

PDT 150

PDT 11/150 SYS EXER
CVKDADO

AH-F239D-MC
FICHE 1 OF 1

NOV 1980
COPYRIGHT © 78-80
MADE IN USA



The main body of the document contains a dense grid of approximately 100 small, illegible tables or data blocks. Each block appears to be a structured data set, possibly a ledger or a series of reports, but the text is too faint to be transcribed. The blocks are arranged in roughly 10 columns and 10 rows.



.REM %

IDENTIFICATION

PRODUCT CODE: AC-F238D-MC
PRODUCT NAME: CVKDADO PDT11/150 SYS EXER
PRODUCT DATE: JULY 1980
MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1978, 1979, 1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT).
1.2	SYSTEM REQUIREMENTS.
1.3	RELATED DOCUMENTS AND STANDARDS.
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES.
1.5	ASSUMPTIONS.
1.6	RUNNING SYSTEMS PROGRAMS.
2.0	OPERATING INSTRUCTIONS.
2.1	LOADING AND STARTING PROCEDURES.
2.2	SPECIAL ENVIRONMENTS.
2.3	OPERATIONAL SWITCH SETTINGS
2.4	PROGRAM OPTIONS.
2.5	EXECUTION TIMES.
2.6	POWER FAIL.
2.7	COMPATABILITY TESTING.
2.8	COPY UTILITY.
2.9	RUNNING WITHOUT CONSOLE.
2.10	CHANGING DEFAULT BAUD RATES.
3.0	ERROR INFORMATION.
3.1	ERROR REPORTING PROCEDURE.
3.2	ERROR HALTS.
4.0	PROGRESS & PERFORMANCE REPORTS.
5.0	DEVICE INFORMATION TABLES.
6.0	DEVICE CONNECTOR LAYOUT.
7.0	SUMMARY OF TESTS.

1.0 GENERAL PROGRAM INFORMATION.

PROGRAM BUG FIXES FROM REV CVKDAD TO CVKDAD:

- A. REPLACE NON-EXISTANT ERROR 140 TO 130.
- B. DO NOT EXAMINE KBD WHILE IN SYNC MODEM TEST.
- C. DO ONLY WORST CASE PATTERNS ON COMPATABILITY TESTS
OTHERWISE DO RANDOM/WCP PATTERNS ON ALTERNATE PASSES.

PROGRAM ENHANCEMENTS FROM REV CVKDAD TO CVKDAD:

- A. NEW DISKETTE DATA PATTERNS.
THE 1'ST WORD OF EACH SECTOR IS ITS TRACK/SECTOR NUMBER.
THE REMAINING WORDS ARE FROM A PATCH-ABLE TABLE CONTAINING WORST CASE PATTERN
...OR RANDOM DATA PATTERNS.
THE PATTERN ALTERNATES BETWEEN WORST CASE PATT ON ODD PASSES
& RANDOM PATTERNS ON EVEN PASSES.
- B. THE DEFAULT BAUD RATES ARE PRINTED ON THE 1'ST PASS STARTUP
FOR THE CLUSTERS, PRINTER, SYNC & ASYNC COMM PORTS.
- C. EXPANDED DISKETTE ERROR MESSAGES.
- D. SET SYNC MODEM TO 9600 BAUD.
- E. FLOPPY RESTORE COMMAND TEST (DXTST4):
IF DX0 DROPPED, CHECK & TEST DX1 INSTEAD OF JUMPING OVER DX1 PORTION.
- F. CLEAR DEVCT AFTER INCREMENTING 'PASS' IN 'EOP' FOR APT.
- G. SET MAX SEEKS TO 500 ONLY IF NOT DOING COMPAT TESTS.
- H. ADD CONTROL-U FUNCTIONALITY TO INPUTTING A NEW DEVICE MAP.
- I. INCREMENT DEVCT MORE FREQUENTLY FOR APT.
- J. CHANGE SECTOR INTERLEAVING FROM EVERY OTHER SECTOR
TO EVERY 3'RD SECTOR TO CATCH INTERLEAVING.
- K. CLEAR ERROR COUNTERS ON RESTARTS OF COMPATABILITY TESTS 1 & 2.
- L. ADDED FILL & EMPTY BUFFER TESTS TO DEVICE MAP.
- M. IMPROVED ABILITY TO DISTINGUISH SEEK ERRORS FROM READ HEADER FAULTS.
- N. TYPE PASS NUMBER WITH EACH ERROR REPORT.
- O. ADDED ABILITY TO RUN WITHOUT CONSOLE.
- P. ADDED ABILITY FOR OPERATOR TO EASILY MODIFY BAUD RATES.
- Q. ADDED ABILITY TO INPUT UNIT SERIAL NUMBER & OUTPUT IT ON EACH ERROR REPORT
& EACH END OF PASS.

1.1 PROGRAM PURPOSE (ABSTRACT).

THE PDT-11/150 SYSTEM EXERCISER TESTS THE PROCESSORS ABILITY TO OPERATE ALL ITS PERIPHERALS IN INTERRUPT MODE AT THE SAME TIME WITH EMPHASIS ON THE DISK SUBSYSTEMS.

IT SHOULD BE NOTED THAT IT IS NOT A DIAGNOSTIC OF THE PERIPHERALS BUT A SERIES OF SYSTEM INTERACTION TESTS.

THE PROGRAM IS DEFAULTED TO EXERCISE THE CLUSTER TERMINALS & THE COMM PORT IN INTERNAL LOOPBACK (MAINT) MODE .
THE DEFAULT CAN BE CHANGED TO EXERCISE ANY OF THE CLUSTER TERMINALS AND/OR THE COMM PORT IN EXTERNAL LOOPBACK.
SEE PROGRAM OPTIONS SEC. 2.4.

TESTING OF THE ACTUAL CLUSTER TERMINALS OR COMM DEVICE IS NOT PERFORMED. THESE DEVICES CAN ONLY BE TESTED EITHER IN EXT. OR INT. LOOPBACK.

THE PRINTER LOGIC WILL BE EXERCISED IF SIZING DETERMINES IT TO BE PRESENT OR IF DATA TERM RDY IS ASSERTED ON ITS CONNECTOR.

THE 'EIS-FIS' LOGIC WILL BE EXERCISED IF SIZING DETERMINES IT TO BE PRESENT.

THE CONSOLE TERMINAL IS NOT TESTED.

