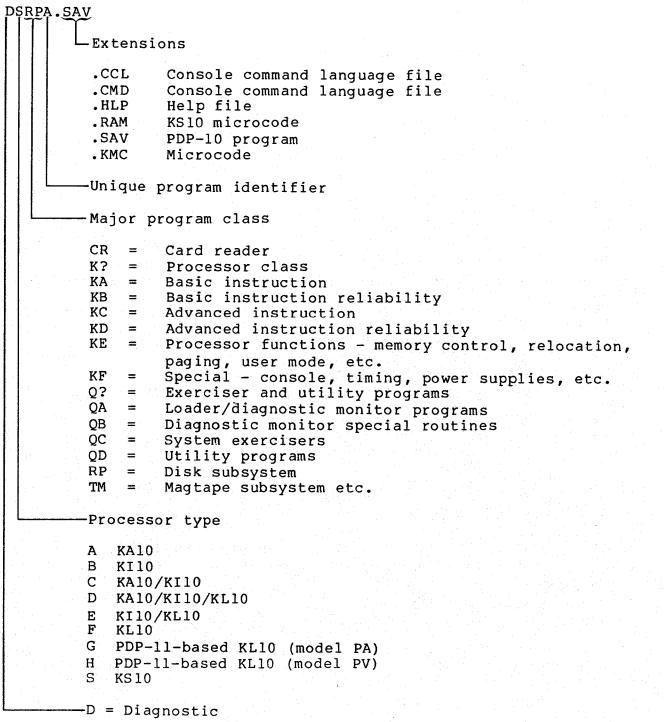
Name	Type	Description
UXXCL3	CHK	Unable to close usage file
UXXCRE	HLT	Cannot create usage file
UXXFAI	CHK	Usage JSYS failure
UXXFIT	INF	Checkpoint file not in correct format for this system, rebuilding.
UXXILL	HLT	USGMES: Illegal function code
UXXMAP	HLT	USGMAP: Call to JFNOFN failed
UXXOPN	HLT	Unable 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

Table of Contents

Summary	Version
KS10 STD	N/A
SMDDT	0.2
SMMAG	0.3
SMMON	0.3
DSKAA	0.1
DSKAB	0.1
DSKAC	0.1
DSKAD	0.1
DSKAE	0.1
DSKAF	0.1
DSKAG	0.1
DSKAH	0.1
DSKAI	0.1
DSKAJ	0.1
DSKAK	0.1
DSKAL	0.1
DSKAM	0.1
DSKBA	0.1
DSKCA	0.1
DSKCB	0.1
DSKCC	0.1
DSKCD	0.1
DSKCE	0.1
DSKCF	0.3
DSKCG	0.2
DSKDA	0.2
DSKEA	0.1
DSKEB	0.1
DSKFA	0.1
DSMMA	0.2
DSRMA	0.2
DSRMB	0.2
DSRPA	0.3
DSTUA	0.2
DSTUB	0.5
DSUBA	0.4

KS10 STANDARD INFORMATION
This module summarizes standard information and procedures which are common to many of the programs in the KS10 Maintenance Library.

PROGRAM IDENTIFICATION CODE
The following figure describes the naming convention used to identify KS10 Maintenance Library programs.



STANDARD PROGRAM STARTING ADDRESSES
Table 1 lists the standard starting addresses for both the diagnostic and utility programs.

Table 1 Standard Program Starting Addresses

Address	Tag	Function
Diagnostic Monitors (SMMON, SMMAG)		
20000	N/A	This is the standard starting address for all KS10 diagnostic monitors.
20001	N/A	This address is used in conjunction with diagnostic monitor control files. Restarting the CPU at 20001 will cause the program currently being run under the monitor to be stopped and the next program in the control file to be started.
20002	N/A	This address is used in conjunction with control switch 15. Restarting the CPU at 20002 will cause the title of the program currently being run under the monitor to be printed. After the title is printed the CPU will halt. The user may restart either the diagnostic monitor or the program at this time.
20003	N/A	This address is used to restart the current program.
Diagnostic Programs		
30000	BEGIN	Stand-alone start
30001	$START	Mode check starting address
30002	DIAGMN	Diagnostic monitor start
30003	SYSEXR	System exerciser start
30004	SFSTRT	Special feature start
30005	PFSTRT	Power fail restart
30006	REENTR	Reenter start
30007	DDTSRT	DDT start
30010	BEGIN1	Start next program pass
30011	SBINIT	PGMINT linkage
30012	RETURN	Return address storage
30013	START1	Optional starting address/instructions
30014	START2	Optional starting address/instructions
30015	START3	Optional starting address/instructions
30016	START4	Optional starting address/instructions
30017	START5	Optional starting address/instructions

STANDARD CONSOLE CONTROL SWITCHES

Table 2 lists and describes the standard functions of the console data switches used to control the operation of most KS10 diagnostics. Exceptions to the standard and right half switches 18 through 35 are described in the individual program summaries.

Table 2 KS10 Standard Control Switch Summary

Switch	Mnemonic	Function
0 (400000)	ABORT	Abort at end of pass
1 (200000)	RSTART	List totals and restart
2 (100000)	TOTALS	List totals and continue
3 (040000)	NOPNT	Inhibit all printing except forced
4 (020000)	PNTLPT	Print on line printer (user, logical DEV)
5 (010000)	DING	Ring terminal bell on error (forced output)
6 (004000)	LOOPER	Enter scope loop on test error
7 (002000)	ERSTOP	Halt on error after reporting error (exec mode), resume normal sequence by pressing CONTINUE. In user mode, this switch causes a CALL AC, EXIT to be executed. Normal test sequence may be resumed by typing .CONT.
8 (001000)	PALERS	Print all errors
9 (000400)	RELIAB	Reliability mode

Table 2 KS10 Standard Control Switch Summary (Cont)

Switch	Mnemonic	Function
10 (000200)	TXTINH	Inhibit comment portion of error messages.
11 (000100)	INHPAG	KS10 - inhibit paging
12 (000040)	MODDVC	Modify device codes
13 (000020)	INHCSH	KS10 - inhibit cache
14 (000010)	OPRSEL	Operator test selections
15 (000004)	CHAIN	This switch used by SMMON, etc., to control chain operations
16 (000002)		Reserved
17 (000001)		Reserved
18 through 35		Refer to the individual diagnostic summaries.

Setting the Console Data Switches

Exec Mode
The switches used to control a single program run on the KS10-based system are derived directly from the console data switches.

The right-half program control switches are redefined when a diagnostic monitor and control file are used to automatically load and sequence a series of diagnostic programs. Refer to the appropriate diagnostic monitor summary module for further information.

User Mode
A diagnostic monitor must be used to run diagnostic programs in user mode. If a single program is run, the program will request that the left and right half switches be entered as octal digits. If less than six digits are typed for either half, the digits typed will be right-justified and zero-filled.

If a diagnostic monitor and control file are used to automatically load and sequence a series of programs, the monitor will request that the left half switches be entered as octal digits. If less than six digits are typed they will be right-justified and zero-filled. The left half digits typed will be shared by all programs in the control file. The right half switches for each program will be supplied by the control file.

DIAGNOSTIC PROGRAM HIERARCHIES
The following tables describe the KS10 Maintenance Library diagnostic hierarchies.

Table 3 KS10 Maintenance Library Utility Programs

Table 4 KS10 Processor Diagnostic Hierarchy

Table 5 KS10 Memory Diagnostic Hierarchy

Table 6 Disk Subsystems Diagnostic Hierarchy

Table 7 Magtape Subsystems Diagnostic Hierarchy

Table 8 Hardcopy Equipment Diagnostics

Table 9 Communications Equipment Diagnostics

Table 10 Unibus Adapter Diagnostic

Table 11 Miscellaneous Diagnostic

MODE:

E indicates that the program may be run in Executive mode.

U indicates that the program may be run in the User mode.

Table 3 KS10 Maintenance Library Utility Programs

Utility	Mode	Title
SMAPT.SAV	E	Special Program
SMBC2.SAV	E U	KS10 Boot Check 2
SMDDT.SAV	E U	KS10 DDT
SMFILE.EXE	U	Special Program
SMMAG.SAV	E U	KS10 Magtape Monitor
SMMON.SAV	E U	KS10 Diagnostic Monitor Standard Control Files SMCPU Processor Diagnostics Run File SMFLT Processor Diagnostics Run File (FLT) SMUSR Processor Diagnostics Run File (TS)
SMTAPE.SAV		Magtape Creator
SUBSM.SAV	E	KS10 Executive Subroutine Program
SUBUSR.SAV	U	KS10 User Subroutine Program

Table 4 KS10 Processor Diagnostic Hierarchy

Diagnostic	Mode	Title
Basic, Advanced, and Reliability		
DSKAA.SAV	E U	Basic Instruction Diagnostic 1
DSKAB.SAV	E U	Basic Instruction Diagnostic 2
DSKAC.SAV	E U	Basic Instruction Diagnostic 3
DSKAD.SAV	E U	Basic Instruction Diagnostic 4
DSKAE.SAV	E U	Basic Instruction Diagnostic 5
DSKAF.SAV	E U	Basic Instruction Diagnostic 6
DSKAG.SAV	E U	Basic Instruction Diagnostic 7
DSKAH.SAV	E	Basic Instruction Diagnostic 8
DSKAI.SAV	E U	Basic Instruction Diagnostic 9
DSKAJ.SAV	E U	Basic Instruction Diagnostic 10
DSKAK.SAV	E U	Basic Instruction Diagnostic 11
DSKAL.SAV	E U	Basic Instruction Diagnostic 12
DSKAM.SAV	E U	Basic Instruction Diagnostic 13
DSKBA.SAV	E U	Basic Instruction Reliability 1
DSKCA.SAV	E U	Advanced Instruction Diagnostic 1
DSKCB.SAV	E U	Advanced Instruction Diagnostic 2
DSKCC.SAV	E U	Advanced Instruction Diagnostic 3
DSKCD.SAV	E U	Advanced Instruction Diagnostic 4
DSKCE.SAV	E U	Advanced Instruction Diagnostic 5
DSKCF.SAV	E U	Advanced Instruction Diagnostic 6
DSKCG.SAV	E U	Advanced Instruction Diagnostic 7
DSKDA.SAV	E U	Arithmetic Reliability

Table 4 KS10 Processor Diagnostic Hierarchy (Cont)

Diagnostic	Mode	Title
Paging And Cache Tests		
DSKEA.SAV	E	Paging Diagnostic
DSKEB.SAV	E	Cache Diagnostic
Supplementary Tests		
DSKFA.SAV	E	Instruction Timing Diagnostic

Table 5 KS10 Memory Diagnostic Hierarchy

Diagnostic	Mode	Title
DSMMA.SAV	E	KS10 1024K Memory Diagnostic
DSMMD.SAV	E U	DECSYSTEM-2020 Memory Diagnostic
Supplementary Tests		
DSMMB.SAV	E U	BLT/FLT 1s and 0s Memory Diagnostic
DSMMC.SAV	E U	FAST AC Diagnostic

Table 6 Disk Subsystems Diagnostic Hierarchy

Diagnostic	Mode	Title
DSRMA.SAV	E	RM03 Basic Diagnostic and Alignment Check
DSRMB.SAV	E U	RM03/RP06 Reliability
DSRPA.SAV	E	RP06 Basic Diagnostic

Table 7 Magtape Subsystems Diagnostic Hierarchy

Diagnostic	Mode	Title
DSTUA.SAV	E	RH11 TM03/TU45 Basic Diagnostic
DSTUB.SAV	E U	Magtape Reliability

Table 8 Hardcopy Equipment Diagnostics

Diagnostic	Mode	Title
DSCDA.SAV	E U	Card Reader Diagnostic
DSLPA.SAV	E U	Line Printer Diagnostic
DSLTA.SAV	E U	Teletype Diagnostic

Table 9 Communications Equipment Diagnostics

Diagnostic	Mode	Title
DSDUA.SAV	E	DUP11 Diagnostic
DSDZA.SAV	E	DZ11 Diagnostic
DSKMA.SAV	E	KMC11 Diagnostic

Table 10 Unibus Adapter Diagnostic

Diagnostic	Mode	Title
DSUBA.SAV	E	Unibus Adapter Diagnostic

Table 11 Miscellaneous Diagnostic

Diagnostic	Mode	Title
DSRHH.SAV	E	RH11 TB Diagnostic (manufacturing)

LOADING PROCEDURES

LOADING FROM DISK PACK - Refer to Procedure 1 and Procedure 2. A boot load readin of the KS10 Maintenance Library (RED PACK) can be performed from the RH11 control unit. Procedure 1 describes a load from drive 0 and Procedure 2 describes a load from other than drive 0.

Procedure 1 Loading the KS10 Maintenance Library from Disk

Step	Command	Procedure
1		Mount the disk pack containing the KS10 Maintenance Library (RED PACK) on drive 0. Write-protect the disk.
2		Make the disk drive READY and ON-LINE.
3	BT1<CR>	Type the command BT1 and carriage return <CR>. SMMON, the diagnostic monitor for disk, will automatically read in and start. Refer to the SMMON command summary.

Procedure 2 Loading the KS10 Maintenance Library from Disk Other Than Drive 0

Step	Command	Procedure
1		Mount the disk pack containing the KS10 Maintenance Library (RED PACK) on the selected drive. Write-protect the disk.
2		Make the disk drive READY and ON-LINE.
3	DS<CR>	Type the command DS and carriage return <CR>. At the completion of the DS command the following dialog will take place. Query Response >>UBA? <CR> >>RHBASE? <CR> >>UNIT? Enter the unit number.
4	BT1<CR>	Type the BT1 command to boot the system.

LOADING FROM MAGTAPE - Refer to Procedure 3 and Procedure 4. A boot load readin of the KS10 Maintenance Library (RED TAPE #3) can be performed from the TM02/03 and RH11 tape control unit. Procedure 3 describes a load from drive 0 and procedure 4 describes a load from other than drive 0.

Procedure 3 Loading the KS10 Maintenance Library from Magtape

Step	Command	Procedure
1		Mount the KS10 Maintenance Library (RED TAPE #3) magtape on transport 0 of the subsystem to be used. Write-protect the tape.
2		Make the transport READY and ON-LINE.
3	MT<CR>	Type the command MT and carriage return <CR>. SMMAG, the diagnostic monitor for magtape, will automatically read in and start. Refer to the SMMAG command summary.

Procedure 4 Loading the KS10 Maintenance Library from Tape Other Than Drive 0

Step	Command	Procedure
1		Mount the KS10 Maintenance Library (RED TAPE #3) magtape on the selected transport to be used. Ensure the tape is write-protected.
2		Make the transport READY and ON-LINE.

Procedure 4 Loading the KS10 Maintenance Library from Tape Other Than Drive 0 (Cont)

Step	Command	Procedure
3	MS<CR>	Type the command MS and carriage return <CR>. At the completion of the MS command the following dialog will take place. Query Response >>UBA? <CR> >>RHBASE? <CR> >>TCU? 0 >>DENS? 1600 >>SLU? Enter the slave number.
4	MT<CR>	Type the MT command to boot the system.

STANDARD ERROR FORMATS

Most KS10 diagnostic programs use one of the following standard error message formats. Unique error message formats are described in the individual diagnostic summary.

ERROR REPORTING

Fatal Halts

The following fatal halt addresses are used to report total inoperation of the KS10 processor. If any of these halts occurs, run a more basic diagnostic that does not use the SUBSM package (DSKAA through DSKAH).

Address	Tag	Reason
2010	NOEXEC	Program not coded for exec mode operation
2011	PLERR	Fatal push list pointer error
2012	PLERR1	Initial push list pointer incorrect
2013	MUOERR	MUUO with LUUO handler wiped out
2014	SMBER	Interrupt without doorbell
2015	SMCER	Clock interrupt without flag set
2016	CPIERR	CPU initialization error
2017	EOPERR	End of program error
2020	LUOERR	Interrupt with LUUO handler wiped out

The following message is typed and a halt or exit is executed.

```
FATAL PROGRAM ERROR AT ######
```

points to the PC of the error handler. Program execution should not be continued until the problem has been corrected.

General Error Message Format

The general error message format specifies up to seven items concerning a test failure.

They are as follows:

1. The PC
2. The present switch settings.
3. The name of the major test sequence being executed.
4. The correct test data results.
5. The actual test data results.
6. The discrepancy.
7. Diagnostic comment concerning the test failure.

A typical printout using the general error message format follows.

```
PC = XXXXXX
SWITCHES = 000000 000000
ERROR IN TITLE-FUNCTION
CORRECT:    XXXXXX XXXXXX
ACTUAL:     YYYYYY YYYYYY
DISCREP:    ZZZZZZ ZZZZZZ
DIAGNOSTIC COMMENT
ADDITIONAL COMMENT
```

PC is the absolute address of the error call instruction.

DISCREP is the octal discrepancy between the CORRECT and ACTUAL test data. (DISCREP is the XOR of CORRECT and ACTUAL).

TITLE, FUNCTION, and DIAGNOSTIC COMMENT portions of the error typeout may be inhibited by setting the TXTINH switch (10). This allows for shorter printouts on repetitive failures.

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 (SMDDT) is always loaded with the KS10 diagnostic routines. Many of these diagnostics use SMDDT for command interpretation and test dispatching (e.g., a diagnostic that uses an $G following a test identification (FRTEST$G) is actually using a SMDDT feature to dispatch to the starting address of the test). SMDDT supports many commands that are useful for controlling diagnostics during maintenance.

DDT<CR> SMMON or SMMAG command to start SMDDT

<CR><LF> PROMPT - SMDDT uses a carriage return followed by a line feed to indicate it is ready to accept a command.

$G Exit SMDDT - 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

SMDDT 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 SMDDT commands.

Table 3 summarizes SMDDT error messages.

Table 1 SMDDT 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 SMDDT 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

Table 2 SMDDT Command Summary (Cont)

Command	Description
Radix for output only)	
$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.
$6T	Set printout mode to sixbit text.
Searching	
a<b>c$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 SMDDT. 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.
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.

Table 2 SMDDT Command Summary (Cont)

Command	Description
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 execute 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.
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

Table 2 SMDDT Command Summary (Cont)

Command	Description
$<	A patch is made by opening an address, typing (altmode) (left 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) (right 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.
Clear Memory	
adr<adr$$Z	PATCH<PATCH+20$$Z Clear memory from address to address.

Table 3 SMDDT Error Messages

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

GENERAL INFORMATION

Code

DSQDD.SAV

Title

SMMAG - DECSYSTEM-2020 Diagnostic Magtape Monitor

Abstract

SMMAG is a variation of SMMON that has been modified to handle magtape as a load medium in both exec and user mode. In exec mode the load medium is restricted to magtape (only). In user mode SMMAG will run with either the TOPS-10 or TOPS-20 operating system and either magtape or disk may be used as the load medium.

SMMAG is command-controlled and can be directed to load and run a single program or execute a control file that will run a sequence of programs. Control files enable SMMAG to be used for the following purposes.

- Rapid checkout of the hardware
- Acceptance testing
- Reliability testing
- Unattended overnight testing.

Hardware Required

KS10 mainframe
Minimum of 32K of memory
Load device: RH11 with TU45 tape drive
Disk subsystem (optional)

NOTE

When under a TOPS-20 monitor, the magtape device must be assigned and density set by a monitor command.

e.g. **@ASSIGN MTA0**
@SET TAPE DENSITY 800 BPI

Preliminary and Associated Programs

SMMAG assumes that the basic instructions and the selected load device are operational.

Restrictions

The diagnostic monitor may be used to call only those programs that follow the prescribed diagnostic formats.

Note

The DECSYSTEM-2020 SUBSM package and SMDDT are automatically loaded on system startup.

Loading and Starting Procedure

Standard (Refer to the KS10 STD module.)
To load SMMAG from the diagnostic magtape, use either the MAGTAP program or the following monitor commands.

TOPS-10

```
.ASSIGN MTA0:
.REWIND MTA0:
.SET DENSITY MTA0: 800(1600) BPI
.SET BLOCKSIZE MTA0: 512
.SKIP MTA0: 3 FILES
.COPY SMMAG.SAV=MTA0:
```

TOPS-20

```
@ASSIGN MTA0:
@REWIND MTA0:
@SET TAPE DENSITY 800(1600) BPI
@SKIP MTA0: 3 FILES
@COPY (FROM) MTA0: (TO) SMMAG.SAV
```

Control Switches

The state of the control switches does not affect the operation of SMMAG unless a control file is being used. A control file lists, as part of each command line, the program to be run and the right half switches to use with that program. This allows the actual (console) right half switches to be used to control the operation of SMMAG. The switches that affect the operation of SMMAG when a control file is in use are listed in Table 1.

OPERATIONAL CONTROL
If SMMAG is loaded from tape the following message will be printed.

SMMAG [DSQDE] DECSYSTEM 2020 DIAGNOSTIC MAGTAPE MONITOR VER 0.2

SMMAG CMD -

If SMMAG is loaded from disk the following message will be printed.

SMMAG [DSQDE] - DECSYSTEM 2020 DIAGNOSTIC MAGTAPE MONITOR - VER 0.2

UBA # -
DRIVE & SLAVE ## -
1600 BPI ? -

NOTE
Drive and Slave are defined by a two digit number. The first digit specifies drive no. of TM02/TM03; the second digit specifies slave no. of transport.

After selection of the load device SMMAG will automatically load SUBSM and SMDDT and print the following.

SMMAG CMD -

Table 2 describes the exec mode device selection commands.
Table 3 describes general SMMAG commands.
Table 4 describes program starting commands.
Table 5 lists SMMAG manual starting addresses.

Table 1 SMMAG Control Switch Summary

Switch	State	Description
9	0	Reduces the iteration count in a control file by a factor of 100 to 1, thus reducing the run time for each program in the file. This is useful for a quick check of the hardware.
	1	Each program listed in a control file is run the specified number of iterations.
15	0	Normal operation.
	1	Inhibit the printing of the test title of each program executed by SMMAG.
18	0	Normal operation.
	1	XPAND - Expand the control sequencing. Refer to the X command, Table 3.

Table 2 Device Selection

Device	Description
0	UBA 0, RH ADR 772440
1	UBA 1, RH ADR 772440
2	UBA 2, RH ADR 772440
3	UBA 3, RH ADR 772440

Table 3 SMMAG General Command Summary

Command	Description
<CR>	Standard command terminator.
$	Altmode - a special command terminator that causes a single program to be loaded but not started.
^Z	A control Z is used to terminate the T command.
D	D<CR> Directs SMMAG to read a control file from the load medium. SMMAG will respond by printing FILE.EXT-. Respond by typing the name of the control file.
F	F<CR> Directs SMMAG to print a directory of the load medium.
G	G<CR> Directs SMMAG to start or restart execution of the program currently loaded in core.
I	I<CR> Directs SMMAG to begin execution of the control file currently in core.
L	L<CR> Directs SMMAG to print a file stored on the load medium. SMMAG will request the name of the file to be printed by printing FILE.EXT-.
R	R<CR> Directs SMMAG to reinitialize itself. SMMAG will begin by requesting the load medium to be used.
S	S<CR> Directs SMMAG to load a single program. SMMAG will request the name of the program by printing FILE.EXT-. Respond by typing the name of the program.
T	T<CR> Directs SMMAG to open a buffer and begin building an internal control file. A control Z (^Z) terminates the T command. Refer to the section on building control files (SMMAG control files) that follows Table 5.
X	X<CR> Directs SMMAG to run through the expanded command set dialogue.

Expanded Command Dialogue
The following additional command sequences are added when either the X command is used or the XPAND switch (18) is set.

TYPE Y OR A FOR SPECIAL USER MODE -

<CR> = no
A<CR> = special user mode after first pass
Y<CR> = special user mode on all passes

Special user mode is a psuedo-user mode. The diagnostic program will be run in user mode (with paging, etc.) and the I/O is trapped back to the diagnostic monitor for processing. This provides a method of checking user mode operation with functional and reliability diagnostics without actually having to use a monitor and timesharing.

Table 4 SMMAG Program Starting Commands

Command	Description
DDT	DDT<CR> Start DDT
PFSTRT	PFSTRT<CR> Power fail restart
REE	REE<CR> Reenter
SFSTRT	SFSTRT<CR> Special features start
START	START<CR> Start diagnostic
START#	START#<CR> Special start. Numbers range from 1 through 5.
STD	STD<CR> Start diagnostic
STL	STL<CR> Start SMMAG
STM	STM<CR> Reinitialize start

Table 5 Standard Manual Starting and Restarting Addresses

Address	Description
20000	SMMAG starting address
20001	If it is desired to abort a test currently in progress or to restart at the next sequential program, the operator may do so by starting at location 20001.
20002	If the diagnostic monitor is running in the mode where titles are not printed [SW 15(1)], and a user program fails such that it is not known which program failed, starting at location 20002 will cause the title to be printed. The computer will then halt at location 20000. The operator may now manually restart the user program or restart the diagnostic monitor.
20003	Program starting and restarting address.

SMMAG Control Files
A control file for SMMAG is an ASCII file consisting of a list of programs to be run. The following apply to constructing a SMMAG control file.

1. A control file can be constructed with any editor program or via the SMMAG T command.

2. A control file can have up to 50 command lines.

3. Each command line consists of five items each separated by a space or tab. The items are as follows.

   a. Program name. If the program name includes an extension, the extension must be included and separated by a period.

      NOTE
      If the special user mode routines are selected, a line that starts with a minus (-) signifies that the program will run in special user mode.

   b. Pass count. The pass count is the number of passes that the program is to run. The pass count may be in the range of 0 to 777777. If 0, the program will run on each pass through the control file.

   c. Switches. This is an octal half word (6 digits) to be used by the program as the right half of the console data switches.

d. Iterations. This is the number, in octal, of iterations the program is to execute. The iteration count may be in the range 0 to 377777. If 0, one iteration is assumed.

e. <CR>. A carriage return terminates the command line and opens the next line for input. If the T command was used to build the control file, a ^Z (control Z) will close the file and return to SMMAG command mode.

Example:

```
DSKAA        10   0         1000<CR>
DSKAB         1   123456    200<CR>
DSKAC         0   000001    1<CR>
^Z
```

4. If the control file is being generated via the T command the following heads will be printed. These act as a guide only and are not actually a part of the control file.

   NAME PASSES RH SWS ITERATIONS

5. Typing errors may be corrected by typing a RUBOUT. The RUBOUT will print three Xs and delete the entire line.

The control file is executed via the I command. The diagnostic monitor will read in and execute the first program on the command list. The program will be iterated the requested number of times and control will then revert to the monitor. The monitor will then proceed to the next program on the list until all programs requested have been executed. When the final program on the command list has been executed, the pass count will be printed and then the monitor will restart with the first program again.

EXAMPLE:

```
SMMAG PASS 000001
SMMAG PASS 000002
etc.
```

A control file will remain in core so that if the monitor is restarted the command list does not have to be read in again unless a new control file or single program is selected.

To use the same control file, type I.

SMMAG ERROR SUMMARY

CMDs REQUIRED
The program was commanded to execute the control file, but the list was empty. Input a program to execute.

ERROR AT
Any load device error will print the message, ERROR AT, followed by the octal address of the error. Consult the listing for an error explanation.

MUUO ERROR
If the diagnostic program being run (in special user mode) causes an MUUO (not trapped I/O), the error message will be printed and the program will halt. The operator may examine the user MUUO locations 17424 and 17425 to determine the cause of the error.

PROGRAM NOT FOUND - PROG.EXT
The program requested is not on the load device.

USER TRAP ERROR
If the diagnostic program being run (in special user mode) causes a trap (PAGE FAIL, PUSHDOWN OVERFLOW or TRAP 3) the message will be printed and the program will halt.

GENERAL INFORMATION

Code	DSQDC.SAV
Title	SMMON - DECSYSTEM-2020 Diagnostic Monitor
Abstract	SMMON is the DECSYSTEM-2020 diagnostic monitor. It runs in either exec or user mode. In exec mode SMMON can load and sequence programs from the disk pack. In user mode, SMMON will run under either TOPS-10 or TOPS-20. The load medium is restricted to disk. SMMON is command-controlled and can be directed to load and run a single program or execute a control file that will run a sequence of programs. Control files enable SMMON to be used for the following purposes. Rapid checkout of the hardware Acceptance testing Reliability testing Unattended overnight testing
Hardware Required	KS10 mainframe Minimum of 32K of memory Load device, RP06 or RM03 disk drive
Preliminary and Associated Programs	SMMON assumes that the basic instructions and the selected load device are operational.
Restrictions	The diagnostic monitor may be used to call only those programs that follow the prescribed diagnostic formats.
Note	The DECSYSTEM-2020 subroutine program SUBSM and SMDDT are automatically loaded on system startup or device specification if they are not already resident in the DECSYSTEM-2020 memory.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	The state of the control switches does not affect the operation of SMMON unless a control file is being used. A control file lists, as part of each command line, the program to be run and the right half switches to use with that program. This allows the actual (console) right half switches to be used to control the operation of SMMON. The switches that affect the operation of SMMON when a control file is in use are listed in Table 1.

OPERATIONAL CONTROL

After the diagnostic monitor is started it will print the following message:

SMMON [DSQDC] - DECSYSTEM 2020 DIAGNOSTIC MONITOR VER 0.1

If the standard diagnostic pack is being used, SMMON will automatically select the disk pack for loading. If the standard diagnostic pack is not being used the following message will be printed:

?DEFAULT DIRECTORY NOT FOUND
UBA#

The appropriate UBA, DISK, and USER area must be supplied. Table 2 defines the devices.

In user mode, the disk is automatically selected and the question is not asked.

After selection of the load device SMMON will automatically load SUBSM and SMDDT and print:

SMMON CMD -

Table 2 describes the device selection commands.
Table 3 describes general SMMON commands.
Table 4 describes program starting commands.
Table 5 lists SMMON manual starting addresses.

Table 1 SMMON Control Switch Summary

Switch	State	Descriptions
9	0	Reduces the iteration count in a control file by a factor of 100 to 1, thus reducing the run time for each program in the file. This is useful for a quick check of the hardware.
	1	Each program listed in a control file is run the specified number of iterations.
15	0	Normal operation.
	1	Inhibit the printing of the test title of each program executed by SMMON.
18	0	Normal operation.
	1	XPAND - Expand the control sequencing. Refer to the X command, Table 3.

Table 2 Device Selection

Device	Description
0	UBA 0, RH address 776700
1	UBA 1, RH address 776700
2	UBA 2, RH address 776700
3	UBA 3, RH address 776700
#	UBA address
?	Identify Disks
DSK:?	Master directory

Table 3 SMMON General Command Summary

Command	Description
<CR>	Standard command terminator.
$	Altmode - a special command terminator that causes a single program to be loaded but not started.
^Z	A control Z is used to terminate the T command.
D	D<CR> Directs SMMON to read a control file from the load medium. SMMON will respond by printing FILE.EXT. Respond by typing the name of the control file.
F	F<CR> Directs SMMON to print a directory of the load medium.
G	G<CR> Directs SMMON to start or restart execution of the program currently loaded in core.
I	I<CR> Directs SMMON to begin execution of the control file currently in core.
L	L<CR> Directs SMMON to print a file stored on the load medium. SMMON will request the name of the file to be printed by printing FILE.EXT-.

Table 3 SMMON General Command Summary (Cont)

Command	Description
R	R<CR> Directs SMMON to reinitialize itself. SMMON will begin by requesting the load medium to be used.
S	S<CR> Directs SMMON to load a single program. SMMON will request the name of the program by printing FILE.EXT-. Respond by typing the name of the program.
T	T<CR> Directs SMMON to open a buffer and begin building an internal control file. A control Z (^Z) terminates the T command. Refer to the section on building control files which follows Table 5.
X	X<CR> Directs SMMON to run through the expanded command set dialogue. Refer to the section on the expanded command which follows Table 5.

Table 4 SMMON Program Starting Commands

Command	Description
DDT	DDT<CR> Start DDT
PFSTRT	PFSTRT<CR> Power fail restart
REE	REE<CR> Reenter user mode
SFSTRT	SFSTRT<CR> Special features start
START	START<CR> Start diagnostic
START#	START#<CR> Special start. Numbers range from 1 through 5.
STD	STD<CR> Start diagnostic
STL	STL<CR> Start SMMON
STM	STM<CR> Reinitialize start

Table 5 Standard Manual Starting and Restarting Addresses

Address	Description
20000	SMMON starting address
20001	If it is desired to abort a test currently in progress or to restart at the next sequential program, the operator may do so by starting at location 20001.
20002	If the diagnostic monitor is running in the mode where titles are not printed [SW 15(1)] and a user program fails such that it is not known which program failed, starting at location 20002 will cause the title to be printed. The computer will then halt at location 20000. The operator may at this time manually restart the user program or restart the diagnostic monitor.
20003	Program starting and restarting address.

Expanded Command Dialogue
The following additional command sequences are added when either the X command is used or the XPAND switch (18) is set.

```
TYPE Y OR A FOR SPECIAL USER MODE -

    <CR>  = no
    A<CR> = special user mode after first pass
    Y<CR> = special user mode on all passes

    Special user mode is a pseudo-user mode.  The diagnostic
    program will be run in user mode (with paging, etc.) and the
    I/O is trapped back to the diagnostic monitor for processing.
    This provides a method of checking user mode operation with
    functional and reliability diagnostics without using monitor
    and timesharing.
```

SMMON Control Files
A control file for SMMON is an ASCII file consisting of a list of programs to be run. The following apply to constructing a SMMON control file.

1. A control file can be constructed with any editor program or via the SMMON T command.

2. A control file can have up to 50 command lines.

3. Each command line consists of five items, each separated by a space or tab. The items are as follows.

   a. Program name. If the program name includes an extension, the extension must be included and separated by a period.

      NOTE
      If the special user mode routines are selected, a line that starts with a minus (-) signifies that the program will run in special user mode.

   b. Pass count. The pass count is the number of passes that the program is to run. The pass count may be in the range of 0 to 777777. If 0, the program will run on each pass through the control file.

   c. Switches. This is an octal half word (6 digits) to be used by the program as the right half of the console data switches.

   d. Iterations. This is the number, in octal, of iterations the program is to execute. The iteration count may be in the range 0 to 377777. If 0, one iteration is assumed.

   e. <CR>. A carriage return terminates the command line and opens the next line for input. If the T command was used to build the control file, a ^Z (control Z) will close the file and return to SMMON command mode.

   Example:

```
DSKAA        10   0         1000<CR>
DSKAB         1   123456    200<CR>
DSKAC         0   000001    1<CR>
^Z
```

4. If the control file is being generated via the T command the following headers will be printed. These act as a guide only and are not actually a part of the control file.

   ```
   NAME   PASSES   RH SWS   ITERATIONS
   ```

5. Typing errors may be corrected by typing a RUBOUT. The RUBOUT will print three Xs and delete the entire line.

The control file is executed via the I command. The diagnostic monitor will read in and execute the first program on the command list. The program will be iterated the requested number of times and control will then revert to the monitor. The monitor will then proceed to the next program on the list until all programs requested have been executed. When the final program on the command list has been executed, the pass count will be printed and then the monitor will restart with the first program again.

EXAMPLE:

```
SMMON PASS 000001
SMMON PASS 000002
etc.
```

A control file will remain in core so that if the monitor is restarted the command list does not have to be read in again unless a new control file or single program is selected.

To use the same control file, type I.

SMMON ERROR SUMMARY

CMDs REQUIRED
The program was commanded to execute the control file, but the file was empty. Input programs to execute.

Disk Pack Errors
Any disk pack error will print the message, ERROR AT followed by the octal address of the error. Consult the listing for error explanation.

MUUO ERROR
If the diagnostic program being run (in special user mode) causes an MUUO (not trapped I/O), the error message will be printed and the program will halt. The operator may examine the user MUUO locations 17424 and 17425 to determine the cause of the error.

PROGRAM NOT FOUND - PROG.EXT
The program requested is not on the load device.

USER TRAP ERROR
If the diagnostic program being run (in special user mode) causes a trap (PAGE FAIL, PUSHDOWN OVERFLOW or TRAP 3), the message will be printed and the program will halt.

GENERAL INFORMATION

Code	DSKAA.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 1)
Abstract	This diagnostic is the first in a series of KS10 processor diagnostics. It is the most basic of the processor diagnostics. This diagnostic assumes the HALT instruction and the computer console to be operative. It makes no further assumptions except that it is loaded into memory correctly. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

CODE	DSKAB.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 2)
Abstract	This diagnostic is the second in a series of KS10 processor diagnostics. The diagnostic tests some of the following instructions: MOVE, COMPARE, HALF WORD and BOOLE. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAC.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 3)
Abstract	This diagnostic is the third in a series of KS10 processor diagnostics. The diagnostic tests some of the following instructions: LOGICAL TEST, HALF WORD and the adder. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAD.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 4)
Abstract	This diagnostic is the fourth in a series of KS10 processor diagnostics. The diagnostic tests register addressing, JFCL, AR FLAGS, ASO, SOS, JRST, AOBJX, JSP, XCT, indirect and indexed addressing. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAE.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 5)
Abstract	This diagnostic is the fifth in a series of KS10 processor diagnostics. The diagnostic tests the FWT, ADD, SUB, PC change and COMPARE instructions. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAF.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 6)
Abstract	This diagnostic is the sixth in a series of KS10 processor diagnostics. The diagnostic tests the BOOLE, HWT, and TEST instructions. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAG.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 7)
Abstract	This diagnostic is the seventh in a series of KS10 processor diagnostics. The diagnostic tests the PUSH, POP, XCT, and BASIC SHIFT/ROTATE instructions. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAH.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 8)
Abstract	This diagnostic is the eighth in a series of KS10 processor diagnostics. The diagnostic tests the PI system, interrupts, LUUOs and input/output. The diagnostic tests the basic functionality of the KS10 processor and microcode.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. If the diagnostic fails to start correctly, try starting at the first test instead of at the beginning of the control sequence. Refer to the listing on microfiche. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program can be determined by examining 30047 "PASCNT".
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.)

OPERATIONAL CONTROL
Once started the program will cycle continually until stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Errors are in the form of HALT instructions. The listing should be consulted to determine the cause of the error. A NOP (JUMP) instruction follows each HALT. This may be useful in constructing a scope loop to cycle on the failing instruction.

GENERAL INFORMATION

Code	DSKAI.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 9)
Abstract	This diagnostic is the ninth in a series of KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, and shift and rotate functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program will run continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKAJ.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 10)
Abstract	This diagnostic is the tenth in a series of KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, logical shift, rotate, and arithmetic shift (single and combined) functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKAK.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 11)
Abstract	This diagnostic is the eleventh in a series of KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, multiply, integer multiply, divide, and integer divide functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKAL.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 12)
Abstract	This diagnostic is the twelfth in a series of KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, multiply, integer multiply, divide, and integer divide functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKAM.SAV
Title	DECSYSTEM-2020 Basic Instruction Diagnostic (Part 13)
Abstract	This diagnostic is the thirteenth in a series of KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, BYTE, BLT, JFFO, and miscellaneous functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL

Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY

Refer to the listing on microfiche.

ERROR SUMMARY

Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKBA.SAV
Title	DECSYSTEM-2020 Basic Instruction Reliability Test (Part 1)
Abstract	This diagnostic tests all of the basic instructions with the exception of fixed point multiply/divide, floating point, and BYTE.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None

Notes

1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs, and other errors of this type are handled by printout on the console terminal.

2. The cycle time depends on memory size and increases as memory size increases.

   A. Normal Operation - Approximately one minute in 32K.

   B. Reliability Mode - Approximately 3 to 5 minutes.

3. The pass count is typed out on the console terminal.

Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Refer to Table 1

Table 1 DSKBA Control Switch Summary

Switch	Description
0-17	Standard (Refer to the 10/10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 12 (MODDUC).
18-26	Not used
27-35	Exec Mode: Select the size of memory to be used. Switches 27-35 correspond to the 9 high-order address bits. The low-order 9 bits are appended as 1s (e.g., 27-35 = 077 equals 77777 or 32K). Switches 27-35 = 000 specifies 32K of the memory is to be used. User Mode The contents of JOBBEL, LOC 44, determine size of memory. Select the desired size when loading the program.

OPERATIONAL CONTROL

Normal operation, with all switches set to zero, is quick-verify mode. To do the reliability test set the "RELIAB" switch. Without the OPRSEL switch (RELIAB switch set or the memory size selected via the switches) the program will run using only 32K. Setting just the OPRSEL switch allows the program to run in quick-verify mode using all available memory.

TEST SUMMARY

In most cases each instruction is tested by simulating the instruction, with a simpler instruction, and then executing it. Random numbers are used as the operands in AC and/or C(E). The results of the simulation and execution are compared, and an error message is printed if the results are not equal.

In both modes (normal operation and reliability mode) a random number is placed in the AC and C(E), and a series of 4 or 5 instructions are executed. The answer in the AC and/or C(E) is checked and an error message is printed if the result is not correct. Each set of 4 or 5 instructions acts on each memory location starting from the end of the area taken up by the program to the designated end of memory.

Also tested are all of the PC-sensitive instructions. These instructions are: JSR, JSA, JSP, JRA, PUSH, POP, PUSHJ, and POPJ. In the majority of cases a JRST or JSP back to the program is placed in every location from the end of the program to the designated end of memory. The program then does a PC-sensitive instruction to the first testing location. When the program returns, a check is made to see that the PC-sensitive instruction went to the right address. If it did not, an error message is printed. The memory address is incremented, and the PC-sensitive instruction repeated. The program also includes a DEFER test and both indirect and indexing are tested.