AFTER THE MEMORY AND EIS-FIS (IF PRESENT) HAVE BEEN TESTED & ALL THE PERIPHERALS ARE BEING EXERCISED & INTERRUPTING AT RANDOM, THE DISK SUBSYSTEM TESTS WILL BEGIN.
SEE TEST SUMMARY SEC. 7.0.

NOTE: IF RUNNING UNDER APT, THE SYNC COMM LOGIC WILL BE TESTED ONLY ON THE 1'ST PASS. THIS IS SO APT 'BREAKS' SO NOT CAUSE FALSE SYNC COMM ERRORS.

THE PROGRAM CONTAINS A UTILITY WHICH CAN COPY THE SYSTEM EXERCISER DISK FROM DX0 ONTO A SCRATCH DISK IN DX1.
THIS ENABLES THE OPERATOR TO CREATE BACKUP COPIES OF THE EXERCISER DISK.
SEE SECTION 2.8.

THE PROGRAM ALSO CONTAINS A COMPATABILITY TESTING OPTION.
SEE SEC. 2.7.

1.2 SYSTEM REQUIREMENTS.

HARDWARE REQUIREMENTS:

PDT-11/150 SYSTEM
8K MEMORY - MINIMUM
CONSOLE TERMINAL (NOT REQ'D AFTER STARTUP)
EXTERNAL LOOPBACK CONNECTORS (OPTIONAL) FOR COMM & CLUSTER TERMINAL PORTS.
(SEE PROGRAM OPTIONS SEC. 2.4)
VT-100 CONSOLE MUST BE SETUP FOR JUMP SCROLL.

SOFTWARE REQUIREMENTS:

THIS EXERCISER IS DESIGNED TO RUN IN ANY OF THE FOLLOWING WAYS:

STAND ALONE
WITH APT MONITOR (INCL. SCRIPT MODE)
WITH RT-11 MONITOR
WITH DIAGNOSTIC SUPERVISOR (NON SCRIPT MODE)

AN XXDP DRIVER IS NOT AVAILABLE FOR THE PDT-11/150
AT RELEASE TIME OF CVKDAD.
I. E. XXDP DOES NOT PRESENTLY SUPPORT THE PDT-11/150.

1.3 RELATED DOCUMENTS AND STANDARDS.

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS	175-003-009-02
APT	MD-11-DZZMA
SYSMAC	MD-11-DZQAC

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

THIS EXERCISER ASSUMES THAT THE POWER UP SELF TEST RUNS ERROR FREE.

1.5 ASSUMPTIONS

THIS EXERCISER ASSUMES THAT THE OPERATOR HAS MODIFIED THE DEVICE MAP (\$DEVN) AT LOC. 1246 IF THE DEFAULTS DO NOT AGREE WITH THE ACTUAL SYSTEM CONFIGURATION. SEE PROGRAM OPTIONS SEC. 2.4.
THE PROGRAM ALSO ASSUMES THE SOFTWARE SWITCH REGISTER IS PROPERLY SET UP (SWREG) AT LOC. 176. SEE SEC. 2.3.

1.6 RUNNING SYSTEMS PROGRAMS

UNLESS THE SYSTEMS PROGRAMS ARE KNOWN TO INITIALIZE THE BAUD RATES OF EACH DEVICE THRU THE PARAMETER REGISTER, THE OPERATOR SHOULD POWER DOWN & UP AGAIN WHEN FINISHED RUNNING THIS EXERCISER. THIS IS TO RESTORE THE PDT-11/150 TO ITS DEFAULT BAUD RATES. OTHERWISE, THE DEVICES WILL ATTEMPT TO RUN AT WHATEVER BAUD RATES

WERE LAST USED BY THIS EXERCISER.

2.0 OPERATING INSTRUCTIONS.

2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED PAPER TAPE.
THE PROGRAM WILL AUTO-START AFTER LOADING.

THE SYSTEM EXERCISER PROGRAM WILL AUTO-START AFTER BOOTING.

EXAMPLE OF STARTUP ASSUMING THAT:

24K WORDS OF MEMORY
EIS-FIS OPTION NOT PRESENT
PRINTER NOT PRESENT
CLUSTER TERM #2 NOT TO BE TESTED
COMM PORT EXT. LOOPBACK
CLUSTER TERM 1 & 3 INT. LOOPBACK.

(STARTUP TYPEOUTS)

CVKDAD PDT-11/150 SYSTEM EXERCISER

SWR = 000000 NEW = 110000 <CR> (HALT ON ERR, ENABLE PERFORMANCE REPORTS)
DEVM = 000017 NEW = 20005 <CR> (CHANGE THE DEFAULTS SEE SEC. 1.5)
SERIAL NO. = 123 NEW = A123<CR> (UP TO 7 CHARS ALLOWED)

24K MEMORY PRESENT
EIS-FIS OPTION NOT PRESENT
TERM #2 TESTING DROPPED
PRINTER NOT PRESENT

PROGRAM DEFAULTS (UNLESS MODIFIED):
CLUSTERS @ 2400 BAUD
PRINTER @ 9600 BAUD
SYNC & ASYNC COMM PORTS @ 9600 BAUD

INSERT SCRATCH DISKS, TYPE 'P' FOR NORMAL TESTING
'240G' FOR NORMAL RESTARTS
'250G' TO COPY SYS EXERCISER DISK
'260G' FOR COMPATABILITY PASS 1: WRITE
'270G' FOR COMPATABILITY PASS 2: READ

(PROGRAM HALTS & WAITS FOR 'P' & CONTINUES....ABOVE NOTE & HALT OMITTED UNDER APT)
THE PROGRAM WILL BEGIN ACTUAL TESTING AT THIS POINT. (SEE SUMMARY OF TESTS SEC. 7.0)

DURING THE TESTS, THE PROGRAM WILL PRINT PROGRESS REPORTS (SEE SEC. 4.1) IF
BIT 12 IS SET IN THE SWREG (SEE SEC 2.3).
END OF PASS & TOTAL ERROR COUNT IS PRINTED AT THE CONCLUSION OF TESTING.
THE PROGRAM JUMPS TO THE BEGINNING & THE ABOVE SEQUENCE IS REPEATED UNTIL HALTED.

2.2 SPECIAL ENVIRONMENTS.

THIS DIAGNOSTIC FOLLOWS THE STANDARD PROCEDURE FOR RUNNING UNDER APT, RT-11 MONITORS, AS DESCRIBED IN THEIR RESPECTIVE PROCEDURES MANUAL AND SYSMAC PACKAGE.

2.3 OPERATIONAL SWITCH SETTINGS

THIS PROGRAM SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (SWREG AT LOC. 176) FROM THE CONSOLE. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE CONSOLE TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: ' SWR=XXXXXX NEW=' (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE CONSOLE:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED THE LAST 6 WILL BE USED. IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 3.
 - C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE THEN A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRAGE RETURN AND A LINE FEED SEQUENCE THEN PROCEED FROM STEP 3 (ERASING ALL PREVIOUS INPUT).
- 4) THE DIAGNOSTIC WILL CONTINUE ON TYPING <CR>.

SOFTWARE SWITCH REGISTER OPTIONS (SWREG)

BIT 15 SET = 100000 = HALT ON ERROR
13 SET = 20000 = INHIBIT ERROR TYPEOUTS
12 SET = 10000 = ENABLE PERFORMANCE REPORTS
10 SET = 2000 = BELL ON ERROR

2.4 PROGRAM OPTIONS.

A DEVICE MAP (\$DEV M) AT LOCATION 1246 (ENTERED BY A CONTROL-G FOLLOWED BY CONTROL-C) IS EXAMINED BY THE PROGRAM TO DETERMINE THE METHOD OF TESTING THE CLUSTER TERMINALS & THE COMM PORT.

THE DEFAULT VALUE = 000017 (INTERNAL LOOPBACK FOR COMM PORT & CLUSTER TERMINALS) IT CAN BE CHANGED TO DO THE FOLLOWING:

\$DEV M (LOC 1246) ENTERED BY ^G^C

BIT 15 SET =	100000 =	DROP PRINTER TESTS
14 SET =	40000 =	DROP CLUSTER TERM #3 TESTS
13 SET =	20000 =	#2
12 SET =	10000 =	#1
11 SET =	4000 =	DROP ASYNC COMM TESTS
10 SET =	2000 =	DROP SYNC COMM TESTS
9 SET =	1000 =	DROP DRIVE 1 TESTS
8 SET =	400 =	DROP DRIVE 0 TESTS
7 SET =	200 =	DROP CLOCK TESTS
6 SET =	100 =	DROP EIS-FIS TESTS
5 SET =	40 =	DROP FILL & EMPTY BUFFER TESTS
4 SET =	20 =	RUN WITHOUT A CONSOLE PRESENT
BIT 3 SET =	10 =	INT. LOOPBACK (MAINT MODE) FOR COMM PORT (DEFAULT)
2 SET =	4 =	INT. LOOPBACK (MAINT MODE) FOR TERM #1 (DEFAULT)
1 SET =	2 =	INT. LOOPBACK (MAINT MODE) FOR #2 (DEFAULT)
0 SET =	1 =	INT. LOOPBACK (MAINT MODE) FOR #3 (DEFAULT)
BIT 3 CLR =		EXT. LOOPBACK FOR COMM PORT.
BIT 2 CLR =		EXT. LOOPBACK FOR TERM #1.
BIT 1 CLR =		EXT. LOOPBACK FOR TERM #2.
BIT 0 CLR =		EXT. LOOPBACK FOR TERM #3.

NOTES:

1. THE CONSOLE IS NOT TESTED.
2. BITS 15-4 SETTINGS WILL OVERRIDE BIT 3-0 SETTINGS.
3. THE PRINTER LOGIC WILL BE TESTED IF BIT 15 = 0 & THE DATA TERM READY IS ASSERTED ON THE CONNECTOR. (A PHYSICAL PRINTER IS NOT REQ'D)
4. A VT-50,52 MAY BE USED INSTEAD OF A PRINTER FOR A VISUAL CHECK.

2.5 EXECUTION TIMES.

ASSUMING ALL DEVICES & 2 DISK SUBSYSTEMS PRESENT, PERFORMANCE REPORTS NOT ENABLED:

1'ST PASS: APPROX. 1 MIN. 45 SEC.

SUBSEQUENT PASSES: APPROX. 10 MIN. 30 SEC.

2.6 POWER FAIL

THERE IS NO POWER FAIL AUTO-RESTART CAPABILITY IN THE PDT-11.

2.7 COMPATABILITY TESTING

PASS 1: ENTERED FROM A '260G'.
WILL WRITE ALL TRACKS & SECTORS ON SELECTED DRIVES
& HALT AFTER AN OPERATOR PROMPT. TYPING 'P' WILL BEGIN PASS 2.

PASS 2: ENTERED BY A 'P' AFTER THE ABOVE HALT, OR A '270G'.
WILL PERFORM SEQUENTIAL & RANDOM READS ONLY, ON THE SELECTED DRIVES.

2.8 COPY UTILITY

TO PROVIDE BACKUP DISKS, A UTILITY IS PROVIDED IN THE PROGRAM TO COPY THE SYSTEM EXERCISER DISK FROM DX0 TO DX1 IN THE FOLLOWING WAY:

1. BOOT THE EXERCISER NORMALLY FROM DX0.
2. ALLOW THE PROGRAM TO HALT AFTER THE WARNING MESSAGE TO USE SCRATCH DISKS.
3. AT THIS POINT, DO A '250G' TO ENTER THE COPY UTILITY.
4. THE OPERATOR WILL BE PROMPTED TO USE A SCRATCH DISK IN DX1 & TYPE 'P' TO PROCEED.
5. WHEN COMPLETED, THERE WILL BE A VERIFYING MESSAGE.
THE OPERATOR CAN THEN DO A 'P' TO COPY ANOTHER IN THE SAME MANNER,
OR A '240G' TO BEGIN NORMAL TESTING. (OPERATOR WILL BE PROMPTED)
6. THE COPY UTILITY CAN BE ENTERED AT ANY TIME BY DOING A 'BREAK' & '250G'.

2.9 RUNNING WITHOUT CONSOLE.

SETTING BIT 4 ON THE DEVICE MAP (SEC. 2.4) ALLOWS THE OPERATOR TO RUN THE EXERCISER WITHOUT A CONSOLE ONCE THE PROGRAM IS UNDERWAY. THE END OF PASS COUNTER WILL INCREMENT NORMALLY WHILE THE END OF PASS & PERFORMANCE REPORT MESSAGES WILL BE SUPPRESSED WHEN THE CONSOLE IS NOT PRESENT. IF AN ERROR OCCURS, THE 2 USER LIGHTS WILL GO ON & THE PROCESSOR WILL HALT. THIS CONDITION TELLS THE OPERATOR THAT AN ERROR HAS OCCURED. THE OPERATOR THEN PLUGS A CONSOLE BACK IN & DOES A PROCEED, 'P'. THE ERROR MESSAGE WILL PRINT OUT & TESTING RESUMES AT WHICH TIME THE CONSOLE CAN BE UNPLUGGED AGAIN.