ERROR SUMMARY

The following are the different error formats with their respective UUOs and error messages.

A. ERROR #1 - ERR AC,E

```
EXAMPLE:                                AC              E
2053 / CAME   AC1,AC2                   ;RESULT         CORRECT
2054 / ERR    AC,RAN1                   ;ORIG C(AC)     ORIG C(E)

AC1=5                                   ;TEST DATA
C(AC1)   = 201532107642
C(AC2)   = 201432107642
C(RAN1) = 777777777777
C(AC)    = 576345670135

ERROR MESSAGE:

                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
AC =     05                             ;AC FIELD OF UUO-1
C(AC) = 201532107642                    ;C(C(AC)) OF UUO-1
COR =    201432107642                   ;C(C(ADDRESS FIELD)) OF UUO-1
         ORIGINAL
C(AC)=   777777777777                  ;C(C(ADDRESS FIELD)) OF UUO
C(E) =   576345670135                   ;C(C(AC)) OF UUO
```

B. ERROR #2 - ERRM AC,E

```
EXAMPLE:                                AC              E
2053 / CAME   AC2,MUD                   ;RESULT         CORRECT
2054 / ERRM   AC,RAN1                   ;ORIG C(AC)     ORIG C(E)

MUD=5033                                ;TEST DATA
C(MUD)   = 201532107642
C(AC2)   = 201432107642
C(RAN1) = 777777777777
C(AC)    = 576345670135

ERROR MESSAGE:

                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
E =      5033                           ;BITS 18-35 (E) OF UUO-1
C(E) =  201532107642                    ;C(C(E)) OF UUO-1
COR =    201432107642                   ;C(C(AC)) OF UUO-1
         ORIGINAL
C(AC)=   777777777777                  ;C(C(E)) OF UUO
C(E) =   576345670135                   ;C(C(AC)) OF UUO
```

C. ERROR #3 - ERRI AC,E

```
EXAMPLE:                                AC              E
2053 / CAME    AC1,AC2                  ;RESULT         CORRECT
2054 / ERRI    RAN1,(AC)                ;ORIG C(AC)     ORIG (E)

AC1=5                                   ;TEST DATA
C(AC1)   = 107742670135
C(AC2)   = 107642670135
C(RAN1)  = 777777777777
C(AC)    = 576345670135

ERROR MESSAGE:
                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
AC =     5                              ;AC FIELD OF UUO-1
C(AC) = 107742670135                    ;C(C(AC)) OF UUO-1
COR =    107642670135                   ;C(C(E)) OF UUO-1
         ORIGINAL
C(AC)=   777777777777                  ;C(C(AC)) OF UUO
E =      670135                         ;C(ADDRESS FIELD)) OF UUO
```

D. ERROR #4 - ERROR AC,E

```
EXAMPLE:                                AC              E
2053 / CAME    AC,RAN
2054 / ERROR   AC,RAN                   ;RESULT         CORRECT

AC=5                                    ;TEST DATA
C(AC)    = 201532107642
C(RAN)   = 201432107642

ERROR MESSAGE:
                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
AC =     5                              ;AC FIELD OF UUO
C(AC) = 201532107642                    ;C(C(AC)) OF UUO
COR =    201432107642                   ;C(C(E)) OF UUO
```

E. ERROR #5 - ER AC,[ASCII/MESSAGE/]

```
EXAMPLE:                                AC              E
2053 / JFCL    10,.+2
2054 / ER      AC,[ASCII/OV/]           ;RESULT         MESSAGE

AC=5
C(AC)    = 201432107642

ERROR MESSAGE:
                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERRORO UUO
AC =     5                              ;AC FIELD OF UUO
C(AC) = 201432107642 OV                 ;C(C(AC)) OF UUO
                                        ;ADDRESS FIELD OF UUO POINTS TO AN
                                        ;ASCII MESSAGE
```

F. ERROR #6 - ERM AC,E

```
EXAMPLE:                                AC              E
2053 / SOJ     AC2,
2054 / ERM     AC1,(AC2)                ;C(AC)          RESULT

C(AC2)=5033                             ;TEST DATA
C(AC)    = 740000005756
C(C(AC2)= 254000004041

ERROR MESSAGE:
                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
E =      5033                           ;BITS 18-35 (E) OF UUO
C(AC)=   740000005756                   ;C(AC) OF UUO
C(E) =   254000004041                   ;C(C(E)) OF UUO
```

G. ERROR #7 - ERRM AC,E

```
EXAMPLE:                                AC              E
2053 / SOG     AC2,
2054 / ERMM    AC1,(AC2)                ;C(AC)          RESULT

C(AC2)=5033                             ;TEST DATA
C(AC1)  = 740000005756

ERROR MESSAGE:

                                        (SOURCE OF NUMBERS PRINTED)
PC =     002054                         ;PC OF ERROR UUO
E =      5033                           ;BITS 18-35 (E) OF UUO
C(AC)=   740000005756                   ;C(AC) OF UUO
```

H. ERROR #11 - EERR ,E

I. ERROR #12 - EERRM ,E

J. ERROR #13 - EERRI ,E

Errors 11, 12 and 13 are the same as errors 1, 2 and 3 except that the AC of the UUO is replaced by C(RAN). In other words C(RAN) will be printed for the original C(E).

GENERAL INFORMATION

Code	DSKCA.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 1)
Abstract	This diagnostic is the first in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, floating scale, floating add/sub, normalized return, and round functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCB.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 2)
Abstract	This diagnostic is the second in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, floating multiply, and floating divide functions. The floating instruction is used to test the exponent calculation functions.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program will cycle continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCC.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 3)
Abstract	This diagnostic is the third in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, and floating point instructions: FIX, FIXR, FLTR. It also tests the double precision moves: DMOVE, DMOVN, DMOVEM and DMOVNM.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program will cycle continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCD.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 4)
Abstract	This diagnostic is the fourth in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, extended addition, and associated logic used by the double precision floating point instructions: DFAD, DFSB, DFMP and DFDV.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDUC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program will cycle continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCE.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 5)
Abstract	This diagnostic is the fifth in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, and precision floating point instructions: DFAD, DFSB, DFMP and DFDV.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCF.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 6)
Abstract	This diagnostic is the sixth in a series of advanced KS10 processor diagnostics. The diagnostic performs logic testing of the KS10 processor, microcode, and double precision fixed point instructions: DADD, DSUB, DMUL and DDIV.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The nonexistent memory and parity stop switches should be reset. These errors, illegal UUOs and other errors of this type are handled by printout on the console terminal. 2. The cycle time of the program is in the millisecond range and is therefore suitable for vibration tests, etc. 3. The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 12 (MODDVC). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKCG.SAV
Title	DECSYSTEM-2020 Advanced Instruction Diagnostic (Part 7)
Abstract	This diagnostic is the seventh in a series of advanced KS10 processor diagnostics. The diagnostic performs testing of the KS10 microcode for extended instruction set.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	The iteration count of the program is printed by the console processor.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 5 (DING), 12 (MODDUC), 14 (OPRSEL), 15 (CHAIN). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program will run continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing or microfiche.

ERROR SUMMARY
Error information may be obtained quickly by printing errors on the line printer.

In the event of print routine failure, the NOPNT switch and the ERSTOP switches may be set to inhibit printout and halt the program with the PC pointing to the error.

GENERAL INFORMATION

Code	DSKDA.SAV
Title	DECSYSTEM-2020 CPU and Memory Reliability Diagnostic
Abstract	This diagnostic test is a comprehensive reliability test of the complete processor and memory subsystem. The program includes arithmetic instruction testing with random operands, random instruction testing, interrupt testing, and memory reliability testing.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	Switch 14 (OPRSEL) must be set to allow the operator to be prompted. The operator then specifies what test control switches are to be used on program startup.

Notes

1. This program uses random numbers and should be run for several hours to ensure that enough numbers are processed to verify system reliability.

2. To increase the duty factor, and decrease the probability of simulator failures, switch 35 (FAST) may be used. However, this is not a complete test of the hardware and should be used with caution.

3. To specify the initial base random number, set switch 18 (RANBAS) (before starting) and respond to the following printout accordingly.

   SPECIFY RANDOM NUMBER BASE -

4. The iteration count of the program is printed as an end of pass count.

Loading and Starting Procedure

Standard (Refer to the KS10 STD module.)

Special feature start (clear all totals) is 30004.

In time-sharing mode the program will run 2 passes and then return to SMMON. To then continually run type "G" to SMMON.

Control Switches

Refer to Table 1.

Table 1 DSKDA Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (Refer to the KS10 STD module). The following switches are not implemented: 12 (MODDVC), 15 (CHAIN).
18	RANBAS	Specify random number base
19	INHCLK	Inhibit clock interrupts
20	INHMEM	Inhibit memory testing
21	INHII	Inhibit instruction interrupt testing
22	INHCI	Inhibit clock interrupt testing
23	INHBLT	Inhibit BLT interrupt testing
24	INHNXM	Inhibit NXM interrupt testing
25		Not used
26		Not used
27	INHNEW	Inhibit double precision instructions
28	INHDFP	Inhibit double floating point test
29	INHFP	Inhibit floating point test
30	INHBYT	Inhibit byte test
31	INHFXD	Inhibit fixed point test
32	INHRAN	Inhibit random instructions
33	SNGFL	Run single fast loop
34	SLOW	Run simulation comparisons (only)
35	FAST	Run instruction comparisons (only)

OPERATIONAL CONTROL
Operational control is via the console control switches. Switch 14 (OPRSEL) when set will cause a prompt for test selection.

TEST SUMMARY
The arithmetic testing is done using pseudo-random numbers and comparing the machine results with each other, and with a hardware operation simulation program.

The program instruction testing program is designed to execute random instructions (non-PC change) in memory, in the fast ACs, and through software simulation. The results of the three groups of instructions are compared for equality. Upon a discrepency the program prints all pertinent information and goes into a repetitive failure loop.

The interrupt portion of the diagnostic tests the priority interrupt system, the processor APR system, and the interrupt ability of most classes of instructions. It also tests nonexistent memory interrupts, and BLT instruction interrupt ability.

The memory reliability portion of the test is used to verify the operations of the memory subsystem. All of the memory, up to 512K, is used if available. A physical memory address test and a randomly selected data patterns test are performed. Memory addressing is verified by using a fast-rate addressing scheme in which the selected address bit will change on every memory access. The fast-rate testing is randomly selected. The selection will have all address bits fast-rated, one address bit fast-rated, or no fast-rate testing.

Simulation Comparison
In this section instructions are executed, then simulated, and the results compared. Actual execution of the instruction should give the same results as simulation of the instruction. Simulation is done by using software routines and psuedo-hardware registers to follow the hardware instruction flows. This routine is controlled by switch 34 (slow).

Instruction Comparison
In this section instructions are tested by performing a divide, a multiply, and then add back in the division remainder. The original operands should be the same as the final results. This routine is controlled by switch 35 (fast).

ERROR SUMMARY
Program errors such as illegal UUOs, parity errors, nonexistent memory, illegal interrupts, etc., are handled by printout of the type of error with as much information as is pertinent.

Instruction Error
If an instruction fails to give the correct results the following will be printed: test title, pass count, type of failure (instruction comparison or simulation comparison), machine results, simulation results, and instruction simulation.

If something happens to the printout or you are not sure what the original and final operands are:

1. ACs are saved in locations SAVAC through SAVAC+17.
2. Original operands are in SAVAC+1, 2 and 3.
3. Results are in AC1, AC2 and AC3.
4. C(AC1)=C(AC), C(AC2)=C(AC+1), C(AC3)=C(E) OR E.

Look in symbol table at end of listing for location of SAVAC.

Simulator Printout
When an error occurs the simulated machine states and registers are printed.

The printout contains the results of the instruction under test; that is, AC, AC+1, E or C(E) for fixed and floating-point instructions, and AC, pointer, C(E) for byte instructions. The following two lines indicate the instruction that failed, initial values, and simulated results. This is followed by the machine times and correct contents of the several registers, after the occurrence of the time pulse.

GENERAL INFORMATION

Code	DSKEA.SAV
Title	DECSYSTEM-2020 Paging Hardware Diagnostic
Abstract	This diagnostic tests the paging hardware of the KS10 central processor.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The program cycle time is a under a minute and is therefore suitable for vibration tests, etc. 2. This diagnostic does not use the INHPAG switch and does not use the cache. 3. The iteration count of the program is printed on the console terminal.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 11 (INHPAG), 12 (MODDVC), 13 (INHCSH), 14 (OPRSEL), 15 (CHAIN).

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
Refer to the listing on microfiche.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSKEB.SAV
Title	DECSYSTEM-2020 KS10 Cache Diagnostic
Abstract	This program was written to recognize and report any errors caused by a fault in the cache logic. The diagnostic consists of seven tests.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The program cycle time is a under a minute. 2. The iteration count of the program is printed on the console terminal.
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 11 (INHPAG), 12 (MODDVC), 13 (INHCSH), 14 (OPRSEL), 15 (CHAIN). There are no right-half switches.

OPERATIONAL CONTROL
Once started the program runs continuously until it is stopped or an error occurs.

TEST SUMMARY
The individual tests performed by this diagnostic are summarized in Table 1.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

Table 1 DSKEB Test Summary

Test	Description
ACCHK	This test verifies that all AC blocks are working correctly. First, each AC in each block is loaded with the block number in the left half, and the AC number in the right half. Thus a unique number is loaded into each AC. The ACs are then checked to ensure that each AC contains the proper number.
EUCHEK	This program checks the transition of the EXEC/USER signal in the cache control logic from the exec state to the user state. This test is used in conjunction with UECHEK, to check the opposite transition of the signal. If both tests succeed, the signal is working correctly. Next, AC 17 is checked to ensure that it can hold all zeros, and then all ones. Then the entire block is tested to ensure that all the ACs in it can hold both zeros and ones. The test is repeated for each of the other blocks.
PHYCHK	This test ensures that the signal NOT PHYSICAL is working correctly. A page is loaded and timed with paging turned off, which yields references that are physical. If any cache hits are found, the signal is bad. When used in conjunction with the other tests that check for cache hits, the opposite state of the NOT PHYSICAL is checked for validity. However, if all the other timing tests fail, this signal is a good place to look for the trouble.
CACCHK	This test makes sure that the page cacheable signal is working correctly. The test checks for both states of cacheable and uncacheable, and performs the test on both exec and user address space.

Table 1 DSKEB Test Summary (Cont)

Test	Description
LOALIT	This test performs the low order address interference checking. The address lines running into the cache (one page) are tested for interference, while at the same time performing a RAM test on the cache RAMS (treating the cache as a memory). This test also allows the checking of the write through, because the memory should end up with different data than what it started with (even though the cache RAM test succeeded).
HOALIT	This test checks the high order address lines of the virtual address and performs an interference check on those lines. A page is loaded into the cache. Then the page number is changed by one bit and executed, while being timed. If the test runs faster than the threshold, cache hits were encountered, indicating the problem is the address bit that changed. If cache misses are encountered, then the address lines have no interference with each other.
UECHEK	This test checks the transition of the EXEC/USER signal in the cache control logic from the user state to the exec state. The test is used in conjunction with EUCHEK, to check the opposite transition of the signal. The test loads a page of memory into the cache by executing in user mode. The program is then put into exec mode, and the same page is executed, while being timed. If the test ran faster than the threshold (500 ns per JRST), then the references received cache hits when they were not supposed to. If however the test ran slower than the threshold, then the references received cache misses, and the state of the EXEC/USER signal is correct.

GENERAL INFORMATION

Code	DSKFA.SAV
Title	DECSYSTEM-2020 Instruction Timing Diagnostic
Abstract	The diagnostic allows the execution times of the different classes of DECSYSTEM-2020 instructions to be timed. The time measurements are then used to ensure that the KS10 processor is operating correctly.
Hardware Required	KS10 mainframe Minimum of 32K of memory
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None
Notes	1. The timing measurements are accurate to within 5 ns. 2. Non-straightfoward time measurements are as follows. **NOTE** **The symbology T(X) means the time required for operation X.** T(main clock) = <T(LSH D35) = T(LSH D15)>/20 This time is the setting of the master clock delays. T(indexing) = T[MOVEI (3)] = T(MOVEI) T(indirecting) = T(MOVEI @3) = T(MOVEI) T(indexing + indirecting) = T[MOVEI @ (3)] = T(MOVEI) T(PUSHJ) = T(PUSHJ + MOVEI) = T(MOVEI)
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Only the following switches are used: 0 (ABORT), 3 (NOPNT), 4 (PNTLPT), 13 (INHCSH). There are no right-half switches.

OPERATIONAL CONTROL
The program makes one pass printing out the timing information and then halts.

TEST SUMMARY
A table is loaded with the instruction to be timed along with any initializing instructions needed. The table is repeatedly executed for one second while the number of iterations of the instruction sequence (the test instruction plus initializing instructions) is counted. The time for the instruction under test is, then, the time for the instruction sequence minus the time for the initializing instructions.

Be cautious in drawing conclusions from the printed data; the data is accurate but the nature of what was timed is sometimes misleading. For example, it would seem reasonable that the time necessary to do an index operation and an indirect operation separately, would be the same time required to do them in one instruction. Because the processor frequently waits for the memory cycle time, this is not always true.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

GENERAL INFORMATION

Code	DSMMA.SAV
Title	KS10 1024K Memory Diagnostic
Abstract	The memory diagnostic tests all of memory unoccupied by the program to make sure that the memory modules and associated hardware function as a complete operating system. The program consists of a series of five tests. The first test verifies the ability of the memory to operate with various bit combination data patterns. The second test verifies the ability of the memory to uniquely address any and every memory location. The third test does a moving inversion pattern using ones and zeros in the data field, then repeats the pattern using ones and zeros in the parity and check bit fields. The fourth test verifies the ability of memory to write and read a single one or single zero bit in a memory array of all ones or all zero words. The fifth test is the same as test 1 except that the combination of all patterns run verifies that there are no single bit errors, and that there are no double bit interactions. Any one, or a combination, of the five tests is selectable from a terminal or from the console data switches. Also, all of memory or any selected module or portion of memory may be tested.
Hardware Required	KS10 mainframe Minimum of 1024K of memory Console terminal
Preliminary and Associated Programs	Refer to the diagnostic hierarchy (KS10 STD module).
Restrictions	None

Notes

1. KS10 does not currently support 1024K of memory; however, the diagnostic is designed with 1024K capabilities should future enhancement occur.

2. Make several passes of the program with data switches 13 (INHCSH) and 14 (OPRSEL) set. To find any solid memory failures, inhibit block transfer (BLT) exercising, fast rate addressing, and read/restore.

3. Make several passes of the program with data switch 9 (RELIAB) set (13 and 14 reset) to find any marginal memory failures.

4. Make several passes of the program in the normal mode, no data switches set, to verify memory system reliability.

5. To thoroughly test the first 16K memory module of a system, use the memory module switches to exchange modules 0 and 1, then reload the program and repeat Notes 2-4 testing module 1.

6. Memory mapping on the KS10 may be limited to unpaged memory by setting switch 11 (INHPAG); i.e., limit maximum tested memory to 112K, set the INHPAG switch. If the switch is not set, the EPMP/UPMP is set up so that relocation equals actual for addresses 112K through 256K.

7. To test memory addressing greater than what is available in consecutive memory, a memory module may be selected to whatever address is desired. Once selected, the memory module may be tested by using 'switch controlled' testing or by using 'segment' testing. Observe the memory map printout and use only memory that actually exists.

Control Switches — Refer to Table 1.

Table 1 DSMMA Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (Refer to the KS10 STD module.)
18	CORENA	Enable reporting of correctable errors as they occur. (This test is for Manufacturing ONLY.)
19	EXTDIN	Extended input format - used to specify and run on selected MA bit in fast rate addressing, plus, if in switches mode, allows type-in selection of specific data patterns for the data patterns test.
20		Unused
21 22 23	WCPSW3 WCPSW2 WCPSW1	Data pattern selection 0 = all patterns 1 = all ones 2 = all zeros 3 = alternate bit pattern 4 = floating ones 5 = floating zeros 6 = floating ones and floating zeros 7 = pseudo-random, parity bit check
24 25 26	TSTSW3 TSTSW2 TSTSW1	Test selection. 0 = physical address, patterns-MOS, address 1 = data patterns 2 = address 3 = WCP 4 = float 1s/0s 5 = data patterns-MOS 6 = physical address 7 = all tests
27	MODUL	Test single module - 128K
28 29	TMIL1 TMIL0	Test what million K 0 = 000000 - 3777777 1 = 4000000 - 7777777 2 = 10000000 - 13777777 3 = 14000000 - 17777777
30 31 32	T128K2 T128K1 T128K0	Test what 128K of selected million 0 = 000000 - 3777777 1 = 400000 - 777777 2 = 1000000 - 1377777 3 = 1400000 - 1777777 4 = 2000000 - 2300000 5 = 2400000 - 2777777 6 = 3000000 - 3377777 7 = 3400000 - 3777777
33	T32K	Enable switches 34 and 35 to select what 32K segment of 128K segment
34 35	T32K1 T32K0	Test what 32K segment of 128K segment 0 = first 32K 00 - 77777 1 = second 32K 100000 - 177777 2 = third 32K 200000 - 277777 3 = fourth 32K 300000 - 377777

OPERATIONAL CONTROL

Console Terminal Control

The test parameters may be selected by input from the console terminal. The desired test and the patterns to be used for the data patterns test may also be specified by the console terminal.

The following are sample inputs and the associated computer printouts. Operator responses follow the -.

```
KS10 MEMORY DIAGNOSTIC (DSMMA)
MEMORY MAP=
FROM TO SIZE/K
000000  077777  32

TEST SELECTION
PHYSICAL ADDRESS, Y OR N (CR) - Y
DATA PATTERNS - 0 FOR ALL, 1 TO 7, Y OR N (CR) - Y
ADDRESS, Y OR N (CR) - Y
MOS WCP - 1 TO 4, Y OR N (CR) - Y
FLOATING ONES/ZEROS, Y OR N (CR) - Y
MOS DATA PATTERNS, Y OR N (CR) - Y
SELECTION COMPLETED, IS IT OK, Y OR N (CR) - Y
```

In the above example the test header was printed, the memory was mapped and the operator was asked what tests are to be run.

At the completion the operator verifies that the selection was correct and then testing commences.

TEST SUMMARY

Memory Mapping
In the first pass of the program, a map of memory is printed as it is seen by the CPU. The purpose of this memory mapping is to determine maximum memory size for the test parameters and also to point out any missing memory addresses or noncontiguous memory modules.

The memory map is printed in the following format:

```
MEMORY MAP=
FROM TO SIZE/K
NNNNNN NNNNNN NN
XXXXXX XXXXXX XX
```

Where:

Ns = the lowest contiguous segment of core memory. This entry should always be the only entry and equal to all memory.

Xs = some higher segment of memory. This will occur for example in a 32K system where the higher module has an address greater than 1.

If there is more than one segment of memory the following is also printed:

```
MAPPING ERROR   ;NON-CONSECUTIVE MEMORY MODULES?
```

If there is any nonexistent memory below 8K the following is printed:

```
MAPPING ERROR   ;NON-X-MEM BELOW 8K?
```

If there is an interleaving problem the following is printed:

```
MEMORY MAP =
FROM TO SIZE/K ADR SEQ
000000 377777 128 XXX7
000000 577777 192 XXX6
000000 577777 192 XXX4

MAPPING ERROR; MEMORY INTERLEAVING ?
```

This indicates that the memory interleaving is faulty.
The first map is done with addresses ending with 7.
The second map is done with addresses ending with 6.
The third map is done with addresses ending with 5.
The fourth map is done with addresses ending with 4.

Memory mapping for nonexistent memory is done with address multiples of 1000(8). Memory mapping of existent memory is done with address multiples of 100(8).

Data Pattern Test
The word patterns to be used for the data patterns test may be specified by input when the program asks for the test selection. Instead of answering Y or N the operator may type in the following:

DATA PATTERNS:

Y = run test, all patterns
N = don't run test
1 = run test, all ones pattern, ONES, all ones
2 = run test, all zeros pattern, ZEROS, all zeros
3 = run test, alternate bit pattern, ALTB, alternate bits, 525252 525252 on the first pass, then 252525 252525 for the 2nd pass. (the same data is used on each pass throughout the memory area tested)
4 = run test, floating ones pattern, FLONE, floating ones, 042104210421 for the first location of the first pass. The data pattern rotates word to word throughout the memory area tested. On the second pass the initial location word is rotated one and then this data word rotates word to word throughout the memory area. This continues on subsequent passes until all data bit combinations are tested (i.e., four passes).
5 = run test, floating zeros pattern, FLZRO, floating zeros, 735673567356 for the first location of the first pass. This data then rotates word to word throughout the memory area tested. On the next pass the initial word is rotated one and the operation continues on subsequent passes until all combinations are tested (i.e., four passes).
6 = run test, both floating ones and floating zeros patterns, Uses the same test code as test 1, the data patterns test, except that the data patterns that are used are selected to verify that every data, parity, and check bit in the storage array can hold 1s and 0s and that there is no interaction between any two bits in the three fields.
7 = run test, pseudo-random, parity bit check pattern, PRAND, pseudo-random parity bit check, 123456701234 for the first location of the first pass. This data then rotates word to word and is two's complemented throughout the memory area tested. On the next pass the initial word is rotated three and the operation continues on subsequent passes until all combinations are tested (i.e., 12 passes).

This pattern causes the parity bit to change word to word and should be the most comprehensive test.

The data patterns test verifies the ability of the memory system to operate with various bit combination data patterns.

The testing sequence used is as follows.

1. Generate the data pattern word.
2. Fill memory with the data pattern.
3. Perform read/restore memory exercises.
4. Test the memory for the correct contents.
5. Perform BLT memory exercises.
6. Test the memory for the correct contents.
7. Set up for fast rate addressing.
8. Perform the fast rate memory exercises.
9. Test the memory for the correct contents.
10. Repeat the fast rate memory exercise using the next address bit.
11. Test the memory for the correct contents.
12. When all the address bits are fast rate exercised, repeat the sequence until all data patterns are completed.
13. Advance to the next test.

Memory Addressing Test
The memory addressing test verifies the ability of the memory system to uniquely address any and every memory location selected for test. This is done by writing the value of each memory location into the right half of itself and the complement of this into the left half. The memory is then exercised and tested. At the completion of this the address pattern word is reversed and the value of each location written into the left half word and the complement of this written into the right half word. The memory is then exercised and tested for correct contents.

The address pattern words used are:

1. C,ADR - LH, complement address/RH, address

2. ADR,C - LH, address /RH, complement address

The testing sequence used is the same as the data patterns test except that the address pattern words are used.

The memory segment is tested using the basic sequence in four passes. The first pass, after the background is written in memory, consists of testing the memory from the low address to the high address, writing the complement of the background. The second pass tests the memory from the high address to the low address, reading the complement of the background, and writing the background back. This leaves the memory filled with the initial background pattern. The third pass tests the memory from the high address to the low address of the segment, writing the complement of the background. The fourth pass tests the memory segment from the low address to the high address, reading the complement of the background, and writing the initial background pattern, leaving the memory filled with the initial background pattern.

If the switches permit, the entire sequence is repeated using each address bit as the fast rate addressing bit. The first pattern used is zeros and ones. This tests the RAMS that hold the data bits. The testing sequence is repeated for a data pattern pair that sets the parity bit and the check bits to zeros and ones.

Floating Ones/Zeros Test
The floating ones/zeros test verifies the ability of the memory system to write and read a single one or a single zero bit in a memory array of all ones or all zero words. This is done by filling the memory array with all ones and then writing a single one in the right-most position of the first word. This word is then read and compared to what was written. The single bit is then rotated left one position and written, read, and compared. This is repeated until all bit positions in the word have been checked. The tested location is then advanced to the next word and the sequence repeated. This continues until all words in the array have been checked. The entire memory array is then tested for correctness.

This process is then repeated using a memory array of all zero words and a floating zero testing word.

Worst Case Patterns Test
The worst case patterns test verifies that every cell of the storage RAM chip can be read and written without affecting any other cell in the RAM.

The memory is first filled with a background pattern. The basic sequence is to first verify the background pattern in the cell with a read. The complement of the background is then written into the cell, and then the cell is read again to verify that the complement was written correctly. The next test cell is then selected.

ERROR SUMMARY

Memory Data Error
When a memory data error occurs the following head is printed:

```
MEMORY DATA ERROR
TN  AS  PAT  ADDRESS  CORRECT  ERROR  FAILED  BITS  PAR
```

Where:

TN = the current test which detected an error.

1 = data patterns
2 = address
3 = worst case patterns
4 = floating 1s/0s
5 = data pattern-MOS
6 = physical address

AS = the current addressing scheme being used to exercise memory (SEQ = sequential, BLT = block transfer, FNN = fast rate on bit NN).

PAT = the current data pattern mnemonic.

ADDRESS = the memory location that contains the data in error.

CORRECT = what the data should have been in that location.

ERROR = the data as read from that location.

FAILED BITS = a binary one indicates that the particular data bit failed, could be a bit pickup or dropout, (printed in octal).

PAR = a P indicates that when the failed location was reread for printout it still contained a parity error.

MEMORY DATA ERROR (sample printout)

TN	AS	PAT	ADDRESS	CORRECT	ERROR	FAILED BITS	PAR
1	SEQ	ALTB	070777	252525 252525	252565 252525	000040 000000	
2	F34	C,ADR	013447	764330 013447	764332 013445	000002 000002	
3A	F31	ZEROS	070777	777777 777777	000000 000000	777777 777777	P
4	SEQ	ONES	070777	777777 777777	777776 777777	000001 000000	

During test 1 address 070777 was found to be in error. From the example, it can be seen that bit 12 was picked up while using the alternate bit data pattern and sequential addressing. During test 2 address 013447 was found to be in error. From the example, it can be seen that what was actually read was the data that corresponds with address 013445 and from this it can be deduced that, while exercising memory with memory address bit 34 changing with every memory access, address 013445 was incorrectly read and then written into the correct location (i.e., bit 34 was slow to set up). During test 3 address 070777 was found to be in error. From the example, it can be seen that all data bits were dropped or that the location failed to read at all. Also, a parity error was found at the failed location. During test 4 address 070777 was again found to be in error. This time bit 17 was dropped.

After each error printout the program continues with the next memory location to test.

Memory Parity Error
When a memory parity occurs the following heading error information is printed (sample printout):

```
MEMORY PARITY ERROR
    TN  PROG   PC    AS   PAT   ADRCON      DATA          PARITY
    3  001057 000006 F32 ALTB   027637  525212 525252     P
```

Where:

TN = the current test being run when the parity error occurred.

PROG = the current subroutine entry point in the test program (the last PUSHJ entry).

PC = the current program counter when the parity error occurred.

AS = the current addressing scheme being used to exercise memory.

PAT = the current data pattern.

ADRCON = the address control word used to access memory, normally points to the current location being tested.

DATA = the contents of the location pointed to by ADRCON normally will be the erroneous data which caused the parity error. If ADRCON points outside memory this will be blank.

PARITY = A P indicates that when the location pointed to by ADRCON was reread for printout it still contained a parity error

In the above sample, a memory parity error occurred during test 3. It can be seen that while exercising the memory using the alternate bit data pattern and fast rate addressing (bit 32 being the current fast rate bit), bit 12 was dropped causing the parity error. Also, when the current testing location pointed to by ADRCON was reread for printout it still contained a parity error. By referring to the program listing it can be seen that the current subroutine was running in the ACs. Which subroutine it was could be determined by the program printout.

If a memory parity occurs when the PC is not in the ACs the following is also printed:

```
PARITY ERROR IN PROGRAM
PROGRAM OPERATION QUESTIONABLE FORM THIS POINT
```

In this case ADRCON and data are probably not valid error indications.

Single Bit Memory Error
When a single bit memory error occurs the following head with error information is printed sample printout:

```
SINGLE BIT MEMORY ERROR
    IN   PROG    PC    AS   PAT    ADDRESS          DATA    BIT #
    5  031342  000006 SEQ  MWC1   00300000   036624 475570 31
```

where:

TN = the current test being run when the error occurred.

PROG = the current subroutine etnry point in the test program, (the last PUSHJ entry).

PC = the current program counter when the error occurred.

AS = the current addressing scheme being used to exercise memory.

PAT = the current data pattern.

ADDRESS = the address control word used to access memory, normally points to the current location being tested.

DATA = the contents of the location pointed to by ADRCON, normally will be the erroneous data which caused the error. If ADRCON points outside memory this will be blank.

BIT# = the bit in the word that was bad.

During test 5, a single bit error occurred. The memory was being addressed sequentially, and MOS worst case pattern 1 was being used. At address 300000, bit 31 failed. Before the program continues after a single bit error, the dip number is determined, and a check is made in the chip error table. If this was the first error for this chip, and there is room in the chip error table, this chip is added to the table, including the slot of the board and the E number of the chip. The error count of the chip is set to 1. If the chip is already in the table, the error count is incremented. If the table overflows with too many bad chips, the following message will be printed only once:

```
SINGLE BIT ERROR TABLE FULL.
ERROR TABULATION WILL CONTINUE BUT NO MORE DIPS WILL BE ADDED.
```

After eight errors for a specific chip have been reported, further typeouts will be suppressed, but error counting for that chip will continue up to a value of 377777.

After each single bit error and printout the program continues the testing sequence.

Nonexistent Memory and Channel 1 Interrupt Errors
When either of these errors occur, one of the following associated headings with error information is printed.

```
NONEXISTENT MEMORY INTERRUPT
    APR             PI              FLAGS   PC      PROG
    011023  013001 000000 000000    3000000 000004  000755

    Or:

    ERROR INTERRUPT
    APR             PI              FLAGS   PC      PROG
    011023 013001 000000 000000     300000  000004  000755
```

Where:

APR = CONI from APR.

PI = CONT from PI.

FLAGS = current program flags when interrupt occurred.

PC = the current program counter when the interrupt occurred.

PROG = the last subroutine entry point in the test program (the last PUSHJ entry).

Illegal UUO
When an illegal UUO is executed the following heading with error information is printed.

```
ILLEGAL UUO EXECUTED
    UUO             FLAGS           PC          PROG
    064240 000000   300000          003677      000433
```

Where:

UUO = the illegal UUO that was attempted to be executed.

FLAGS = the program flags

PC = the program counter at which the UUO occurred.

PROG = the last subroutine entry point in the test program (the last PUSHJ entry).

Program Checksum Error
At the completion of a testing sequence the program is checksummed and this checksum is compared to the checksum obtained before the program was executed. If the checksums agree the program continues normal operation. If the checksums do not agree the following is printed:

```
ERROR IN PROGRAM, CHECKSUMS DO NOT AGREE
PROGRAM OPERATION QUESTIONABLE FROM THIS POINT
```

The program then continues operation or restarts depending upon switch 7, ERSTOP.

The program checksum is obtained by adding together all program code from BEGIN1 to ENDSLD of the program. All changeable words are at the end of the program after ENDSLD, (all variable data). The purpose of this is to verify that the program is indeed valid since when testing a memory module in which the program also resides, the program could get changed due to interaction between the testing area of core and the program data of core. This could cause erroneous error reporting.

At the completion of the testing sequence or after each individual test, as selected by the switches, the error totals are printed. This printout facilitates data bit error determination and also address failure correlation. The printout provides the error totals per test, the number of parity errors, the data bits with pickup and dropout errors, and the address bits with associated data bit pickup and dropout errors. Also, the number of single bit errors per chip is reported.

A sample printout follows.

```
TEST COMPLETION, PASS COUNT  0
ERROR TOTALS:
       PHY ADR  PATTERNS     ADDRESS      WCP     FLOAT  ECC ERRORS
          0          0          0          0         0      571

PARITY ERRORS:  0

DATA BIT FAILURES
     BIT       PICKUP              DROPOUT              ECC
      11          0                   0                205
      36          0                   0                190
      40          0                   0                  1
      41          0                   0                175

ADDRESS BITS WITH DATA FAILURES
     BIT       PICKUP              DROPOUT              ECC
      14
      15
      16
      17          0                   0                571
      18          0                   0                190
      19          0                   0                175
      20          0                   0                395
      21
      22          0                   0                381
      23          0                   0                571
      24          0                   0                571
      25          0                   0                366
      26          0                   0                366
      27          0                   0                  1
      28          0                   0                381
      29          0                   0                206
      30          0                   0                381
      31          0                   0                206
      32          0                   0                381
      33          0                   0                571
      34          0                   0                366
      35          0                   0                176
                  0                   0                365

      ECC ERROR FREQUENCY
      SLOT        E# / ERRORS
      2           175/190
      3           209/175
      4           74/205, 196/1

      END PASS 1.
```

After the totals printout the error counters are zeroed and the program checksummed. Test operation then continues with the first/next test.

GENERAL INFORMATION

Code	DSRMA.SAV
Title	DECSYSTEM-2020 RH11-RM03 Basic Device Diagnostic
Abstract	DSRMA isolates solid RM03 faults to the faulty module or group of modules within the DCL or drive electronics. Scope looping capabilities have been designed into each test. Testing is done on a "start small and build up" basis. The diagnostic starts by testing all control bus cycles and ends by doing full-speed data transfer operations to various disk addresses. Positioning logic is also tested. In addition to the straight line diagnostic tests, the diagnostic also contains a head alignment/verification subprogram that is invoked by setting the proper sense switch. This diagnostic runs only in exec mode.
Hardware Required	KS10 mainframe Minimum of 48K of memory RH11 Massbus controller 1 to 8 RM03 disk drives
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	None

Notes

1. Run time (per drive basics)

   a. If no operator intervention tests are run, the diagnostic will complete a pass in approx. 2 minutes 40 seconds, with one drive selected and no errors occurring.

   b. The dual-port tests take approximately 2 minutes to execute once the drive is cabled up.

2. The following sequence must be followed to ensure complete testing of a single-ported device.

   a. Cable the controller to the active port, lock the device on that port, and run one pass of the diagnostic with the manual intervention tests selected.

   b. If multiple passes are desired, it is not necessary to run the manual intervention tests following the first pass.

3. The following sequence must be followed to ensure complete checkout of dual-ported drives.

   a. Cable the controller to either port, lock the drive on that port, and run a pass of the diagnostic with the manual intervention tests selected. This tests one port.

   b. Cable the controller to the other port, lock the drive on that port, and run a pass of the diagnostic. This time the manual intervention tests are not necessary. This tests the other port.

   c. Run the dual port test as instructed by the diagnostic. This tests the dual-port arbitration logic in the drive. The dual port test cable #7010507-02 is required.

Loading and Starting Procedure — Standard (Refer to the KS10 STD module.)

Control Switches — Refer to Table 1.

Table 1 DSRMA Control Switch Summary

Switch	Descriptions
0-17	Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 9RSTART), 2 (TOTALS), 9 (RELIAB), 11 (INHPAG), 12 (MODDUC), and 15 (CHAIN).
18	Trace program's progress test by test
19	List all possible modules if not running in text inhibit mode
20	If set, request UBA address parameters
21	Run the dual-port tests
22-23	Not used
24	Trace tests that are omitted (skipped)
25	Pause until this switch is reset
26	Run the test specified by switches 27-35
27-35	If SW 26 is set the selected test will run continuously The following test numbers will run the specified RH11 test: Switches (27-35) — RH11 Test 157 — RHTST1 Tests RHWC 160 — RHTST2 Tests RHBA 161 — RHTST3 Tests RHDB 162 — RHTST4 SILO Test 1 163 — RHTST5 SILO Test 2 164 — RHTST6 SILO Test 3

OPERATIONAL CONTROL
The diagnostic furnishes all necessary instructions for operation as it runs. Sense switch options may also be printed at run time.

The diagnostic will first poll the system and report which RM03 and controller configurations exist. After the system configuration report, the user is allowed to select any number of drives for test (out of those found).

A drive may be selected for test, whether or not it was detected at configuration time. Following the completion of configuration, the diagnostic goes to its test dispatcher to determine which tests are to be run.

TEST SUMMARY
Table 2 describes the individual tests performed by this diagnostic.

NOTE
The following table correlates the RH test numbers with the sequential test numbers. An RH11 test error will be reported by its RH11 test number (e.g., RHTST1) rather than the diagnostic test number (e.g., 157).

Diagnostic No.	RH11 No./Test
157	RHTST1/RHWC
160	RHTST2/RH3A
161	RHTST3/RHDB
162	RHTST4/SILO Test 1
163	RHTST5/SILO Test 2
164	RHTST6/SILO Test 3

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

Dispatcher
In order to understand the dispatcher, it is important to understand the following facts about the test structure.

1. There are 265 (octal) in-line standalone tests, some of which require operator intervention. These tests may be selected for individual operation by SW 27-35.

2. The dual-port test is a standalone series of 23 (octal) subtests that are self-documenting with complete instructions.