NOTE: THE CONSOLE MUST NOT BE REMOVED DURING PRINTOUT OF A MESSAGE. DOING SO WILL CAUSE THE UNIT TO HANG UNTIL THE CONSOLE IS RE-CONNECTED.

2.10 CHANGING THE DEFAULT BAUD RATES.

THE FOLLOWING LOCATIONS HOLD THE DEFAULT BAUD RATES USED BY THE PROGRAM:

DEVICE	LOCATION	DEFAULT
PRINTER	3520	9600 BAUD
SYNC MODEM	3522	9600 BAUD
ASYNCR MODEM	3524	9600 BAUD
CLUSTER TERM #1	3526	2400 BAUD
CLUSTER TERM #2	3530	2400 BAUD
CLUSTER TERM #3	3532	2400 BAUD

TO MODIFY THE BAUD RATES, LOAD THE FOLLOWING OCTAL NUMBERS INTO THE ABOVE LOCATIONS:

BAUD RATE	NUMBER	BAUD RATE	NUMBER
50	0	1800	4000
75	400	2000	4400
110	1000	2400	5000
134.5	1400	3600	5400
150	2000	4800	6000
300	2400	7200	6400
600	3000	9600	7000
1200	3400	19200	7400

3.0 ERROR INFORMATION.3.1 ERROR REPORTING PROCEDURE.

TYPICAL ERROR PRINTOUTS ARE SHOWN BELOW:

PASS # 2
SERIAL NO. = A123
TERM #3 DATA COMP ERR
ERROR # ERR PC EXPECT RECVD
000011 006516 000200 000100

PASS # 5
SERIAL NO. = A123
DX1 SOFT ERR - DATA COMP
DXTST # ERR PC RXCS RXES RXSA EXPECT RECVD # RETRIES
000003 010636 100337 000230 003405 100520 100120 4

PASS # 8
SERIAL NO. = A123
DX0 HARD ERR - AFTER WRITE CMD
ERROR # DXTST # ERR PC RXCS RXES TRACK SECTOR #RETRIES
000016 000001 011246 100337 000230 59 18 10

WHERE ALL VALUES TYPED ARE OCTAL EXCEPT :
#RETRIES, TRACK, & SECTOR WHICH ARE IN DECIMAL.

BITS 15, 13, & 10 OF THE SWITCH REGISTER (SWREG) CONTROL THE
SEQUENCE OF EVENTS AFTER AN ERROR IS CAUGHT.

BIT 15 SET: CAUSES THE PROGRAM TO HALT IN THE ERROR ROUTINE.
IF THE PROGRAM IS CONTINUED, IT WILL PROCEED
FROM WHERE IT HALTED.

BIT 13 SET: DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 SET: CAUSES THE BELL TO RING ON ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G <^G> FUNCTION.
REFER TO SECTION 2.3 FOR DETAILS.

NOTE: SINCE THERE ARE NO SPECIFIC TEST NUMBERS, APT WILL REPORT ALL ERRORS AS TEST 0 ERRORS.

3.2 ERROR HALTS.

THE ONLY HALT IN THE EXERCISER IS IN THE ERROR ROUTINE, AND
IS EXECUTED ONLY IF BIT 15 OF THE SWITCH REGISTER (SWREG) IS SET WHEN AN ERROR OCCURS.

4.0 PROGRESS & PERFORMANCE REPORTS

THE FOLLOWING REPORTS ARE ENABLED WITH BIT 12 SET IN THE SWITCH REGISTER (SWREG LOC 176) & WILL APPEAR FOLLOWING LOADING & STARTING AS DESCRIBED IN SECTION 2.1 (ASSUMING ALL DEVICES ARE TO BE TESTED, NO ERRORS & CONSOLE CONNECTED):

MEM TESTS DONE	(ALL MEMORY FROM LOCATION 0 TO THE BEGINNING OF THE RT-11/XXDP MONITOR (WHEN AVAIL) HAS BEEN TESTED.)
EIS-FIS TESTS DONE	(1000. NUMBER OF PASSES OF THE EIS-FIS INSTRUCTION EXERCISER HAS BEEN EXECUTED.)
CLOCK RUNNING	(100 INTERRUPTS HAVE BEEN DETECTED BEFORE EXERCISING THE NEXT DEVICE. THE CLOCK CONTINUES RUNNING IN INTERRUPT MODE.)
PRINTER RUNNING	(5 LINES HAVE BEEN PRINTED WITH ERROR BIT CHECKING ONLY. THE PRINTER RUNS CONTINUOUSLY IN INTERRUPT MODE AT 9600 BAUD. ITS ACTUAL PERFORMANCE IS A VISUAL CHECK IF PRINTER CONNECTED. NO OTHER CHECKING PERFORMED IF EXTERNAL LOOPBACK USED.)
SYNC COMM DONE	(CHARACTERS 0 THRU 377 HAVE BEEN TRANSMITTED & RECEIVED WITH ODD PARITY ENABLED. INTERRUPTS ARE THEN DISABLED & THE SYNC COMM IS NO LONGER EXERCISED. THIS IS SO THAT THE ASYNC COMM CAN BE EXERCISED FOR THE DURATION OF THE PASS.)
ASYNC COMM RUNNING	(CHARS 0 THRU 377 HAVE BEEN TRANSMITTED & RECEIVED BEFORE EXERCISING THE NEXT DEVICE. IT CONTINUES RUNNING IN INTERRUPT MODE AT 9600 BAUD WITH ODD PARITY ENABLED & REPEATS THE 0 THRU 377 CYCLE)
TERM #1 RUNNING	(CHARACTERS 0 THRU 377 HAVE BEEN TRANSMITTED & RECEIVED BEFORE EXERCISING THE NEXT DEVICE. THE TERMINAL KEEPS RUNNING CONTINUOUSLY AT 2400 BAUD IN INTERRUPT MODE & REPEATS THE 0 THRU 377 CYCLE)
TERM #2 RUNNING	(SAME AS #1)
TERM #3 RUNNING	(SAME AS #1)
DX0 TRK 0 DONE	(DRIVE 0, TRACK 0 IS WRITTEN WITH A DATA PATTERN, READ & DATA COMPARE PERFORMED)
DX1 TRK 0 DONE	(DRIVE 1, TRACK 0 IS WRITTEN WITH A DATA PATTERN, READ & DATA COMPARE PERFORMED)
DX0 TRK 1 DONE	(ETC)
DX1 TRK 1 DONE	(ETC UNTIL...)
DX0 DATA PATT DONE	(ALL TRACKS HAVE BEEN WRITTEN WITH A DATA PATTERN IN INTERRUPT MODE.)
DX1 DATA PATT DONE	(SAME AS DX0)
	(FOR 1'ST PASS TRACKS 0-4, 40-44; 72-76(10) DONE ONLY.)
DX0 RANDOM SEEKS DONE	(500 RANDOM SEEKS WITH READ & DATA COMPARE HAVE BEEN PERFORMED ON DRIVE 0)
DX1 RANDOM SEEKS DONE	(SAME AS DX0....10 RANDOM SEEKS ONLY FOR 1'ST PASS QUICK VERIFY)
END OF PASS #1	TOTAL ERRORS: 0
	TOTAL SOFT ERRORS: 0
SERIAL NO. = A123	TOTAL ERRORS THIS PASS: 0
	SOFT ERRORS THIS PASS: 0

(...& ENTIRE PROCESS REPEATS)

5.0 DEVICE REGISTERS.

PARAMETER REGISTER:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

PORT SELECT			BAUD RATE			USR LT2		USR LT1		PAR TYP		PAR EN		CHAR LENGTH		MAIN
*****													177420 (WR ONLY)			

LINE CLOCK:

VECTOR: 100

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

*****													177546		

CONSOLE: ADDR: 177560-177566 VECTOR: 60/64
 CLUSTER #1: 176500-176506 300/304
 #2: 176510-176516 310/314
 #3: 176520-176526 320/324

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

TKS								RCVR DONE		RCVR IE							

TKB													KB DATA BUFFER				

TPS								XMIT RDY		XMIT IE							

TPB													PRINTER DATA BUFFER (WRITE ONLY)				

PRINTER:

VECTOR: 200

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PRSR	ERR								PRIN	PRIN							177514
									RDY	IE							
PRB																	177516
																	PRINTER DATA BUFFER (WRITE ONLY)

ASYNCR MODEM:

VECTOR: 330/334

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RCSR	DSI	RING	CTS	CAR		SEC	DSET		RCVR	RCVR	DSET		SEC	RTS	DTR		176610
				DET		REC	RDY		DONE	IE	IE		XMIT				
RBUF	ERR	OR	FR	PAR													176612
		ERR	ERR	ERR													RECEIVER DATA BUFFER
XCSR									XMIT	XMIT						BRK	176614
									RDY	IE							
XBUF																	176616
																	TRANSMITTER DATA BUFFER (WRITE ONLY)

SYNC MODEM:

VECTOR: 340/344

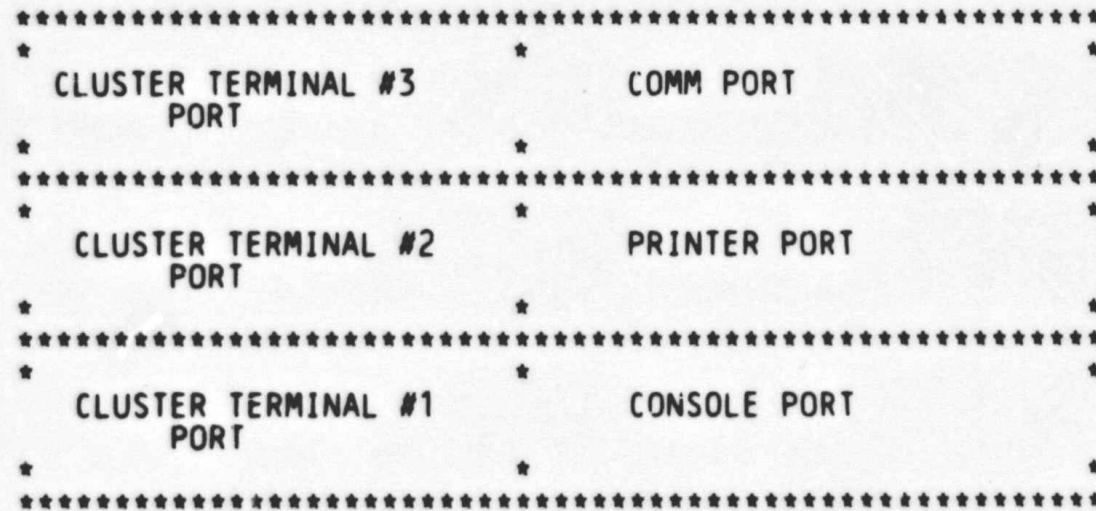
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RCSR	DSC	RING	CTS	CAR	REC	SEC	DSET	STRP	RCVR	RCVR	DSET	SRCH	SEC	RTS	DTR		176620
				DET	ACT	REC	RDY	SYNC	DONE	IE	IE	SYNC	XMIT				
RBUF	ERR	OR		PAR													176622 (RD ONLY)
		ERR		ERR													RECEIVER DATA BUFFER
PCSR		SYNC			WORD	PAR	EVEN										176622 (WR ONLY)
		CHAR			LENGTH	IE	PAR										SYNC REG
XCSR	DNA			MA	MA			MAST	XMIT	XMIT	DNA	SEND	H/F				176624
				MODE	CLK			RST	RDY	IE	IE		DUP				
XBUF																	176626
																	TRANSMITTER DATA BUFFER (WRITE ONLY)

DISK:

VECTOR: 264

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RXCS	ERR								CTL	DRV		UNIT		FUNCTION		GO	177170
									DONE	IE		SEL					
RXDB	DATA BUFFER															177172	
RXES									DRV		DEL	RNF	CRC	LOST	INV	ID	177172
									RDY		DATA			DATA	ADDR		
RXSA	TRACK 0 - 114(8)											SECTOR 1 - 32(8)					177174

6.0 DEVICE CONNECTOR LAYOUT



772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827

```
.TITLE CVKDAD          PDT11-150 EXERCISER
:*COPYRIGHT (C) 1979
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY GARY PAPAZIAN
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
:*
```

.SBTTL OPERATIONAL SWITCH SETTINGS

```

:
:      SWITCH          USE
:      -----          ---
:      15              HALT ON ERROR
:      13              INHIBIT ERROR TYPEOUTS
:      12              ENABLE PERFORMANCE REPORTS
:      10              BELL ON ERROR
:
:
```

.SBTTL BASIC DEFINITIONS

```
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
```