The dispatch algorithm works as follows.

a. Read the console switches.

b. If pause switch (002000) is set, go to step a.

c. If the dual-port option switch is set (040000), run the dual-port tests on dual-ported RM03s only, then return to step a.

d. If the operator intervention switch is ON, set a flag that allows operator intervention tests to be executed when encountered. (Flag is checked locally at each test.)

e. If SW 26 is set (001000) (Manual Test Selection) execute the test whose number is specified in switches 27-35. If illegal, give an error message. The test is executed one time and then return is made to step a.

f. Execute the next in-line test of the 265 (octal) available one time, and then return to step a. When all tests have been executed, end of pass is typed and the sequence will be repeated until the diagnostic is stopped, or the abort switch is raised.

NOTE
When an in-line test is being run, it is executed on all drives being tested before return is made to the dispatcher.

Use of the Diagnostic
A typical use of the diagnostic would be as follows.

Run the diagnostic until it fails on a test and then select that test from the switches so that it will be continuously run and the trouble can be isolated or modules replaced. After replacing a module or group of modules the entire diagnostic should be rerun in case the symptoms have changed.

Due to the structure of the diagnostic, troubleshooting should be done using the earliest possible test that failed. If several tests indicate malfunctions, check them out one at a time.

Table 2 DSRMA Test Summary

Test	Description
1	DUAL DRIVE SELECTION TESTS All drives on the bus are powered up except the drive being tested. The diagnostic will do a blind read of the drive type register (register 06) to guarantee that no other drive on the Massbus responds to this drive's device number. If the test is unsuccessful, the drive type and serial number registers of the responding device on the bus will be printed in order to aid in isolating the faulty device. In this particular case, the drive being tested is not the faulty one and it is necessary to determine which drive on the bus is incorrectly responding by shutting the drives off (one at a time) until the problem disappears. That drive is the one that has faulty hardware and its modules (as called out) should be replaced, provided it is an RM03. If the device is not an RM03, run the diagnostic for the device, as there is something wrong with device selection logic for that device. Either the device select bits are not getting over properly or the logic that looks for a match between device selection bits and their logical address is malfunctioning. There is a remote possibility that there are actually two drives on the Massbus with the same address.

Table 2 DSRMA Test Summary (Cont)

Test	Description
2	DEMAND/TRANSFER HANDSHAKE TEST This test does a blind read of the drive-type register (register 06) and looks only for the proper operation of demand and transfer. Data and parity errors are ignored. Should a problem occur, check the demand transfer function along with the device selection logic.
3	TEST ABILITY TO READ BACK 0S IN C00-C15 The following sequence is used for reading zeros over the control bus: 1. Write device register 20. This is a fully writable device register and writing it with 1s will cause the RM03 holding register to fill up with 1s. 2. Because the program receives the complement of the data contained in the RM03 holding register, the data seen by reading register 13 should be all 0s in C00-C15. Parity errors are ignored in this test. Failure possibilities include: (IF), no (ILR), register select error, CTOD fault, handshake timing failure, and transmitters not getting enabled. Also, (CBI) faulty CTOD line level and faulty transmitter chips in failing bit position.
4	TEST ABILITY TO READ BACK 1S IN C00-C15 The following sequence is used for trying to read 1s over the control bus. Read drive register 13 (register 13 is the drive's tri-state bus-holding register). Although it is not one of the implemented registers on the Massbus, it is not flagged as illegal when it is read. Since this register was set to 0 with the Massbus initialization, and since the program sees the complement of the data in the holding register, 1s should be seen at the control bus. The problem could be that MASINIT is not working. If it is not, replace CS,CBI. This is also difficult because the error may be due to logic on CS,DS dragging down the bus.
5	VERIFY THAT DRIVE TYPE IS VALID FOR DEVICE This test reads the drive type register to verify that it is returning a legal value for this drive. The RM03 has the following legal device types: 1. 020024 single-ported RM03 2. 024024 dual-ported RM03
6	WRITING 1S OVER THE CONTROL BUS The holding register (register 13) is written with 1s and then read-back. If the write was successful, the data read back should be 0s, since reading register 13 returns the complement of its contents. A stuck GO bit in this test will cause the RMR. However, RMRs do not inhibit the loading of the holding register.
7	WRITING 0S OVER THE CONTROL BUS The following procedure is used to verify that 0s can be written over the control bus. 1. Write 1s to the holding register (worked in previous test). 2. Write 0s to the holding register. 3. Read the holding register and verify that the data is all 1s. This is true because reading the holding register (register 13) returns the complement of the data it contains. This test has the property that an RMR caused by a stuck GO bit will not inhibit the writing of the holding register.

Table 2 DSRMA Test Summary (Cont)

Test	Description
10	VERIFY THAT RHCLR RESETS HOLDING REGISTER The following sequence is used to verify that the holding register clears. 1. Issue a MSTCLR to initiate the system. 2. Write 1s to a writable register (register 12). 3. Issue a MSTCLR to clear the holding register (all bits). 4. Read the holding register and verify that it is correct. Data read should be all 1s since reading the holding register returns the complement of its contents.
11	WRITE A FLOATING 1 DOWN THE CONTROL BUS This test writes a 1 down the control bus using the holding register. **NOTE** **Writing a floating 1 pattern results in a floating 0s pattern read back because of the holding register's inverting property.**
12	WRITE FLOATING 0S TO CONTROL BUS BITS C00-C15 This test writes a floating 0 down the control bus using register 13 (holding register). **NOTE** **Writing a floating 0 pattern results in a floating 1s pattern being read back because of the inversion property of the holding register.**
13	TEST THAT PARITY NETWORK CAN GENERATE A 1 The following sequence is used to verify that the drive can generate a 1 for control bus parity. 1. RHCLR to set the holding register to 0. 2. Read the holding register and verify that a 1 comes back in control bus parity position. Data should be 177777 because of the inversion property of the holding register.
14	TEST THAT PARITY NETWORK CAN GENERATE A 0 The following sequence is used to verify that the drive can generate a 0 for parity. 1. RHCLR to clear errors and holding register 2. Write the holding register with 177776 3. Read the holding register and verify that the parity bit is a 0. The data from the holding register should be 000001 because of its inverting property.
15	PATTERN TEST PARITY NETWORK CHIPS The operation of the parity generator can be verified completely by a set of floating 1s and 0s written into and read from the holding register. The holding register is read and parity is checked after each read. Remember that reading the holding register returns the complement of what was written into it
16	VERIFY CONTROL REGISTER GUARANTEED BITS Read the drive's control register (register 00) and verify that the bits, normally guaranteed to be 1 and 0 are correct. The expected register contents should be C11 = 1, C12 = 0 1. IF port MUX, REGISTER SEL bits, DVA flip-flop fault 2. CBI bad register select line 3. CS, DS some other register gating its contents onto the bus
17	VERIFY THAT GO IS NOT STUCK SET Issue a MSTCLR and then read verify that the GO bit (register 00 C00) is 0

Table 2 DSRMA Test Summary (Cont)

Test	Description
20	DATA TEST FUNCTION BITS WITHOUT SETTING GO The following sequence is used. 1. RHCLR to clear go bit. 2. Write a pattern from table to register 00. 3. Read register 00 and verify that pattern is correct (including the go bit). (IF) IF GO bit is set, some bit is probably cross-coupled to it. If bits fail to be set, there is probably faulty control register logic. A remote possibility exists that some other register is dragging down the bus.
21	PARTIAL TEST OF REGISTER SELECT AND DECODE This test verifies that no register select bits are stuck at 0. It uses the following algorithm which essentially floats a 1 down the register select lines. 1. Let N = 0. 2. RHCLR to INIT. 3. Write 00 to the control register. 4. Write a 70 to register (2**N). 5. Read the control register and verify that it is still 00. If not, there is a register select problem. 6. Repeat from Step 2 for all Ns up to and including 4.
22	SELECT REGISTER 15 FOR FIRST TIME Test is actually looking for a register decode fault by reading register 15 and checking that there are 0s in C00-C02, C04-C06, C08-C09. These are guaranteed 0 for this register.
23	CHECK FOR PERSISTENT REGISTER 15 ERRORS The following is done. 1. RHCLR should clear register 15. 2. Write register 15 with 177777 (should also clear it). 3. Read verify register 15 and verify that no persistent errors occurred. (All bits should be zero). Faults are easily traced back from the error flip-flop.
24	VERIFY OPERATION OF REGISTER 15 IMPLEMENTED BITS Three subtests are required. 1. RHCLR/write 1s/read/verify that data are 1s. 2. RHCLR/write 1s/write 0s/read/verify that data are 0s. 3. RHCLR/write 1s/RHCLR/read/verify that data are 0s. The implemented bits are: C03, C07, C09-C14.
25	TEST SINGULARITY OF REGISTER 15 IMPLEMENTED BITS This test verifies that no bits in register 15 interfere with one another. The following sequence is used. 1. RHCLR to clear errors. 2. Write a pattern from a table to register 15. 3. Read register 15 and verify that pattern is correct.
26	ADDRESS REGISTER 12 FOR THE FIRST TIME This test is looking for register 12 selection problem and does it in the following way. 1. RHCLR to clear errors. 2. Write 177777 to register 12. 3. Read register 12 and verify that C12-C15 are 0. Suspect a register select or decode fault.

Table 2 DSRMA Test Summary (Cont)

Test	Description
27	WRITE 1S AND 0S TO C00-C09 OF REGISTER 12 Subtest A 1. RHCLR to initialize. 2. Write 1777 to register 12. 3. Read verify that C00-C09 are 1. Subtest B 1. RHCLR to initialize. 2. Write 1777 to register 12 (sets bits). 3. Write 0s to register 12. 4. Read/verify that C00-C09 of register 12 are 0.
30	FLOATING PATTERNS TO REGISTER 12 This test verifies that no bits in register 12 interfere with one another by floating 1s and 0s down the implemented bits of register 12 (C00-C09).
31	ADDRESS REGISTER 11 FOR THE FIRST TIME The following sequence is used to gain some confidence that we have selected the correct register. 1. RHCLR to clear errors. 2. Write register 11 with 177777. 3. Read register 11 and verify that C00-C06, C08-C09, C13-C15 are 0. The bits tested for 0 are not implemented, so a failure in this test probably indicates a register-select or register-decode failure.
32	WRITE 1S AND 0S TO IMPLEMENTED BITS OF C07, C10-C12 This test verifies that all implemented bits of register 11 can be set to 1 and 0. The implemented bits are C07, C10-C12.
33	FLOAT PATTERNS TO REGISTER 11, IMPLEMENTED BITS This test verifies that none of the implemented bits of register 11 interfere with one another. The implemented bits are C07, C10-C12.
34	VERIFY THAT RHCLR DOES NOT AFFECT REGISTER 11 Write 1s to register 11 implemented bits, then issue RHCLR and verify that no register 11 bits were cleared.
35	0S AND 1S TEST OF THE DA REGISTER Four subtests are implemented as follows. Subtest A 1. RHCLR 2. Write 177777 to register 05. 3. Read/verify that all bits are set. Subtest B 1. RHCLR 2. Write 177777 to register 05. 3. RHCLR (RHCLR should not clear DA). 4. Read/verify that all bits are set. Subtest C 1. RHCLR 2. Write 177777 to register 05. 3. Write 0s to register 05. 4. Read/verify that all bits are 0. Subtest D 1. RHCLR 2. Write 0s to register 05. 3. Write 1s to register 05. 4. Read/verify that all bits are set.
36	FLOAT 0S AND 1S TO C00-C15 OF REGISTER 05 This test verifies that none of the bits in register 05 interfere with one another.
37	Test has been deleted.
40	Test has been deleted.

Table 2 DSRMA Test Summary (Cont)

Test	Description
41	VERIFY THAT DRIVE NOT GENERATING PERSISTENT CBPE RHCLR and then read/verify that C03 of register 02 is 0.
42	VERIFY THAT WRITING A REGISTER DOES NOT CAUSE PARITY ERROR Two subtests are run in the following sequence. Subtest A 1. RHCLR 2. Read/verify that (PAR) C03 register 2 is 0. 3. Write 0s to register 05 (parity bit sent will be a 1). 4. Read register 02 and verify that no parity error was detected. Subtest B 1. RHCLR 2. Read/verify that (PAR) C03 register 02 is a 0. 3. Write a 1 to C00 of register 05 (Parity bit sent will be a 0). 4. Read register 02 and verify that no parity error was detected.
43	ADDITIONAL WRITE PARITY TESTS This tests that parity check logic within the RM03 is operating properly for a number of patterns.
44	DATA TEST C03 OF REGISTER 02 Three subtests are used to verify that C03 of register 02 is operational. 1. RHCLR/write 1/read-verify that C03 set. 2. RHCLR/write 1/WRITE 0/read-verify that C03 clear. 3. RHCLR/write 1/CHCLR/read-verify that C03 clear.
45	TEST REGISTER 02 (ER1) FOR STUCK 1S This test verifies that following an RHCLR and a write of 0s to register 02, all bits are 0. Stuck bits are easily traced back from the set error flip-flop.
46	TEST REGISTER 02 FOR 1S, 0S, RESET FUNCTIONS The following 4 subtests are required. 1. RHCLR/write 1s/read VERIFY. 2. RHCLR/write 1s/RHCLR/read/verify. 3. RHCLR/write 1s/write 0s/read verify. 4. RHCLR/write 1s/read verify.
47	FLOAT 1S AND 0S TO C00-C15 OF REGISTER 02 This test guarantees that there is no cross-coupling between any of the bits C00-C15 in register 02.
50	VERIFY THAT DEVICE CAN DETECT PARITY ERROR This test writes 0s to the control register but does it with incorrect parity. The program then reads error register 1 and verifies that the device detected the parity error.
51	VERIFY THAT ILR NOT SET FOR REGISTERS 00-17 This test reads each of the legal registers in the drive and after each read, verifies that an illegal register condition was not detected (C01 register 02).
52	VERIFY ILR SETS FOR REGISTERS 20-37 This test reads each of the illegal registers in the drive and after each read, verifies that an illegal register condition was detected (C01 register 02).
53	TEST FOR A PERSISTENT COMP ERROR Verify that ERROR C14 register 01 is not set after error registers 1 and 2 are cleared.
54	VERIFY COMPOSITE ERROR SET PATHS This test verifies that the setting of any error bit in the device will cause ERROR C14 register 01 to be set.

Table 2 DSRMA Test Summary (Cont)

Test	Description
55	VERIFY THAT SYSTEM IS NOT STUCK IN M-MODE Verify that, following an RHCLR, the m-mode bit is clear (register 03 C00).
56	DATA TEST C00 REGISTER 03 The M-Mode bit is data-tested using three subtests. 1. RCHCLR/WRITE 1/READ-VERIFY. 2. RHCLR/WRITE 1/WRITE 0/READ-VERIFY. 3. RHCLR/WRITE 1/RHCLR/READ-VERIFY.
57	TEST FOR WRITABLE REGISTER INTERFERENCE This test guarantees that no writable register interferes with any other register because of a register select decoding problem. The following algorithm is used. 1. Pick a writable register (call it X). 2. Write appropriate bits of register X (from table). 3. Pick another register (call it Y). 4. Write 0s to all bits of register Y. 5. Read register X and ensure that it did not change. If it did, a register addressing problem exists between X and Y and the test will loop on this pair. 6. Do steps 2 through 6 letting Y vary 00-37 but never equal to X.
60	DRIVE CLEAR WITHOUT THE GO BIT Issue a drive clear command without setting the GO bit and verify that this operation did not cause an error to occur. If this test fails, there is probably something wrong with the command decoder or the microsequencer logic.
61	VERIFY THAT DRIVE CLEAR IS INOPERATIVE WITHOUT GO. This test essentially verifies that the microsequencer is sensitive to the state of the GO bit.
62	VERIFY THAT DRIVE CLEAR IS OPERATIONAL There are two possible failure modes. 1. If no ER1 bits cleared, we have U SEQ fault, GO will not set, improper command decode as possible faults. 2. If some ER1 bits clear, it is safe to assume that drive clear has partially worked but caused an error to occur.
63	USE DRIVE CLEAR TO VERIFY DEVICE SELECT LOGIC This test is actually the complement of test 1. The test verifies that the drive being tested responds to no other device code but its own. The following algorithm is used. 1. Call drive being tested drive X. 2. Call some other, drive Y. 3. RHCLR issued to drive X. 4. Write 1s to register 02 in drive X. 5. Issue a drive clear to drive Y. 6. Read register 02 in drive X and verify that it has not been modified. 7. Repeat steps 3 through 6 for all possible Ys not equal to X. Drive being tested is responding to some other device code.
64	PARTIAL TEST OF NO OP COMMAND Issue a NO OP command and then verify: 1. It did not cause an error. 2. The GO bit cleared after command execution.
65	VERIFY OPERATION OF DEBUG CLOCK FUNCTION Verify that debug enable in the MAINT REGISTER (C14 register 03) functions correctly and that debug clock also functions correctly (C15 register 03).

Table 2 DSRMA Test Summary (Cont)

Test	Description
66	VERIFY OPERATION OF DEBEND (C14 REGISTER 03) This test verifies that the debug enable bit is fully operational. Failure is an indication that M-Mode bit is not interlocked properly with debug clock bit in the maintenance register logic.
67	CHECK PARTIAL OPERATION OF GO AND DRY Verify that GO C00 register 00 and DRY C07 register 01 are complements when GO = 0.
70	CHECK PARTIAL OPERATION OF GO AND DRY Verify that DRY (C07 register 01) and GO (C00 register 00) are complements when GO is set.
71	VERIFY OPERATION OF PIP C13 REGISTER 01 Using the maintenance register, the functionality of PIP is checked. Possible faults: Maintenance logic malfunction associated with on-cyl bit; (CS) Status logic malfunction associated with PIP (IF).
72	CHECK FUNCTIONALITY OF MOL FROM MAINT REGISTER Using the maintenance register feature, the MOL bit in the status register (C12 register 01) is tested for proper operation. Possible faults: Status logic fault associated with MOL (IF); Maintenance logic fault associated with (UN-RDY) C09 register 03 (CS).
73	VERIFY FUNCTIONALITY OF WRL STATUS THROUGH MAINT REGISTER Using the maintenance register feature, verify that WRL C11 register 01 is functioning properly. Possible faults: Faulty logic associated with status logic for WRL (IF) Faulty logic associated with maintenance logic for W-PROT (CS).
74	SET ER2 DEV CK AND SKI THROUGH MAINT REGISTER Using maintenance register features, test the operation of DEV CK C07 register 15 and SKI C14 register 15. Possible faults: Faulty maintenance register logic associated with bits C07, C06 (CS); Faulty error reg logic associated with register 15 C14, C07 (IF).
75	VERIFY THAT DEV CK CAUSES UNSAFE Using maintenance register features, verify that the occurrence of DEVCK C07 register 15 causes UNSAFE C14 register 02 to set.
76	A TEST OF DIAGNOSTIC MODE INTERLOCK This test verifies proper reset operation of flip-flops in the maintenance register (register 03). It is designed to prove that when the drive is taken out of maintenance mode, C03, C06-C09 are cleared. All faults would be local to maintenance register. In particular: If PIP = 0 Fault is C08 register 3 ON CYL If MOL = 1 Fault is C09 register 3 UN RDY If WRL = 1 Fault is C03 register 3 W PROT If SKI = 1 Fault is C07 register 3 SK ERR If DEV CK = 1 Fault is C06 register 3 DR FLT
77	VERIFY THAT VV C06 REGISTER 01 NOT STUCK AT 1

Table 2 DSRMA Test Summary (Cont)

Test	Description
100	SET VV WITH A PACK ACKNOWLEDGE COMMAND This test verifies that VV (C06 register 01) can be set with a pack acknowledge command. Since a pack acknowledge is being used for the first time in this test, the diagnostic also verifies that issuing the command does not cause an error to occur. If an INVCMD error occurs, problem is on (CS). If some other error condition occurs, scope back from error flip-flop to isolate to the module.
101	Test has been deleted.
102	SET VV WITH A READIN PRESET COMMAND This test verifies that VV (C06 register 01) can be set with a readin preset command. Since a readin preset is being used for the first time in this test, the diagnostic also verifies that a composite error did not occur. If an INVCMD error occurs, problem is on (CS). If some other error occurs, scope back from error flip-flop to isolate to the module.
103	VERIFY THAT DPR IS A 1 (C08, REGISTER 01) Read status register and verify that C08 is a 1. It is hardwired to 1 in this device.
104	VERIFY THAT WE ARE NOT PROGRAMMABLE Toggle MOL via the maintenance register and then verify that (C09 register 01) PGM is 0. The diagnostic should be locked on one of the ports because the operator was told to do so at start-up time (during system configuration).
105	VERIFY THAT READIN PRESET CLEARS PROPER THINGS
106	Test has been deleted.
107	GO INTO OFFSET MODE USING OFFSET CMD This test places the drive in offset mode by issuing an offset command (15). The offset mode flip-flop OFS-MD is C00 register 01. Since this is the first time the offset command is used, the diagnostic checks for the occurrence of a composite error after the command is issued. Trace a composite error failure back from the appropriate error latch to isolate the fault to a module.
110	CLEAR OFS MD BY WRITING INTO REGISTER 12 This checks out one of the clear paths to the offset mode flip-flop.
111	FORCE AN INVCMD ERROR C12 REGISTER 15 This error condition is forced by issuing an offset command while VV = 0.
112	VERIFY THAT GO DOES NOT SET WITH A PARITY ERROR This test verifies that the GO bit will not set if a parity error is detected when the control register is written.
113	VERIFY THAT REGISTER 03, REGISTER 04 CAN BE WRITTEN WHILE GO = 1 Writing the maintenance register (register 03) or attention summary register (register 04) while the GO bit is set should not cause RMR to occur. Problem is on IF and is either a register decode fault or an error in the PLA that causes RMR.
114	CAUSE AN RMR TO OCCUR This test causes an RMR (C02 register 02) to occur by writing into the DA register (register 05) while GO = 1.

Table 2 DSRMA Test Summary (Cont)

Test	Description
115	ISSUE A SEEK WITH U SEQUENCER STOPPED This test issues a seek command to verify that GO is set, and no errors were caused by the operation. If an invalid command error occurs, the fault is on CS. Any other error condition will require scoping back from the error flip-flop to isolate the faulty module.
116	CAUSE IAE WITH A SEEK TO AN ILLEGAL CYL Test is designed to cause an IAE (C10 register 02) by issuing a maintenance mode seek command to an illegal cylinder.
117	VERIFY ADDRESS DECODE FOR LEGAL DISK ADDRESSES This test uses a maintenance mode seek command to verify that all of the legal addresses decode as legal. This test is repeated while varying SECT, TRK, CYL through their legal values. SECT (0-35) TRK (0-4) CYL (0-1466)
120	VERIFY ADDRESS DECODE FOR ILLEGAL ADDRESSES This test uses a maintenance mode seek command to verify that the disk address decode logic does not allow any illegal addresses to be referenced without flagging IAE (C10 register 02). This test is repeated while varying SECT, TRK, CYL through their illegal values one at a time. SECT (36-377) TRK (5-377) CYL (1467-1777)
121	VERIFY THAT ATTN IS NOT STUCK SET
122	CAUSE ATTN TO SET WITH COMPOSITE ERROR
123	SET AND CLEAR THE ATTN FLIP-FLOP
124	VERIFY THE POSITION OF THE PSEUDO ATTN BIT This test is designed to verify that the pseudo attention bit for the drive being tested is placed on the bus in one position only, and that the position is correct. If step 6 of this test fails, the following is possible. 1. The drive under test is indicating its position in either wrong or multiple bus positions 2. Some other device on the bus has an attention bit that was not dropped in response to the RHCLR.
125	CLEAR ATTN BY WRITING PSUEDO-ATTN BIT FOR THE DRIVE Possible faults: Position decode failure (IF); Incorrect device select receivers from drive (CS); Faulty device select switch (unit select) (drive).
126	ADDITIONAL TEST OF PSEUDO-ATTN POSITION DECODE This test is designed to verify that the drive being tested does not respond to writes of other drives' pseudo-bits.
127	CLEAR ATTN USING MBA CLEAR PATH
130	CLEAR ATTN BY WRITING THE GO BIT Verify that ATTN can be cleared by setting the GO bit provided. No error condition exists.
131	TEST OF ANOTHER ATTN SET PATH Verify that ATTN is set if a register is written while a composite error exists.

Table 2 DSRMA Test Summary (Cont)

Test	Description
132	SET ATTN BY TOGGLING MOL This test toggles MOL in maintenance mode via (C09 register 03) UN-RDY. This should cause ATTN to set.
133	VERIFY THAT WRITING GO WITH ERROR SETS ATTN This test is very similar to test 131 but is not actually redundant because of the implementation scheme.
134	SETTING GO WITHOUT ERROR SHOULD NOT SET ATTN
135	VERIFY ATTN AS SEEN AT THE CONTROLLER With ATTN logic in the drive working, this test merely verifies that its state is properly observed at the controller. If a failure occurs, status registers from all drives on the bus are snapshot and dumped because there is the possibility that some drive other than the one being tested is sending ATTN to the RH. If the status register KSDUMP fails to show up the drive asserting ATTN, it will be necessary to power down the drives on the bus, one at a time, until the faulty drive is identified.
136	VERIFY OPERATION OF THE UNIT-SELECT PLUG This test is designed to verify that correct things happen if the unit-select plug is removed and reinserted. It is not necessary to check for the presence of ATTN. This is known to work.
137	VERIFY OPERATION OF WRITE-PROTECT SWITCH This test requires operator intervention. It asks the operator to operate the write-protect switch and verifies the switch status through the status register WRL C11 register 01
140	CHECK FOR PROPER STATE OF DRQ C11 REGISTER 06 This tests the drive type register for the presence of DRQ and then, after reporting its state, asks the operator if the result is correct.
141	VERIFY OPERATION OF START-STOP SWITCH This test uses operator dialogue in conjunction with reading drive registers to determine whether or not the drive can be cycled up and down using the start-stop switch. After each operation, the drive is checked and proper state of MOL and any errors except SKI. However, SKI may occur.
142	FUNCTIONAL TEST OF PORT-SELECT LOGIC AND SWITCH This is a group of eight subtests that exercise the port-select logic and switch, through all of their logical combinations. Each of the subtests is described at the start of the test. The subtests all require manual intervention and are self-documenting as they are executed. Faults would be localized to IF, or port-select switch
143	VERIFY OPERATION OF POWER-UP SEQUENCE This verifies that when the drive is powered up, ATTN is set and an AC LOW error condition exists. AC LOW is indicated by UNSAFE = 1 and DEV CK = 0 (this is because of implementation). The operator is instructed to power up the drive and the diagnostic will then check for errors.
144	A BASIC LOOKAHEAD REGISTER SELECT CHECK The diagnostic reads register 07 and verifies that C00-C05, C11-C15 are 0. These bits are not implemented, so if something is read back, a register selection problem is at fault.
145	VERIFY THAT M MODE STOPS US FROM SEEING LOOKAHEAD This test is now obsoleted because of hardware implementation

Table 2 DSRMA Test Summary (Cont)

Test	Description
146	VERIFY THAT LOOKAHEAD REGISTER IS TRANSITIONING This test cycles the drive up and reads the lookahead register at least 50 times looking for a transition of any type
147	VERIFY THAT ALL LEGAL SECTORS CAN BE READ With the drive on-line and in 18-bit mode, searches for each legal sector (0-35) by reading the lookahead register at least 1000 times for each one.
150	VERIFY THAT LOOKAHEAD IS IN 18-BIT FORMAT This test verifies that the lookahead register is in 18-bit mode when it is supposed to be. This is accomplished by placing the drive in 18-bit mode and verifying that no sector greater than 35 is detected by reading the lookahead register 400 times.
151	TEST OF THE OFFSET COMMAND This test issues an offset command while in maintenance mode and verifies proper operation. If a composite error occurs, trace back from the error flip-flop to isolate the failing module
152	TEST OF RETURN TO CENTERLINE COMMAND This test issues a return to centerline command in maintenance mode and verifies proper operation. If a composite error occurs, trace back from the error flip-flop to isolate the failing module.
153	VERIFY THAT MICRO-SEQ HANDLES ILF CODES CORRECTLY This test issues each of the possible illegal commands and after each, reads status and verifies proper operation. If any error but ILF occurs, trace back from error flip-flop to isolate to failing module.
154	TEST MICRO-SEQ RESPONSE TO FUNCTION WITHOUT GO BIT This test issues all functions 00-76 without the GO bit to ensure proper device decode and response. If an error occurs, scope back from error flip-flop to isolate problem to the faulty module.
155	ADDITIONAL MICROSEQUENCER CHECKS This test verifies proper microsequencer response to the pack-acknowledge and release commands when issued in maintenance mode.
156	TEST FOR BUS BITS STUCK AT 1
157	RHSTS1 FLOAT 1s AND 0s TO C00-C15 OF RHWC. This test guarantees that there is no cross coupling between any of the bits C00-C15 in RHWC
160	RHSTS2 FLOAT 1s AND 0s TO C00-C45 OF RHBA This test guarantees that there is no cross-coupling between any of the bits C00-C15 in RHBA.
161	RHSTS3 FLOAT 1s AND 0s TO C00-C15 OF RHDB This test guarantees that there is no cross-coupling between any of the bits C00-C15 in RHDB.
162	RHSTS4 SILO TEST 1 Test the silo buffer in the RH11 controller. A read is attempted from an empty silo. Data late (DLT), transfer error (TRE) and special condition (sc) should set. Then loading a 1 into TRE should clear DLT, TRE and SC.
163	RHSTS5 SILO TEST 2 Test the IR and OR bits of RHCS2. At the beginning IR should be set and OR reset. First an all 0s word is loaded into the silo via RHDB, and then an all 1s word is loaded into the silo. OR should become set in 30 microseconds. Again, the time is not checked but OR should be set. The output from the silo should be an all 0 word and then an all 1 word.

Table 2 DSRMA Test Summary (Cont)

Test	Description
164	RHSTS6 SILO TEST 3 Tests the silo buffer by filling it with a count of 0 to 65 and then checking if IR is down and OR is high, and reading and verifying the silo output.
166	FULL SPEED SEEK TO CYL0 This test verifies that a full speed seek to CYL0 operates without error.
167-174	Not implemented
175	VERIFY THAT FULL SPEED RECAL WORKS WITHOUT ERROR
175-206	Not implemented
207	FULL SPEED SEARCH A full speed search is issued to CYL0, SURF0, SECTOR0. COMPOSITE error is then checked.
216	CHECK CYL ADR, AND TAG BUS PATH Uses maintenance mode to check cylinder address generation and its application on the drive bus.
217	CAUSE IAE WITH SEARCH TO ILLEGAL CYLINDER
220	CHECKS LAST SECTOR DECODE
222	VERIFIES THAT SEEKS TO LEGAL CYLINDER DO NOT CAUSE IAE Checks that issuing maintenance mode seek commands to CYL0 with the legal track/sector address combinations do not cause IAE. Test also verifies that head addresses do not go out on the bus bits by accident.
223	VERIFIES THAT SEEKS TO ILLEGAL ADDRESSES CAUSE IAE Issues an M Mode seek to all possible legal addresses and verifies that IAE sets.
224	SECTOR COMPARATOR TEST Using a full speed search, test verifies that search gets the correct sector.
225	SEEK TO PRIME CYLINDERS FOLLOWED BY A RECAL This test is functional. It is designed primarily to check lines on the tag bus. Obvious drive errors may be detected by this test.
226	VERIFY SEEKS FOR ALL POSSIBLE PLUS AND MINUS DIFFERENCES Test is designed to check difference in calculation logic and velocity logic in drive.
227	VERIFY OPERATION OF OCC FROM ROM
230	VERIFY THAT DATA COMMANDS DO NOT CAUSE ILF
231	VERIFY THAT RUN IS COMING BACK IN RESPONSE TO OCC Test issues a read command in maintenance mode and verifies that run go, as seen through the maintenance register, sets within 20 ms.
232	CAUSE OPI WITH RUN TIMEOUT
233	RESETTING RUN TIMER WITH RESET GO
234	FUNCTIONAL TEST THE WRITE PROTECT CIRCUITRY Three subtests are done in DIAG mode. Control of WRT PROT is done through the maintenance register. The three subtests check the following conditions. 1. WRT and WRL produce WLE. 2. Not WRT and WRL produce not WLE. 3. WRT and not WRL produce not WLE.
235	VERIFY OPERATION OF MAINT EBL REGISTER 03 C13
236	CHECK INCREMENTATION OF DA AND DCY WITH EB1
237	TEST OPERATION OF LBT LOGIC WITH 3 SUBTESTS
240	TESTS LBT ADDRESS DECODE LOGIC

Table 2 DSRMA Test Summary (Cont)

Test	Description
241	CAUSE AN AOE CONDITION
242	TEST OF AOE LOGIC WITHOUT RUN LINE
243	TEST OF AOE WITH INCORRECT ADDRESS
244	CLEAR OFFSET WITH A RECAL COMMAND
245	VERIFY THAT WRT DATA AND WRT HEADER COMMANDS CLEAR OFFSET
246	VERIFY THAT READ COMMANDS DO NOT CLEAR OFFSET
247	CLEAR OFFSET MODE WITH IMPLIED SEEK
274	CHECK BASIC TIMING USING PROM STROBE This test verifies that PROM strobe comes up at the right time and stays up for 4-bit clocks.
275	TEST SOME WRITE DATA LOGIC This test verifies that the hardware generates 0s during the time between the rising of PROM strobe and the rising edge of write clock. This allows the hardware to get in sync. The data is checked through bit C03 of the maintenance register.
276	CHECK TIMING OF A FORMAT OPERATION This test uses maintenance mode to stop the microsequencer clock and to step the device through a write header and data operation. Depending on the position within the sector, either 16- or 18-bit clocks will be issued when the ROMCLK routine is called. Only in the data area will 18-bit mode be used. All else is done in 16-bit mode.
277	VERIFY OPERATION OF WRT HD AND DAT UP TO DATA FIELD This is a series of six subtests that verify operation of the write header and data command from the pre-header gap to the start of the data field. The test completes at the start of the data field in order to minimize the size of the scope loop.
300	CHECK FOR STUCK 1S AND 0S ON DATA BUS This test issues a maintenance mode write header and data command and checks that the data bits on the Massbus going from controller to drive are not stuck at 1s or 0s. Look for faulty interface, data path and shift register problems.
301	CHECK OF WRITE DATA PATH USING FLOATING PATTERNS This test verifies that there are no bits cross-coupled on write data bus by testing data integrity with floating patterns. 1. Start a maintenance mode transfer of 1 sector. 2. Issue 432 clocks to step to data area. 3. Issue 36 clocks to step over first PDP-10 word. 4. Start clocking PDP-10 half words out of memory (floating patterns), each time verifying that data was correct. 5. Repeat step 4 until 18 PDP-10 words have been tested. 6. Abort the transfer.
302	TRANSFER STANDARD DATA PATTERNS CHECKING FOR PARITY This test passes 50 (10-bit) data patterns across the Massbus and checks for parity errors after each transfer. The error report contains: 1. The number of of 18-bit transfers that failed. 2. Data that was sent over the Massbus. 3. Data that was read out of the maintenance register.

Table 2 DSRMA Test Summary (Cont)

Test	Description
303	CAUSES A DRIVE TIMING ERROR DTE Verifies that the drive can detect a drive timing error. The error is forced in the following way. 1. Start a one sector maintenance mode write header and data command. 2. Cause a sector pulse (which causes a drive timing error). 3. Read ER1 and verify that DTE is set.
304	VERIFY READ HEADER SYNC BYTE DETECTION Verify that for a read header and data command, the sync byte detector works properly.
305	VERIFY THE READ DATA PATH INTEGRITY Supply necessary data patterns via a maintenance mode read header and data operation and check data that appears in memory against the expected data (18 bits at a time). Data checking is not performed until the data field is reached. Once at the data field, the following four subtests are performed. 1. One Massbus transfer of 0s. 2. One Massbus transfer of 1s. 3. Thirty-four Massbus transfers of floating patterns. 4. Six Massbus transfers of 0s and parity patterns.
306	CHECK DRIVE PARITY NETWORK Send various data patterns from drive to controller over the Massbus data path using a maintenance mode read header and data operation. The following two subtests are performed. 1. Sends 1s and 0s. Possible faults include parity network and parity transmit faults. 2. Supply other data patterns to verify parity generation logic in drive; faults only include parity generator.
307	CHECK OF WRITE DATA TIMING This test uses maintenance mode to stop the microseqencer clock and to step the device through a write data operation. Depending on the position within the sector, either 16- or 18-bit clocks will be issued when the ROMCLK routine is called. Only in the data area will 18-bit mode be used. All else is done in 16-bit mode.
310	CHECK OF READ DATA TIMING This test uses maintenance mode to stop the microsequencer clock and to step the device through a read data operation. Depending on the position within the sector, either 16- or 18-bit clocks will be issued when the ROMCLK routine is called. Only in the data area will 18-bit mode be used. All else is done in 16-bit mode.
311	VERIFY THAT ECC GENERATION WORKS CORRECTLY In 18-bit mode, issues a write header and data command, issue enough clocks to step up to the ECC field, clock out and read the next 32 bits of ECC data and verify that it is correct.
312	CHECK HEADER LOGIC WHILE SUPPLYING CORRECT DATA This test starts a one sector maintenance mode read header and data operation. It clocks through the header area supplying correct data and verifies the following. 1. Data area is up. 2. No HCE, HCRC or FMT errors were detected. 3. Header data integrity is ignored at this time.

Table 2 DSRMA Test Summary (Cont)

Test	Description
313	TEST HEADER CRC LOGIC Issues a maintenance mode read operation and supplies wrong CRC data. This should cause HCRC to be detected. 1. Send 224 0s for header gap. 2. Send 8-bit sync byte. 3. Send 32 0s for header. 4. Send 8 0s for first half of header CRC. 5. Send 8 1s (wrong data) for remainder of header CRC. 6. Send 8 0s to move slightly into gap. 7. Read status and verify that HCRC is set.
314	TEST HEADER COMPARE LOGIC This test verifies that the drive can detect header data errors. This is done by supplying the header data in an M-mode read header and data operation. The diagnostic supplies the wrong cylinder information. This results in a HCE and a HCRC error.
315	TEST OF FER ERROR DETECTION LOGIC This test verifies that a format error can be detected. It starts a maintenance mode read header and data operation in 18-bit mode, then supplies the header data with the 16-bit FMT bit set. This should cause a format error (FER). HCRC and HCE are incidental to this test.
316	TEST OPERATION OF HCI LOGIC Test initiates a read header and data command in maintenance mode (30-sector mode). With the header compare inhibit bit (HCI) set, the data supplied by the diagnostic supplies incorrect header, format, CRC information. After completing transfer through the header area, status is checked. HCE, FER, HCRC should all be 0.
317	FORCE A DCK USING READ HDR AND DATA CMD Test starts a 1-sector maintenance mode read header and data operation, with HCI = 1. It clocks through the sector supplying correct data up until the ECC character. This should cause DCK=1 to assert.
320	FULL SPEED WRITE HEADER AND DATA TEST This test issues a full speed write header and data command to the first disk address. This has worked previously in maintenance mode in test no. 277. Disk timing and interface logic are used for the first time.
321	SELECT EACH HEAD FOR THE FIRST TIME This is the same as 320, but each head is used for the write commands.
322	WRITE THEN READ HEADERS AND DATA AT FULL SPEED FROM BLOCK 0 This test checks for errors occurring during a read header and data operation. Data errors cannot be isolated to read faults or write faults. Header errors without data errors indicate DS board failures.
323	WRITE THEN READ IT FULL SPEED FROM ALL HEADS This test checks for errors occurring during a read header and data operation. The test is essentially the same as test 322, except various addresses are used and all heads tested. At completion of this test writing and reading from each head has been successful. During this test it is impossible to determine whether errors occurred during the read or write operation. HCEFER without data errors indicate DS faults.
324	CHECK HEADER DATA AT VARIOUS DISK ADDRESSES This test checks that headers can be written and read from various disk addresses.

Table 2 DSRMA Test Summary (Cont)

Test	Description
325	HEAD SELECTION AND INTERFERENCE TEST This test checks that writing a disk surface does not interfere with headers written on another disk surface.
326	CYLINDER SELECTION AND INTERFERENCE TEST The test checks that writing a header on a disk cylinder does not interfere with headers written on another disk cylinder. The test is similar to test 325, only the cylinder number is the variable instead of the header number.
327	FULL SPEED WRITE DATA COMMAND Test checks the full speed write data command. At this point only full speed write header and data commands have been used. Test is similar to test 320.
330	FULL SPEED READ DATA COMMAND Test checks the full speed read data command. At this point only full speed read header and data commands have been used. Test is similar to test 322, but read data instead of read header and data is used.
331	TEST LAST BLOCK TRANSFERRED STATUS Test checks that a write header and data to CYL 1466, HEAD4, SECT35 (last block) causes LBT to set in the status register.
332	DETECTION OF AOE AT FULL SPEED Test checks that a 2-sector transfer starting at CYL 1466 HEAD4, SECTO35 (last block) causes the addressing overflow error bit to set.
333	FULL SPEED READ WITH OFFSET Test checks that a full speed read with offset and HCI=1 does not cause errors other than DCK.
334	CHECK MAINT MODE ECC CORRECTION This test does a read header and data command and will force a known correctable error condition. The data will be all 0s and the error is forced by insertion of an erroneous 1 in the data stream. The final conditions should be as follows. Data check = 1 ECC hard = 0
335	FORCES AN ECC HARD ERROR This test does a read header and data command and will force a data error larger than the burst length.

GENERAL INFORMATION

Code	DSRMB.SAV
Title	DECSYSTEM-2020 KS10/RH11 - RM03/RP06 - Reliability Diagnostic
Abstract	DSRMB runs on any DECSYSTEM-2020 KS10/RH11, RP06/RM03 system and provides engineering, field service, and production with the following: Full speed data transfers which use most of the drive's features (i.e., ECC correction, implied seeks, spiral transfers, etc.). A random parameter data/mechanical reliability test (FRTEST) to provide a simulation of a drive in a system environment. The ability to format a RP06/RM03 disk pack The ability to loop (or freeze) on a particular sequence of instructions. The capability of simultaneous operations to several drives across a Massbus controller. Dual-port operation of the RP06/RM03s. On-line operation of the program using dump mode I/O to read and write. Special protection is given to guard against accidental data destruction in user mode.
Hardware Required	KS10 mainframe Minimum of 32K of memory One RH11 Massbus controller connected to a UBA (Unibus Adapter). 1 to 8 RP06/RM03 disk drives: 1 RP06/RM03 single port 1 RP06/RM03 dual port Magtape as program-loading device.
Preliminary and Associated Programs	Refer to diagnostic hierarchy (KS10 STD module).
Restrictions	DSRMB does not support simple test routines for use with DDT as was the practice with previous disk reliability diagnostics. The structure of this program dictates that it must be thoroughly understood before it would be possible to code simple write/read loops in DDT. The operator data test has been furnished to support simple write/read loops to a specific disk address.
Notes	1. This program has not been run on many of the possible system configurations! 2. To print on logical device DEV in user mode, first assign the disk as DEV then in the program set switch PNTLPT (switch 4).
Loading and Starting Procedure	Standard (Refer to the KS10 STD module.)
Control Switches	Refer to Table 1.

OPERATIONAL CONTROL

Table 2 is a list of commands available to the operator while the program is running.

Typing a number of a test instead of the actual test name is acceptable and proper test dispatching will take place. The test name will be typed back to the operator before dispatching is executed.

Typing anything except a test number or name defined will cause an error message to be printed.

TEST SUMMARY
Table 3 lists routines that may be selected only by operator dialogue.

Table 4 summarizes the tests performed by DSRMB.

ERROR SUMMARY
Refer to the listing on microfiche.

Table 1 DSRMB Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (Refer to the KS10 STD module.) The following switches are not implemented: 11 (INHPAG), 12 (MODDVC), 15 (CHNN).
18	DELTRK	Pack deletion/protection in effect suspends any operator-selected pack deletion/protection
19	VARIAB	Allow operator to change test parameters
20	ACTSEK	Use implied seeks Force actual seeking instead of implied seeks
21	FREEZE	Allow the operator to execute the last 20 commands repeatedly
22	INHULD	Inhibit drive unloads on a hard error abort
23	ALLADR	Select all disk addresses before changing drive numbers
24	ALLDRV	Select all drives on an MBC before selecting a new MBC
25	ONESEC	Limit transfer size to 1 sector
26	INHRET	Automatic error recovery enabled inhibit error recovery procedure
27	MODUBA	Modify Unibus/UBA addresses
28	SNCON	Use switches to control tests
29		Not used
30-35		Test # (In octal) 00 TOTAL 01 RONLY 02 WONLY 03 OPDTST 04 STEST 05 SEKTST 06 INTTST 07 FRTEST 10 FORMAT 11 SEQGEN 12 ACCEPT 13 PTIME 14 RCTEST

Table 2 DSRMB Command Summary

Command	Description
A	A reports drives available for testing.
H	H causes a help message to be typed.
S	S reports drives selected for testing.
R	R reports program run time, number of disk blocks (sectors) read and written, along with the number of position commands issued and test being run.

Table 3 DSRMB Test Summary

Routine	Description
MAPOUT	This routine reports the pack BAT block contents and allows the operator to update the pack BAT block (no update is possible in user mode). The operator may also use this service to eliminate known soft error spots from testing or to include/exclude any hard error spots on the pack from testing.
DELETE	This service routine allows the operator to eliminate areas of a pack from testing or to use only specific areas of the pack for testing. This service is called from MAPOUT if the operator desires to remove soft spots from testing.
PAKINT	This script will automatically format, map and create the BAT blocks for the selected RP06/RM03 disk drives. The script can be run in exec mode or in user mode under a TOPS-10 monitor. The purpose of this script is to limit the dialogue necessary to complete this operation, and avoid user errors. An additional feature has been added to limit the program start-up dialogue in user mode. A question is asked (SHORT STARTUP DIALOG Y OR N ?) If a Y is typed the program will be initialized automatically.
CONFIG	This routine allows the operator to re-map the Massbus and report the results. Also enables the operator to add or delete drives from the test list without retyping the entire list. Write-enabling a drive in user mode is accomplished with this service (in user mode, all drives are initially write-protected at run time). Typing H to WHAT DRIVE will give the operator a list of acceptable commands to CONFIG.
H	Typing H to WHAT TEST will give the operator the complete list.

Table 4 DSRMB Error Summary

Test	No.	Description
TOTAL	00	This is the order of test execution SESTST RONLY WONLY STEST INTTST PTIME RCTEST FRTEST
RONLY	01	RONLY - Read-Only Test RONLY reads all areas of a disk pack. No data comparison can be made on the data read because the type of data stored on the pack is not known. No write data transfers will be issued during this test (i.e., no damage to the data on the pack should occur). Any detected data errors will be reported and an error recovery will be issued. Errors corrected by using the ECC logic will be reported with both the original data and the correct data given.

Table 4 DSRMB Error Summary (Cont)

Test	No.	Description
WONLY	02	WONLY - Write-Only Test This is a data transfer write only test. It will write data on all unprotected disk pack areas and destroy any previous data on the pack. Either hardware or software (by an operator command) write protection will inhibit a drive from being selected to run this test. Any type of data that happens to reside in memory will be used for this test - the operator requests a specific data pattern to be used. Setting console switch VARIAB will enable the operator to specify the data pattern to be used.
OPDTST	03	OPDTST - Operator-Selectable Data Test This test allows the operator to specify the type of data to be used for writing and reading data "to/from the drive pack. OPDTST will select all unprotected pack areas for testing. Data will be written, read, and verified for correctness on all selectable areas on all drives. OPRVT - Operator-Variables Data Test This section allows the operator to select transfer parameters such as disk address, data pattern to be used, pass iteration count, and data transfer size. Write once and read 1000 times Simply write and read Write only Read only (no data comparison) Get the seek parameters OPRDRV - Routine to set up the selected drive parameters
STEST	04	STEST - Surface Test STEST selects all unprotected pack areas and uses several different data patterns to verify that the disk pack can hold data of different types reliably. The data is written, read, and verified for correctness. Data patterns used for this test are: 1. floating 1s 2. floating 0s 3. alternate bits pattern 4. random data 5. binary up-count 6. disk worst-case data 7. channel pattern 8. channel parity STST2 - This section of STEST checks the reliability of the hardware to perform spiral-type data transfers. This section starts by writing data at sector 11 of the selected test track to the middle of the next track. The hardware should keep track of the surface number and update it when the spiral occurs. The read and verify section issues two read transfer requests to check the entire write transfer. STST3 - This section of STEST uses the remaining test patterns with full track data transfers and no spiraling. NODRIVE - Routine to remove an untestable selected drive from the test list, then return to the select routine called.

Table 4 DSRMB Error Summary (Cont)

Test	No.	Description
SEKTST	05	SEKTST - Mechanical Positioning Test Mechanical motion reliability tests: 1. Issue a recalibrate, then seek to the max cylinder. Repeat 100 times. 2. Seek between cylinder 000 and cylinder 128. Repeat 100 times. 3. All 1 cylinder forward seeks starting at cylinder 000 and ending at max cylinder. Follow with a reverse 1 cylinder seek test starting at the max cylinder and ending at cylinder 000. Repeat 10 times. 4. Incremental seeks starting at cylinder 000 (i.e., 0 to 1, 0 to 2, 0 to 3, etc.). Follow with an incremental reverse seek sequence starting at max cylinder. 5. Random seek test - seek to 500. Random cylinders on the test drive. 6. Servo noise test. Seek to a reference cylinder then seek to cylinders N+4, N+1, N+3, N+2, and N+5. Repeat the sequence until all cylinders have been referenced. 7. Cylinder difference test - using cylinder 0 as a reference with HCI set, issue a read headers and data command to cylinder N. Verify the cylinder position by checking the header data.
INTTST	06	INTTST - Track/Data Interaction Test This test checks for track interaction failures at the cylinders where the write current is stepped. The write current is stepped every 200 octal cylinders from cylinder 0 for RP06s. First seek to the reference cylinder N and write a worst-case data pattern on all tracks at REF CYL, REF CYL-1, and REF CYL+1. Then read/verify the data on those tracks. The reference cylinder is the cylinder where the write current is stepped. Next, write a parity pattern on all tracks of the REF CYL and read/verify the data. Go back and re-verify the data on REF CYL-1 and REF CYL+1. Repeat ten times. Repeat the sequence until all current change cylinders have been referenced. Next, get a reference cylinder and write a parity pattern on either side of it and verify the data. Re-verify that the data on the reference cylinder has not changed. Repeat the sequence until all current change cylinders have been referenced. NOTE Pack deletion/protection is inhibited during this test. INTTKS - This routine will update the surface selected number and exit with DSKAD and .SURF updated. RTN +1 if new surface number selected. RTN +2 if at end of selected cylinder. HISIDE - Routine to select a REF CYL+1 by adding 2 to the selected CYL number in the DSKAD address RTN +1 is the only return. INSIDE - Routine to select a REF CYL by adding -1 to the DSKAD address cylinder number RTN +1 is the only return.

Table 4 DSRMB Error Summary (Cont)

Test	No.	Description
		INTREF - Routine to select a REF CYL from the defined table INTTAB where REF CYL = LH of entry +1. RTN +1 if DSKAD is sbelow REF cylinder. RTN +2 if at the end of the table. RTN +3 if DSKAD points to the REF CYL. INTADR - Routine to test for a selected range of addresses specified by INTTAB table. RTN +1 if not in range (DSKAD too low or high). RTN +2 if at the end of the table. RTN +3 if DSKAD is in range. LOSIDE - Routine to select a REF CYL-1 from the INTTAB defined table. RTN +1 if at end of table. RTN +2 if REF CYL-1 is found.
FRTEST	07	FRTEST - Fast Random Parameter Transfer Test This test is designed to randomly select disk areas for random selected data patterns to keep the positioner in maximum motion. Data transfer size can be limited to one sector if program switch ONESEC is set. Setting this switch will keep the positioner moving.
FORMAT	10	FORMAT - Disk Pack Formatter/Verifier This routine provides facilities for formatting/verifying disk packs in either 16-bit (PDP-11 mode) or 18-bit (PDP-10 mode) format. Options ask the operator to process entire disk packs or just a section of a disk pack. Formatting/verifying is done on a track by track basis (i.e., an individual sector cannot be processed; the entire track must be processed). Format a track Exit format routine Verify a formatted track by reading headers and data Report the finish time
SEQGEN	11	SEQGEN - This is a test script generator SEQGEN is capable of creating a test script containing up to 10 tests from the test list or executing an existing test script that was previously generated from SEQGEN or preloaded by the program. The preloaded scripts are as follows. TOTAL = all tests ACCEPT = drive acceptance testing MECH = all mechanical tests (no data) TIMING = all timing tests RUNLST = run the script-generated list Test typed names for validity. Script table is now filled with valid names - execute the table.

Table 4 DSRMB Error Summary (Cont)

Test	No.	Description
ACCEPT	12	This is the acceptance script list. SEKTST RONLY RCTEST FRTEST INTTST STEST RCTEST WONLY RONLY FRTEST
PTIME	13	PTIME - Disk Mechanical Timing Test PTIME will measure the times of recals, seeks, and disk rotation. If VARIAB (switch 19) is set, the operator may select seeks to be timed. NOTE This test has been designed to perform a relative measurement of seek times from drive to drive or from a previous run of the drive selected for test. It is not designed to measure the drive's performance against the seek time specification. PTIM0 - Mechanical timing test - part 0. Measure rotational velocity 50 (decimal) times. Print high, low range, and average times in milliseconds. Rotational velocity test Save timebase readings from CHAN 4 INTR service Search for sector 0 again Calculate the elapsed time PTIM1 - Mechanical timing test - part 1 Do 100 recals and print the high, low, range, and average times in milliseconds PTIM2 - Mechanical timing test - part 2 Time all single cylinder forward seeks. Print high, low, range, and average times in milliseconds. PTIM3 - Mechanical timing test - part 3 Time all single cylinder reverse seeks. Print high, low, range, and average times in milliseconds. PTIM4 - Mechanical timing test - part 4 Time incremental (oscillating) forward seeks. Print high, low, range, and average times in milliseconds. PTIM5 - Mechanical timing test - part 5 Time incremental (oscillating) reverse seeks. Print high, low, range, and average times in milliseconds. PTIM6 - Mechanical timing test - part 6 Time 500 (decimal) random seeks with random stalls between operations, print high, low, range, and average times in milliseconds. PTIM7 - Mechanical timing test - part 7 Time the average seek time by seeking to a reference cylinder for the particular drive type 100 times and print high, low, range and average times in milliseconds. The reference cylinder for RP04/05s is 136 (decimal) and cylinder 255 for RP06s.

Table 4 DSRMB Error Summary (Cont)

Test	No.	Description
		PTIM8 - Mechanical timing test - part 8 Time maximum seek time (seek from cylinder 0 to 420, or 814). 100 repetitions; prints high, low, range, and average times in milliseconds. PTIMOP - Routine to allow position timing between any two operator-selected cylinders. PTMCLR - Clears the high, low, and average counters used by PTIME PTMSAV - Saves the value derived from the timing routine PTIMER - Routine to measure event time for a given position command in AC1 when the routine is called. Returns with the measured time in SAVTIM. Calculate the elapsed time from the time the I/O was issued in LDREG (RDTIMC) and the time reading from CHAN 5 interrupt service (RDTIMD). PWAIT - This routine will stall the execution of the program until the I/O in progress has been completed or 1 second has elapsed since the I/O was issued - the operation has timed out. PTMPNT prints out high and low, range, and average times PTMCNV converts the count in AC0 to milliseconds and prints it.
RCTEST	14	RCTEST - Random Command Test This test causes an overlap of position and data transfer commands to selected drives. The commands to be issued are: 1. Read - no data compare 2 Read Headers 3. Write - write any data to be used 4. Seek - seek to selected random address 5. Search - search to selected random disk address All data transfers are limited to two sectors in length.

GENERAL INFORMATION

Code

DSRPA.SAV

Title

DECSYSTEM-2020 KS10/RH11-RP06 Basic Device Diagnostic

Abstract

DSRPA is the RH11/RP06 basic device diagnostic, designed to isolate solid RP06 faults to the faulty module or group of modules within the DCL or drive electronics. Scope looping capabilities have been designed into each test.

Testing is done on a "start small and build up" basis. The diagnostic starts by testing control bus cycles and ends by doing full-speed data transfer operations to various disk addresses. Positioning logic is also tested.

In addition to the straight line diagnostic tests, the diagnostic also contains a head alignment/verification subprogram that is invoked by setting the proper sense switch.

This diagnostic operates in exec mode only.

Hardware Required

KS10 mainframe
Minimum of 48K of memory
RH11 Massbus controller
1-8 RP06 disk drives

Preliminary and Associated Programs

Refer to diagnostic hierarchy (KS10 STD module).

Restrictions

Notes

1. An asterisk (*) in the test column of Table 2 indicates that the listing (on microfiche) contains a recommended troubleshooting procedure.

2. Run time (per drive basis)

   a. If no operator intervention tests are run, the diagnostic will complete a pass in approximately three minutes.

   b. The dual-port tests take approximately three minutes to execute once the drive is cabled up.

   c. Excluding warm-up and set-up time, head alignment verification takes approximately five minutes.

3. The following sequence must be followed to ensure complete testing of a single-ported device.

   a. Cable the controller to the active port, lock the device on that port and run one pass of the diagnostic with the manual intervention tests selected.

   b. If multiple passes are desired, it will not be necessary to run the manual intervention tests following the first pass.

Loading and Starting Procedure

Standard (Refer to the KS10 STD module.)

Control Switches

Refer to Table 1.

Table 1 DSRPA Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (Refer to the KS10 STD module.) The following switches are not implemented: 1 (RSTART), 2 (TOTALS), 9 (RELIAB), 15 (CHAIN).
18		Trace program's progress test by test.
19		List all possible modules if not running in text inhibit mode.
20		Not used
21		Run the dual-port tests.
22		Run the head alignment verification test.
23-24		Not used
25		Pause until this switch is reset.
26		Run the test specified by SW 27-35.
27-35		Specify the test that will be continuously run if SW 26 is set (1000).
		Switches (27-35) RH11 Test
		157 RHTST1 Tests RHWC
		160 RHTST2 Tests RHBA
		161 RHTST3 Tests RHDB
		162 RHTST4 SILO Test 1
		163 RHTST5 SILO Test 2
		164 RHTST6 SILO Test 3
		167 RHTST7 SILO Test 4
		170 RHTS10 SILO Test 5

OPERATIONAL CONTROL
The diagnostic furnishes all the necessary instructions for operation as it runs. The sense switch options may also be printed at run time.

The diagnostic will first poll the system and report which RP06 and controller configurations exist.

The user is allowed to select drives for test. A drive may be selected for test, whether or not it was detected at configuration time. Following the completion of configuration, the diagnostic goes to its test dispatcher to determine which tests are to be run.

Dispatcher
In order to understand the dispatcher it is important to understand the following facts about the test structure.

1. There are 363 (octal) in-line stand-alone tests, some of which require operator intervention. These tests may be selected for individual operation by SW 27-35.

2. The dual-port test is a stand-alone series of 23 (octal) subtests that are self-documenting with complete instructions.

3. Head alignment-verification is a stand-alone subprogram, but requires the use of a CE pack and DDU tester.

   The dispatch algorithm works as follows.

   a. Read the console switches.

   b. If pause switch (002000) is set, go to step a.

   c. If the dual-port option switch is set (040000), perform and run the dual-port tests on dual-ported RP06s only. Then return to step a.

   d. If the head alignment option switch is set (020000), perform head alignment-verification on the drives selected for test. Then return to step a.

e. If the operator intervention switch is ON, set a flag that allows operator intervention tests to be executed when encountered (flag is checked locally at each test).

f. If SW 26 is set (001000) (manual test selection), execute the test whose number is specified in SW 27-35. If illegal, give an error message. The test is executed one time and then return is made to step a.

g. Execute the next in-line test of the 363 available, one time, and then return to step a. When all tests have been executed, end of pass is printed and the sequence will be repeated until the diagnostic is stopped or the abort switch is raised.

NOTE
When an in-line test is being run, it is executed on all drives being tested before return is made to the dispatcher.

Use of the Diagnostic
A typical use of the diagnostic would be as follows.

Run the diagnostic until it fails on a test and then select that test from the switches so that it will be continuously run and the trouble can be isolated or modules replaced. After replacing a module or group of modules rerun the entire diagnostic in case symptoms have changed.

Due to the structure of the diagnostic, troubleshooting should be done using the earliest possible test that failed. If several tests indicate malfunctions, check them out one at a time.

TEST SUMMARY
The individual tests performed by this diagnostic are described in Table 2.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

Table 2 DSRPA Test Summary

Test	Description
1*	DUAL DRIVE SELECTION All drives on the bus are powered up except the drive being tested. The diagnostic will do a blind read of drive register number 6 (the drive type register) to guarantee that no other drive on the Massbus responds to this drive's device number. If the test is unsuccessful, the drive type and serial number registers of the responding device on the bus will be printed in order to aid in the device isolation.
2*	DEMAND/TRANSFER TEST The drive being tested is now powered up. The diagnostic does a blind read of register number 6 in the drive (drive type) looking only for proper operation of Demand and Transfer over the Massbus. Improper operation would result in a control bus timeout. Data and parity errors are ignored at this point in the testing.
3	Not implemented.
4*	TRANSCEIVER ENABLE TEST This test reads the drive type register looking for a legal device type. It is the first time 1s and 0s are read. This test is actually to guarantee that the transceivers are enabled during a control bus read cycle and that the C to D (direction of transfer signal) is not incorrect during the control bus read cycle. In getting this test operational, drive-type register problems are debugged as a by-product. The program reads the drive-type register. Successful operation indicates that the entire control bus read cycle is operational.
5*	RESET/WRITE/READ 1S TO OFFSET REGISTER Master reset, then write/read/verify 1s from the low-order 7 bits of the offset register. This verifies that the receivers pass 1s.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
6*	RESET/WRITE/READ 0S TO OFFSET REGISTER Write 1s in bits 0-6 of offset register. This is known to work from test 5. Reset and write 0s; read and verify that bits 0-6 have gone to 0.
7*	RESET/WRITE 0S/WRITE 1S/READ/VERIFY OFFSET REGISTER This test is similar to test 5, but it writes 1s to flip-flops that are guaranteed to contain 0s, thus verifying that transmitters pass 1s, that reset is not stuck in clear direction, and that flip-flops are being loaded.
10*	WRITE 1S/WRITE 0S/READ/VERIFY OFFSET REGISTER This test is similar to test 6; however, this test guarantees that the flip-flops are 1s before the 0s are written, thus verifying that the transmitters pass 0s to the flip-flops and the flip-flops clear via the data inputs.
11*	WRITE 1S/RESET/READ/VERIFY OFFSET REGISTER Write 1s to the flip-flops, issue a reset (Massbus clear) to verify that the reset pulse is clearing the proper bits. This test also verifies that the reset pulse is getting over the Massbus to the DCL properly.
12*	RESET/WRITE/READ/VERIFY 1S TO ERROR REGISTER NO. 3 Master reset, then write/read/verify 1s to error register no. 3 (ERR-3). This verifies that the receivers pass 1s.
13*	RESET/WRITE/READ 0S TO ERROR REGISTER NO. 3 Write 1s to the register. This is known to work from test 12. Reset/write 0s/read/verify that the bits were cleared. This verifies that the flip-flops and receivers work with 0s.
14*	RESET/WRITE 0S/WRITE 1S/READ/VERIFY ERROR REGISTER NO. 3 This is similar to test 12 except that this test writes 1s to flip-flops that are guaranteed to contain 0s, thus verifying that the transmitters are passing 1s, that reset is not stuck in the clear direction, and that flip-flops are loading via their data inputs.
15*	WRITE 1S/WRITE 0S/READ/VERIFY ERROR REGISTER NO. 3 This test is similar to test 13 except that it is certain that the flip-flops are 1s when the 0s are written, thus verifying that the transmitters pass 0s to the flip-flops and the flip-flops clear via the data inputs.
16*	WRITE 1S/RESET/READ/VERIFY ERROR REGISTER NO. 3 Write the flip-flops with 1s. Issue a reset (Massbus clear) and verify that the flip-flops are cleared by the reset.
17*	RESET/WRITE/READ/VERIFY 1S TO ERROR REGISTER NO. 2 Master reset, then write/read/verify 1s from the high-order two bits of error register no. 2 (ERR-2). This verifies that the receivers pass 1s.
20*	RESET/WRITE/READ 0S TO ERROR REGISTER NO. 2 Write 1s in the flip-flops. This is known to work from test 17. Reset/write/read/verify that the flip-flops are cleared. This verifies that the flip-flops and receivers work with 0s.
21*	RESET/WRITE 0S/WRITE 1S/READ/VERIFY ERROR REGISTER NO. 2 This is like test 17 except that this test writes 1s to flip-flops that are guaranteed to be 0, thus verifying that the transmitters are passing 1s, that reset is not stuck in the clear state, and that flip-flops are loaded via their data inputs.
22*	WRITE 1S/WRITE 0S/READ/VERIFY ERROR REGISTER NO. 2 This test is similar to test 20 except that this time, it is guaranteed that the flip-flops are 1s when the 0s are written, thus verifying that the transmitters pass 0s to the flip-flops and the flip-flops are clear via the data inputs.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
23*	WRITE 1S/RESET/READ/VERIFY ERROR REGISTER NO. 2 Write 1s to the flip-flops, issue a reset and verify that the flip-flops are cleared by the reset pulse.
24*	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 15 ERROR Master reset, read and verify that bit 15 is not stuck on a 1. This verifies that there is not a persistent (off cylinder) error that invalidates further testing of the control bus.
25	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 14 ERROR Master reset, read and verify that bit 14 is not stuck on a 1. This verifies that there is not a persistent SEEK INCOMPLETE error that invalidates further testing of the control bus.
26	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 06 ERROR Master reset, read and verify that bit 06 is not stuck on a 1. This verifies that there is not a persistent DC LOW error that invalidates further testing of the control bus.
27	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 05 ERROR Master reset, read and verify that bit 05 is not stuck on a 1. This verifies that there is not a persistent DC LOW error that invalidates further testing of the control bus.
30	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 04 ERROR Master reset, read and verify that bit 04 is not stuck on a 1. This verifies that there is not a persistent 35 VF error that invalidates further testing of the control bus.
31	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 01 ERROR Master reset, read and verify that bit 01 is not stuck on a 1 indicating a persistent WT OFS condition.
32	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 00 ERROR Master reset, read and verify that bit 00 is not stuck on a 1. This verifies that there is not a persistent dc unsafe error that invalidates further testing of the control bus.
33	TEST FOR PERSISTENT ERROR REGISTER NO. 3 BIT 13 ERROR Master reset, read and verify that bit 13 is not stuck on a 1. This verifies that there is not a persistent OPE error that invalidates further testing of the control bus.
34	TEST THAT ERROR REGISTER NO. 3 CAN BE LOADED WITH 1S Master clear, write 1s/read/verify that all bits in the register are set. This just checks flip-flops in the register. All bits are known to clear from previous tests.
35	TEST THAT ERROR REGISTER NO. 3 CAN BE LOADED WITH 0S Write 1s/write 0s/read/verify that all register bits are 0. This just checks that all flip-flops in the register can be loaded with 0s. Bits are known to set from previous tests.
36*	SINGULARITY OF ERROR REGISTER NO. 3 BITS (FLOAT 1S) Float a 1 down the register to guarantee that no bits interfere with one another. Before each 1 is loaded, the register is cleared.
37*	SINGULARITY OF ERROR REGISTER NO. 3 BITS (FLOAT 0S) Float a 0 down the register to guarantee that no bits interfere with one another. Before each bit is loaded, the register is preset to 1s.
40	ERROR REGISTER NO. 3 DIRECT RESET TEST Load the register with 1s. Master clear the register (Massbus clear) and verify that the register is in fact cleared. This test tests the ability of all flip-flops to clear.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
41	TEST THAT PARITY NETWORK CAN GENERATE A 1 Offset register is cleared with a master reset and then read over the control bus. The parity bit should be a 1 since the data field is 0s.
42	TEST THAT PARITY NETWORK CAN GENERATE A 0 A single bit is written to error register no. 3 (bit 00). The register is then read back over the control bus. This test verifies that the parity bit is a 0 since the data field contains an odd number of 1s.
43	TEST PARITY NETWORK WITH VARIOUS PATTERNS Load and read a series of seven patterns over the control bus using error register no. 3. These patterns have been selected to guarantee that the parity networks themselves have no internal failures that will cause them to make a wrong decision.
44	Not implemented.
45*	RESET/READ/VERIFY C01 ERROR REGISTER NO. 1 Master reset, read and verify that the illegal register (ILR) bit is not stuck on a 1. It is necessary to have the ILR bit operational for further register selection tests.
46*	WRITE/READ/VERIFY A 1 TO C01 ERROR REGISTER NO. 1 Write/read and verify that a 1 can be written to the ILR flip-flop. C01 of error register no. 1.
47	WRITE A 1/RESET/READ/VERIFY C01 ERROR REGISTER NO. 1 This test is similar to test 45 except that it guarantees the flip-flop is set when the reset is applied.
50	WRITE A 1/WRITE A 0/READ VERIFY C01 ERROR REGISTER NO. 1 This test ensures that C01 error register no. 1 can be cleared via its data input.
51	Not implemented.
52*	SEE THAT READING REGISTERS S 00-17 DOES NOT CAUSE ILR Although only REG SEL 04 seems important in determining whether or not an illegal register is selected, testing all the legal registers may catch coupling between the register select bits.
53*	VERIFY OPERATION OF CONTROL REGISTER C03 This test gets C03 checked out along with using the control register for the first time, which will check part of the register select logic. Four discrete subtests are tried. 1. Write/read/verify a 1 to C03 2. Write a 1/write a 0/read/verify C03 3. Write a 0/write a 1/read/verify C03 4. Write a 1/reset/read/verify that C03 did not clear
54*	RESET/READ/VERIFY A 0 IN C03 ERROR REGISTER NO. 1 It is necessary to get this bit working in order to do some later register tests. This is a tricky bit to test since it is the parity error detection bit and we are writing registers to test the bit. This resets the bit and verifies that the flip-flop is not stuck on a 1.
55*	RESET/WRITE 0/VERIFY C03 ERROR REGISTER NO. 1 Writing 0s to error register no. 1 and then checking C03 ensures that the drive's parity detection network is not detecting a parity error when the drive receives a parity bit of 1.
56*	RESET/WRITE 1 TO C01 ERROR REGISTER NO. 1/READ/VERIFY C03 ERROR REGISTER NO. 1 Writing a single 1 to C01 of error register no. 1 and verifying that (the parity bit) C03 error register no. 1 did not set ensures that the drive's parity detection network is not detecting a parity error when the drive receives a parity bit of 0.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
57*	RESET/WRITE A 1/READ/VERIFY C03 ERROR REGISTER NO. 1 It has been proven that writing a single 1 does not cause a parity error. Now reset and write a 1 to C03 error register no. 1 to ensure that the flip-flop can be set.
60*	RESET/WRITE A 1/WRITE A 0/READ/VERIFY C03 ERROR REGISTER NO. 1 This test ensures that the C03 error register no. 1 bit can change from the 1 to 0 state (can be cleared by writing a 0).
61*	RESET/WRITE A 1/RESET/READ/VERIFY C03 ERROR REGISTER NO. 1 This test ensures that the C03 flip-flop clears with direct reset.
62*	CAUSE A PARITY ERROR. SEE IF IT IS DETECTED This tests the parity detection logic by having the controller write the register with incorrect parity. C03 error register no. 1 should set since it is the parity error flip-flop.
63*	GUARANTEE THAT GO BIT C00 CONTROL REGISTER IS NOT STUCK Master reset, read, verify that the go bit is not stuck at 1. This is a necessary prerequisite for testing C03 in maintenance register.
64*	VERIFY OPERATION OF C00 MAINTENANCE REGISTER Testing this bit is a prerequisite test for testing C03 maintenance register. This bit (C00 maintenance register) is the diagnostic maintenance mode bit. It must be operational before other bits in the maintenance register can be tested. A group of four tests are performed to verify the operation of this bit. 1. Write a 1/read/verify C00 is a set 2. Write a 1/write a 0/read/verify C00 is clear 3. Write a 0/write a 1/read/verify C00 is set 4. Write a 1/reset/read/verify C00 is clear
65*	RESET/WRITE 1S/READ/VERIFY C01-C04 This test will catch register select failures. Writing C01-C04 should not cause any bits to set since the diagnostic bit is clear.
66	VERIFY COMPLETE OPERATION OF C01-C04 MAINTENANCE REGISTER These four subtests check complete operation of C01-C04 of the maintenance register, in conjunction with the maintenance mode bit C00 maintenance register. 1. Set M-M bit (M-M bit = maintenance mode bit C00) Write 1s to C01-C04/read/verify 2. Set M-M bit Write 0s to C01-C04/read/verify 3. Set M-M bit Write 0s/write 1s/read/verify C01-C04 4. Set M-M bit Write 1s/reset/read/verify C01-C04
67*	PARTIAL TEST OF C03 DESIRED SECTOR/TRACK ADDRESS REGISTER This test is essentially to make C03 desired sector/track address register readable and writable so the bit can be used later in register selection tests. The reset test for this flip-flop will be done later. This test could turn up register select problems since this is the first time we are addressing desired sector/track address. This test comprises the following three subtests. 1. Write a 1/read/verify C03 2. Write a 1/write a 0/read/verify C03 3. Write a 0/write a 1/read/verify C03

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
70*	PARTIAL TEST OF C03 DESIRED CYLINDER ADDRESS REGISTER This test is to verify that C03 is readable and writable so the bit can later be used to test the register selection logic. This test could identify register select problems since this is the first time the desired cylinder address register is addressed. The test comprises the following three subtests. 1. Write a 1/read/verify C03 2. Write a 1/write a 0/read/verify C03 3. Write a 0/write a 1/read/verify C03
71*	VERIFY THAT C03 ERROR REGISTER NO. 2 IS NOT STUCK AT 1 Issue a reset and verify that C03 error register no. 2 is not stuck in the set state or held set by a persistent (current switch unsafe) error from the VENDOR logic.
72	VERIFY COMPLETE OPERATION OF C03 ERROR REGISTER NO. 2 REGISTER This test comprises a series of four subtests to ensure that the bit is working properly and can be used in register select tests. The subtests are as follows. 1. Write a 1/read/verify C03 2. Write a 1/write a 0/read/verify C03 3. Write a 0/write a 1/read/verify C03 4. Write a 1/reset/read/verify C03
73*	TEST SELECTION OF DCL WRITABLE REGISTERS The object of this test is to guarantee that no writable DCL register interferes with any other register because of a register selection problem or a register select decoding problem.
74*	VERIFY THAT DRIVE CLEAR WORKS This test verifies that drive clear operates. 1. Put drive in maintenance mode (set diagnostic bit maintenance register). 2. Issue a drive clear with the go bit set. 3. Read maintenance register and verify that the drive is no longer in maintenance mode. Drive clear should cause the maintenance mode bit to clear.
75	VERIFY SOME GO BIT CONTROL LOGIC This tests some of the gating to the D input of the STO GO flip-flop (maintenance register). It ensures that issuing a drive clear command without the GO bit does not cause a drive clear. It does the following. 1. Put drive in maintenance mode. 2. Issue a drive clear with the GO bit set, but writes the register with incorrect parity. 3. Read the maintenance register to see if drive is still in maintenance mode. It should be, because the parity error should inhibit the drive clear from taking the drive out of maintenance mode.
76	VERIFY ADDITIONAL GO BIT CONTROL LOGIC This tests the logic to inhibit execution of a command when a parity error occurs while writing the control register. It is done in the following manner. 1. Put the drive in maintenance mode. 2. Issue a drive clear with the GO bit set, but write the register with incorrect parity. 3. Read the maintenance register and see if drive is still in maintenance mode. It should be because the parity error should inhibit the drive clear from taking the drive out of maintenance mode.
77*	TEST FOR POSSIBILITY OF MULTI-DRIVE RESPONSE This test is effectively the complement of test 1. This test ensures that the drive being tested responds to no other drive address but its own.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
100*	ENSURE THAT A PERSISTENT MSE DOES NOT EXIST Reset/write 0/read/verify that C04 error register no. 2 (read and write) is not stuck at 1.
101	Not implemented.
102*	VERIFY THAT A MULTIPLE HEAD SELECT PROBLEM DOES NOT EXIST Reset/write 0/read/verify that C10 error register no. 2 is not a 1.
103*	VERIFY THAT A NO HEAD SELECT PROBLEM DOES NOT EXIST Reset/write 0/read/verify that C10 error register no. 2 is not a 1.
104*	VERIFY THAT A PLO UNSAFE DOES NOT EXIST Reset/write 0/read/verify that C13 error register no. 2 is not at 1.
105*	VERIFY THAT PERSISTENT INDEX ERROR DOES NOT EXIST Reset/write 0/read/verify that C11 error register no. 2 is not stuck at 1.
106	VERIFY THAT C08 ERROR REGISTER NO. 2 WRTRU IS NOT STUCK AT 1 Reset/write 0/read/verify C08 error register no. 2.
107*	VERIFY THAT C05 ERROR REGISTER NO. 2 TRADF IS NOT STUCK AT 1 Reset/write 0/read/verify that there is not a persistent transition detector failure.
110*	VERIFY THAT C06 ERROR REGISTER NO. 2 TRANSU IS NOT STUCK AT 1 Reset/write 0/read/verify that C06 error register no. 2 is not at 1.
111*	VERIFY THAT C02 ERROR REGISTER NO. 2 WSUNS IS NOT STUCK AT 1 Reset/write 0/read/verify C02 error register no. 2.
112*	VERIFY THAT C01 ERROR REGISTER NO. 2 CFAIL IS NOT STUCK AT 1 Reset/write 0/read/verify C01 error register no. 2.
113*	VERIFY THAT C00 ERROR REGISTER NO. 2 WCUNS IS NOT STUCK AT 1 Reset/write 0/read/verify C00 error register no. 2.
114	CHECK THAT C07 ERROR REGISTER NO. 2 ABS IS NOT STUCK AT 1 Reset/write 0/read/verify C07 error register no. 2.
115	VERIFY OPERATION ERROR REGISTER NO. 2 AND ITS MULTIPLEXERS This test merely verifies that the register flip-flops and their associated multiplexers go through their various transitions properly. The following subtests are performed. 1. Reset/write 1s/read/verify 2. Reset/write 1s/write 0s/read/verify 3. Reset/write 0s/write 1s/read/verify 4. Reset/write 1s/reset/read/verify
116	SINGULARITY OF ERROR REGISTER NO. 2 BITS (FLOAT 1S) Float a 1 down the register to guarantee that no bit interferes with another. Before each 1 is loaded, the register is cleared.
117	SINGULARITY OF ERROR REGISTER NO. 2 BITS (FLOAT 0S) Float a 0 down the register to guarantee that no bit interferes with another. Before each bit is loaded, the register is reset to 1s.
120	Not implemented.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
121	DATA TEST OF THE IMPLEMENTED BITS OF OFFSET REGISTER The implemented bits are C00-C07 and C10-C12. Although C15 is also implemented, it is read-only and indeterminate and will be ignored during this sequence of tests. The following four subtests are used to data-test this register. 1. Reset/write 1s/read/verify 2. Reset/write 1s/write 0s/read/verify 3. Reset/write 0s/write 1s/read/verify 4. Reset/write 1s/reset/read/verify (C00-C07 only) C10-C12 are cleared by reading (reset) and will have to be tested later.
122	SINGULARITY OF OFFSET REGISTER (FLOAT 1S) Float a 1 down the implemented bits of offset register to guarantee that no bit interferes with another. Before the 1s are loaded, the register is cleared.
123	SINGULARITY OF OFFSET REGISTER (FLOAT 0S) Float a 0 down the implemented bits of offset register to ensure that no bit interferes with another. Before each floating pattern is loaded, the register is set to 1s.
124	Not implemented.
125	DATA TEST IMPLEMENTED BITS OF DESIRED CYLINDER ADDRESS REGISTER The following three subtests do basic data tests of the desired cylinder address register implemented bits. 1. Reset/write 1s/read/verify C00-C09 2. Reset/write 1s/write 0s/read/verify C00-C09 3. Reset/write 0s/write 1s/read/verify C00-C09
126	SINGULARITY OF DESIRED CYLINDER ADDRESS REGISTER BITS (FLOAT 1S) Float a 1 down the register's implemented bits to guarantee that no bit interferes with another. Before each 1 is loaded, the register is cleared.
127	SINGULARITY OF DESIRED CYLINDER ADDRESS REGISTER BITS (FLOAT 0S) Float a 0 down the implemented bits of desired cylinder address register to guarantee that no bit interferes with another.
130	Not implemented.
131	DATA-TEST THE IMPLEMENTED BITS OF DESIRED SECTOR/TRACK ADDRESS REGISTER The implemented bits are C00-C04 and C08-C12. The following subtests are used to check this register and its associated multiplexers. 1. Reset/write 1s/read/verify 2. Reset/write 1s/write 0s/read/verify The clearing and incrementing of this register will be verified in another test.
132	SINGULARITY OF DESIRED SECTOR/TRACK ADDRESS REGISTER BITS (FLOAT 1S) Float a 1 down the implemented bits of desired sector/track address register to guarantee that no bit interferes with another. Before the 1s are loaded, the register is cleared.
133	SINGULARITY OF DESIRED SECTOR/TRACK ADDRESS BITS (FLOAT 0S) Float a 0 down the implemented bits of desired sector/track address to guarantee that no bit interferes with another. Before each pattern is written, the register is loaded with 1s.
134	Not implemented.