001100

```
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
```

*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776

177774
177772
177570
177570

```
HT= 11              ;;CODE FOR HORIZONTAL TAB
LF= 12              ;;CODE FOR LINE FEED
CR= 15              ;;CODE FOR CARRIAGE RETURN
CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776         ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774     ;;STACK LIMIT REGISTER
PIRQ= 177772       ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570       ;;HARDWARE SWITCH REGISTER
DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
```

*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

```
R0= %0              ;;GENERAL REGISTER
R1= %1              ;;GENERAL REGISTER
R2= %2              ;;GENERAL REGISTER
R3= %3              ;;GENERAL REGISTER
R4= %4              ;;GENERAL REGISTER
R5= %5              ;;GENERAL REGISTER
R6= %6              ;;GENERAL REGISTER
R7= %7              ;;GENERAL REGISTER
SP= %6              ;;STACK POINTER
PC= %7              ;;PROGRAM COUNTER
```

*PRIORITY LEVEL DEFINITIONS

828 000000
829 000040
830 000100
831 000140
832 000200
833 000240
834 000300
835 000340

PR0= 0 ;: PRIORITY LEVEL 0
PR1= 40 ;: PRIORITY LEVEL 1
PR2= 100 ;: PRIORITY LEVEL 2
PR3= 140 ;: PRIORITY LEVEL 3
PR4= 200 ;: PRIORITY LEVEL 4
PR5= 240 ;: PRIORITY LEVEL 5
PR6= 300 ;: PRIORITY LEVEL 6
PR7= 340 ;: PRIORITY LEVEL 7

836
837
838 100000
839 040000
840 020000
841 010000
842 004000
843 002000
844 001000
845 000400
846 000200
847 000100
848 000040
849 000020
850 000010
851 000004
852 000002
853 000001

;* "SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1

.EQUIV SW09,SW9
.EQUIV SW08,SW8
.EQUIV SW07,SW7
.EQUIV SW06,SW6
.EQUIV SW05,SW5
.EQUIV SW04,SW4
.EQUIV SW03,SW3
.EQUIV SW02,SW2
.EQUIV SW01,SW1
.EQUIV SW00,SW0

854
855
856
857
858
859
860
861
862
863
864
865
866 100000
867 040000
868 020000
869 010000
870 004000
871 002000
872 001000
873 000400
874 000200
875 000100
876 000040
877 000020
878 000010
879 000004
880 000002
881 000001

;* DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1

.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8

882
883

```
884 .EQUIV BIT07,BIT7
885 .EQUIV BIT06,BIT6
886 .EQUIV BIT05,BIT5
887 .EQUIV BIT04,BIT4
888 .EQUIV BIT03,BIT3
889 .EQUIV BIT02,BIT2
890 .EQUIV BIT01,BIT1
891 .EQUIV BIT00,BIT0
892
893 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
894 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
895 000010 PFSVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
896 000014 TBITVEC=14 ;: "T" BIT
897 000014 TRTVEC= 14 ;:TRACE TRAP
898 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
899 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
900 000024 PWRVEC= 24 ;:POWER FAIL
901 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
902 000034 TRAPVEC=34 ;: "TRAP" TRAP
903 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
904 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
905 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
906
907
908
909
```

```
910 ;:*****
911 ;* PDT-11/150 DEVICE VECTORS
912 ;:*****
913
914 000200 PRVEC= 200 ;:PRINTER
915
916 000100 CLKVEC= 100 ;:LINE CLOCK
917
918 000330 AMRVEC= 330 ;:ASYNC MODEM RECVR
919 000334 AMXVEC= 334 ; XMITR
920
921 000340 SMRVEC= 340 ;:SYNC MODEM RECVR
922 000344 SMXVEC= 344 ; XMITR
923
924 000264 RXVEC= 264 ;:DISK
925
926 000300 TK1VEC= 300 ;:CLUSTER TERM #1 VECTORS
927 000304 TP1VEC= 304
928 000310 TK2VEC= 310 ; #2
929 000314 TP2VEC= 314
930 000320 TK3VEC= 320 ; #3
931 000324 TP3VEC= 324
```

932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973

177420
177546
177514
177516
176610
176612
176614
176616
176620
176622
176622
176624
176626
177170
177172
177172
177174
176500
176502
176504
176506
176510
176512
176514
176516
176520
176522
176524
176526

```

:*****
:      PDT-11/150 DEVICE REGISTERS
:*****
PARAM= 177420      ;PARAMETER REGISTER      (WRITE ONLY)
CLK= 177546      ;LINE CLOCK
PRS= 177514      ;PRINTER STATUS REG
PRB= 177516      ;      BUFFER      (WRITE ONLY)
AMRC= 176610      ;ASYNC MODEM RECVR STATUS REG
AMRB= 176612      ;      BUFFER
AMXC= 176614      ;      XMITR STATUS REG
AMXB= 176616      ;      BUFFER      (WRITE ONLY)
SMRC= 176620      ;SYNC MODEM RECVR STATUS REG
SMRB= 176622      ;      BUFFER      (READ ONLY)
SMPAR= 176622      ;      PARAMETER REG (WRITE ONLY)
SMXC= 176624      ;      XMITR STATUS REG
SMXB= 176626      ;      BUFFER      (WRITE ONLY)
RXCS= 177170      ;DISK CSR
RXDB= 177172      ;      DATA BUFF
RXES= 177172      ;      ERROR & STATUS REG
RXSA= 177174      ;      TRK & SECTOR ADDR REG
TK1S= 176500      ;CLUSTER TERMINAL #1
TK1B= 176502
TP1S= 176504
TP1B= 176506
TK2S= 176510      ;      #2
TK2B= 176512
TP2S= 176514
TP2B= 176516
TK3S= 176520      ;      #3
TK3B= 176522
TP3S= 176524
TP3B= 176526