Table 2 DSRPA Test Summary (Cont)

Test	Description
135	VERIFY THAT ERROR REGISTER NO. 1 C15 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent data check error does not exist.
136	VERIFY THAT C14 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent unsafe condition does not exist.
137	VERIFY THAT C13 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent OPI does not exist.
140	VERIFY THAT C12 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent DTE error does not exist.
141	VERIFY THAT C11 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent WLE does not exist.
142	VERIFY THAT C10 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent IAE error does not exist.
143	VERIFY THAT C09 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent AOE does not exist. The desired address registers are set to 0 in order to localize the failure to a smaller number of modules.
144	VERIFY THAT C08 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent HCRC error does not exist.
145	VERIFY THAT C07 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent header compare error does not exist.
146	VERIFY THAT C06 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0/read/verify that a persistent hard ECC error does not exist.
147	VERIFY THAT C05 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent write clock fail error does not exist.
150	VERIFY THAT C04 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent format error does not exist.
151	VERIFY THAT C02 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 Reset/write 0s/read/verify that a persistent register modify refused error condition does not exist.
152	VERIFY THAT C00 ERROR REGISTER NO. 1 IS NOT STUCK AT 1 1. Write 0s to the control register (a NOP) 2. Reset/write 0s/read/verify that a persistent ILF condition does not exist.
153	DATA TEST OF FLIP-FLOPS AND MULTIPLEXERS OF ERROR REGISTER NO. 1 The flip-flops and multiplexers of error register no. 1 are data-tested using the following four subtests. 1. Reset/write 1s/read/verify 2. Reset/write 1s/write 0s/read/verify 3. Reset/write 0s/write 1s/read/verify 4. Reset/write 1s/reset/read/verify
154	SINGULARITY OF ERROR REGISTER NO. 1 BITS (FLOAT 1S) Float a 1 down the register to guarantee that no bit interferes with another. Before each 1 is loaded, the register is cleared by writing 0s into it.

Table 2 DSRPA Test Summary (Cont)

Test	Description
155	SINGULARITY OF ERROR REGISTER NO. 1 BITS (FLOAT 0S) Float a 0 down the register to guarantee that no bit interferes with another. Before each floating pattern is loaded, the register is preset to 1s.
156	Not implemented.
157*	TEST THAT A PERSISTENT COMPOSITE ERROR DOES NOT EXIST Master reset, then read the status register and ersure that composite error is not set.
160*	VERIFY THAT COMPOSITE ERROR CAN BE DECODED This test verifies that composite error can be read back successfully. It will accept any path at this point in time. Reset/write 1s to ER1, 2, 3/read/verify that C14 status register is set.
161*	VERIFY THAT EACH ERROR REGISTER NO. 1 BIT CAUSES COMPOSITE ERROR This test verifies that the setting of each bit in error register number 1 (register 02) produces a composite error indication in the status register (register 01).
162*	VERIFY THAT EACH ERROR REGISTER NO. 2 BIT CAUSES COMPOSITE ERROR This test verifies that the setting of each bit in error register number 2 (register 14) causes a composite error in the status register (register 01).
163	VERIFY THAT EACH ERROR REGISTER NO. 3 BIT CAUSES COMPOSITE ERROR This test verifies that the setting of each bit in error register number 3 (register 15) causes a composite error in the status register (register 01).
164	VERIFY THAT DRY IS COMPLEMENT OF THE GO BIT This test ensures that DRY and GO are complements when GO is reset. Reset/read status register/verify that C07 (DRY) is 1.
165	TESTING FOR PROPER HEAD LOAD OPERATION This test verifies that the head load sequence completes and also that the drive status is correct at completion of the head load sequence.
166*	TESTING FOR PROPER HEAD UNLOAD OPERATION This test verifies that the head unload sequence will operate correctly and error free.
167	VV AND PACK ACK OPERATIONAL TESTS This is a series of four subtests designed to ensure that the volume valid status bit is working in conjunction with the pack acknowledge command. The subtests are as follows. 1. Issue pack acknowledge with drive cycled up and ensure that VV gets set. 2. Verify that issuing pack acknowledge does not cause errors. 3. Take drive off-line with VV set and ensure that it stays set. 4. With VV set, cycle drive down and up and ensure that VV gets cleared.
170	TEST VV/COMP ERR INTERLOCK This test verifies that VV will not set if a composite error exists when the command is issued. The following sequence is issued. 1. Cycle drive down and up to clear VV 2. Write an error register to force composite error 3. Issue a pack ACK command 4. Read status and control register 5. Ensure that the GO bit is clear.
171	SEE THAT READIN PRESET WILL SET VV This ensures that readin preset sets VV and does not cause composite error in doing so.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
172*	TEST STO GO RESET LOGIC This verifies that the GO and STO GO reset logic is partially operational. The following sequence is used. 1. Clear the previous errors. 2. Issue pack ACK to set VV. 3. Issue another pack ACK to transfer STO GO to GO. 4. Read the control register and ensure that GO is clear.
173	VERIFY THAT MAINTENANCE MODE DISCONNECTS SECTOR CLOCK This is a basic test to verify that going into maintenance mode actually does something effective. The look ahead register is used here as a tool. The following method is used. 1. Power up the spindle. 2. Go into maintenance mode. 3. Read the look-ahead register for reference. 4. Read the look-ahead register 20 more times to ensure that it does not change. This ensures that the register is disconnected from the drive's index and sector clock.
174	TEST THAT THE GO BIT CAN BE SET This test ensures that the GO bit can be set. The following sequence is used. 1. Issue the master reset. 2. Put drive in maintenance mode (no sector clocks can occur). 3. Issue a seek command (GO should stay hung because a seek command cannot complete without sector clocks). 4. Read the control register and verify that GO is set.
175	CHECK THAT RHCLR WILL CLEAR HUNG GO BIT A hung GO bit is simulated by issuing a seek while in maintenance mode. The following sequence is used. 1. Issue RHCLR to clear the errors. 2. Put the drive in maintenance mode (stops sector clock from coming). 3. Issue a seek (to set the GO bit). 4. Issue another RHCLR (to reset the GO bit). 5. Read control register and verify that GO is clear.
176	TEST THAT GO AND DRY ARE COMPLEMENTS This test ensures that GO and DRY are complements when GO is set. We set GO by issuing a seek while in maintenance mode.
177	FLOAT A 1 DOWN C01-C05 OF CONTROL REGISTER Test merely verifies that all bits can be set. Some have already been tested.
200	TESTING OF PIP WITH A SEEK COMMAND This test is designed to check the operation of PIP in conjunction with a seek command issued in maintenance mode. Three subtests are required. 1. Read PIP with no command to ensure that it is not stuck at 1. 2. Issue a maintenance mode seek with no GO bit and ensure PIP does not set. 3. Issue a maintenance mode seek with GO bit and ensure PIP does not set.
201	VERIFY THAT READIN CLEARS OFFSET REGISTER This test verifies that the readin preset command clears C10-C12 of the offset register.
202	VERIFY THAT READIN CLEARS DESIRED CYLINDER REGISTER This test verifies that the readin preset command clears C00-C09 of the desired cylinder address register.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
203	VERIFY THAT READIN WILL CLEAR DESIRED TRACK/SECTOR REGISTER This test verifies that issuing a readin preset command will clear C00-C04 and C08-C12 of the desired track/sector address register.
204	A TEST OF THE CONTROL REGISTER MULTIPLEXER This test verifies that the control register multiplexer (RG6) is actually returning 0s in the guaranteed-to-be-0 bit positions which are C06-C10 and C12-C15.
205*	TEST THAT ILLEGAL COMMANDS CAUSE ILF ERROR This test issues the illegal commands to guarantee that the drive recognizes ILF for each one. The GO bit is tested after each operation also to ensure that it got reset.
206	DESIRED SECTOR/TRACK ADDRESS MULTIPLEXER TEST Read the desired sector/track address register and verify that the guaranteed 0 bits (C05-C07 and C13-C15) are 0s.
207	LOOK-AHEAD REGISTER MULTIPLEXER TEST Read the look ahead register and verify that the guaranteed 0 bits (C00-C03 and C11-C15) are 0.
210*	DRIVE-TYPE REGISTER GUARANTEED 0S TEST Read the drive-type register and verify that the guaranteed 0 bits are correct. They are: C00-C03, C05-C10, C12, C14-C15.
211	DRIVE-TYPE REGISTER GUARANTEED 1S TEST Read and verify that C01, C04 and C13 of the drive-type register are 1.
212*	DESIRED CYLINDER ADDRESS REGISTER GUARANTEED 0 TEST Read the desired cylinder address register and verify that C10-C15 are 0.
213*	CURRENT CYLINDER ADDRESS REGISTER GUARANTEED 0 TEST Read current cylinder address register and guarantee that C10-C15 are 0.
214	A BASIC RESET TEST A test to verify that an index will clear the low-order seven flip-flops of the extension counter. The following sequence is used: 1. Put in maintenance mode (20-sector mode). 2. Issue index (a general reset). 3. Issue 127 sector clocks to fill low-order seven bits). 4. Issue index to try to reset the low-order seven flip-flops. 5. Issue 127 sector clocks to transfer low-order seven flip-flops to a readable area. 6. Read the sector counter. 7. Verify that fraction is 00.
215	SECTOR CONTROL TEST Verify that the sector counter flip-flops do not count too rapidly using the following sequence. 1. Put in maintenance mode (20-sector mode). 2. Issue index (to clear the counter). 3. Issue 127 sector clocks. 4. Read the look-ahead register. 5. Verify that the sector counter is still at sector 00 extension 00.
216	SECTOR COUNTER TEST Verify that the sector counter is not counting too slowly using the following test sequence. 1. Put in maintenance mode (20-sector mode). 2. Issue index (clears counter). 3. Issue 128 sector clocks (now into first extension). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 00 extension 20.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
217	SECTOR COUNTER TEST Verify that sector counter is counting properly using the following sequence. 1. Put into maintenance mode (20-sector mode). 2. Issue index (clears counter). 3. Issue 255 sector clocks (to end of extension 20). 4. Read the sector counter register. 5. Verify that the sector counter is at sector 00 extension 20.
220	SECTOR COUNTER TEST Verify that the sector counter is counting properly using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clear the register). 3. Issue 256 sector clocks. (to start of extension 40). 4. Read the sector counter register. 5. Verify that the sector counter is at sector 00 extension 40.
221	A SECTOR COUNTER TEST Verify that the sector counter is counting correctly using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clear register). 3. Issue 511 clocks (to end of extension 40). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 00 extension 40.
222	A SECTOR COUNTER TEST Verify that the sector counter is counting properly using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index. 3. Issue 512 sector clocks (to start of extension 60). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 00 extension 60.
223	A SECTOR COUNTER TEST Verify that the sector counter is counting correctly using the following sequence. 1. Go into maintenance mode (22-sector mode). 2. Issue index (to clear the register). 3. Issue 608 sector clocks (to end of extension 60). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 00 extension 60.
224	A SECTOR COUNTER TEST Verify that the sector counter counts correctly using the following sequence. 1. Go into maintenance mode (22-sector mode). 2. Issue index (clear counter). 3. Issue 609 clocks (steps out of extension 60). 4. Read the sector counter. 5. Verify that the sector counter is at the sector 01 extension 00.
225	A SECTOR COUNTER TEST Verify that the sector counter is counting correctly using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index (to clear the counter). 3. Issue 671 clocks (step to the end of extension 60). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 00 extension 60.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
226	A SECTOR COUNTER TEST Verify that the sector counter is counting correctly using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clear counter). 3. Issue 672 sector clocks (to start of sector 01 extension 00). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 01 extension 00.
227	A SECTOR COUNTER TEST Verify that the low-order nine bits of the sector fraction counter can be cleared with index. This particular subtest will only test the 128 and 256 weighted flip-flops. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index (to clear the counter). 3. Issue 511 sector clocks (sets low-order nine bits). 4. Issue index (should clear). 5. Read the register. 6. Verify that the sector counter is at sector 00 extension 00.
230	A SECTOR COUNTER TEST Verify that the low-order nine bits of the sector fraction counter can be cleared by index. This test will pick up the resets of the 1 through 64 weighted bits. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue the index pulse. 3. Issue 511 sector clocks (sets low-order nine bits). 4. Issue second index (should clear). 5. Issue 127 sector clocks (moves invisible bits to where they can be seen). 6. Read the sector counter. 7. Verify that the sector counter is at sector 00 extension 00.
231	A SECTOR COUNTER TEST Verify that index can clear all of the low-order ten bits of the sector fraction counter. This test picks up the high-order bit (512 weight). The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clears counter). 3. Issue 512 clocks (set high-order bit). 4. Issue index (should clear). 5. Read the sector counter. 6. Verify that the sector counter is 0.
232	A SECTOR COUNTER TEST Verify that the high-order three stages of the sector fraction counter edge load to 0s properly at sector pulse time. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clears sector counter). 3. Issue 672 clocks (goes to first sector, causing sector pulse. The fraction counter should have edge loaded with 0s). 4. Read the look-ahead register. 5. Verify that the sector counter is at sector 01 extension 00.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
233	A SECTOR COUNTER TEST Verify that the low-order seven flip-flops of the sector fraction counter edge load to 0s at the occurrence of sector pulse. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index (to clear counter). 3. Issue 672 clocks (causes sector pulse which should edge load the low-order seven bits with 0s). 4. Issue 127 clocks (steps the low-order 7-bit magnitude to a visible area in the register). 5. Read the look-ahead register. 6. Verify that the sector counter is at sector 01 extension 00.
234	A SECTOR COUNTER TEST Verify that the sector counter and overflow decode work in 20-sector mode. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index. 3. Issue 12767 sector clocks (to end of second last sector). 4. Read the sector counter. 5. Verify that the sector counter is at sector 22 extension 60.
235	A SECTOR COUNTER TEST Verify that the sector counter and overflow decode work correctly in 20-sector mode. The following sequence is used. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clears counter). 3. Issue 12768 sector clocks (to start of last sector). 4. Read the sector counter. 5. Verify that the sector counter is at sector 23 extension 00.
236	A SECTOR COUNTER TEST Verify that the sector counter and overflow decode are working correctly in 20-sector mode using the following sequence. 1. Go into maintenance mode (20-sector mode). 2. Issue index (clear counter). 3. Issue 13567 (takes you to invalid sector if decode fails). 4. Read the sector counter. 5. Verify that the sector counter is at sector 23 extension 00.
237	A SECTOR COUNTER TEST Verify that the sector counter and overflow decoder work properly in 22-sector mode using the following sequence. 1. Go into maintenance mode (22-sector mode). 2. Issue index (clear the counter). 3. Issue 12788 sector clocks (to end of second last sector). 4. Read the sector counter. 5. Verify that the sector counter is at sector 24 extension 60.
240	A SECTOR COUNTER Verify that the sector counter and overflow decode work properly in 22-sector mode using the following sequence. 1. Go into maintenance mode (22-sector mode). 2. Issue index (to clear counter). 3. Issue 12789 sector clocks (to start of last sector). 4. Read the sector counter. 5. Verify that the sector counter is at sector 25 extension 00.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
241	A SECTOR COUNTER TEST Verify that the sector counter and overflow decode function properly in 22-sector mode using the following sequence. 1. Go into maintenance mode (22-sector mode). 2. Issue index (clear the counter). 3. Issue 13489 sector clocks (into invalid sector if overflow fails). 4. Read the sector counter. 5. Verify that the sector counter is at sector 25 extension 00.
242	A SECTOR COUNTER TEST Verifies that the four low-order bits of sector flip-flops clear with index pulse. The following sequence is used. 1. Go into maintenance mode (22-sector mode). 2. Issue index (clear the register). 3. Issue 9135 clocks (set low-order four flip-flops in sector register). 4. Issue index (should clear the register). 5. Read the sector counter. 6. Ensure that the sector counter is at sector 00 extension 00.
243	A SECTOR COUNTER TEST Verify that the high-order bit of the sector counter will clear with index. The following sequence is used. 1. Go into maintenance mode (22-sector mode). 2. Issue index (clears counter). 3. Issue 9744 clocks (sets high-order flip-flop). 4. Issue index (should clear flip-flops). 5. Read the sector counter. 6. Verify that the sector counter is at sector 00 extension 00.
244	Not implemented.
245*	VERIFY THAT SECTOR CLOCKS COME DOWN FROM DRIVE Under program control, the only thing the diagnostic can tell about sector clock is that it is present or not present. This test checks for its presence using the following sequence. 1. Go into maintenance mode (stops look-ahead register). 2. Issue a diagnostic index pulse (clears the look-ahead register). 3. Take drive out of maintenance mode with a master clear. This allows sector clock to increment the look-ahead register. 4. Read the look-ahead register up to a maximum of 200 times (more than enough to see it increment). 5. If no change occurs (look-ahead is still sector 00 extension 00), it is safe to assume that it is not incrementing, because either sector clock is not coming down or index coming down from the drive is stuck high.
246*	TEST FOR THE PRESENCE OF INDEX PULSE This test is merely to ensure that index pulse is coming down from the drive. The test is based on the fact that if index pulse is missing, the sector counter will count up and stick at sector 23 (octal). This is the maximum count possible in 20-sector mode. The following sequence is used. 1. Go into 20-sector mode. 2. Read the look-ahead register up to 50 times looking for a nonsector 23 value indicating that index pulse has come.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
247*	CHECK PIP AND ILF WITH POSITIONING COMMANDS Verify the seek, offset, return to centerline, recalibrate, cause the PIP bit to set and do not cause ILF to be set. The following sequence is used. 1. Go into maintenance mode. 2. Set DA = 0 and DCY not equal to CCY. 3. Issue a position command. 4. Read the control, status, and error register no. 1. 5. Test for presence of PIP and that ILF is not set.
250*	VERIFY THAT A LEGAL CYL ADDRESS DECODES CORRECTLY This test verifies that a maintenance mode seek can be issued to every legal cylinder without getting any invalid address errors. The following sequence is used. 1. Issue master clear (clear errors). 2. Go into maintenance mode. 3. Load desired address register with 0s. 4. Load the desired cylinder register with cylinder number. 5. Issue a seek command (strobes the IAE flip-flop). 6. Read control, status, ER1 and DCY. 7. Verify that IAE is not set.
251*	VERIFY THAT ILLEGAL CYLINDER ADDRESS DECODES CORRECTLY Verify that issuing a seek to an invalid cylinder causes an IAE error. The following sequence is used. 1. Issue master clear (to clear errors). 2. Go into maintenance mode. 3. Load DA register with 0s. 4. Load desired cylinder register with 815 (illegal address). 5. Issue a seek command (to strobe IAE flip-flop). 6. Read error register number 1. 7. Verify that IAE is set.
252	VERIFY THAT A LEGAL TRACK ADDRESS DECODES CORRECTLY Verify that issuing seeks to each of the legal track addresses does not cause an IAE to occur. The following sequence is used. 1. Issue master clear (to clear errors). 2. Go into maintenance mode. 3. Clear desired cylinder address register. 4. Load sector = 0 and desired track. 5. Issue a seek command (strobe the IAE flip-flop). 6. Read ER1 and verify that IAE is not set.
253	VERIFY THAT AN ILLEGAL TRACK ADDRESS CAUSES IAE Verify that an attempt to seek to track 19 will cause IAE to set. The following sequence is used. 1. Master clear (to clear errors). 2. Go into maintenance mode. 3. Write 0s to desired cylinder register. 4. Write 0 to desired sector and set up for track 29. 5. Issue a seek command 6. Read ER1 7. Verify that IAE is set.
254	VERIFY THAT LEGAL SECTORS DO NOT CAUSE IAES Verify (in 20-sector mode) that issuing seeks to the legal sectors does not cause IAE to set. The following sequence is used. 1. Master reset (to clear errors). 2. Go into maintenance mode. 3. Go into 20-sector mode. 4. Load 0s to desired cylinder register. 5. Load DA register with track = 0 current sector. 6. Issue a seek command (strobe the IAE flip-flop). 7. Read ER1. 8. Verify that IAE is not set.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
255	VERIFY THAT ILLEGAL SECTOR CAUSES IAE Verify (in 20-sector mode) that issuing a seek to sector 20 causes IAE to be set. The following sequence is used. 1. Issue master clear (clears errors). 2. Go into maintenance mode. 3. Go into 20-sector mode. 4. Write 0s to desired cylinder address register. 5. Write DA register with track = 0 and sector = 20. 6. Issue a seek command (strobe IAE flip-flop). 7. Read ER1. 8. Verify that IAE is set.
256	VERIFY THAT LEGAL SECTORS DO NOT CAUSE IAE Verify (in 22-sector mode) that seeking to legal sectors does not cause IAE. The following sequence is used. 1. Master clear (clear errors). 2. Go into 22-sector mode. 3. Go into maintenance mode. 4. Write 0s to desired cylinder register. 5. Write DA register with track = 0 current sector. 6. Issue a seek (strobe the IAE flip-flop). 7. Read ER1. 8. Verify that IAE is not detected.
257	VERIFY THAT ILLEGAL SECTOR CAUSES IAE Verify (in 22-sector mode) that issuing a seek to sector 22 causes IAE to occur. The following sequence is used. 1. Issue master reset (to clear errors). 2. Go into 22-sector mode. 3. Go into maintenance mode. 4. Write 0s to desired cylinder address register. 5. Write track = 0 and sector = 22 to DA register. 6. Issue a seek command (strobe the IAE flip-flop). 7. Read error register no. 1. 8. Verify that IAE is set.
260	SEEK CONTROL WITH IAE SELECTED This test verifies that signal SS4 BAD ADR L does the following. 1. Causes SEEK GO CLEAR to clear the GO bit. 2. Inhibits the setting of SS0 SEEK GO. The following test sequence is used. 1. Issue master clear (to clear errors). 2. Go into maintenance mode and set CCY to 0. 3. Load DCY with illegal cylinder (815). 4. Issue a seek. 5. Read ER1 and control register. 6. Issue RHCLR. 7. Read DCY and CCY. 8. Verify that go is 0 and that CCY is not 815.
261*	CURRENT CYLINDER ADDRESS REGISTER TEST Data-test the current cylinder address register by transferring various patterns into it from the desired address register. The following test sequence is used. 1. Issue master clear (clears errors). 2. Go into maintenance mode. 3. Load a pattern into the desired cylinder address register. 4. Issue a seek command. 5. Issue a master clear (terminates seek transfers DCY to CCY). 6. Read control register, DCY, CCY. 7. Verify that DCY copied to CCY correctly.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
262*	VERIFY THAT SUBTRACTOR WORKS CORRECTLY FOR CCY = DCY This test verifies that the subtractor does not come up with a difference when CCY = DCY (done for all cylinders). The following test sequence is used. 1. Issue a master clear. 2. Go into maintenance mode. 3. Load DCY with cylinder. 4. Issue a seek command. 5. Issue a master clear (terminate seek transfer DCY to CCY). 6. Go back into maintenance mode (DCY now equals CCY). 7. Issue a seek command. 8. Read the control register. 9. Verify that the GO bit is clear. It should be clear because the difference should be 0. Issuing a seek when DCY = CCY should inhibit the seek from occuring and generate SSO SK GO CLR H (SSO) to clear GO.
263	Not implemented.
264*	A FULL-SPEED RECALIBRATE TEST This test issues a recalibrate, waits for completion, tests for errors following completion. The recalibrate can only be tested at full speed (functionally).
265*	TEST RESET OF CURRENT CYLINDER ADDRESS REGISTER This test verifies that all flip-flops in the CCY register will reset with a recalibrate pulse. The following sequence is used [two patterns are required (377 and 1400 octal)]. 1. Issue a master reset (to clear errors). 2. Go into maintenance mode. 3. Load a pattern into desired cylinder address register. 4. Issue a seek command. 5. Issue a master clear (to transfer DCY to CCY). 6. Issue a recalibrate (should clear CCY). 7. Read the current cylinder register. 8. It should be 0.
266	TEST SUBTRACTOR LOGIC WITH A SERIES OF PATTERNS This test applies a set of patterns to CCY and DCY and looks for the fact that a difference was detected. The patterns come from a look-up table. The patterns chosen will catch a very high percentage of subtractor faults.
267*	BASIC SEEK TEST This test merely issues a 1-cylinder seek at full speed and tests for the following situations. 1. Drive is not hung. 2. No errors are caused. 3. DCY = CCY = 1.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
270	TEST OF SKI LOGIC This test verifies that the seek incomplete logic is capable of detecting such an error condition. The following sequence is used to test the SKI logic. 1. Clear the errors. 2. Recalibrate (positioner to cylinder 0). 3. Seek to cylinder 1 (positioner to cylinder 1). 4. Load 400 into the CCY (hardware now out of synch). 5. Issue a seek to cylinder 1. This should cause an SKI error. The positioner is already over cylinder 1 but it thinks a 4-cylinder reverse seek must be done because CCY is sitting at 400. 6. Read error register and CCY register. 7. Verify that SKI has occurred. 8. Verify that CCY is now clear. This is true because SKI causes the drive to initiate an automatic recalibrate pulse which causes a RESET REGISTERS to occur and reset CCY if the sequence works correctly.
271*	A TEST OF SUBTRACTOR DIFFERENCE LOGIC This set of two subtests causes seeks with all possible positive and negative subtractor differences. Philosophy: These are functional tests designed to detect but not diagnose a class of seek control logic faults. Since the output of the difference logic is not visible, this test has to be functional. If we slip position during the test sequence, the error will accumulate and at some point in the sequence, force the positioner to one of the guard bands causing a SKI to occur. Unfortunately, the only thing that can be said about a fault detected in this test is that a positioner malfunction exists. The following test sequence is used. Subtest 1 1. N = 1 2. RHCLR 3. Recalibrate. 4. Seek to CYL-N and test for error. 5. Seek to CYL-0 and test for error. 6. N = N + 1 7. Done if N = 814. If not done go to step 4. Subtest 2 1. N = 813 2. RHCLR 3. Recalibrate. 4. Seek to CYL-814. 5. Seek to CYL-N and test for error. 6. Seek to CYL-814 and test for error. 7. N = N - 1 8. Done if (N < 0). Otherwise go to step 5. Due to the way in which this test could fail, the standard scope loop technique is not used. Upon detecting an error, this test will report, clear and remember the fact that an error was committed. After cycling through the entire test, it will be repeated if loop on error has been selected and an error was detected somewhere along the way.
272*	VERIFY THAT A NO-OP COMMAND IS OPERATIONAL Issue a NO-OP command in maintenance mode and ensure that the GO bit is not set. This gives some confidence that the command was not decoded incorrectly.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
273*	VERIFY OPERATION OF OPI ERROR LOGIC The following sequence is used to verify part of the OPI error detection logic. 1. Issue a master clear (to clear errors). 2. Issue a recalibrate (synchronize positioner). 3. Go into maintenance mode. 4. Issue a search mode for surface = 0 sector = 0. 5. Issue an index. 6. Issue an index. 7. Verify that OPI is not set too early. 8. Issue an index. 9. Verify that OPI did set.
274	VERIFY THAT RMR CAN OCCUR Cause an RMR condition and verify that the drive responds correctly using the following sequence. 1. Issue a master clear. 2. Issue a recalibrate. 3. Go into maintenance mode (allows seek to hang). 4. Seek to cylinder 1 (sets GO bit). 5. Write 0 to DCY (should cause RMR). 6. Read R1 and DCY. 7. Verify that RMR occurred. 8. Verify that DCY was not loaded.
275*	VERIFY THAT WRITING ATA DOES NOT CAUSE RMR Write the attention summary register with the GO bit set and verify that RMR does not occur. The sequence is as follows. 1. Issue master clear. 2. Recalibrate. 3. Go into maintenance mode (allows seek to hang). 4. Seek to cylinder 1 (sets GO bit). 5. Write the ATA register with 0s. 6. Read ER1. 7. Verify that RMR did not set.
276*	BASIC SEARCH FUNCTION IN MAINTENANCE MODE This test is a basic verification of the search command decode. The following test sequence is used. 1. Issue a master clear, (to clear errors). 2. Recalibrate (causes CCY = 0). 3. Set the DA register for cylinder = 0. 4. Go into maintenance mode (no sector pulse freezes command). 5. Issue a search. 6. Read control, status, and error registers. Test for correct status: GO bit = 1 PIP = 0 Composite error = 0
277	OPERATION WHILE SEARCHING FOR ILLEGAL SECTOR This test issues a search for an illegal sector. The status of the drive is then checked to verify that performance was correct under this set of conditions. The search should terminate immediately because of IAE and the GO bit should reset via the class B error reset condition. The following test sequence is used. 1. Clear errors. 2. Recalibrate (sets CCY = 0). 3. Set DCY = 0. 4. Set DA register to track = 0 sector = 30. 5. Go into maintenance mode. 6. Issue a search command. 7. Read device registers and verify status.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
300*	SEARCH FOR SECTOR 0 This is the first full-speed search command. A functional search for sector = 0, cylinder = 0, surface = 0. The following sequence is used. 1. RHCLR 2. Recalibrate. 3. Set DA = DCY = 0. 4. Search. 5. Wait until done. 6. Read device registers.
301	SEARCH COMBINED WITH IMPLIED SEEK Verify that a search causes an implied seek by looking at the current cylinder address register when completed. Except for the implied seek to cylinder number 1, this test is identical to the previous test.
302	VERIFY OPERATION OF THE SECTOR COMPARATOR This test cannot run in user mode because system scheduling will cause it to fail. Search for all possible sectors in 22-sector mode. This test searches for all sectors from every other sector on the disk. The following algorithm is used. 1. RHCLR 2. Recalibrate. 3. Synch = 0 (a synch point to start search from). 4. .SECT = 0 (sector = 0, track = 0) 5. RHCLR (clear errors) 6. DCY = 0 (clear desired cylinder address) 7. DA register loaded with .SECT 8. Keep reading look-ahead register until sector is found (synch). 9. Now in synch. Issue a search command. 10. Wait until done. Then read the look-ahead register. If error, report and loop to step 5. 11. Verify that the look-ahead register = .SECT. If error, report and loop to step 5. 12. .SECT = .SECT + 1 If .SECT $\leq$22., go to step 5. 13. Synch = synch + 1. Advance to next synch point. If synch $\leq$22. go to step 4 (next permutation). 14. Done with test (approximately 484 searches).
303*	TEST OPERATION OF WRITE LOCK SWITCH Verify with the status register and operator intervention that the write lock switch works correctly.
304*	TEST OPERATION OF UNLOAD/STAND-BY Verify through operator intervention and status information that the unload command works correctly.
305*	ISSUE A 0 MICROINCH OFFSET A full-speed offset operation to test for a hung device or a composite error caused by offsetting. The test sequence is as follows. 1. Master clear 2. Recalibrate. 3. DCY = 0 4. DA = 0 5. OFFSET = 0 6. Issue an offset command. 7. Wait until done and test for errors.
306*	FULL SPEED RETURN TO CENTERLINE TEST Test to verify that the return to centerline command does not cause a composite error or hung device. The sequence is as follows. 1. Issue a master clear. 2. Recalibrate. 3. DCY = DA = OFFSET = 0 4. Issue two return to centerline commands. The first will take drive out of offset mode if necessary. 5. Wait until done and verify that device did not hang and composite error did not set.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
307*	ISSUE THE POSSIBLE OFFSETS Issue the possible offsets and return to centerline commands at full speed to check for hung device or composite error. The test sequence is as follows. 1. Issue a master clear. 2. Recalibrate. 3. DCY = DA = 0 4. Load the offset register with a value. 5. Issue an offset command. 6. Verify that no error occurred. If there is an error, loop back to step 1 after reporting. 7. Issue a return to centerline command. 8. Verify that no error occurred. If there is an error, loop back to step 1 after reporting. 9. Update offset value from table. If not finished, go back to step 1.
310*	ISSUING A SEEK WHILE IN OFFSET MODE If the drive is in offset mode when a seek is requested, it is supposed to postpone the seek, issue an automatic return to centerline, and then proceed with the seek. The logic to do this is on (SS0). There is no way to guarantee that this operation works correctly, but this test does the operation looking for hung devices or composite error being caused by this operation. The following test sequence is used. 1. Issue a master clear. 2. Recalibrate. 3. DCY = 0 = DA. 4. Offset = 600 microinches 5. Issue an offset command. 6. Seek to cylinder number 1. 7. Wait until done. 8. Read the device registers. 9. Verify that: a. Device did not hang b. Composite error did not set c. Seek took place (CCY = 1).
311*	VERIFY THAT ATA IS NOT STUCK SET Verify that the ATA flip-flop is not stuck at 1. The following test sequence is used. 1. Issue a master clear (should reset ATA). 2. Write ATA register with 377 (should reset ATA). 3. Read the status register. 4. Verify that ATA is clear.
312	VERIFY THAT ATA CAN BE SET The following test sequence is used. 1. Issue a master clear. 2. Write the ATA register with 377. 3. Issue recalibrate (causes ATA). 4. Read the status register. 5. Verify that ATA is set.
313	VERIFY THAT RHCLR WILL RESET ATA 1. Issue a master clear (clear ATA). 2. Write the ATA register with 377 (clear ATA). 3. Issue recalibrate (sets ATA). 4. Issue a master clear (RHCLR to clear ATA). 5. Read the status register. 6. Verify that ATA is clear.
314	VERIFY RESITTING OF ATA BY WRITING REGISTER 04 The following test sequence is used. 1. Issue a master clear (clears ATA). 2. Issue recalibrate to set ATA. 3. Write ATA with proper bit for drive being tested. 4. Read the status register. 5. Verify that ATA is clear.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
315	VERIFY OPERATION OF PSEUDO-ATA POSITION It is already verified that writing ATA register with 377 will reset the ATA flip-flop. This is a test to verify the position decode. The idea is to cause an attention and then write all bits of the ATA register except the drive asserting attention. If position decoder is working, ATA will still be set. The following sequence is used. 1. Issue a master clear. 2. Recalibrate (causes ATA). 3. Write the ATA bits for drives not under test. 4. Read the status register for drive being tested. 5. Verify that ATA has not cleared.
316*	BASIC ATA REGISTER READ CYCLE The following sequence is used. 1. Issue a master clear. 2. Write ATA with 377 (clears all ATAs in all drives). 3. Read the status registers for all drives into cores. 4. Read ATA summary register. 5. Verify that ATA summary register is 0.
317	TEST FOR PROPER PSEUDO-ATA POSITION This test verifies that when the test drive asserts ATA, it comes back in one and only one position in the ATA register and also that the position is correct. The following test sequence is used. 1. Issue a master reset. 2. Recalibrate (to set ATA). 3. Read the ATA register 4. Verify that it is correct: a. ATA is asserted for this drive in proper position. b. No other ATAs asserted (multiple position decode).
320	VERIFY THAT SK GO CLR SETS ATA Verify that the completion of a seek sets ATA. The following sequence is used. 1. Issue a master reset (clears errors). 2. Recalibrate. 3. Issue a master reset (resets ATA). 4. Seek to cylinder 0 (causes ATA). 5. Read the status register. 6. Verify that ATA is set.
321	VERIFY THAT SRCH COMPL WILL SET ATA Verify that the completion of a search will set ATA using the following sequence. 1. Issue a master reset (clear errors). 2. Recalibrate. 3. Issue a master reset (clears ATA). 4. Issue a search for cylinder 0, sector 0, surface 0. 5. Read the status register. 6. Verify that ATA is set.
322	VERIFY THAT SETTING GO WHILE ERR CAUSES ATA Verify that attempting to set the GO bit while the drive has a composite error causes ATA to be set. The following sequence is used. 1. Issue a master reset (clears errors). 2. Recalibrate. 3. Write a DCK error into ER1. 4. Write 377 to ATA register to clear ATA. 5. Issue a NO OP with the GO bit set (should set ATA). 6. Read status register and verify that ATA was set.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
323	VERIFY THAT ATA CAN BE CLEARED WHILE ERROR PERSISTS The following test sequence is used. 1. Issue a master reset (clears errors). 2. Recalibrate (sets ATA). 3. Write a DCK error to error register no. 1. 4. Write the ATA register to reset ATA flip-flop. 5. Read the status register. 6. Verify that ATA is cleared.
324	ATA BEHAVIOR FOR CYCLE ON/OFF, UP/DOWN This test comprises 3 subtests designed to check the response of the ATA logic to various cycle states of the drive. Subtest A 1. Ensure that device is on-line. 2. Recalibrate. 3. Issue an unload (uses OFFCHK routine). 4. Read status and verify that ATA was not set. Subtest B (Exec Mode Only) 1. Power the drive off, then on, but do not load heads. 2. Read the status register. 3. Verify that ATA was set by power-up. 4. Verify that AC LOW is asserted in ER3. 5. Issue a drive clear (uses Massbus clear). 6. Verify that AC LOW is cleared. Subtest C (Exec Mode Only) 1. Verify that the heads are unloaded (via OFFCHK). 2. Load Ready (via ONCHK). 3. Read the status register. 4. Verify that the transition of MOL has set ATA.
325	DATA TRANSFER COMMAND DECODE CHECKS This test verifies that the data transfer commands are decoded in the drive. The following test sequence is used for each for the six data transfer commands available. 1. RHCLR (to clear errors) 2. Recalibrate. 3. DA = DCY = 0 (a valid address). 4. Go into maintenance mode (20-sector mode). 5. Issue a data transfer command. 6. Snapshot appropriate drive registers. 7. Abort the transfer. 8. RHCLR 9. Verify that GO = 1, PIP = 0, ILF = 0. 10. Repeat test for all data transfer commands.
326	TEST OF DATA BUS CONTROL SIGNALS This test issues a controlled sequence of events to check the following. 1. OPI logic. 2. Run line and command decode for write header and data. 3. Occupied line (RH10 only). 4. Exception logic. 5. EBL logic. The test is generally structured as follows (ten subtests). 1. Start a maintenance mode write data command. 2. Issue index pulses to verify OPI logic. When OPI works correctly, the command is decoded correctly and run must have been asserted by the controller. 3. When the OPI condition occurs, a class B error is caused which will force the transfer to shut down. By observing the states of the DA register, done, and exception, the diagnostic is able to isolate control line faults to the proper modules.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
327*	VERIFY OPERATION OF WRT HD + DAT UP TO DATA FIELD This is a series of eight subtests that verify operation of the write header and data command from the preheader gap to the start of the data field. The test completes at the start of the data field in order to minimize the size of the scope loop. The subtests consist of these procedures. 1. Check 39 bytes of 0s in the sector gap. 2. Check for proper sync byte. 3. Check that the eight bytes of header and key words are 0 since the transfer is to cylinder 0, surface 0, sector 0. 4. Check that header CRC bytes are 0. 5. Check that the 11 bytes of 0s in the head gap are correct. 6. Check that the post header synch byte is correct. 7. Verify that the data envelope is not up too soon. 8. Verify that the data envelope is up on time.
330*	CHECK FOR STUCK 1S AND 0S ON DATA BUS Issue a maintenance mode write header and data command. Check that the data bits on the Massbus going from controller to drive are not stuck at 1s or 0s. 1. Start the transfer. 2. Issue 496 shift clocks to step to data area. 3. Clock first 18 bits out and verify that they are 0. 4. Clock second 18 bits out and verify that they are 1s.
331	CHECK OF WRITE DATA PATH USING FLOATING PATTERNS Verify that there are no bits cross-coupled on the write data bus by testing data integrity with floating patterns. 1. Start a maintenance mode transfer of 1 sector. 2. Issue 962 clocks to step to data area. 3. Issue 36 clocks to step over first PDP-10 word. 4. Start clocking PDP-10 half words out of memory (floating patterns), each time verifying that data was correct. 5. Repeat step 4 until 18 PDP-10 words have been tested. 6. Abort the transfer.
332*	TRANSFER STANDARD DATA PATTERNS CHECKING FOR PARITY This test passes 52 (18-bit) data patterns across the Massbus and checks for parity errors after each transfer.
333	Not implemented.
334	VERIFY SECTOR TIMING OF DATA ENV, ECC ENV, EBL Test clocks through an entire sector in maintenance mode to verify that the above signals transition at exactly the right time. Eight subtests do the job as follows. Clocks / DATA ENV / ECC ENV / DONE 1. 495 / 0 / 0 / 0 2. 1 / 1 / 0 / 0 3. 4607 / 1 / 0 / 0 4. 1 / 0 / 1 / 0 5. 31 / 0 / 1 / 0 6. 1 / 0 / 0 / 0 7. 15 / 0 / 0 / 0 8. 1 / 0 / 0 / 1 Testing around transition points optimizes test execution time but the diagnostic cannot tell exactly how much timing is incorrect. This would make the test unreasonably long.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
335	CAUSE A DRIVE TIMING ERROR (DTE) This test verifies that the drive can detect a drive timing error. The error is forced in the following way. 1. Start a 1-section maintenance mode write header and data command. 2. Cause a sector pulse (which causes a drive timing error). 3. Read ER1 and verify that DTE is set.
336	CHECK HEADER LOGIC WHILE SUPPLYING CORRECT DATA Start a 1-sector maintenance mode read header and data operation. Clock through the header area supplying correct data and verify the following. 1. Supply 312 0s for preheader gap. 2. Supply 8-bit synch char 31 (octal). 3. Supply 80 0s for header field. 4. Supply 88 0s for post-head gap. 5. Supply 8-bit synch char 31 (octal). 6. Verify that hardware status is correct.
337	TEST HEADER CRC LOGIC Issues a maintenance mode read operation and supplies wrong CRC data. This should cause HCRC to be detected. 1. Send 312 0s for header gap. 2. Send 8-bit sync byte. 3. Send 64 0s for header and key words. 4. Send 8 0s for first half of header CRC. 5. Send 8 1 bits (wrong data) for rest of header CRC. 6. Send 8 0s to move slightly into gap. 7. Read status and verify that HCRC is set.
340	TEST HEADER COMPARE LOGIC Verify that the drive can detect header data errors. This is done by supplying the header data in a maintenance mode read header and data operation. The diagnostic supplies the wrong cylinder information. This will result in a HCE and a HCRC error. The test outline is as follows. 1. Send 312 0s for preheader gap. 2. Send 8-bit synch byte. 3. Send 8 bits of 1s for wrong cylinder information. 4. Send 72 bits of 0s for rest of header data and header CRC word. 5. Verify that HCE was detected. The HCRC error is incidental to this test.
341	TEST OF FER ERROR DETECTION LOGIC Verify that a format error can be detected. Start a maintenance mode read header and data operation in 20-sector mode, then supply the header data with the 22-sector FMT bit set. This should cause a format error (FER). HCRC and HCE are incidental to this test. The test outline is as follows. 1. Send 312 0s for preheader gap. 2. Send 8-bit synch byte. 3. Cause error in next two bytes: a. 12 bits of 0s b. 1 bit of 1 for 11-format c. 3 bits of 0s to fill out last byte. 4. Send 64 bits of 0s to complete header area. 5. Verify that FER has occurred. HCE and HCRC may also appear but are insignificant in this test.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
342	TEST OPERATION OF HCI LOGIC Test initiates a read header and data command in maintenance mode (20-sector mode) with the header-compare inhibit bit (HCI) set. The data supplied by the diagnostic supplies incorrect header, format, CRC information. After completing transfer through the header area, status is checked. HCE, FER, HCRC should all be 0. The test outline is as follows. 1. Send 312 0s for preheader area. 2. Send 8-bit synch byte. 3. Send 8-bit byte with wrong cylinder (binary = 00000001). 4. Send 8-bit byte with wrong FMT (binary = 00010000). 5. Send 64 0s which completes header area with wrong CRC. 6. Verify that HCE = FER = HCRC = 0.
343	VERIFY THAT SYNCH DETECTOR NOT STUCK HIGH Start a maintenance mode read header and data operation. The diagnostic supplies all 0s for data area. After 512 clocks (data of 0s) and data ENV bit is examined. It should be at 0. If it is at a 1, the synch byte detector has erroneously detected a synch byte.
344	VERIFY THE READ DATA PATH INTEGRITY Supply necessary data patterns via a maintenance mode read header and data operation and check data that appears in the RH buffer register against the expected data (18 bits at a time). Data checking is not performed until we step to the data field. Once at the data field, the following four subtests are performed. 1. 1 Massbus transfer of 0s. 2. 1 Massbus transfer of 1s. 3. 36 Massbus transfers of floating patterns. 4. 8 Massbus transfers of 0s and parity patterns.
345	CHECK DRIVE PARITY NETWORK Send various data patterns from drive to controller over the Massbus data path using a maintenance mode read header and data operation. There are two subtests. 1. Sends 1s and 0s. Possible faults include parity network and parity transmit faults. 2. Supply other data patterns to verify parity generation logic in drive. Faults only include parity generator.
346*	TEST ATA TRANSMITTER FOR STUCK AT 1 Verify that no drive on the bus has an attention transmitter stuck at 1, (constantly asserting ATA). 1. Issue RHCLR. 2. Write the ATA register with 377 to clear ATA flip-flops. 3. Verify that ATA (CONI DATA) is not asserted.
347	TEST THAT DRIVE SENDS ATA LINE TO RH Issue a recalibrate and verify that ATA (CONI DATA) is asserted.
350	VERIFY THAT ECC GENERATION WORKS CORRECTLY In 18-bit mode, issue a write header and data command, issue enough clocks to step up to the ECC field, clock out and read the next 32 bits of ECC data and verify that it is correct.
351*	FULL-SPEED WRITE HEADER AND DATA TEST IN 18-BIT MODE This is a full-speed write header and data operation to cylinder 0 surface 0 sector 0. This transfer has worked previously in maintenance mode. This test uses disk clocking and the MDLI interface logic for the first time. Because of the variety of faults that could turn up in this test, module callout may not be optimum.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
352	VERIFY THAT EACH HEAD CAN BE SELECTED Issue a full-speed write header and data to the first sector (sector 0) on every surface of cylinder 0. This is the first time any head other than 0 has been selected and the hope is that only head-related problems show up (NHS, MHS, WCU, etc.). OPI errors could possibly come up during this test also. This would be an indication of a faulty track comparitor on SS5.
353*	DISK ADDRESS LOGIC (INCREMENTATION) TEST Using full-speed write header and data tests, verify that the various disk address registers can increment properly.
354*	VERIFY THAT LBT DETECTOR WORKS Do a 1-sector write header and data to the last block on the disk (cylinder = 814, surface = 18, sector = 19). At the end of the transfer, verify that LBT has set and that DA = 0 and DCY = 815.
355*	VERIFY THAT AOE CAN BE DETECTED Do a full-speed write header and data operation to the last disk sector (sector 23 surface 20 cylinder 1456). The transfer will be two sectors long, which will overflow the disk and cause AOE.
356	WRITE/READ HEADERS AT FULL SPEED Write, then read headers and data at full speed using cylinder 0, surface 0, sector 0. This is a functional test, so module callout will be impossible. This is the first time headers and data have been read at full speed. The operation has worked previously in maintenance mode.
357	WRITE THEN READ HEADERS FROM VARIOUS ADDRESSES Write, then read/verify headers from various disk areas. During the read, HCI is set to allow header recovery.
360	WRITE THEN READ HEADERS FROM VARIOUS ADDRESSES This test after writing headers reads them with HCI = 0. The previous test has just checked to see that headers were read and written correctly. This test will look for header errors HCE, FER. Either of these would be an indication that comparitor logic on the SS board is faulty and cannot match headers and/or formats.
361*	FUNCTIONAL READ WRITE TEST Full speed exercise at first disk address. 1. Write headers and data. 2. Read data.
362	VERIFY SECTOR TIMING OF DATA ENV, ECC ENV, EBL Test clocks through an entire sector in maintenance mode (16-bit) to verify that the above signals transition at exactly the right time. Eight subtests do the job as follows. **Clocks — DATA ENV — ECC ENV — DONE** 1. 495 — 0 — 0 — 0 2. 1 — 1 — 0 — 0 3. 4095 — 1 — 0 — 0 4. 1 — 0 — 1 — 0 5. 31 — 0 — 1 — 0 6. 1 — 0 — 0 — 0 7. 15 — 0 — 0 — 0 8. 1 — 0 — 0 — 1 Testing around transition points optimizes test execution time but the diagnostic cannot tell exactly how much timing is incorrect. This would make the test unreasonably long.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
363	LOGICAL ADDRESS PLUG TEST This test checks for the proper drive response when the logical address plug is removed and reinserted. The expected response is as follows. With plug removed, handshake fails, can no longer access the drive. With plug reinserted: 1. ATA = 1 2. MOL = 1 3. OPE = 1 4. VV = 0.
364	Not implemented.
DUAL 1	VERIFY THAT PROGRAMMABLE MODE OPERATES CORRECTLY Read and verify that both ports' status registers yield programmable bit. This test also fails if reading the status register captures the port. It should not do so.
DUAL 2	TEST OPERATION OF CAPTURE AND RELEASE ON PORT A The test is designed to do the following. 1. Go into neutral. 2. Seize port A with read of control register. Verify with a read of the port B status. 3. Wait approximately 1/2 second and verify (by reading port B status) that drive has not yet dropped into neutral because of a 1-shot fault. 4. Wait approximately 1 second more and verify that the drive has gone into neutral (the 1-shot did time out) causing automatic release.
DUAL 3	TEST OPERATION OF CAPTURE AND RELEASE ON PORT B This test is designed to do the following. 1. Go into neutral. 2. Seize port B with read of control register. Verify with a read of port A status. 3. Wait approximately 1.5 second and verify that the drive has gone into neutral (the 1-shot did time out) causing automatic release.
DUAL 4*	VERIFY ABILITY TO HOLD PORT A INDEFINITELY Repeatedly capture port A for some period of time greater than 1.5 second and verify that the port stays captured for the entire time. The following sequence is used. 1. Go into neutral. 2. Capture port A by reading the control register. 3. Read port B status and verify that drive has port A captured. 4. Repeat steps 2 and 3 for a period of time longer than the 1-shot time (approximately 2 seconds).
DUAL 5*	VERIFY ABILITY TO HOLD PORT B INDEFINITELY Repeatedly capture port B for some period of time greater than 1.5 seconds and verify that the port stays captured for the entire period of time. The following sequence is used. 1. Go into neutral. 2. Capture port B by reading the control register. 3. Read port A status and verify that port B has been captured. 4. Repeat steps 2 and 3 for a period of time longer than the 1-shot time (approximately 2 seconds).
DUAL 6*	VERIFY THAT READING REGISTER 00 ONLY CAUSES PORT A CAPTURE 1. Go into neutral. 2. Read each of the Massbus registers (excluding control register) and after each read, verify that port A has not been captured by checking status on port B.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
DUAL 7*	VERIFY THAT READING REGISTER 00 ONLY CAUSES PORT B CAPTURE 1. Go into neutral. 2. Read each of the Massbus registers (excluding control register) and after each read, verify that port B has not been captured by checking the status on port A.
DUAL 10*	VERIFY THAT WRITE TO OTHER ATAS DOES NOT CAPTURE PORT Write ATA registers for all drives except the one under test and verify that no port has been captured.
DUAL 11	CAPTURE PORT A BY WRITING ITS CONTROL REGISTER Writing any register will capture a port if drive is in neutral (the selection of the control register is arbitrary). 1. Go into neutral. 2. Write control register on port A. 3. Read status from port B and verify that port A was captured.
DUAL 12	CAPTURE PORT B BY WRITING ITS CONTROL REGISTER Writing any register will capture the port if drive is in neutral. The selection of the control register is arbitrary. 1. Go into neutral. 2. Write the control register on port B. 3. Read status from port A and verify that port B was captured.
DUAL 13	VERIFY THAT RELEASE FROM PORT A DOES NOT SET ATA With no request on port B, a release from port A should not set the ATA flip-flop on port B. 1. Go into neutral with ATAs for both ports clear. 2. Read control register for port A to capture. 3. Wait until we flip back into netural. 4. Verify that ATA has not been set for port B.
DUAL 14	VERIFY THAT RELEASE FROM PORT B DOES NOT SET ATA With no request on port A, a release from port B should not set the ATA flip-flop on port A. 1. Go into neutral with ATAs for both ports clear. 2. Read control register for port B to capture. 3. Wait until we flip back into neutral. 4. Verify that ATA has not been set for port A.
DUAL 15	RELEASE, WITH RELEASE COMMAND, FROM PORT A Verify that we can release from port A with a release command and that issuing the release command does not cause composite error. 1. Go into neutral. 2. Read control register on port A to capture it. 3. Issue a release command. 4. Read status register on port B and verify that release worked correctly and did not cause composite error.
DUAL 16	RELEASE, WITH RELEASE COMMAND, FROM PORT B Verify that we can release from port B with a release command and that issuing the release command does not cause composite error. 1. Go into neutral. 2. Read control register from port B to capture the port. 3. Issue a release command. 4. Read status register on port B and verify that release command worked correctly and did not cause composite error.