```

974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026

010000
000000
050000
060000
020000
030000
030000
040000

000000
000400
001000
001400
002000
002400
003000
003400
004000
0C4400
005000
005400
006000
006400
007000
007400

```
*****
*          PARAMETER REGISTER EQUATES          WRITE ONLY
*****

;BIT 14-12 PORT SELECT
:
CONSOL= 10000      ;SELECT CONSOLE TERMINAL
CT1= 0             ;CLUSTER TERMINAL #1
CT2= 50000        ;CLUSTER TERMINAL #2
CT3= 60000        ;CLUSTER TERMINAL #3
FRT= 20000        ;PRINTER PORT
AMOD= 30000       ;ASYNC COMM PORT
SMOD= 30000       ;SYNC COMM PORT
ULITE= 40000      ;USER LIGHTS

;BIT 11-8 BAUD RATE (DELTA = 400)
:
B50= 0             ;50 BAUD
B75= 400           ;75
B110= 1000        ;110
B134= 1400        ;134.5
B150= 2000        ;150
B300= 2400        ;300
B600= 3000        ;600
B1200= 3400       ;1200
B1800= 4000       ;1800
B2000= 4400       ;2000
B2400= 5000       ;2400 (CLUSTER DEFAULT)
B3600= 5400       ;3600
B4800= 6000       ;4800
B7200= 6400       ;7200
B9600= 7000       ;9600 (PRINTER, SYNC & ASYNC DEFAULT)
B19200= 7400      ;19200

;IF 110 BAUD SELECTED, 2 STOP BITS ARE ASSUMED.

;BITS 4-5 PARITY CONTROL
:
EPAR= 60          ;EVEN PARITY ENABLE
OPAR= 20          ;ODD PARITY ENABLE (DEFAULT)

;BITS 3-2 CHARACTER LENGTH (DELTA = 4)
:
CHAR5= 0          ;5 BITS/CHAR
CHAR6= 4          ;6
CHAR7= 10         ;7
CHAR8= 14         ;8 (PRINTER, TERMINAL, MODEM DEFAULT)

;BIT 1 MAINTENANCE BIT
:
MAINT= BIT1       ;ENABLE MAINT MODE
                ;PRINTER & DISK DO NOT HAVE MAINT MODE
```

1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059

100000
040000
020000
010000
002000
001000
000200
000100
000040
000010
000004
000002

100000
040000
020000
010000

000200
000100
000001

```
*****  
*          ASYNC MODEM BIT DEFINITIONS          *  
*****  
:RCSR  
:DSI= BIT15          :DATA SET INTERRUPT  
:RING= BIT14        :RING  
:CTS= BIT13         :CLEAR TO SEND  
:CDET= BIT12        :CARRIER DETECTED  
:SREC= BIT10        :SECONDARY RECD/SUPERVISORY RECD  
:DSRDY= BIT9        :DATA SET RDY  
:DONE= BIT7         :RECVR DONE  
:IE= BIT6           :RECVR IE  
:DSIE= BIT5         :DATA SET IE  
:SXMIT= BIT3        :SECONDARY XMIT/SUPERV XMIT  
:RTS= BIT2          :REQ TO SEND  
:DTR= BIT1          :DATA TERMINAL RDY  
  
:RBUF  
:ERR= BIT15         :ERROR  
:ORERR= BIT14       :OVERRUN ERROR  
:FRERR= BIT13       :FRAMING ERROR  
:PARERR= BIT12      :PARITY ERROR  
  
:XCSR  
:RDY= BIT7          :XMIT RDY  
:IE= BIT6           :XMIT IE  
:BREAK= BIT0        :BREAK
```

1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002

100000
040000
010000

000000
040000

000000
002000
004000
006000

001400
001000

```
*****  
* SYNC MODEM BIT DEFINITIONS  
*****  
:RCSR  
:DSC= BIT15 ;DATA SET CHANGE  
:RING= BIT14 ;RING  
:CTS= BIT13 ;CLEAR TO SEND  
:CDET= BIT12 ;CARRIER DETECTED  
:RACT= BIT11 ;RECVR ACTIVE (SYNC DETECT)  
:SREC= BIT10 ;SECONDARY RECD/SUPERVISORY RECD  
:DSRDY= BIT9 ;DATA SET RDY  
:STSYN= BIT8 ;STRIP SYNC  
:DONE= BIT7 ;RECVR DONE  
:IE= BIT6 ;RECVR IE  
:DSIE= BIT5 ;DATA SET IE  
:SRSYN= BIT4 ;SEARCH SYNC  
:SXMIT= BIT3 ;SECONDARY XMIT/SUPERV XMIT  
:RTS= BIT2 ;REQ TO SEND  
:DTR= BIT1 ;DATA TERMINAL RDY  
  
:RBUF  
:ERR= BIT15 ;ERROR  
:ORERR= BIT14 ;OVERRUN ERROR  
:PARERR= BIT12 ;PARITY ERROR  
  
:PARCSR PARAMETER CONT REG  
:BIT 14 SYNC CHAR  
:SYNC2= 0 ;2 SYNC CHARS (DEFAULT)  
:SYNC1= BIT14 ;1  
  
:BITS 10-11 WORD LENGTH  
:CHR5= 0 ;5 BITS/CHAR  
:CHR6= 2000 ;6  
:CHR7= 4000 ;7  
:CHR8= 6000 ;8 (DEFAULT)  
  
:BITS 8-9 PARITY CONTROL  
:ENVPAR= 1400 ;ENABLE EVEN PARITY  
:ODDPAR= 1000 ;ENABLE ODD PARITY (DEFAULT)
```


1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156

100000
010000
004000
000400
000200
000100
000040
000020
000010

;XCSR

;DNA= BIT15 ;DATA NOT AVAIL
;MM= BIT12 ;MAINT MODE
;MCLK= BIT11 ;MAINT CLOCK
;MR= BIT8 ;MASTER RESET
;RDY= BIT7 ;XMIT RDY
;IE= BIT6 ;XMIT IE
;DNAIE= BIT5 ;DATA NOT AVAIL IE
;SEND= BIT4 ;SEND
;FC DUP= BIT3 ;1=HALF DUPLEX, 0=FULL DUPLEX (DEFAULT = 0 FOR TESTS)

;DISK BIT DEFINITIONS

;RXCS

;ERR= BIT15 ;ERROR
;DONE= BIT7 ;CONTR RDY
;IE= BIT6 ;DRIVE IE
;DX1= BIT4 ;UNIT SELECT: 0=DX0, 1=DX1
;USEL= BIT4 ;UNIT SELECT

;FUNCTIONS (INCLUDES BIT0 = GO)

;FBUF= 1 ;FILL BUFFER
;EBUF= 3 ;EMPTY BUFFER
;WSEC= 5 ;WRITE SECTOR
;RSEC= 7 ;READ SECTOR
;INITAL= 11 ;INITIALIZE
;RSTAT= 13 ;READ STATUS
;WDDSEC= 15 ;WRITE DELETED DATA SECTOR
;RESTOR= 17 ;RESTORE TO TRACK 0