*Listing on microfiche contains troubleshooting procedure.

Table 2 DSRPA Test Summary (Cont)

Test	Description
DUAL 17	FIRST TEST TO VERIFY REQUEST HOLD LOGIC 1. Go into neutral and reset all ATAs. 2. Capture port B by reading port B control register. 3. Request port A by writing port A control register. 4. Read port A status and verify that it is not captured. Port B should be in possession for 1 second. 5. Wait for greater than a second to give auto-release from port B a chance to happen. 6. Read status from port A and verify that ATA has set (because of the request from step 3).
DUAL 20	SECOND TEST TO VERIFY REQUEST HOLD LOGIC 1. Go into neutral and reset all ATAs. 2. Capture port A by reading port A control register. 3. Request port B by writing port B control register. 4. Read port B status and verify that it is not captured. Port A should be in possession for 1 second. 5. Wait for greater than a second to give auto-release from port A a chance to happen. 6. Read status from port B and verify that ATA has set (because of the request from step 3).
DUAL 21	CHECK AUTO UNLOCK FEATURE FROM PORT B Verify that the test does not hang on a port if it is requested but not used. 1. Clear all ATAs and go into neutral. 2. Capture port A by reading the control register. 3. Request port B by reading its control register. 4. Wait approximately 3 seconds. Drive should time out on port A, flip to port B because of the request, timeout on port B and flip back to neutral. 5. Read status and verify that the tests are not hung on one of the ports.
DUAL 22	CHECK AUTO UNLOCK FEATURE FROM PORT A Verify that the test does not hang on a port if it is requested but not used. 1. Clear all ATAs and go into neutral. 2. Capture port B by reading the control register. 3. Request port A by reading its control register. 4. Wait approximately 3 seconds. Drive should timeout on port B, flip to port A because of the request, timeout on port A and flip back to neutral. 5. Read status and verify that the test is not hung on one of the ports.
DUAL 23	VERIFY THAT RHCLR WILL NOT RESET ATA IN NEUTRAL 1. Cause ATA on port A (recalibrate). 2. Cause ATA on port B (recalibrate). 3. Issue RHCLR. 4. Read status from both ports and verify that neither ATA bit has been reset.
ALIGN	HEAD ALIGNMENT VERIFICATION TEST This test will operate on all drives under test. The overall head alignment for a drive is verified in the following manner. 1. Recalibrate the positioner. 2. Seek to cylinder 496. 3. Verify the alignment of heads 0 through 18. 4. Seek to cylinder 8. 5. Verify the alignment of heads 0,1 and 17,18. 6. Seek to cylinder 800. 7. Verify the alignment of heads 0,1 and 17,18. 8. Seek to cylinder 496. 9. Reverify the alignment of heads 0 through 18. The following algorithm is used to verify the alignment of a particular head.

Table 2 DSRPA Test Summary (Cont)

Test	Description
	1. Offset the positioner to +600 microinches. 2. Record the value of the sign bit. 3. Move the positioner towards the track centerline in 25 microinch increments until the sign bit changes. Upon detecting a sign change, record the offset value. 4. Offset the positioner to -600 microinches and repeat steps 2 and 3 above. 5. Average the two values of offset and consider the head to be out of alignment if the average offset has turned out to be in excess of 75 microinches (absolute value) on cylinder 496, 150 microinches on cylinders 8 and 800. NOTES 1. Positive offsets indicate a head position toward the spindle side of the actual track centerline. 2. By default, normal printout mode (SW 10 reset), the program will print a complete table of offsets for all heads. In short printout mode (SW 10 set), information will be printed for only those heads which are out of alignment. 3. The off-line tester must be interfaced. 4. An industry-standard alignment pack is required. 5. The test has two parts: adjust mode and verify mode. If the question (adjust or verify) is answered with adjust, the program does not test head alignment, but merely furnishes a method (self-documenting) of positioning back and forth between cylinders 496 and 0. Cylinder 496 is used to scope the DIE bits and cylinder 0 to adjust the head.

SETUP PROCEDURES:

1. Install a CE pack with drive write-protected.
2. Slide the logic assembly forward.
3. Remove drive dc power.
4. Connect the head alignment unit (PN-211292) interface cable or tester's head alignment cable to the B04 slot of the logic assembly.
5. Turn drive dc power back on and load heads.
6. Press the calibrate switch on the head alignment unit or the tester's control panel.
7. Verify that the tester's ALIGN INVALID lamp is illuminated. If the setup does not work, refer to the appendix of the maintenance manual for possible faults and solutions.
8. To prevent index error, place a jumper between D04-R19 and D04-R39. This will prevent index errors. The jumper is to be removed when the head alignment test is complete.

THERMAL EQUILIBRIUM REQUIREMENTS (MINIMUM):

1. With the wind tunnel in the closed position, the drive must operate in track following or seek mode for 20 minutes. Any pack may be used for the first 15 minutes but the CE pack must be mounted for the last 5 minutes.
2. The CE pack must reach equilibrium by being in the computer room for 1 hour, or by being used for the entire 20 minute cycle described in the first step.

GENERAL INFORMATION

Code — DSTUA.SAV

Title — DECSYSTEM-2020 RH11-TM02/03-TU45/TU77 Basic Diagnostic

Abstract — DSTUA is the RH11-TM02-03-TU45 Massbus magtape system basic diagnostic written to run on the DECSYSTEM-2020. The diagnostic will test up to 64 drives - the maximum number that can be connected to a single RH controller. This diagnostic tests multiple drives serially; that is, the program runs one drive through a set of tests, and then runs another drive through the same set of tests, and so on.

Hardware Required

KS10 Mainframe
Minimum of 48K of memory
RH11-TM02/TM02 - up to four TU45s

Preliminary and Associated Programs — Refer to diagnostic hierarchy (KS10 STD module).

Restrictions — All drives to be tested must be connected to the same RH and each TM02 or TM03 must have at least one slave attached.

Notes

1. To test multiple drives on a single RH simultaneously, (using the interrupt system) run the multi-drive exerciser - the RH11-TM02/03-TU45/TU77 Magtape Diagnostic Exerciser program (DSTUB).

2. Each part of the diagnostic will run with preassigned defaults (record size, data mask, etc.). To change the defaults, set the OPRSEL switch (14), and start the test. Questions concerning the relevant parameters will be asked. Parameters that have been changed will remain in effect until changed. Note that if the record size has been changed in part B1, the record size will also be changed for part B2. The same is true for data mask, optional data pattern, etc. If the same question is asked before either part, the response typed will affect both parts. In exec mode, if a response is not received for each question within a few minutes, the program will timeout and assume the default.

3. Run times are in minutes except as noted (all switches 0).

	TM02/TU45	TM03/TU45
Part B1	1.5/3	2/3.5 (cache on/off)
Part B2	3	3.5

Loading and Starting Procedure

Exec Mode Operation

1. Mount tape of known quality on the transports to be tested.

2. Place the transports to READY, WRITE ENABLE, and REMOTE.

3. Set the appropriate data switches.

4. Refer to the standard loading and starting procedure in the KS10 STD module.

Control Switches — Refer to Table 1.

OPERATIONAL CONTROL

Manual Drive Configuration
The first configuration question printed after starting the diagnostic will be:

SPECIFY TM (3=TM03, 2=TM02, CR WILL TEST ALL ON-LINE DRIVES)

TM02 If a 2 is typed the next question printed will be:

TM02 #'s (CR WILL TEST ALL ON-LINE DRUS):

In response to this question, the TM to be tested will be entered by its number (0-7). If more than one, they must be separated by commas. For each TM02 typed in, the following question will be printed:

TM02 #X, SLV TYPE (TU16, TU45, or TU77):

Respond by typing in the drive type. The next question printed will be:

TM02 #X, SLAVE #'s

In response to this question, type the slave number (0 to 7 for TU45 or 0 to 3 for TU77). If more than one, separate them by commas.

TM03 If a 3 is typed the next question printed will be:

TM03 #'s (CR WILL TEST ALL ON-LINE DRUS):

In response to this question, the TM to be tested will be entered by its number (0-7). If more than one, they must be separated by commas. For each TM03 typed in, the following question will be printed:

TM03 #X, SLV TYPE (TU16, TU45, OR TU77):

Respond by typing in the drive type. The next question printed will be:

TM03 #X, SLAVE #'s

In response to this question, type the slave number (0 to 7 for TU45 of 0 to 3 for TU77), if more than one, separate them by commas.

Automatic Drive Configuration
If just a carriage return <CR> is typed in answer to the first question, the program searches RH11 (3772440) until an on-line drive is found. The program then enters in the configuration and all on-line drives.

The program will automatically determine whether the controller on the RH11 is a TM02 or TM03 based upon the value of the drive type register bit 5 (0 = TM02, 1 = TM03).

NOTE
This auto-mode should not be used unless you are reasonably confident that the drive type bits work properly. The program will not automatically enter in the configuration any on-line drive on which an error occurs in the read register routines.

PART B1 - CONTROLLER TESTING
Before running this part of the diagnostic, make sure all the drives are idle (not rewinding, etc.) and are at or near BOT.

If the OPRSEL switch (SW 14) is on, the following questions will be asked (with the following defaults):

1. FIXED RECORD SIZE (4-1000(8), CR = 100):

On all fixed record tests in the diagnostic, the number typed here will be used for the number of words per record. Typing a number less than 4 will give you 100 octal (the default); typing a number greater than 1000 will give you 1000 octal.

2. DATA COMPARE MASK (CR = CHECK ALL BITS):

   The data compare mask is a 36-bit octal number indicating what bits the program is to check when doing data compare. Type 0 to ignore data altogether. Type -1 or carriage return to check all bits (the default). A 1 in a bit position means that bit is required to be correct. In compatible tape format mode, the last four bits are always printed AS-SI, but are always ignored during data compare.

3. OPTIONAL DATA PATTERN:

   The 36-bit pattern typed here will be used as one of the patterns during the wraparound tests, basic tape-motion tests, and reliability. If a carriage return is typed, the most recent optional data pattern will be printed (initially 123456,654321).

4. IS THE OPI TIME EXTEND ECO IN - Y OR N <CR>

   This question allows or disallows Test 75 to be run (the inital default is N.) If this is a working ECO'ed TM02, request Test 75 to be run.

Part B1 tests mostly the TM02 or TM03 controller, and is composed of tests 1 through 77. Tests 1 and 2 are run once per pass, whereas tests 3 through 77 are run first on the TM02 or TM03 with the lowest address in the configuration. Then tests 3 through 77 are run on the TM02 or TM03 with the next to the lowest address, and so on, until all the TM02s or TM03s have been tested.

Note that even though the tests only test the TM02 or TM03, there must be a slave on the TM02 or TM03 under test and it must be on-line and write-enabled. The controller needs several signals from a slave to work properly: SWC (slave write CLK), MOL (on-line), WRL (write-locked), and BOT (beginning of tape).

After all TM02s or TM03s have been tested, one pass of Part B1 has been run. If the abort switch (SW 0) is not set, another pass of part 1 is begun. If abort is set, control returns to the console software package. For descriptions of each test, see Table 2.

NOTE

All tests described in Table 2 are not necessarily done. This depends on the values of assembly switches and options selected.

PART B2 - SLAVE TESTING

Before running this part of the diagnostic, make sure all the drives are idle (not rewinding, etc.) and are at or near BOT.

If the OPRSEL switch (SW 14) is on, the first three questions described for Part B1 will be repeated. In addition the following questions will be asked:

4. CLOSE SKEW WINDOW ON WRITES - Y OR N <CR>
   If the close skew windows maintenance mode is to be used while writing during test 46 through test 72, type Y. Otherwise, type N. The timeout default is N.

Part B2 tests and times the drives. It is composed of tests 100 through 173. Tests 172 and 173 are for TM03 only. The slave with the lowest address on the lowest address TM02 or TM03 is tested first. Then the slave with the next lowest address on the lowest address TM02 or TM03 is tested, and so on, until all the slaves on the lowest address TM02 are tested. Then the lowest slave on the next to the lowest address TM02 or TM03 is tested, and so on, until all slaves have been tested.

After all the slaves have been tested, one pass of part B2 has been run. If the abort switch (SW 0) is not set, another pass of part B2 is run. If abort is set, control returns 0 to the console software package.

For descriptions of each test, see Table 3.

NOTE

All tests described in Table 3 are not necessarily done. This depends on the values of assembly switches and options selected.

TEST SUMMARY
The diagnostic is divided into the following parts:
Abbreviation Description

B1 Basic part 1 (TM02 or TM03 controller testing) (test 1 through test 77). Refer to Table 2.

NOTE
An on-line, write-enabled slave is required on the TM02 or TM03 to run test B1.

B2 Basic part 2 (slave testing) (tests 100 through 171 for TM02, tests 100 through 173 for TM03). Refer to Table 3.

B Basic (runs B1, B2)

Table 4 summarizes the DDT symbolic locations.

Table 1 DSTUA Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (Refer to the KS10 STD module.) The following tests are not implemented: 11(INHPAG) and 12 (MODDVC).
18	TRASW	Print the current test number.
19	DOOPT	Do operator intervention tests.
20	CHGDRV	Do change addresses.
21	TSTEOT	Do test EOT mark.
22	PHEDSW	Print header block information during interchange/compatability tests.
23	PTIMSW	Print times during timing test even if written spec.
24	BORDSW	Do not print board callout information.
25	INHRSW	Inhibit retries during data transfers.
26	LOOPTS	Loop on current test.
27	SPCTST	Loop on the test specified by switches 28-35.
28-35	TSTMSK	Specify the test that will be continuously run if switch 27 is set.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03)

Test	Description
1	TEST THE RH This test verifies that the slave sync response works while doing a read or write of the following registers: MTCS1, MTWC, MTBA, MTCS2 and MTDB.
2	TEST TM02 AND TM03 RESPONSE This test ensures that each TM02 or TM03 controller responds to the address selected by the address select plug. All TM02s or TM03s should be tested at this time. Pull all the controller address select plugs (slave addresses don't matter) on the current RH (power down all other devices on this RH). Sequentially plug in all 8 addresses into one TM02 or TM03 when asked to. Then plug all 8 addresses into the next TM02 or TM03 and so on. When all TM02s or TM03s have been fully tested, type altmode to the question and the next test will start. When a TM02 or TM03 is set to address 2, for instance, addresses 0, 1, 3, 4, 5, 6, and 7 should evoke no response (no TRA). Address 2 is checked for a response (TRA). Loop on error loops with the error-causing address select plug left in. Each address 0 through 7 is polled to produce a scope loop on all 8 addresses. On the scope, 8 pulses will be seen on the demand line, then a pause. The first pulse will be controller address 0, etc.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	Note that if switch 20 (CHGDRV) is not set, the program will not ask for a change to any of the address select plugs, but will test the controller response on its current address. If there are TM02s or TM03s on 2 and 5 in the configuration, the test will expect a response from 2 and 5, and no response from 0, 1, 3, 4, 6, and 7. Note also that if there are other devices on the current RH not in the configuration, unexpected response errors will be received during this test. The TM02 or TM03 when on 2 responds to both 2 and 5, and the TM02 or TM03 when on 5 correctly responds to 5. This test will not detect the error unless this test is run with the OPRSEL switch set. Loop on Error will loop, polling all addresses 0 through 7 without requesting a change to any address plugs. TM02 or TM03 Loop - The start of the loop to test all the TM02 or TM03 controllers. Although no slaves are directly tested, the TM02 or TM03 needs the medium on line (MOL) and slave write clock (SWC) signals from some slave. Note that this test requires that the selected drive not be at EOT (end of tape) and the tape must be write-enabled. When the drive status register is read, the EOT bit and the WRL bit must be 0 or an error will occur. The error printout will show good data in binary (0, 1, and X for don't care) good in octal, bad, and bad bits. The bad column is the data as actually read from the register. The bad bits column will contain a 1 only if that bit was bad, not just if the good and bad columns are different. If a CPAR (control bus parity error) occurs on a register read, the register contents when printed will be preceded by a P. Loop on error will loop on reading the register that caused the error. NOTE If this test fails with a 000040 in the error register, the problem is probably a missing or marginally seated bit fiddler board. After Test 4 a warning is printed if the drive with the lowest number address select plug on the TM02 or TM03 under test is not on line-and write-enabled. As mentioned before, a drive must be on-line and write-enabled for the TM02 or TM03 to be tested.
6	WRITE EVEN PARITY TO THE FRAME COUNTER This test ensures that the TM02 or TM03 will detect bad parity (CPAR error) received from the RH. Seventeen patterns are used: 0s and rotating 1s (100000, 40000, 20000, ..., 2, 1). One or more controller clears are issued between each pattern to clear errors. The following bits are checked: Register Bits — Octal Position — State DRER CPAR (bit 32) — 000010 — 1 DRER other bits — 177767 — 0 DRDS ATA (bit 20) — 100000 — 1 DRDS ERR (bit 21) — 040000 — 1 Loop on error will loop on the pattern that caused the error. Tests 3 through 77 test the TM02 or TM03. The first TM02 or TM03 is tested fully before the second TM02 or TM03 is tested, etc.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
3	TEST OF R/W REGISTERS IN RH11 This test checks the R/W registers in the RH11 by writing bit patterns to the MTWC, MTBA, and MTDB registers then reading them back and assuring pattern integrity.
4	STATIC WRITE REGISTER TESTING This test checks the TM02 or TM03 writable registers' ability to accept and return data to the RH. Good CBus parity should be generated at all times during this test. After each write and read, the test reads the error register (DRER). An error occurs if any bits are non-0. The second mask contains a 1 whenever a 1 may be written (the GO bit is tested later). The first mask contains a 1 wherever a 1 was written in that position. The following table (which can also be found in the listing) indicates what bits are written. **Software Register Name — First Mask — Second Mask** DRFC: - Frame Counter 177777 177777 DRCS1:- Control 000076 177776 DRMR: - Maintenance 000037 177777 DRTC: - Tape Control 017777 177777 Before each register is written, all four registers are written with their respective second mask to check for register selection and addressing errors. The first register tested is the frame counter (DRFC) (register 05) because all bits are writable/readable. The patterns used for all registers are: 000000, and a rotating 1 (100000,40000,20000,...,2,1). If an error occurs (data, parity, or no TRA), the program will Loop on Error on the register and pattern which caused the error. The other registers tested are: control register (DRCS1), maintenance register (DRMR), and tape control register (DRTC). Since all of the bits are not writable/readable, the masks indicate what bits will be written and read. The error printout will show good data in binary (0, 1, and X for don't care), good in octal, bad, and bad bits. Good in octal is what actually was written into the register. Bad was what was actually read from the register. The bad bits column contains a 1 only if that bit was bad, not just if the good and bad columns are different. If a CPAR (control bus parity error) occurs on a register read, the register contents when printed will be preceded by a P. Note that if a bit is stuck high, a CMUX line may be shorted to ground on any of the following boards: M8903-TCCM (TM02 only) or M8933-TCMM (TM03 only) M8905-MAINT REGISTER, M8909-MBI or any of the three Massbus cable cards (TM02 only) or M8933-TCCM (TM03 only). Note also that if any of the lines which select the following registers are stuck in the asserted state, you will get a data error when trying to read a frame count of 0: (DRCS1, DRDS, DRDT, DRTC, or DRSN). In the 5 registers, at least one bit is statically high, and this high will be ORed with the 0 of the frame count, resulting in a data error. The DRER, DRMR, DRAS, and DRFC register selection is tested in test 7. The DRCK register is tested during the WRP0 wraparound test.
5	READ TM02/TM03 REGISTERS AND CHECK KNOWN BITS This test reads each register and checks the bits which have a known state after a controller clear. The following table of masks indicate what bits are checked and which state is correct. The first mask contains a 1 if that bit has a known state after a controller clear. The second mask contains a 1 if the known state is a 1.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	Before each register (except the maintenance register) is read, the four writable registers are written with their respective second mask from test 4, to check for register selection and addressing errors. When the maintenance register is read, none of the four registers is written into.

Register Number	Name	Description	TM02 TU45 First Mask	TM02 TU45 Second Mask	TM03 TU45 First Mask	TM03 TU45 Second mask
0	DRCS1	(Control)	177701	004000	177701	004000
1	DRDS	(Status)	167715	000600	167715	000600
2	DRER	(Error)	177777	000000	177777	000000
3	DRMR	(Maint)	000077	000000	000077	000000
4	DRAS	(Attn Sum)	177777	000000	177777	000000
5	DRFC	(Frame Count)	000000	000000	000000	000000
6	DRDT	(Drive Type)	175774	140010	175774	140050
7	DRCK	(Checksum)	177000	000000	177000	000000
10	DRSN	(Serial Number)	000000	000000	000000	000000
11	DRTC	(Tape Cntl)	140000	100000	140000	100000

7	TM02/TM03 ATA BIT ON ALL ADDRESSES Again, unplug all TM02 or TM03 address select plugs and sequentially plug all 8 plugs into the current TM02 or TM03. For each address, the test writes all 0s with bad parity into the frame counter. The ATA bit must then be 1. The test then checks that there is a 1 in the correct position in the attention summary register (DRAS). The test writes a 1 into all other bit positions and makes sure the DRAS bit from the current TM02 or TM03 is still up. The test next writes a 1 into that position, and checks that the bit has been reset to 0. Loop on error will loop on the address at which the error occurred. Note that if switch 20 (CHGDRV) is not set, the test will be run but no question to change the address select plug will be asked. The test will be run testing only the ATA bit in the position corresponding to the current TM02's or TM03's address.
10	MAKE TM02/TM03 GENERATE EVEN CBUS PARITY This test writes the CBus even parity maintenance mode (MREP) into the maintenance register, then reads the frame counter. The test expects the RH to detect a CBus parity error (CPA) and register access error (RAE). 17 patterns are loaded into the frame counter: 000000, and rotating 1s (100000, 40000, 20000, ..., 2, 1). Loop on error will loop on the particular pattern that caused the error.
11	READ TM02/TM03 NONEXISTENT REGISTERS This test reads registers 16 through 23. After each read, the test expects the illegal register address bit (ILR) to be high. The test also expects the status register bit (ERR) to be 1. Loop on error loops on the particular register address which caused the error.
12	WRITE TM02/TM03 NONEXISTENT REGISTERS Same as test 11 but writes registers 16 through 23.
13	TEST THE NO-OP COMMAND This test sets the GO bits (function bits all 0). GO should immediately reset and the error register should be 0. DRY (which is the inversion of GO) should be 1.
13A	RH11 SILO TEST 1 Test the silo buffer in the controller. A read is attempted from an empty silo. Data late, transfer error and special condition should set and then be cleared by loading a 1 into TRE.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
13B	RH11 SILO TEST 2 Test the IR and OR bits of RHCS2. Initial IR should be set and OR reset. An all 0s word and then an all 1s word is loaded into the silo via RHDB. OR should set in 30 microseconds (timing is not checked). The output from the silo should be a word of all 0s and then a word of all 1s.
13C	RH11 SILO TEST 3 Test the silo buffer by filling it with a count of 0 through 65. Then check that IR is low and OR is high. Finally, read and verify the silo output.
13D	RH11 SILO TEST 4 67 words are written into the silo and the CS2 register is checked for a data late condition. Even after the 67th word is input, the first output word should be the first one input.
13E	RH11 SILO TEST 5 The silo is loaded with 0, 1, 2, 3, then after OR is set a clear to CS2 is done, then a 4 is loaded into the silo. After OR is set, two reads from the silo are done and the last DLT in CS2 should be set.

PREPARATION FOR THE WRAPAROUND MODES

The selected slave on the TM02 or TM03 must not be at BOT during the testing of the wrap modes. Thus, immediately after test 14, if the BOT bit of the lowest numbered slave is high, the program executes an erase without checking for errors (erase will be tested later). If the BOT bit is still 1, the program asks the operator to manually move the tape off BOT.

THE WRAPAROUND MODES

There are several (four for TM02, five for TM03) data wraparound modes to enable the program to test the data paths of the TM02 or TM03. The read wraparound simulates a read from tape. The data frames come from the program writing into the maintenance register (DRMR) rather than from reading tape. The write wraparounds simulate a tape write by having the TM02 or TM03 put the data frames into the maintenance register, rather than onto the slave bus to be written onto tape.

The wraparound test exercises the TM02 or TM03 for all formats, densities and data patterns. Loop on error will loop on the particular pattern that caused the error. Loop on current or specific test will loop on a particular format and density, but will cycle through all the data patterns.

If the test detects a data bus parity error during the wraparound transfers, the error printout will indicate at which word of the transfer the error occurred (word 1 is the first word of the transfer). The transfer will continue and do the data compare. The data bus parity error is not a fatal error, so if no other error printouts accompany this error printout, you can assume that all the data was transferred correctly.

14-27 WRP3 READ WRAPAROUND ROUTINE

Test	Format	Density TM02	TM03	Frames Per Word	Per Operation
14	CORDMP	200 bits/in	800 bits/in	5	RD FWD
15	7TRK	200 bits/in	(not used)	6	RD FWD
16	ASCII	200 bits/in	(not used)	5	RD FWD
17	COMPAT	200 bits/in	800 bits/in	4	RD FWD
20	CORDMP	1600 bits/in	1600 bits/in	5	RD FWD
21	ASCII	1600 bits/in	(not used)	5	RD FWD
22	COMPAT	1600 bits/in	1600 bits/in	4	RD FWD
23	CORDMP	1600 bits/in	1600 bits/in	5	RD REV
24	ASCII	1600 bits/in	(not used)	5	RD REV
25	COMPAT	1600 bits/in	1600 bits/in	4	RD REV
26	CORDMP	(not used)	800 bits/in	5	RD REV
27	COMPAT	(not used)	800 bits/in	4	RD REV

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	After the tape control register (DRTC-REG11) is loaded properly, the routine checks the drive status (DRDS-REG01) PES bit (phase encoded status). This bit should be low for NRZI and high for PE mode. The RH channel command list is set up and the buffer is cleared. The routine then checks that the slave write clock in the maintenance register is toggling. If the SWC bit is not toggling, the routine aborts with an error message. The maintenance register (DRMR) maintenance operation field is set up and the maintenance mode bit (15) is set. The routine then loads the RH11 secondary command register with the DTES (disable transfer error stop) bit, a block count of -1 (1777) and function 71 for read FWD (77 for READ REV). Then the routine waits for ACCL (in the tape control register) to go low (the tape being at bot will cause ACCL to time out). When ACCL goes low, the maintenance clock (MC) will be low. NOTE For a TM02 the state of the DRMR data parity bit is ignored by the hardware and is always written with a 0. For a TM03, since WRAP mode will transmit a parity bit across the bit fiddler module and incorrect parity will cause a vertical parity error, control is set up in the routine to generate appropriate parity into DRMR-REG03 when loading the data field. The routine then writes each frame of data twice into the data field of DRMR, causing the MC bit to be inverted with each write. The high-order frame data bit is written into the high-order bit of the register, etc. If the MC bit does not toggle at any point during the test, the routine aborts because the error is fatal. When all of the data frames have been written, the routine writes the maintenance mode end of record (MMEOR) and MM bit in the DRMR, signifying the end of record, and then clears all of the DRMR bits. The routine then waits for the status bit DRY to go high (GO goes low). A data compare is done with the words which the channel has written into the data buffer (RD FWD only). The routine figures out from the channel logout data how many words were transferred and gives an appropriate error if the wrong number of words was transferred. In test 23, after all patterns are tested, an additional transfer is made using the function write check FWD (51) and a data pattern of 0. Then another transfer is made using the function write check REV (57). These functions should execute exactly like READ FWD and READ REV respectively. Loop on error will loop on the function during which the error occurred.
30-36	WRP2-WRITE WRAPAROUND (BIT FIDDLER) ROUTINE Density **Test Format TM02 TM03 Frames/Word** 30 CORDMP 200 bits/in 800 bits/in 5 31 7TRK 200 bits/in (not used) 6 32 ASCII 200 bits/in (not used) 5 33 COMPAT 200 bits/in 800 bits/in 4 34 CORDMP 1600 bits/in 1600 bits/in 5 35 ASCII 1600 bits/in (not used) 5 36 COMPAT 1600 bits/in 1600 bits/in 4 The WRP2 routine operates much like the WRP3 routine except that WRP2 writes the DRMR twice (toggling the maintenance clock high then low), then reads the next frame of data from the DRMR data field. Also, the DRMR data parity bit is read and checked for odd parity (on the 9 data bits). WRP2 assembles each word and writes the word back into the data buffer location that the word originally came from. Note also that as with any write, the RH11 secondary block address register is loaded with the desired frame count.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	In NRZI mode, after all the data frames have been read from the DRMR data field, the program must still toggle the MC clock over the CRC and LRC characters, even though it cannot actually read these characters. The MC bit must be cycled 8 (decimal) additional times (by writing the DRMR 16 times), 4 more for the 3 dead frames and 1 CRC frame, and then 4 for the 3 dead frames and 1 LRC frame. Then WRP2 writes the MMEOR code, then 0, into the DRMR indicating the end of the record, as in WRP3. After every 4th, 5th, or 6th frame of data, depending on the format (1 word), the program assembles a whole word and writes this word back into the same memory address that the channel read the word from. In PE mode, the program (WRP2) does not read each preamble character, but must toggle the MC bit over the preamble. For each character of the preamble, the MC bit must change state 4 times (toggle 2 times). There is also an extra character at the beginning of the preamble, used internal to the WRAP mode. Thus, there are a total of 42 preamble characters (1 extra character, forty 0s, and a 1) so the MC bit must be toggled 84 times. However, after the first toggle at any point during the test, WRP2 expects to see the first frame of data throughout the preamble, although only checking after the first and last preamble toggle. Data errors are reported as detected whether in frames, words, or both. A maximum of 8 frame errors and 8 data errors are reported (set at assembly time).
37-45	WRP1 PARTIAL WRITE WRAPAROUND ROUTINE Density TM02 TM03 Frames/Word Test Format Density Frames/Word 37 CORDMP 200 bits/in 800 bits/in 5 40 7TRK 200 bits/in (not used) 6 41 ASCII 200 bits/in (not used) 5 42 COMPAT 200 bits/in 800 bits/in 4 43 CORDMP 1600 bits/in 1600 bits/in 5 44 ASCII 1600 bits/in (not used) 5 45 COMPAT 160 bits/in 1600 bits/in 4 The WRP1 routine operates in NRZI exactly the same as does the WRP2 routine in NRZI. The main difference between WRP1 and WRP2 in PE mode is that in WRP1 you see each frame of the preamble and postamble, and their complements. You also see the complement of the data frames. After GO is set and ACCL (in the DRTC register) goes low, WRP1 begins to write the maintenance register (DRMR) two times (one toggle of the MC bit) and to check the data and parity bit. The MC bit is toggled twice (by writing the DRMR register 4 times) without checking the data, to skip over the extra character. To toggle over the forty 000 frames, toggle the MC bit 80 times. After an odd number of toggles (1, 3, 5, ...), the 000 character will appear (all 9 bits 0). After an even number of toggles (2, 4, 6, ...), the complement of the 0 character, or 777 will appear. After complementing the above operation, toggling the MC bit once more will produce the all 1s preamble character (777). Toggling again will give the complement of the 777 character (000). WRP1 then checks to make sure the frame counter has not changed before toggling the MC bit over the data frames. For each data frame the MC bit must be toggled twice. After the first toggle the data frame will appear (for example, 765). After the second toggle the complement of the data frame will appear (all 9 bits complemented).

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	After all the data frames, all the postamble frames and their complements will be listed, just as with the preamble (one 777 character followed by forty 000 characters). The first toggle after the last data frame should produce 777, the next 000, then 000, 777, 000, 777, etc. After the postamble, the frame counter should still be 0. WRP1 writes the maintentance mode end of record (MMEOR), then 0 to end the transfer. DRY should then come up within 15 ms. WRP1 then does a data compare.
47-64	WRP0 GLOBAL WRITE WRAPAROUND ROUTINE Density Test Format TM02 TM03 Frames/Word 47 7TRK 200 bits/in (not used) 5 50 ASCII 200 bits/in (not used) 5 51 COMPAT 200 bits/in (not used) 4 52 CORDMP 556 bits/in (not used) 5 53 7TRK 556 bits/in (not used) 6 54 ASCII 556 bits/in (not used) 5 55 COMPAT 556 bits/in (not used) 4 56 CORDMP 800 bits/in 800 bits/in 5 57 7TRK 800 bits/in (not used) 6 60 ASCII 800 bits/in (not used) 5 61 COMPAT 800 bits/in 800 bits/in 4 62 CORDMP 1600 bits/in 1600 bits/in 5 63 ASCII 1600 bits/in (not used) 5 64 COMPAT 1600 bits/in 1600 bits/in 4 The main difference between WRP0 and the other WRAPS is that in the other WRAPS, the program causes the MC bit to toggle by writing the DRMR (maintenance register). In WRP0, the MC bit toggles at the actual data rate using the SWC signal as the frequency reference. Each time the MC bit changes state, the next data frame is available in the DRMRM data field. In this test ACCL is not tested because the ACCL checking routine takes too long at the faster data rates. The data frames are dumped as-is into core to be reassembled into words later. At the end of a good NRZI transfer, the PEFLRC error will come up (DRER bit 28), as well as a CS2 drive exception error. The CRC character is calculated by the program and compared to the hardware calculated CRC in the DRCK check register. At the end of the WRP0 NRZI transfer, the DRFC (frame counter) is loaded with 0 and read. If non-0, the DRCK (check character) register is selected all the time. The check character register is not checked until now because the previous tests are unable to force a bit on in the DRCK register. For a TM02, in PE mode, the program has trouble keeping up with the data rate. Only every other frame will appear each time the MC bit changes state. The program fills in the missing frames before assembling words out of the good frame data. This fill is required so that a bit dropout error in PE mode will seem nonrepeating when actually a particular bit may be dropping on every word. Note that with a format which has 5 frames per word (CORDMP and ASCII), every frame will eventually appear (the sequence will be 1, 3, 5, 2, 4, 1, 3, 5, ...). If a format with an even number of frames per word is used, the even frames will never appear (COMPAT format will give you 1, 3, 1, 3, ...).

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	For a TM03, since WRP0 for both NRZI and PE mode only transmits every 9th data character across the Massbus, only every 9th frame will be seen each time the MC bit changes state. For example, if the record size is 100 (octal) and core dump format (5 frames per word) is used, 43 (octal) of the 9th frame will be seen in the entire WRP0 operation. The amount of the 9th frame available is calculated by the formula (record size X frame no.)/9 (decimal). Also, in order to do data compare, the program will fill in the missing frames before assembling words out of the good frame data. The maximum size transfer at 800 and 1600 bits/in densities is 100 words.
65-66	WRP4 GLOBAL WRITE WRAPAROUND ROUTINE Density Test Format TM02 TM03 Frames/Word 65 CORDMP (not used) 800 bits/in NRZI 5 66 COMPAT (not used) 800 bits/in NRZI 4 This wrap is used from TM03 only. WRP4 performs the same function as WRP0 with the exception that the write CRC generator will not be clocked in NRZI mode. The purpose of this wraparound is to allow the generation of the crc errors during the maintenance mode testing to enable crc error correction sequencing (see test 70 - CRC error correction test). WRP4 will also transmit every 9th data character across the Massbus; however, only every 9th frame will be seen each time the MC bit changes state. NOTE The maximum record size is 100.
67	FORCE DBUS PAR ERRORS, RH TO TM02/TM03 This test is not used with the RH11.
70	CRC ERROR CORRECTION This test ensures that the CRC error correction logic works. The test contains two parts. PART 1- SINGLE BAD TRACK, EVERY FRAME This test simulates a bad track on tape resulting in a CRC error and subsequent correction of data in the failing track. It writes a known data pattern (all 0s, all 1s and 525252,528282) with a data bit altered in each of the following data tracks. Note that only COMPAT format is allowed. 010020,,040104 Track 1 on 002004,,010021 2 040100,,200420 3 100200,,401040 5 200401,,002000 6 401002,,004000 7 004010,,020042 8 020040,,100210 9 The test proceeds as follows. Step 1 - Execute WRP0 in NRZI mode with a known data pattern and record size. At completion of WRP0 check for CRC error: If error = 1, halt or loop on error. If error = 0, continue. Step 2 - Execute WRP4 in NRZI mode with the same record size and data pattern; however, alter data in one whole track as shown above. This will generate a CRC error. At completion of WRP4 check CRC error bit: If error = 0, halt or loop on error. If error = 1, continue.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
	Step 3 - Execute WRP3 read reverse in NRZI mode with the same record size and data pattern used in step 1 (no need to check data). The purpose of this step is to initiate the CRC error correction cycle. At completion of WRP3, the bad track is found and all bits in bad track are inverted. Also, no error should be generated. Step 4 - Execute WRP4 again with the same data as in step 2. At completion of WRP4, CRC error should be cleared and the received data should match the data pattern transmitted in step 1. If not, halt or loop on error. The program repeats the above steps for each track and each data pattern mentioned above. Part 2 - MULTIPLE BAD TRACKS, EVERY FRAME This test checks that CRC error correction is not performed. When multiple bad tracks are detected the test proceeds as follows: Step 1 - Same as Step 1 in Part 1. Step 2 - Same as step 2 in Part 1 except more than one track is altered. Step 3 - Same as step 3 in Part 1. Step 4 - Execute WRP4 again with the same data as in step 2. However, at completion of WRP4, the CRC error should not be corrected. The CRC bit and ERR bit should be 1. If not, loop on error. NOTE Only drive clear can be used to clear errors existing between the four steps. Controller clear is absolutely disallowed because it will create an INIT PULSE, which will ruin the entire CRC error correction cycle.
71	CAUSE ILLEGAL FUNCTIONS This test loads all the illegal functions into the control register and expects the illegal function error bit (ILF) to be 1 as well as the status ERR bit (other error bits must be 0). The GO bit must have been loaded with 1 to cause the error. The following codes are sequentially loaded into the control register (other codes are legal and so generate no error): 5 37 55 13 41 63 15 43 65 17 45 67 23 47 73 35 53 75 Loop on error loops on the particular function that caused the error.
72	ILLEGAL FORMAT ERROR AND DRV CLR TEST TM02 TM03 Legal Illegal Legal Illegal 0-3 4-17 0,3,14,16 1,2,4-13,15,17 The test starts just like WRP3 did, with no tape motion. Immediately after the GO bit is loaded, the FMT error and ERR bits should come up and DRY should reset for each of the illegal formats. The test then tries to reset the error with the drive clear command rather than with control clear. No error should remain after the drive clear. Loop on error will loop on the particular format in which the error occurred.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
73	TEST RMR DURING WRP3 This test ensures that if an attempt is made to write any of the writable registers (DRCS1, DRFC, DRTC) an ARMR error will be received (and the ERR bit) and the data in the register will not be affected (VPE error is expected in TM03). To test each register, the TM02 or TM03 is set up for a WRP3 operation, and the GO bit is set. Then the register under test is loaded with data that will be distinguished from what was in the register first. The program expects to see RMR and ERR bits, and checks to see that the data in the register did not change. The program does a controller clear after each register is tested. Loop on error will loop on the particular register at which the error occurred.
74	TEST OPI ON WRT AND READ (WRAP) This test checks times; i.e., how long OPI error (operation incomplete) takes to occur on writes and reads. The program sets the TM02 or TM03 up for a WRP2 and does not toggle the MC bit. OPI time on write is the time between the loading of the GO bit and the OPI coming up. To time OPI on a read, the program sets up the TM02 or TM03 for a WRP3 but does not toggle the MC bit. OPI time is from the GO bit until OPI comes up. NOTE **For a TM03, OPI error will identify any operation other than rewind that fails to execute within seven seconds.** Loop on error will loop on the particular mode (write or read) during which the error occurred.
75	OPI TIME EXTEND TEST This test checks that the OPI error will not occur in the middle of a long record write. The test first sets prepares for a standard WRP2 write-wraparound transfer, begins the transfer, but does not toggle the maintenance clock. The test instead does the following to toggle the maintenance clock (MC): Wait 250 ms, toggle the MC, wait 200 ms, toggle the MC, wait 500 ms, read the error register. The test will see OPI if the OPI time was not extended (due to the maintenance clock toggling simulating tape frames being written), but will not see OPI time was extended. An error will print out if the OPI time was not extended. Loop on error will loop on the whole test. This test will only be run if Y is answered to the fourth question asked at the beginning of the program. If the OPRSEL switch is not set, the question will not be asked and this test will not be run.
76	TEST REV MOTION AT BOT - NEF ERROR This test positions the tape at BOT and executes a space reverse of 1 record. The test then checks that 0 gets an NEF error and the status register error bit. If the tape is at BOT, the test is run. If the tape is not at BOT, the test does a rewind without checking for errors. If BOT is then 1, the test is run. If BOT is still 0, the test asks to position the tape at BOT if the DOOPT (19) or CHGDRV (20) switches are set. If neither is set, the test will be skipped. Loop on error will loop only on the space reverser part of the test.

Table 2 DSTUA Test Summary for Part B1 (TM02/TM03) (Cont)

Test	Description
77	TEST NEF ERROR IN NRZI, TOO FEW FRAMES This test ensures that a record of fewer than 13 (decimal) frames may not be written. The test attempts to write a record (at 200 bits/in for a TM02, 800 bits/in for a TM03) of 1 through 12 (decimal) frames, each time expecting to see an NEF error and the ERR bit. Loop on error will loop on the particular frame size that did not result in an NEF error. Test 77 concludes part B1, TM02/TM03 testing. If any TM02s or TM03s have not yet been tested on this pass, part B1 is now run on those TM02s or TM03s. If all TM02s or TM03s have been tested, the console switches are read. If abort (SW 0) is set, control goes to the control software; if not, another pass of B1 is run.

Table 3 DSTUA Test Summary Part B2 (Drives)

Test	Description
100	CHK STATUS BITS FOR NONEXISTENT SLAVES This test asks to have all address select plugs of all slaves on the current TM02 or TM03 pulled, except for the current slave under test. Put the current slave off-line, with the address select plug in. After a controller clear, the following bits are checked: SLA=0, MOL=0, SSC=0, SPR=1, DRV type=011(8) for TU16, 012 for TU45. With any other drive selected, the bit states should be the same as the bit states in the previous test. The test will indicate what drive was selected when the error occurred. The test also asks for the drive's serial number (4 decimal digits maximum). An error will be printed if the typed-in serial number does not match the serial number of the selected, off-line slave. Loop on error will loop selecting the slave address at which the error occurred. This test will be done only if switch 19 (DOOPT) or 20 (CHDRV) is set.
101	Not implemented
102	CHK ON-LINE DYNAMIC STATUS BITS The first part of the test asks for the current slave to be put off-line if not already off-line, and then does a controller clear. Then the test asks for the slave to be put on-line again. The following bits are checked: SLA=1, MOL=1, ATA=1, SS=1, SPR=1, and the 9-bit drive type 012(octal) for TU45. The test then deselects the slave under test and checks that SL=0. The test then selects again the slave under test. SLA should still equal 0 because deselecting the slave should clear SLA. (On the TU45, deselecting the slave will not clear SLA, so this last SLA check is not done on TU45s). Any errors are then printed. The second part of the test does a controller clear, then asks for the slave to be put off-line. The following bits are checked: SLA=0, SSC=1. The test then does a controller clear and checks that SS=0 after the initialization. Then the test asks for the slave to be put on-line again. This test will be done only if switch 19 (DOOPT) or 20 (CHGDRV) is set.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
103	TEST SLAVES' RESPONSE TO ALL ADDRESSES This test first asks for the slave under test to be put on MTA0. The other slave address plugs on this TM02 must be unplugged. The test then polls all slaves 0 through 7, expecting to see SPR=1 for MTA0, and SPR=0 for MTA1 through MTA7. Any errors are then printed, and a request for the slave to be changed to MTA1. The test then polls again 0 through 7, expecting this time SPR=1 for MTA1, and SPR=0 for the other numbers, and so on. After the slave has been tested on MTA7, all slave address plugs should be put back where they were. Loop on error loops with the slave set to whatever address caused an error, while the program polls all numbers 0 through 7. This test is done only if switch 20 (CHGDRV) is set.
104	CHECK THE FREQ OF SWC2 IN MAINT REG This test checks the frequency of the slave write clock in the slave (SWC2 in the maintenance register). The test uses a special timing loop in the accumulators to measure 400 cycles of SWC2 and then divides the time by 400 before printing.
105	TEST ERASE AND REWIND, AND THE TCW BIT This test checks the status bits during and after a rewind to ensure the interrupt system operates properly. A rewind with a 10 second wait for BOT is done before subtest 1. If the drive was not already near BOT, timeout errors will occur. A longer wait is not wise because rewind has not yet been tested and the program would appear to hang if the rewind did not work. The first subtest sets up the tape control register and checks that the TCW bit (in the tape control register) is 1. Then the test does an erase and checks to see that the TCW bit is 0 after ACCL goes low during the erase. Note that for TM03, there is no need to check the TCW bit since this bit is redefined as the SLA bit. As soon as the erase is finished, the program does a rewind and waits for DRY to come up. After DRY, ATA should be 1. The program then writes the proper bit in the ATTN summary register to clear ATA, checks that SSC is 0, deselects the slave under test, and waits for that bit to come up again. When ATA goes high, the program checks that SSC is 1, does a drive clear (note that the drive is still deselected) and checks that SSC is 0. The program then reselects the slave and checks that BOT is 1 and PIP is 0. The second subtest does a rewind while at BOT and reads the status registers, making sure that the SSC bit is set. Loop on error will loop on the failing subtest.
106	TEST THE READ-IN PRESET CMD This test checks that the preset TM03 function works. If there is no slave on MTA0 which is on the current TM02 or TM03, a request will be made to put one there just for this test. The test then does an erase to move the tape off BOT if the drive was at BOT. The test loads 6777 into the tape control register, and then does a read-in preset. The tape control register should be set to 1000 (800 bits/in, CORDMP, ODD PAR, MTA0. The test then waits for PIP to go low and then checks to make sure BOT is high. This test will be skipped if switch 20 (CHGDRV) is not set and there is no slave set to address 0. If there is already a slave on 0, the test will be run. This test is run only once per TM02. (If more than 1 drive is to be tested on this TM02, the test is skipped after the first slave is tested.)

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
107	TEST THE WRITE-LOCK FEATURE This test will request that the write ring be removed from the slave and then tries to erase on the tape. The WRL bit, the NEF error bit, and the status register ERR bit should all be 1s. This test will be done only if switch 19 (DOOPT) or 20 (CHGDRV) is set.
110	TEST THE REWIND OFF-LINE COMMAND (UNLOAD) This test does an erase, then the unload command. The test then waits for DRY and expects SSC to be 1 at this time on TU16s. SSC is not checked on TU45s (should be 0). The program then waits for PIP to go low, then checks that BOT is high and MOL is low. This test will be done only if switch 19 (DOOPT) or 20 (CHGDRV) is set.
111	TEST THE UNS ERROR BIT This test executes a no-op command while the drive is off-line. The test first does an unload to put the drive off-line. If the drive is still on-line, the test requests that the drive be put off-line. The test does a no-op, and checks for the UNS error and the status register error bit. The test then asks for the drive to be put back on-line. Loop on error will loop on the no-op command, and the checking for UNS and error. This test will be done once per TM02 or TM03, and only if switch 19 (DOOPT) or 20 (CHGDRV) is set.
112	TAPE MARK, SPACE REV TEST This test first makes sure the TM bit in the status register is 0, then writes a tape mark. The program waits for DRY, then checks the TM bit which should be 1. The test then does a drive clear (so that the TM will not reset) and a space reverse of 1 record (the frame counter is set to 1777777). The test then waits 10 ms (if TU45) and checks the TM status bit. The TM bit should be 0 because the drive set pulse should reset TM. The test then waits for DRY and again checks that TM is set to 1. The test then does a controller clear and makes sure the TM bit is reset. B1 Loop on error loops on the whole test at once; that is, WRITE TM, SPACE REV, WRITE TM, SPACE REV, etc. At the end of the test, a rewind with a long wait for BOT is done. If this test is looped on error the tape position will be rather far from BOT. NOTE The tests that follow expect the tape to be positioned close to BOT.
113	WRITE BAD PARITY TO TM02 OR TM03; TRY TO SET GO BIT This test tries to write the erase command into the control register (DRCS1) with bad parity in the CBus lines. GO should not set and CPAR bit should be 1. Loop on error will loop on the whole test.
114-122	SIMPLE MOTION AND DATA XFER TEST These tests exercise all the tape motion commands at 200 bits/in 800 bits/in and 1600 bits/in, using all formats. The size and pattern used for data records are the parameters you typed in at the start of the program except that if the size is an even number, the test will add 1 to make the size odd to more rigorously test the LRC logic.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description

Test	Density TM02	TM03	Format
114	200 bits/in	800 bits/in	CORDMP
115	(not used)	(not used)	
116	200 bits/in	(not used)	ASCII
117	200 bits/in	800 bits/in	COMPAT
120	1600 bits/in	1600 bits/in	CORDMP
121	1600 bits/in	(not used)	ASCII
122	1600 bits/in	(not used)	COMPAT

Each test is composed of 10 subtests, each of which will loop on error on the whole subtest. The exception is subtests 8 and 9. An error in either will indicate in which subtest the error occurred. Loop on error will loop on both subtests so that the tape will be positioned correctly.

NOTE

Errors during writes and reads will not be retried. A bad tape will cause the test to fail on a good drive. If this test fails and you think the drive is working, try another tape.

Subtests

1. LONG REWIND (7 min timeout)

2. Write the ALTPAT pattern and check the LRC. Also checks that ATA is 0 after the data operation unless EOT is up or ERR is up (in which case ATA must be 1).

3. REWIND
   READ FWD

4. SPACE REV
   READ FWD
   Also checks that ATA is 0 after the data operation unless EOT is up or ERR is up (in which case ATA must be 1).

5. READ REV
   SPACE FWD
   Also checks that data is 0 after the data operation unless EOT is up or ERR is up (in which case ATA must be 1).

6. WRITE TAPE MARK

7. WRITE THE ALTPAT PATTERN

8. REWIND
   SPACE FWD 3 records

   NOTE

   This test expects the SPACE FWD to terminate early with a FCE error and ERR. Frame counter is initially set to 1777775, and should stop at 177777 due to the tape mark.

9. SPACE REV 2 RECORDS
   SPACE FWD 1 record (or 2 records if the space rev spaced 2 records)

   NOTE

   See note under previous subtest. The only difference is that the frame counter is initially set to 177776.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
	10. LONG REWIND After each write data at 200 bits/in or 800 bits/in, the program calculates the LRC character (where noted above) and compares the calculated character with the character in the 9-bit data field in the maintenance register. The LRC is calculated on the basis of the frame DTA pattern and the CRC character as read from the check character register (CRC generation was checked in the WRP0 write wraparound test). Note that the parity bit is the high order bit in the CRC character in the check character register, but the parity bit is the low order bit in the maintenance register data field. An LRC character error will print out in maintenance register (low order parity bit) format. NOTE Time ranges involved in all timing tests are made to meet the requirement of ECO M8921-00008 (TU45 only).
123	TIME WRITE FWD AT BOT This test measures the relevant times for a write at BOT: ACL delay, shutdown delay, and settledown delay. ACCL delay is the time between GO setting and ACCL going to 0. Bl shutdown delay is the time between when the frame counter overflows to 0 and SHUTDOWN becomes 1. Settledown delay is the time between when SHUTDOWN becomes 1 and SHUTDOWN becomes 0. The test does a long rewind (7 minute maximum wait) before doing the write.
124	TIME WRT FWD NOT AT BOT This test measures only the ACCL delay (see test 123) for a write not at BOT (should be much shorter than the previous test at BOT).
125	TIME READ FWD AT BOT This test measures the relevant times for a read forward at BOT: ACCL delay, shutdown delay, and settledown delay (see test 123). The times are measured the same way as the write forward at BOT except for shutdown, which is the time between when the final frame count is reached and when shutdown becomes 1. The test does a long rewind (7 minute maximum wait) before doing the read. The test also does not check for data errors. Test 123 must previously have been run before starting this test manually.
126	TIME READ FWD NOT AT BOT This test measures only the ACCL delay (see test 123) for a read not at BOT (should be much shorter than the previous test at BOT). The test does not check for data errors. Test 123 must previously have been run before starting this test manually.
127	TIME READ REV This test checks the same three parameters as does test 123 except that this test times the read reverse not at BOT. Test 123 must previously have been run before starting this test manually.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
130	TIME TURNAROUND, FWD TO REV This test measures the time the drive takes to change from a forward command to a reverse command. The program first executes a write forward. As soon as DRV is raised, the program executes a read reverse. Turnaround time is the time from when DRY goes high the write until ACCL goes low during the read.
131	TIME TURNAROUND, REV TO FWD The program first writes a good record, then reads reverse, then reads forward. Turnaround time is the time from when DRY goes high after the read reverse until ACCL goes low during the read forward.
132	TIME GAP SIZE - STOP HALF This test measures the time the tape takes to stop at the end of writing a record (a measure of the distance into the interrecord gap). The test writes a record and waits for SHUTDOWN to go low, which indicates that the tape has stopped. The test then does a read reverse and measures the time from GO = 1 until the frame counter increments to 1, indicating that the first frame has been read.
133	TIME GAP SIZE - START HALF This test does a read forward over the record written in the previous test, and measures the time between GO = 1 and the frame counter increment to 1. This gives a measure of tape travel during startup.
134	GAP SIZE - INTERRECORD This test will measure the interrecord gap (IRG) on the fly. The time should not be equal to the start half plus the stop half, because the acceleration and deceleration delays are not used here. (The time here should be less than the sum.) The test writes two records, then reads reverse two records. (Do the second read reverse as soon as DRY is raised from the first one.) The gap time is the time between GO = 1 for the second read reverse, and the frame counter increments to 1 during the second read reverse.
135	TIME GAP SIZE WITH VARIABLE DELAYS BETWEEN WRITES This test writes 17 records, with 16 different delay times between each record. The delay time between the 1st and 2nd is about 1 ms, 2nd and 3rd 2 ms, 3rd and 4th 3 ms, etc. The test then reads reverse over all 17 records, checking the time between DRY = 1 from the previous record and the frame counter increments to 1 on the next record. The times which are not within range are printed as out of range errors. Loop on error will loop on the whole test (17 writes, 17 read reverses).
136 - 142	DATA RATES These tests time the drive data rates at the following densities: Test 136: Tape CNTL Density 0; Density TM02 200 bits/in; Density TM03 (not used); No. of Words (Decimal) 160; Format CORDMP Test 137: Tape CNTL Density 1; Density TM02 556 bits/in; Density TM03 (not used); No. of Words (Decimal) 556; Format COMPAT Test 140: Tape CNTL Density 2; Density TM02 800 bits/in; Density TM03 (not used); No. of Words (Decimal) 640; Format CORDMP Test 141: Tape CNTL Density 3; Density TM02 800 bits/in; Density TM03 800 bits/in; No. of Words (Decimal) 640; Format CORDMP Test 142: Tape CNTL Density 4; Density TM02 1600 bits/in; Density TM03 1600 bits/in; No. of Words (Decimal) 1280; Format CORDMP Each test writes four inches at the above densities, and times from when the frame counter begins incrementing to the time at which DRY becomes 1. The test divides the time by 4 before printing so you get the time for one inch plus 1/4 of the time between the last frame and DRY becoming 1.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
143	TIME ERASE This test ensures that erase will erase the proper amount of tape. The test times between GO = 1 and GO = 0 (DRY = 1).
144	TIME WRITE TAPE MARK This test does the same thing as the previous test except that the command executed is write tape mark.
145	TEST CAPSTAN SPEED WITH 800 BITS/IN MASTER/SKEW TAPE This test asks you to mount an 800 bits/in master tape on the drive (write locked) to test the capstan speed. The test reads 8 inches of tape and measures the time from the time the frame count increments to 1, to the time the frame counter increments to the last frame (the time between the first frame and the last frame). The test divides the time by 8 before printing, so for TU45s at 751 in/s, you should get 13.33 ms. The test then asks you to put the scratch tape back on the drive. This test will be done only in exec mode and if switch 19 (DOOPT) or 20 (CHGDRV) is set.
146 - 154	COMPLEX MOTION TESTS These tests write many different patterns on tape and exercise the spacing function for various formats and densities. Errors on writes and reads will be retired. The transfer size used during this test is the size typed in at the beginning of the program (the default here will be 101) except that if an even number of words was typed in , the test will add 1 to result in an odd record size. An odd record size will better test the LRC logic.

Test	Density TM02	Density TM03	Format	Exercise Spacing?
146	200 bits/in	(not used)	CORDMP	YES
147	not used	(not used)		
150	200 bits/in	(not used)	ASCII	NO
151	200 bits/in	(not used)	COMPAT	NO
152	556 bits/in	(not used)	CORDMP	YES
153	800 bits/in	800 bits/in	CORDMP	YES
154	1600 bits/in	1600 bits/in	CORDMP	YES

Each test first rewinds and writes a tape mark on tape, then 40 records in the sequence WRT, READ REV, READ FWD, and then a tape mark. The data patterns used are the same as the patterns used for the wraparound modes. Loop on error during a read will loop on the read operation that caused the error by inserting a space APE in the proper direction. On a write, loop on error will cause the drive to write continuously. If the test is not in CORDMP (tests 147 through 151), the test is finished. If the test is in CORDMP, the spacing exercising of the test will begin.

Note that the routines used to write and read data are the same routines used during reliability, interchange, etc. Tape and data errors will be retried (unless the inhibit retries switch is set - INHRSW). If retries are not inhibited, the test will retry each data operation and will loop on error only after exhausting all the retries. Loop on error will loop on the operation at which the error occurred.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
	The spacing subtest consists of first rewinding the tape, writing a tape mark, and then WRT, READ REV, READ FWD 40(octal) records, starting at BOT. The data patterns are 0, 1, 2, 3, 4, ... 37 (right justified in the 36-bit word, other bits 0) then write another tape mark. These particular patterns will allow the test to read a record and tell how many records there are from BOT. To exercise the spacing commands, space 1 record (over the tape mark), space forward n records, read forward, and rewind. N ranges from 1 to as many records as were written (40) in steps of 1. Loop on Error will loop on the whole above sequence for a constant N. If the drive spaces the wrong number of records, the error printout will tell how many the drive should have spaced (good) vs. how many the drive did space (bad). After each write data at 200 bits/in, the program calculates the LRC character and compares the calculated character with the character in the 9-bit data field in the maintenance register. The LRC is calculated on the basis of the frame data pattern and the CRC character as read from the check character register (CRC generation was checked in the WRP0 write wraparound test). Note that the parity bit is the high order bit in the CRC character in the check character register, but the parity bit is the low order bit in the maintenance register data field. An LRC character error will print out in maintenance register (low order parity bit) format. At the end of each test, the tape statistics (number of records written, etc) are printed out only if there were retry errors or dead tracks (in PE). Setting the totals switch will print the current value of the statistics. Loop on error is inhibited during the initial write APE mark operation of the spacing subtest. Also, looping on an error on the write data operation, will result in long blank spots on the tape because the loop on error operations are SPACE REVERSE, ERASE, and WRITE (the same as normal write retries). These blank spots may cause OPI errors if the spacing part of the test is allowed to run.
155	TEST TAPE MOTION MAINTENANCE MODE: CRIPPLE RECEPTION OF OCC This test uses the main mode CROCC (17). The program loads CROCC (cripple reception of OCC) into the MAN REG. The test then executes a write. DTE (drive timing error) and the status register ERR bit should come up shortly thereafter.
156	TEST WRITE TAPE MARK IN WRP0 This test writes a tape mark during the WRP0 (mode 7) wraparound mode, by setting the maintenance register to WRP0 and loading the WRT tape mark command into the drive control register. While waiting for DRY, the CSITM (correctable skew, illegal tape mark) bit should come up.
157	TEST ILL CHK CHAR IN NRZI This test uses the maintenance mode ILCCI (illegal CHRK CHAR, 1 dead track-mode 21). The program first executes an erase to ensure that it is not at BOT, and then does a normal WRP0 wraparound transfer (using the WRP0 routine). The program sets up the maintenance mode ILCC1 and then executes a WRT FWD (data pattern = -1, COMPAT FORMAT, 4 words). The error bits CORCRC and PEFLRC should come up in the error register, and ERR should come up. Looping on error loops only on WRP0 and write forward, not on the ERASE.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
160	TEST ILLEGAL TAPE MARK - NRZI This test uses the maintenance mode ITM12 (illegal tape mark, 2 dead tracks-mode 23). The program loads the MAINTENANCE REGISTER with the MAINTENANCE MODE, then executes a write forward (data pattern = -1, COMPAT format 4 words). After DRY comes up, the CSITM and INC errors, and ERR should be up. The LRC character (as seen in the maintenance register data field) should be 146 for TU45. The CRC character (as seen in the check character register) should be 777.
161	TEST DEAD TRKS IN PE MODE This test tests the dead track maintenance modes (ILCC1-21 and ITM12-23). The code of the subtest is executed twice, once for each maintenance mode. Both modes do 1000(octal) frames (200 words of COMPAT format), PE mode, and a special pattern so that the data frames come out like 377,003,377,003,... The program begins to execute a write forward normally. After setting the GO bit, the test waits until the frame count = 0, which indicates that all the data frames have been written, and the record is just about to reach the read head for read-after-write. The test then loads the MAINTENANCE REGISTER with the proper mode to cause 1 or 2 dead tracks on the read-after-write. After DRY comes up, the test checks the error bits. ILCC1 (1 dead track) should get a CORCRC ERROR and 002 should be in the check register (the PEFLRC error is "don't care"). ITM12 (2 dead tracks) should get an INC ERROR and 046 should be in the check register (CORCRC, PEFLRC, and NSG ERRORS are "don't care"). ATA must also be up because of the error. NOTE If the test is run and extra bits appear in the check register, the tape may be bad and these extra bits may be the result of real-live dead tracks.
162	TEST INCORRECT PREAMBLE - PE This test tests the INC PRE (incorrect preamble-27) maintenance mode. The test first sets up for a normal write forward of 40(octal) frames). As soon as the frame counter begins to change, indicating that a good preamble has been written, the test loads the maintenance register with the INC PRE MAINT MODE. After DRY comes up, the test expects to see INC and PEFLRC, in the error register (NSG is "don't care"). For a TM03, this test is skipped because PE format error is now derived by a rather complex algorithm in which the error is treated for read and write operations.
163	TEST EVEN-ODD PARITY TAPE READ/WRITE This test ensures that the test can write and read even or odd parity in NRZI (200 bits/in) on tape. First the test writes a record of 252525,,525252 odd parity. Then it reads reverse even parity. The data should all be correct, but an INC ERROR and ERROR should be received. Loop on error will loop on the above pair of operations. Next the test writes a record of 0s on tape, even parity. The drive should prevent an all 0 frame from being written on tape even parity. (To do so would result in a tape character with no flux change in any channel.) The drive forces one of the data bits and the parity bit to a 1. On the read reverse in odd parity an INC ERROR will appear and the data should be 040100,,200400. ATA must also be up because of the error. Loop on error will loop on the above write, read reverse sequence.
164	Not implemented

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
165	FORCE DBUS PAR ERRORS TM02/TM03 TO RH This test causes the RH11 to detect DBus parity errors. The test uses the following patterns: 0, 400000, 200000 , ... , 2 , 1 at 200 bits/in. (Note that since the data bus is only 18 bits wide, we only need rotate the 1 through half the 36-bit word.) First the test writes a good record, then read reverse the record with the maintenance register set up in the maintenance mode to generate even parity. The test makes sure that the data bus parity error (DBPE) bit in the CS2 is set. After checking the CS2 bits, the test does a data compare to ensure that the data transferred was accurate. Loop on error will loop on the sequence WRITE, READ REV, before the particular pattern at which the error occurred. This test is done only once per TM02.
166	REEL SERVO TEST AT BOT This test spaces back and forth continuously over a number of records near BOT to check that the reel servo will keep the proper amount of tape in the vacuum column. The test first writes 30(decimal) records starting at BOT. Each record is written at 200 bits/in core dump, and contains 16(decimal) words, so that the tape contains 1 record per inch (including the interrecord gap). The test then begins a loop spacing the tape back and forth over a varying number of records. The number of records spaced varies from 5 to 21 with an increment of 1. The loop is as follows: 1. If medium on line (MOL) is not asserted: wait for MOL (in case the drive went off-line [lost vacuum] on the previous loop). 2. Rewind to make sure of sync. 3. Space forward N records. 4. Space reverse N records. 5. Increment the loop size (N) and go back to step 1. If an error occurs in step 3 or 4, the printout will indicate the direction of motion and the size of the loop in inches. Loop on error will loop on the size that caused the error (steps 1 through 4 above). If the drive loses vacuum due to servo response errors, the program will type "waiting for TM02/TU16 to go on-line...". Load the tape again, put the drive on-line, and the test will continue.
167	TEST END OF TAPE This test checks that the drive can only space one record at a time when EOT is up. First, the test just writes records at 200 bits/in (for TM02) or 800 bits/in (for TM03) until EOT comes up (about 10-15 min for a 2400 ft tape). If EOT is not seen the test will write off the end of tape. The first subtest writes one more record on the tape to ensure that the tape is well past EOT. The test then does a controller clear and ensures that EOT is still set. Next the test space reverses 3 records (one at a time) and checks that EOT is 0. Then the test spaces forward 3 records and checks that EOT is 1. Loop on error will loop on this subtest except for the initial write of the extra record past EOT.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
	The second subtest writes 3 more records on tape. The data patterns are 1,2, and 3 (000000,,000001, etc). Next it attempts to space reverse 2 records. The drive should stop after 1 record with a FCE (frame count error) and the frame counter equal to 177777. Then READ REV a record, and expect to see data = 2 and ATA (attention) up. If the data is wrong, the error message will calculate how many records were spaced and just print that information (i.e., if the data was 1, you know 2 records were spaced). Loop on error will loop on this subtest except for the initial write 3 records. The third subtest writes as many records as necessary to see EOT come up (should already be up). Then the test rewinds and checks that EOT is 0 when the drive reaches BOT. Loop on error will loop on this whole subtest, but it will be useful only if a special tape that has the EOT marker located close to BOT is used. This test will be done only if switch 18 (TSTEOT) is set.
170	TEST IDNB BIT AND MEASURE DURING NRZI This test ensures that a PE read will recognize the IDB (ID Burst) near BOT, and that an NRZI write will erase the IDB. The first subtest rewinds the tape to BOT and writes a 1600 bits/in record (100 words, all 0) and checks to make sure that the IDB burst bit is 1. The second subtest rewinds the tape, spaces forward at 1600 bits/in and checks that IDB is 1. The third subtest rewinds the tape and writes 100 words at 800 bits/in all 0. IDB should also be 0. The fourth subtest rewinds the tape and spaces forward a record at 1600 bits/in. IDB should also be 0. Loop on error on this subtest will loop on the sequence: REWIND, WRITE FWD at 800 bits/in, REWIND, SPACE FWD at 1600 bits/in. The write is added to the loop so that scoping may be used to find out why the write does not erase the IDB, although the SPACE FWD detects whether the IDB was actually erased. Loop on error will loop on any subtest that detected an error.
171	TEST EAODTE BIT (TC REGISTER) This tests the enable abort on data transfer error bit (in the tape control register). The test first writes a record of 200 words, all 1s (pattern is -1), odd parity. Then the test read reverses the record with the EAODTE bit on. The test passes if the drive stops reading with less than 8 words slipping through. This test is done once per TM02.
172	AUTO DENSITY SELECT TEST This test tests the automatic density select feature on the TM03. It contains two parts: the first subtest writes a record in PE and reads reverse in PE to make sure that the record being written is good. Then it rewinds to BOT to change density to NRZI and tries to read forward in NRZI. At the completion of test, IDB and PES bits should be 1, INC/VPE and COR/CRC bits should be 0. This indicates that the drive automatically selects the proper density-PE from BOT and actually performs the read forward operation in PE rather than NRZI.

Table 3 DSTUA Test Summary Part B2 (Drives) (Cont)

Test	Description
	The second subtest writes a record in NRZI and reads reverse in NRZI to make sure that the record being written is good. Then it rewinds to BOT to change density to PE and tries to read forward in PE. At the completion of test, IDB, PES, INC/VPE, and COR/CRC bits should be 0. This indicates that the drive automatically selects the proper density NRZI from BOT and actually performs the read forward operation in NRZI rather than PE. This test is skipped on TM02.
173	TEST NEF CAUSED BY NOT CHANGING DEN AT BOT This test ensures that density changes cannot be allowed whenever tape is not at BOT. The first subtest writes a record in PE starting from BOT,, then changes density (not at BOT) to NRZI and tries to write another record in NRZI. NEF and ERR bits should be 1 at the completion of the last write operation. The second subtest writes a record in NRZI starting from BOT, changes density (not a BOT) to PE and tries to write another record in PE. NEF and ERR bits should be 1 at the completion of the last write operation. This test is skipped on a TM02.

Table 4 DSTUA Symbol Summary

Symbol	Description
TM02	Current TM02 or TM03 under test (right-justified, left half of the word).
TU16	Current TU16 or TU45 under test (right-justified, right half of the word).
ERRORF	Error flag. If 1, will cause loop on error.
DATSIZ	Default no. words/records (set up during option select).
OPTIMSK	Contains a 1 for bits you care about data comparing (set up during option select).
RELRPT	No. passes each density, reliability (set up during option select).
TITSKW	Close skew window: 0 = No, 3 = Yes (set up during option select).
ADRDAT	Contains the first location used as the data buffer. This location may be patched to move the data buffer around.
PHYADR	Contains the actual physical (22-bit) address being used as the data buffer. This is needed because when running paged, ADRDAT will always contain 400000.
DATTAB through DATTAB+ NUMPAT-1)	The table of data patterns used for wrap modes and complex motion tests. Reliability uses DATTAB through DATTAB+RELPAT-1 plus the multiple word patterns.

GENERAL INFORMATION

CODE

DSTUB.SAV

Title

DECSYSTEM-2020 RH11-TM02/TM03-TU45/TU77 Magtape Reliability Diagnostic

Abstract

DSTUB is the RH11-TM02/TM03-TU45 Unibus adapter magtape system reliability diagnostic written to run on the KS10. The diagnostic will test up to 64 drives - the maximum number that can be connected to a single RH controller. This diagnostic tests multiple drives serially; that is, the program runs one unit through a set of tests, and then runs another drive through the same set of tests, and so on.

Each part of the diagnostic will run with preassigned defaults (record size, data mask, etc.) as noted in the description of each part. To change the defaults, set the OPRSEL switch (14), and start the test. Questions concerning the relevant parameters will be asked. Parameters that have been changed will remain in effect until the parameters are changed again. If the same question is asked before several parts, the response typed in will affect all the parts. In exec mode, if a response is not received for each question within a few minutes, the program will timeout and assume the default.

Hardware Required

KS10 Mainframe
Minimum of 48K of memory
RH11-TM02/TM03 - up to four TU45s

Preliminary and Associated Programs

Refer to diagnostic hierarchy (KS10 STD module)

Restrictions

Drives to be tested must be connected to the same RH11 and each TM02 or TM03 must have at least one slave attached.

Loading and Starting Procedures

Exec Mode Operation

1. Mount tape of known quality on the transports to be tested.
2. Place the transports to READY, WRITE ENABLE and REMOTE.
3. Set the appropriate data switches.
4. Refer to the standard loading and operating procedure in the KS10 STD module.

User Mode Operation

1. Gain access to the time-sharing system.
2. Assign transports to be tested.
3. Mount tape of known quality.
4. Place the transports to READY, WRITE ENABLE, and REMOTE.
5. Set the data switches if desired.

Control Switches

Refer to Table 1.

Notes

1. Run times (all switches 0) Run times for TM02 and TM03 are the same.

Part	TU45
Reliability (one pass, one density, 2400 ft tape)	20 min
Interchange write (write a "bunch")	10 s
Interchange read (read a "bunch")	30 s

2. To aid in scope-looping a problem, the instructions SCOPE and SCOPE2 are located at various places in the program. SCOPE selects the serial number register which provides sync capability on the serial number register select line. Select occurs wherever SCOPE appears in the listing. However, when testing the serial number register, SCOPE is not quite as useful.

   SCOPE2 selects the serial number register, then the checksum register, then the serial number register. The two pulses distinguish SCOPE2 from SCOPE when more than one sync point is desired.

   Massbus INIT is another useful signal to sync on. Massbus INIT occurs wherever the listing contains RHCLR or MTACLR in the basic diagnostic.

OPERATIONAL CONTROL

MANUAL DRIVE CONFIGURATION
The first configuration question printed after starting the diagnostic will be:

SPECIFY TM (3 = TM03, 2 = TM02, CR will test all on-line drives)?"

TM02 If a 2 is typed, the next question printed will be: INPUT TM02 #'s -

In response to this question, the TM to be tested will be entered by its number (0-7). If more than one, they must be separated by commas. For each TM02 typed in, the following question will be printed.

TM02 # X; SLV TYPE (TU16, TU45, or TU77)

Respond by typing in the drive type. The next question printed will be: TM02 #X SLAVES:

In response to this question, type the slave number (0-7). If more than one, separate them by commas.

TM03 If a 3 is typed, the next question printed will be: INPUT TM03 #'s-

In response to this question, the TM to be tested will be entered by its number (0-7). If more than one, they must be separated by commas. For each TM03 typed in, the following question will be printed:

TM03 #X; SLV TYPE (TU16, TU45, or TU77)

Respond by typing in the drive type. The next question printed will be: TM03 #X SLAVES:

In response to this question, type the slave number (0-7). If more than one, separate them by commas.

AUTOMATIC DRIVE CONFIGURATION
If just a carriage return <CR> is typed in answer to the TM02 or TM03 question, the program then enters in the configuration and all on-line drives.

The program automatically determines whether the controller on the RH11 is TM02 or TM03 based on the value of the drive type register bit 5 (0 = TM02, 1 = TM03).

NOTE

This auto mode should not be used unless the drive type bits work properly. The program will not automatically enter in the configuration any on-line drive on which an error occurs in the read register routines.

The maximum number of slaves able to be simultaneously tested may be changed at assembly time by redefining UNTMAX to be that number (currently 10 octal). The configuration routine will not allow more than UNTMAX slaves to be entered into the configuration table. (UNTMAX controls the statistics table size.)

USER MODE DRIVE CONFIGURATION
In user mode, only one drive may be entered in the configuration.

The desired drive must be on-line and write-enabled to allow the diagnostic to do an OPEN on the drive.

In both TOPS-10 and TOPS-20, the drive must be assigned.

In TOPS-10 the question "type MTAPE to be tested" is asked in the form: MTXN<CR>. Type the logical name of the drive; i.e., MTA0, MTC5, etc.

In TOPS-20, the question "type MTAPE to be tested" is asked in the form: MTAN:<CR>':. Type the logical name of the drive; i.e., MTA0:, MTA3:, etc. If the drive exists, the slave type will be asked, and then the RH #, TM02 or TM03 #, slave #, and serial number will be given. If the drive does not exist (according to the monitor) the program will give an error message and ask the above questions again.

TEST SUMMARY

Table 2 summarizes the test and routines available in DSTUB.

Table 3 summarizes the commands which may be used with the OPERATOR routine.

Table 4 summarizes the DDT symbolic locations.

Table 1 DSTUB Control Switch Summary

Switch	Mnemonic	Description
0-17		Standard (refer to the KS10 STD module)
18	TRASW	Trace Program
19	PHEDSW	Print header for compatability and interchange
20	INHRSW	Inhibit error retries on data ops
21	LOOPTS	Loop on current test
22	SPCTST	Loop on specific test
23-25		Not used

Table 2 DSTUB Test Summary

Test	Description	Cross Ref.
^D	Enters DDT	5.
C	Compatibility Test	4.
IR	Interchange Read Test	4.
IW	Interchange Write Test	4.
O	Operator Function Select Test	1.
R	Reliability Test (1600 bits/s and 800 bits/s)	2.
R1	Reliability Test (1600 bits/s only)	2.
R2	Reliability Test (200 bits/s only)	2.
R5	Reliability Test (556 bits/s only)	2.
R8	Reliability Test (800 bits/s only)	2.
U	Unknown Tape Read Test	3.

Table 3 DSTUB Command Summary

Command	Description
ER	Erase
RF	Read forward
RR	Read reverse
RW	Rewind
SFN	Space forward (N, if present must be between 1 and 9 and indicates no. of record to be spaced. If just SF, record 1 will be printed.
SRN	Space reverse (same as for SFN)
UF	Unknown read forward (No length or data checking)
UR	Unknown read reverse (no length or data checking)
WF	Write forward
WM	Write tape mark

Table 4 DSTUB Symbol Summary

Symbol	Description
TM02	Current TM02 or TM03 under test (right-justified, left half of the word).
TU16	Current TU45 or TU77 under test (right-justified, right half of the word).
ERRORF	Error flag. If 1, will cause loop on error.
FASTNR	On interchange, fast mode writes this many bunches.
DATSIZ	Default no. words/records (set up during option select).
OPTMSK	Contains a 1 in bits for data comparing (set up during option select).
RELRPT	No. of passes each density, reliability (set up during option select).
TITSKW	Close skew window 0 = No, 3 = Yes (set up during option select).
ADRDAT	Contains the first location used as the data buffer. This location may be patched to move the data buffer around.
PHYADR	Contains the actual physical (22-bit) address being used as the data buffer. This is needed because when running paged, ADRDAT will always contain 400000.
DATTAB through DATTAB +NUMPAT-1	Reliability uses DATTAB through DATTAB+RELPAT-1 plus the multiple word patterns.

TEST DESCRIPTIONS

1. OPERATOR FUNCTION SELECT TEST (TEST 1)
   This test allows typing in a sequence of operations to be performed permitting scope looping on a particular set of operations and parameters.

   To select this test, type 0 to the WHAT TEST question. The program prints a heading and (in exec mode) asks what TM (no need to specify TM02 or TM03). Type the TM and <CR> (carriage return), or just <CR> for the default (the last one tested). Then the test asks what slave. Type the slave address number <CR>, or just <CR> for the default (the last one tested). The next parameter to be specified is the sequence of operations. The test prints the available operations and puts the carriage on the next line. Type the desired sequence separated by commas. Type <CR> immediately to use the last sequence typed. Type ALTMODE to go to DDT.

   Then the test asks a number of questions, depending on what operations were requested:

   a. TYPE THE DENSITY (CR FOR 1600 bits/in):

      Specify the density to be used (the first digit is sufficient; i.e., 8 for 800 bits/in).

      NOTE
      For TM03, density can only be changed at BOT. Only 800 bits/in and 1600 bits/in are allowed on TM03.

   b. FORMAT (CR,0 = CORDMP, 1 = 7TRK, 2 = ASCII, 3 = COMPAT):
      Type the desired format.

      NOTE
      7-track and ASCII format are not allowed on TM03.

   c. PARITY (CR,0 = ODD; E = EVEN):

      Type the desired parity (CR = 0). This question will not be asked if 1600 bits/in density was specified.

d. RECORD SIZE (4-1000(8), CR = 100):

Type the desired size in words.

e. CLOSE SKEW WINDOW ON WRITES Y OR N (EXEC MODE ONLY)

Type Y only if the CLOSE SKEW WINDOW maintenance mode is to be run on writes. Otherwise, type N.

f. DATA PATTERN (CR FOR THE LAST ONE):

Type data used for writes and reads.

g. DATA MASK(CR = -1):

Type desired data mask.

h. REPT COUNT (CR = 1):

Type the desired number of repetitions in decimal.

The test will clear the statistics then execute a sequence of commands beginning at the current position of the tape, until the repeat count is satisfied, until ^C is typed, or until the restart switch is turned on. If the repeat count was satisfied, the test asks for a new command sequence. Type ^C or set the restart switch if the test is finished.

If the tape reaches EOT during the test, the test will complete the command sequence, rewind the tape, and begin the command sequence again. Turning on the TOTALS switch will cause the statistics to print out.

Note that timing counter is used when executing each sequence of commands. If the drive ready bit does not come up within the limit of the timing counter (approximately 1 s) due to problems occurring in drive, error message "ACCL bit did not come up in time" will be printed. This will eliminate the usual process of waiting 7 s for OPI error to come up and has the drive hung for a long period of time. The timing counter has no effect on the test if the drive is working.

Note that an SF6 command will attempt to space 6 records at once. If a tape mark is read (or BOT is reached or EOT is past), all 6 records will not be spaced.

To loop over one particular record that failed, for instance, in the unknown read test, find the number of that record from the printout (for example, record number 69). Run the 0 test with the command string of RW (rewind). Then use the command SF with a repeat count of 68. The tape will be positioned immediately before the 69th record. Then use the commands UF, SR with a repeat count of 500 or so to loop on the record (-1 to loop forever).

If a DDT breakpoint is to be hit after each operation, put a breakpoint at OBRK.

2. RELIABILITY TESTS:
The reliability diagnostic writes non-random data patterns on tape (including the alternate pattern selected during option select) in random size records (for the list of patterns, see section 9.0 called table of data patterns). The reliability diagnostic consists of four tests: test 2 through test 5.

The reliability tests will be run if any of the following responses are typed to the WHAT TEST question:

Type	For Density
R	1600 bits/in, then 800 bits/in (test 2 and test 3)
R1	1600 bits/in only (test 2)
R8	800 bits/in ONLY (test 3)
R5	556 bits/in ONLY (test 4) (TM02 only)
R2	200 bits/in (test 5) (TM02 only)

If the OPRSEL switch (SW 14) is on, you will be asked the following questions:

a. DATA COMPARE MASK (CR = CHECK ALL BITS):
The data compare mask is a 36-bit octal number indicating what bits the program is to check when doing data compare. Type 0 to ignore data altogether, or -1 or just carriage return to check all bits. A 1 in a bit position means that bit is to be corrected. In COMPAT tape format mode, the last four bits are always printed as-is, but are always ignored during data compare.

b. OPTIONAL DATA PATTERN:
The 36-bit pattern typed here will be used as one of the data patterns. If carriage return is typed, the result will be the most recent optional data pattern (initially 123456,,654321).

c. CLOSE SKEW WINDOW ON WRITES Y OR N <CR>
If the Close Skew Window maintenance mode is to be used, while writing, type Y. Otherwise, type N. The default is N.

In user mode, the question will not be asked about the skew window and the window will not be tightened.

d. START TESTING USING FORMAT (CR,0 = CORDMP; 2 = ASCII; 3 = COMPAT):
When any reliability section is started, the format specified is used as the first format to be tested. The default density is core dump.

NOTE
ASCII format is not allowed on TM03.

e. TYPE NR OF THE TAPE PASSES/DENSITY (DECIMAL) (CR = 1):
One tape pass is defined as write, read reverse, read forward until the EOT mark is seen. Type the desired number of tape passes for each density.

The program writes, reads reverse, then reads forward the current pattern and pseudo-random record size on the lowest numbered TM02 or TM03 and lowest numbered slave. The program then writes, reads reverse, and reads forward the current pattern and record size on the next to the lowest slave on that TM02 or TM03, and so on until all slaves on all TM02s or TM03s have been written. Note that if a nonrecoverable write error occurs after 10 write retries, the program will skip the read reverse and read forward operations. The program then selects the next pattern, computes another random record size, moves the data buffer up in core another 8K, and repeats the above sequence.

After all the patterns have been written, the program writes a tape mark, does a space REV and then space FWD, and checks after each operation that the status bit TM is asserted. Now, the next format in the sequence core dump, ASCII (TM02 only), COMPAT is used. If a drive reaches the EOT mark, the tape is rewound and the above sequence continues on the remaining drives until all drives have reached EOT. One pass has now been completed. Another is started (continuing with the current pattern) at the current density if the number of passes completed is less than the number of passes requested during the option select questions.

At the end of the desired number of tape passes of each density, the statistics (number of records written, number of errors, etc.) are printed out, and the next density of the reliability test is started if desired. After all the densities that were requested are run, control will pass to the console package software if the abort switch (0) is set. If the switch is not set, the reliability test will run for another program pass.

To see the statistics during the test, use the statistics printing switch (SW 2).

If the program gets a non-retry (FATAL) type of error (off-line error, etc.) or gets 10 (decimal) nonrecoverable retry type errors, the drive is ignored for the rest of the test.

The set of patterns used for reliability begins at DATTAB and continues through DATTAB+RELPAT-1 (139 single-word patterns). The set of patterns includes 0s, 1s, alternating 1010..., rotating 1s, field of 0s, rotating 0 in field of 1s, and some worst-case patterns used during the interchange and compatibility tests. The test also uses some additional patterns that repeat over several PDP-10 words. The format is always COMPAT for these multiple-word patterns.

Because the OPI time extend ECO may not be in. The maximum record size at 200 bits/in is not larger than 1777(octal) words. The maximum record size at 556 bits/in is 3777(octal) words. The minimum record size is 4 words.

3. UNKNOWN TAPE READ (TEST NUMBER 6)
This test is used to read a tape without knowing the format of the data on the tape, or the size of the records on the tape. In other words, the test just checks for vertical parity, CRC, and LRC in NRZI mode, and for vertical parity and dead tracks in PE mode.

If the OPRSEL switch (SW 14) is set, the following questions will be asked:

a. READ REVERSE ALSO - Y or N <CR>
Type Y to read reverse as well as read forward; type N to just read forward. The initial default is Y.

b. TYPE THE DENSITY OF TAPE (CR FOR 1600 bits/in):
The initial default density when the program is restarted is also 1600 bits/in. Typing the first digit of the density is sufficient.

NOTE
200 bits/in and 556 bits/in are not allowed on TM03.

c. TAPE FORMAT (CR, 0 = CORDMP; 2 = ASCII; 3 = COMPAT):
Type the desired format. Note that this question is important only if it is necessary to examine the data after each read with DDT.

NOTE
ASCII format is not allowed on TM03.

d. DO YOU WANT SYSERR RECORDING Y OR NO <CR>
In exec mode, the tape is read to see if there is a PE IDB burst at BOT. If there is and in NRZ density, or if there is not and you are in PE density, a warning will print out and ask if the diagnostic is to continue. Type Y to continue or N to restart the diagnostic. If nothing is typed, the program will timeout and assume Y.

The tape is read (read FWD if N was typed to question 1, read FWD, read REV, space FWD) until EOT is seen or until 2 tape marks in a row are seen (whichever comes first). If the frame counter shows that less than 13 decimal frames were read (in exec mode), or if less than 4 words were read (in user mode), a warning is printed out along with any other errors that occurred.

In exec mode, after the tape is read once, control will pass to the console package if the ABORT switch (0) is set. If the switch is not set, the tape will be read again. In user mode, the test will always abort after one pass.

After each read the record data will be present in the data buffer for examination by DDT. The routine will handle any size record, but if the record is greater than 7776 (octal) words, the first word of the record will be destroyed by the 7777th word, etc.

4. INTERCHANGE AND COMPATIBILITY TESTS (IW,IR,C)
This test allows writing a tape on one drive and reading and data comparing the tape on another drive at 1600 bits/in or 800 bits/in (selectable). The header and data format is compatible with the other PDP-10 diagnostic interchange tests (including the TM10A and TM10B magnetic tape reliability diagnostic program).

The interchange tests are mainly used when writing a whole tape (or at least a lot of tape), and transferring the tape to other drives to be read. The compatibility test is mainly used when many drives write and read the same tape. The format of the tape is identical and so either test may process a tape written by the other.

If the OPRSEL switch (SW 14) is set when any of these tests are run, the following questions are asked:

a. TYPE THE DENSITY OF TAPE (CR FOR 1600 bits/in):
Type the density. Initially, the density is 1600 bits/in. Only 800 bits/in and 1600 bits/in are legal densities on IW,IR and C test.

b. CLOSE SKEW WINDOW ON WRITES Y OR N <CR>
This question will only be asked during the IW and C tests and only if in exec mode. To get the close skew window maintenance mode to be run while doing writes, type Y otherwise N. The initial and timeout defaults are N.

c. FAST MODE Y OR N <CR>
This question will only be asked during interchange write. If Y is typed, 5 bunches maximum will be written. If N, as many complete bunches as will fit on the tape before EOT is reached are written. The initial and timeout defaults are N.

d. VERIFY TAPES AFTER WRITE Y OR N <CR>
This question will only be asked during interchange write. Type Y if interchange read is to be run immediately after interchange write. Otherwise, type N; the default is N.

When in exec mode and at the start of reading a tape (IR or C), the tape is read to see if there is a PE IDB burst at BOT. If there is and in NRZ density, or if there is not and in PE density, a warning will print out and ask if the diagnostic is to continue. Type Y to continue or N to restart the diagnostic. If nothing is typed, the program will time out and assume Y.

The first record of a set of records (a bunch) is a header record containing the writing drive's type and serial number, processor type and serial number, date, and a comment. If the PHEDSW (switch 19) is set, this header will print out as each header is written or read (once per tape during interchange write).

e. TYPE DATE AS MM-DD-YY
This is asked in exec mode only.

f. COMMENT (18 CHARS):
Type any comment up to 18 characters maximum or type <CR> to continue. The program will then type:

MOUNT TAPES ON ALL DRVS, WRT ENABLED; TYPE CR:

g. DO YOU WANT TO USE REDUCED-SIZE FORMAT
(N = NORMAL FORMAT)?

The interchange write, interchange read and compatibility test provide the option of using either the normal size format or the reduced-sized (half of the normal size) format to read or write bunches of records. To use the reduced-size format, type Y to the question DO YOU WANT TO USE REDUCED-SIZE FORMAT (N = NORMAL FORMAT)?

IW-INTERCHANGE WRITE TEST (TEST NUMBER 7)

This test will write bunch format tapes on all the drives at the current density. When all the bunches are written the statistics are printed. Interchange read will be run if you typed Y to the verify tapes question. If not, the console switches are read. The program then runs another pass of interchange write (if the ABORT switch is not set) or aborts the console software (if ABORT is set).

h. DATA COMPARE MASK (CR = CHECK ALL BITS):
The data compare mask is a 36-bit octal number indicating what bits you want the program to check when doing data compare. Type 0 to ignore data altogether, or -1 or just carriage return to check all bits. A 1 in a bit position means you require that bit to be correct. In COMPAT tape format mode, the last four bits are always printed as-is, but are always ignored during data compare.

i. DO YOU WANT TO USE REDUCED-SIZE FORMAT
(N = NORMAL FORMAT)?
This question is asked again if the response to the question VERIFY AFTER WRITE?, was yes.

IR-INTERCHANGE READ TEST (TEST NUMBER 10)

If the OPRSEL switch (SW 14) is set, the following questions are asked:

a. TYPE THE DENSITY OF TAPE (CR FOR 1600 bits/in):
Type the density. Initially, the density is 1600 bits/in. Only 800 bits/in and 1600 bits/in are legal densities on IW, IR and C test.

b. DATA COMPARE MASK (CR = CHECK ALL BITS):
The data compare mask is a 36-bit octal number indicating what bits the program is to check when doing data compare. Type 0 to ignore data altogether, -1, or just carriage return to check all bits. A 1 in a bit position requires that bit to be corrected. In compatible tape format mode, the last four bits are always printed as -1s, but are always ignored during data compare.

The program then types:

MOUNT INTERCH TAPES ON ALL DRIVES IN THE CONFIGURATION;
TYPE CR:

c. DO YOU WANT TO USE REDUCED-SIZE FORMAT
(N = NORMAL FORMAT)? Y OR N -
The interchange write, interchange read and compatibility test provide the option of using either the normal size format or the reduced-sized (half of the normal size) format to read or write bunches of records. Make sure that response agrees with the format in which the tape was previously written.

If the OPRSEL switch (SW 14) is not set, the program will only ask the last question above and then continue.

This test then reads tapes written in standard bunch format. The test reads on all drives as many bunches as were written (until two tape marks in a row have been read). After all bunches are read, the statistics are printed and the console switches are read. If the OPRSEL switch (SW 14) is not set, the test is over and the program runs another pass of interchange read (if the ABORT switch is not set) or aborts to the console software (if ABORT is set). If interchange read was run to verify the tapes and the abort switch was not set, interchange write, then interchange read is run again.

If OPRSEL switch is set, the test asks to rotate the tapes (move the tape on the first drive to the second, etc.) and the interchange test runs again. The tapes are to be rotated one time less than the number of drives in the configuration (not at all if there is only one drive). After the tapes have been rotated the proper number of times, the console switches are read. The program runs another pass of interchange read (if the ABORT switch is not set) or aborts to the console software (if ABORT is set).

C-COMPATIBILITY TEST (TEST NUMBER 11)

This test assumes there is one tape that is to be moved from one drive to the next on all drives in the configuration.

The first question (after the OPRSEL switch-selected questions above) asked is:

CMD (CR, R = READ TAPE; W = BEGIN WRITING NEW TAPE; E = WRT EARLY EOT):

R Will read all the bunches on the tape and ask the CMD question below.

W Will write a bunch starting at BOT and ask the CMD question below.

E Will ask WRT EOT AFTER HOW MANY BUNCHES (DECIMAL): Type the number of bunches to be saved on the tape. The test will write a logical EOT mark (2 tape marks) after that many records. It will then ask the above CMD question again. The erase will abort if a logical EOT is seen before the requested number of bunches.

CMD (W = WRTBNH; CR,R-REWIND AND EXIT; E = ERASE):

W Writes a bunch and asks the CMD question again.

NOTE

If physical EOT is aborted, the number of bunches on tape remains the same as before the current W command.

R Rewinds the tape, types the statistics, and asks to move the tape to the next drive in the configuration. After all have been tested, the console switches are read. The program runs another pass of compatibility test if the ABORT switch (SW 0) is not set. It aborts to the console software if ABORT is set.

E Erase asks how many bunches from the end (decimal) are to be erased (going toward BOT from the current tape position). If 2 is typed and tape is positioned after the fifth bunch, there will be three bunches left. The test will then rewind the tape, and begin the C test on the next drive in the configuration.

A compatibility tape will consist of a number of bunches of records, each bunch containing a header descriptive record, a number of data records of different lengths, followed by a tape mark. If the bunch is the last bunch on tape, two tape marks will follow the bunch.

A compatibility program should read and data compare all the bunches on tape (until two tape marks are seen in a row), erase the second tape mark, append another bunch, then a second tape mark. However, if the drive sees EOT, the program should reverse to the previous tape mark, append another tape mark (thus aborting the write), and inform the operator that the tape is full.

There are three possible formats for tapes (each incompatible with the other) - 7-track 800 bits/in NRZ, 9-track 800 bits/in NRZ, and 9-track 1600 bits/in PE. The tape format should be clearly marked on the tape, as most drives must be told what they are reading.

Bunch format - 9-track (normal size)

A bunch will contain 199(10) records followed by 1 tape mark. The first record will be a header record of 48(10) frames (see header format). The next 66(10) records will contain 24(10) frames per record. The 1st and 34th record will contain the first 9-track pattern repeated to fill out the record, the 2nd and 35th record will contain the 2nd 9-track pattern repeated, etc.

The next set of 66(10) records will contain 528(10) [=1020(8)] frames per record, with the same data format as the previous paragraph.

The third set of 66(10) records will contain 12024(10) [=27370(8)] frames per record, with the same data format as in the previous paragraph.

At 1600 bits/in, approximately 53 feet of tape will be written. At 800 bits/in, approximately 96 feet of tape will be written.

Bunch format - 7-track 800 bits/in NRZ (normal size)

A bunch will contain 169(10) records followed by one tape mark. The first record will be a header record of 48(10) frames (see paragraph on header format). The next 56(10) records will contain 24(10) frames per record. The 1st and 29th record will contain the first 7-track pattern repeated, the 2nd and 30th record will contain the 2nd 7-track pattern repeated, etc.

The next set of 56(10) records will contain 528(10) frames per record with the same data format as in the previous paragraph.

The third set of 56(10) records will contain 12024(10) frames per record, with the same data format as in the previous paragraph.

At 800 bits/in, approximately 80 feet of tape will be written.

HEADER FORMAT

The header will be a PDP-10 type SIXBIT descriptive record containing information about the writing drive and processor. Each test string may be 6 characters long, left-justified, with unused bytes set to 0 (space). The first character will be written onto and read from tape before the second character, etc.

If string 6 is non-0, strings 6, 7, and 8 will be taken as a SIXBIT comment (18 characters max, unused characters should be zeroed).

String	Text String	Example	Meaning
1	DRIVE TYPE	TU45	
2	DRV SER NR	5039	
3	PDP TYPE	PDP-10	
4	PROC SER NR	1	
5	DATE	100284	OCT. 2, 1984
6	COMMENT	THIS I	
7	COMMENT	S A CO	
8	COMMENT	MMENT.	[THIS IS A COMMENT.]

NOTE

Bunch size with reduced-size format equals half of bunch size with normal format. Header size and data patterns are the same for both formats.

5. REFER TO THE LISTING ON MICROFICHE

ERROR SUMMARY

ERROR LOOPING STRUCTURE

The test in the diagnostic is structured so that the user of the program may loop on as small a section of the current test as possible to provide in most cases a tight scope loop. However, this tight scope loop is only possible (unless DDT is used) if the subtest to be looped on causes an error. It is possible to loop on a whole test (even if no error occurs) which may contain one or more error loops.

```
TST<N>:   TSTSET          ;THE START OF TST<N>
                               ; INITIALIZES VARIABLES

          LOOPST          ;SETS UP TO LOOP ON
                               ; CURRENT TEST

          LOOPSE          ;SETS UP TO LOOP ON ERROR
                               ; ON SUBTEST 1

          (SUBTEST 1)
          PNTERR          ;PRINTS OUT ERRORS
          LOOPFE          ;IF AN ERROR OCCURRED, LOOPS
                               ; BACK TO MOST RECENT LOOPSE
                               ; (IN THIS CASE SUBTEST 1)

          LOOPSE          ;SETS UP TO LOOP ON ERROR
                               ; ON SUBTEST 2

          (SUBTEST 2)
          PNTERR
          LOOPFE          ;IF LOOP ON ERROR, LOOPS
                               ; ON SUBTEST 2

          LOOPCK          ;LOOPS BACK TO MOST RECENT
                               ; LOOPST IF YOU WANT TO
                               ; LOOP ON CURRENT TEST
```

While looping on error, the loop on error routine checks to see if EOT (end of tape) is asserted. If tape is at EOT and a test which normally will hit EOT (EOTTFL flag is 0) is not being run, LOOPFE will rewind the tape before continuing to loop on test.

ERROR HANDLER AND PRINTOUTS

Register 16 (named E) is used as an index in the error stack. When an error occurs, an entry is put in the error stack. The error is actually reported the next time a PNTERR instruction is executed. This stacking of errors is necessary because critical registers may change state while the error is being reported. An entry in the error stack includes this critical data.

On normal format printout, some register names have parentheses and some do not. The names without parentheses are the critical stacked registers. The names with parentheses are not stacked, and may contain data other than what the registers contained at the time of the error. For instance, if an error occurs during an operation with the GO bit high, the control register will probably be printed later when the GO bit is low. However, the data in the command bits will still be valid. To see the true value of GO at the time of the error, look at the status register DRY bit (not GO) because the status register is saved at the time of the error.

Whenever the program detects an error, a fault instruction (UUO) is executed, which looks like fault $C,<ADDRESS>. The <ADDRESS> is a pointer to the error table which contains an entry for every possible error. Each entry in the error table consists of flags to tell what format the error should be printed out in, text describing the error, and board isolation data. The $C indicates that the program should continue after the error is stacked. (the $C may also be in the error table.) If the $C were not there, the fault routine would return directly to the routine that called the routine that executed the fault (the next address on the P STACK). Thus, if $C is not there, the current routine will abort.

On word data errors, A WD ADR of 1 is the first word of the transfer. On frame data errors, frame 1 word 1 means the first frame of the transfer.

When there are more than 8 word data errors (defined by DERMAX at assembly time - see paragraph on assembly parameters), the first 7 are printed, followed by the last word in error. This feature allows the user to infer whether the data errors persisted until the end of the transfer or whether the errors stopped in the middle of the transfer.

ERROR CLEARING ROUTINES

To initialize the RH and magtape, the program does Massbus INITS, by the instruction MTACLR.

MTACLR also reads the error register and puts an entry on the error stack if the error register is not clear after the Massbus INIT.

In the reliability, compatibility, interchange, and unknown read sections, a Massbus INIT is done only at the beginning of the test and after the printing of any errors. Drive clears are done normally but only if the ATA, ERR, or SSC bits are up.

PROCESSOR-DETECTED ERRORS

A NXM error will cause the program to fail. If a parity error is detected, the memory containing the program and the data buffer are accessed. A printout occurs if a parity error is detected and the word is written back into memory to clear the parity error. The diagnostic then continues.

GENERAL INFORMATION

Code — DSUBA.SAV

Title — DECSYSTEM-2020 Unibus Adapter Exerciser

Abstract — This diagnostic is designed to run on a DECSYSTEM-2020 in exec mode, and will verify that the Unibus adapters function properly. A minimum of one, and preferably at least two, Unibus exerciser modules are required to achieve complete testing. However, a stand-alone mode of operation exists which does not require anything to be placed on the Unibus. The program was written for manufacturing, but nothing precludes its use in the field.

Hardware Required — KS10 mainframe
Minimum of 128K of memory, 256K recommended
1 or 2 Unibus adapters (UBA) (Rev. 2 or higher)
0 to 12 Unibus exerciser modules (UBE) (preferably at least 2)

Preliminary and Associated Programs — Refer to diagnostic hierarchy (KS10 STD module)

Restrictions — If SW 20 (STANDA) is not set and the Unibus exerciser modules are not in place, the diagnostic will only run the first 24 tests. The operator will receive the following message:

WARNING: NO EXERCISERS ON BUS, - UBA MEDIATED TESTS ABORTED

IF SW 20 (STANDA) is set, the diagnostic will run all tests without Unibus exerciser modules.

It is important to remember that the KS10 CPU has absolutely no control over the KS10 bus parity line. If the operator does not disable parity checking via the 8080 console, the occurrence of a parity error in the Unibus adapter will automatically and irrevocably stop the diagnostic. However, if parity is disabled, the program will check for assertions of the SMPE bit in the UBA status register (assuming of course that it is possible to access this register), and provides fairly extensive scope looping facilities for this kind of error.

Notes

1. Note that all of the Unibus exerciser modules must have jumper W1 installed, except for the last one on the Unibus. Also, be sure that jumper CA1 to CB1 is removed from each slot occupied by an exerciser module.

2. For further reference, consult the Unibus systems exerciser diagnostic, (MD-11-DZKUA), the Unibus exerciser module diagnostic, (MD-11-DZKUB), the bus exerciser engineering specifications (M7855-0-8), the Unibus adapter technical drawings (M8619-0-UBA-1-H), and the KS10-Based DECSYSTEM-2020 Technical Manual (EK-0KS10-TM)

Loading and Starting Procedure — Standard (Refer to the KS10 STD module.)

Control Switches — Standard (Refer to the KS10 STD module.)
Right half switches are as follows:

18 — Allows start at operator-selected test.

19 — Loop on selected test.

20 — Stand-alone mode - use non-Unibus exerciser mediated tests only.

21 — Check parity even if it seems to be stuck high.

22-35 — Not used

OPERATIONAL CONTROL
The program will ask for the number of the Unibus adapter to be tested, and autosize for Unibus exerciser modules on the adapter in question, and will then commence execution. Execution will start at test 1 and proceed through the tests in numerical order. If the Unibus adapter is not functioning so that autosizing is not possible, the operator must restart the diagnostic and enter all parameters manually. The only exceptions to this default flow of control, are when an error occurs in the basic accessing tests, (1, 2, 3, 21, 22), in which case all remaining tests are aborted. Another exception is when the autosizing routine fails to find any exercisers on the Unibus, in which case the exerciser mediated tests are aborted. The switches allow the operator to perform an operator-selected test, to loop on a test, bypass exerciser-mediated tests, and input Unibus exerciser parameters manually. All the standard diagnostic switches are also used (ring bell on error, loop on error, etc.).

TEST SUMMARY
Table 1 summarizes the tests performed by DSUBA.

ERROR SUMMARY
Standard (Refer to the KS10 STD module.)

Table 1 DSUBA Test Summary

Test	Description
1	Checks that the status register of the Unibus adapter (UBA) currently being tested can be successfully accessed without causing a page failure or parity error. If the status register cannot be accessed, the diagnostic is aborted, and unless the force parity testing switch was set, a stuck high condition on the parity bit inhibits further parity testing.
2	Checks the ability of the CPU to access the UBA maintenance register. As this is a write only register, we only check that we can write to it without causing a page failure or setting parity high.
3	Checks that we can successfully write to and read from the UBA paging registers, and that parity does not get set as we do so.
4	Writes and reads several test patterns into and out of the UBA status register. Only the read/write bits are tested, which includes the high and low PIA bits (bits 30-32 and 33-35 respectively), and the disable transfer if bad memory data bit (no. 28). If an error is detected, the test is repeated with the maintenance bit set, thus providing a quick check of the 4 X 4 memory locations.
5	Checks that we can write several test patterns into and out of the UBA status register without causing a parity error.
6	Checks the addressing lines for the 64-word UBA paging RAM by writing the number of a page into that specific page, doing the same for all pages, and then checking that nothing was over-written. This procedure is done both top-down and bottom-up.
7	Checks the read/write bits for each location in the UBA paging RAM. There is no longer any concern about a RAM parity bit. The error routine types out the bits in the format in which they are written, not the read format.
10	Checks that both the nonexistent device and the time-out bits are set high on a write to a nonexistent I/O device. If both bits are in fact set, then the ability of UBA INIT to clear these bits is tested.
11	Checks out the WRIO -> NPR word transfer to KS10 memory wrap-around test. Odd word transfers are done both to a field of 1s and a field of 0s to check the read-pause-write operation.
12	Uses the maintenance wrap-around mode of the UBA to initiate NPR transfers to the low byte of the even word of KS10 memory.

Table 1 DSUBA Test Summary (Cont)

Test	Description
13	Uses the UBA maintenance wrap-around mode to initiate read-reverse transfers to KS10 memory. Both even and odd word transfers are read-pause-write operations, and are done alternately to a field of 1s and a field of 0s.
14	Uses the UBA maintenance wrap-around mode to initiate full 36-bit fast transfers to KS10 memory. The even word transfer is stored in the UBA and should not disturb memory, and the odd word transfer should cause both words to be written into memory at once.
15	Uses the UBA maintenance wrap-around mode to initiate a word NPR transfer from KS10 memory. Both even and odd words are tested.
16	Uses the maintenance wrap-around mode of the UBA to initiate NPR transfers to the high byte of the even word of KS10 memory. This is a read-pause-write operation, and the transfers are made to both a field of 1s and a field of 0s.
17	Uses the maintenance wrap-around mode of the UBA to initiate NPR transfers to the low byte of the odd word of KS10 memory. This is a read-pause-write operation, and the transfers are made both to a field of 1s and a field of 0s.
20	Uses the maintenance wrap-around mode of the UBA to initiate NPR transfers to the high byte of the odd word of KS10 memory. This is a read-pause-write operation, and the transfers are made to both a field of 1s and a field of 0s.
21	Uses the wrap-around mode of NPR transfers to do WRIO-NPR transfers to @MEMLOW for all pages of the UBA paging RAM. If the data does not show up in @MEMLOW, then an effort is made to see if some other location was disturbed. The number of the paging RAM is written into the location, and should end up in the LH word, with the address in the RH.
22	Uses the wrap-around mode to write into all KS10 memory locations from MEM low to MEM max. This test uses paging RAM location zero exclusively, and writes the address of location into each location of memory. This test only does transfers to a few select locations, and if an error occurs the program will determine if another memory location was disturbed.
23	Uses the wrap-around mode to write into all KS10 memory locations from MEM low to MEM max. This test uses paging RAM location zero exclusively, and writes the address of a location into each location of memory. There is no attempt to clear memory or reflush cache after each write. Note: This test transfers to 256K max.
24	Checks that the upper two bits of Unibus data can be disabled when the disable bit is set in the paging RAM. As a Unibus exerciser can only transfer the lower 16 bits, this test must be done in wrap-around mode.
25	Checks that the I/O registers of the Unibus exercisers can be successfully accessed without causing I/O page fails. NOTE: If this test fails, all other UBE tests are aborted.
26	Checks that the data register of all the UBEs on the Unibus can be written into and read from without losing any data.
27	Is similar to test 25, except that it checks the data paths to the UBE cycle count registers.
30	Is similar to test 25, except that it checks the UBA address register data paths.
31	Is similar to test 25, except that it checks the data path to the UBE control register 1.

Table 1 DSUBA Test Summary (Cont)

Test	Description
32	Is similar to test 25, except that it checks the data path to the read/write bits of the UBE control register 2.
33	Sets up the UBEs to do two half-word transfers to one word of KS10 memory. Only one transfer and one pattern is used for each UBE on the bus. This test is intended to verify that the basic control signals work, not to check either the data or addressing lines.
34	Sets up the UBEs to do four byte transfers to one word of KS10 memory. Only one transfer and one pattern is used for each UBE on the bus. This test is intended to verify that the basic control signals work, not to check either the data or addressing lines.
35	Sets up the UBEs to do two read-reverse transfers to one word of KS10 memory. Only one location and one pattern is used for each UBE on the bus. This test is intended to verify that the basic control signals work, not to check either the data or addressing lines.
36	Sets up the UBEs to do one fast transfer to one full word of KS10 memory. Only one transfer and one pattern is used for each UBE on the bus. This test is intended to verify that the basic control signals work, not to check either the data or addressing lines.
37	Sets up the UBEs to do one half-word transfer from one word of KS10 memory. Only one location and one pattern is used for each UBE on the bus. This test is only intended to verify that the basic control signals work.
40	Does a WRIOB to the low byte of the data register of each UBE on the bus, and then does a RDIOB from the same register. Only a test pattern of all 1s is transferred. This test is only intended to verify the operation of the basic control signals.
41	Does a WRIOB to the high byte of the data register of each UBE on the bus, and then does a RDIOB from the same register. Only a test pattern of all 1s is transferred. This test is only intended to verify the operation of the basic control signals.
42	Causes a UBE to do an NPR request that results in a full 36-bits worth of data being read from KS10 memory at once, and then being transferred to the UBE one half-word at a time. Only one test pattern and one memory location is used for each UBE. This test is intended only to check the basic control signals.
43	Checks that the interrupt vectors in the VECNUM table are in fact correct. Each UBE is made to interrupt the KS10 CPU, and an error is generated only if the resultant vector is not the same as the one given, or if no interrupt is generated at all. If the actual vector is incorrect, it is automatically autosized and printed out during the error call.
44	Causes a Unibus exerciser to perform multiple NPR transfers from MEM low to MEM max. Only one exerciser is used in this test, and routine BLOCKT is used to work the UBE. Note that this test transfers up to 256K max.
45	Checks that having the AC low line asserted on the Unibus (by a UBE) will set the AC/DC LOW bit in the UBA, and that if this bit does get set, that it can be cleared satisfactorily. (Note only one UBE is used in this test)
46	Causes a Unibus exerciser to do two NPR word transfers to KS10 memory without relinquishing control of the bus.
47	Checks the ability of the UBA to translate from Unibus BR4 level to any KS10 PI level. A Unibus exerciser is set up to interrupt at BR4, and the UBA status register is sequentially set up for PI levels 1 to 7.

Table 1 DSUBA Test Summary (Cont)

Test	Description
50	Checks the ability of the UBA to translate from Unibus BR5 level to any KS10 PI level. A Unibus exerciser is set up to interrupt at BR4, and the UBA status register is sequentially set up for PI levels 1 to 7.
51	Checks the ability of the UBA to translate from Unibus BR6 level to any KS10 PI level. A Unibus exerciser is set up to interrupt at BR6, and the UBA status register is sequentially set up for PI levels 1 to 7.
52	Checks the ability of the UBA to translate from Unibus BR7 level of any KS10 PI level. A Unibus exerciser is set up to interrupt at BR7, and the UBA status register is sequentially set up for PI levels 1 to 7.
53	Checks that the interrupt-pending bit of the Unibus adapter gets set when a UBE is trying to interrupt. The KS10 PI system is turned off, and the bits are read at leisure. No interrupts occur.
54	Checks that the interrupt pending bits of the Unibus adapter get set when a UBE is trying to interrupt. The KS10 PI system is turned off, and the bits are read at leisure. No interrupts occur.
55	Goes through all combinations of two different BR levels, and checks that the Unibus arbitrator logic functions correctly. The first two UBEs listed in UBETBL are set up to interrupt the CPU and are both set off at the same time by SIMUGO.
56	Is similar to test 45, except that it tests the ability of the UBA to arbitrate between BRs and NPRs. (The NPR should always get priority.)