;RXES: ERROR & STATUS

;RDY= BIT7 ;DRIVE RDY
;DD= BIT5 ;DELETED DATA DET.
;RNF= BIT4 ;RECORD NOT FOUND
;CRC= BIT3 ;CRC ERROR
;LDATA= BIT2 ;LOST DATA
;INVADR= BIT1 ;INVALID ADDR
;ID= BIT0 ;INITIALIZE DONE

1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207

000174 000174
000174 000000
000176 000000
000100 000100
000100 000102
000102 000002
000200 000200
000200 000137 003574
000240 000240
000240 000137 003574
000250 000250
000250 000137 003556
000260 000260
000260 000137 003536
000270 000270
000270 000137 003544
001000
001000
000024 000200
000044 000044
000044 001000
001000
001000 000000
001002 001170
001004 000024
001006 000226
001010 000000
001012 000030

```
::*****  
;* SWITCH REGISTER  
*****  
DISPREG: .=174 .WORD 0 ;SOFTWARE DISPLAY REG  
SWREG: .WORD 0 ;SOFTWARE SWITCH REG  
.  
.=100  
102 ;SETUP CLOCK VECTOR AREA TO DO RTI  
2 ;IF ENABLED  
.  
.=200  
JMP START ;USE ONLY ONCE. WILL BE OVERLAID BY  
;PRINTER VECTOR ADDR.  
.  
.=240  
JMP START ;RESTART ADDR  
.  
.=250  
JMP START3 ;ENTER HERE FOR COPY UTILITY  
.  
.=260  
JMP START1 ;COMPATABILITY PASS 1  
.  
.=270  
JMP START2 ; PASS 2  
.  
.  
.=1000  
SBTTL APT PARAMETER BLOCK  
:  
*****  
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT  
:  
*****  
.$X= ;:SAVE CURRENT LOCATION  
.=24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM  
200 ;:FOR APT START UP  
.=44 ;:POINT TO APT INDIRECT ADDRESS PNTR.  
$APTHDR ;:POINT TO APT HEADER BLOCK  
.=.$X ;:RESET LOCATION COUNTER  
:  
*****  
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC  
:INTERFACE SPEC.  
$APTHD:  
$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
$MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)  
$STMT: .WORD 20. ;:RUN TIM OF LONGEST TEST  
$PASTM: .WORD 150. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
$UNITM: .WORD 0 ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
;SETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
```

1208
1209
1210
1211
1212
1213
1214 001100
1215 001100 001100
1216 001100 000000
1217 001102 000
1218 001103 000
1219 001104 000000
1220 001106 000000
1221 001110 000000
1222 001112 000000
1223 001114 000
1224 001115 001
1225 001116 000000
1226 001120 000000
1227 001122 000000
1228 001124 000000
1229 001126 000000
1230 001130 000000
1231 001132 000000
1232 001134 000
1233 001135 000
1234 001136 000000
1235 001140 177570
1236 001142 177570
1237 001144 177560
1238 001146 177562
1239 001150 177564
1240 001152 177566
1241 001154 000
1242 001155 002
1243 001156 012
1244 001157 000
1245 001160 177607 000377
1246 001164 077
1247 001165 015
1248 001166 000012
1249
1250
1251
1252
1253
1254 001170
1255 001170 000000
1256 001172 000000
1257 001174 000000
1258 001176 000000
1259 001200 000000
1260 001202 000000
1261 001204 000000
1262 001206 000000
1263 001210

.SBTTL COMMON TAGS

*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
*USED IN THE PROGRAM.
.=1100
\$CMTAG: .WORD 0 ;; START OF COMMON TAGS
\$TSTNM: .BYTE 0 ;; CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 ;; CONTAINS ERROR FLAG
\$ICNT: .WORD 0 ;; CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 ;; CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 ;; CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 ;; CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 ;; CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 ;; CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 ;; CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 ;; CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 ;; CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 ;; CONTAINS 'BAD' DATA
 .WORD 0 ;; RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 ;; AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 ;; INTERRUPT MODE INDICATOR
 .WORD 0
\$SWR: .WORD DSWR ;; ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD DDISP ;; ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 ;; TTY KBD STATUS
\$TKB: 177562 ;; TTY KBD BUFFER
\$TPS: 177564 ;; TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 ;; TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 ;; CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
\$TPFLG: .BYTE 0 ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
\$BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
\$QUES: .ASCII /?/ ;; QUESTION MARK
\$CRLF: .ASCII <15> ;; CARRIAGE RETURN
\$LF: .ASCIZ <12> ;; LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: ;; APT MAILBOX
\$MSGTY: .WORD AMSGTY ;; MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;; FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;; TEST NUMBER
\$PASS: .WORD APASS ;; PASS COUNT
\$DEVCT: .WORD ADEVCT ;; DEVICE COUNT
\$UNIT: .WORD AUNIT ;; I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;; MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ;; MESSAGE LENGTH
\$ETABLE: ;; APT ENVIRONMENT TABLE

```
1264 001210 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
1265 001211 000 $ENVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
1266 001212 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
1267 001214 000000 $USWR: .WORD AUSWR ::USER SWITCHES
1268 001216 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
1269 : * BITS 15-11=CPU TYPE
1270 : * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
1271 : * 11/70=06,PDQ=07,q=10
1272 : * BIT 10=REAL TIME CLOCK
1273 : * BIT 9=FLOATING POINT PROCESSOR
1274 : * BIT 8=MEMORY MANAGEMENT
1275 001220 000 $MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
1276 001221 000 $MTYP1: .BYTE AMTYP1 ::MEM. TYPE,BLK#1
1277 : * MEM.TYPE BYTE -- (HIGH BYTE)
1278 : * 900 NSEC CORE=001
1279 : * 300 NSEC BIPOLAR=002
1280 : * 500 NSEC MOS=003
1281 001222 000000 $MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
1282 : * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
1283 001224 000 $MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
1284 001225 000 $MTYP2: .BYTE AMTYP2 ::MEM. TYPE,BLK#2
1285 001226 000000 $MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
1286 001230 000 $MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
1287 001231 000 $MTYP3: .BYTE AMTYP3 ::MEM. TYPE,BLK#3
1288 001232 000000 $MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
1289 001234 000 $MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
1290 001235 000 $MTYP4: .BYTE AMTYP4 ::MEM. TYPE,BLK#4
1291 001236 000000 $MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
1292 001240 000300 $VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
1293 001242 000000 $VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
1294 001244 176500 $BASE: .WORD ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
1295 001246 000017 $DEV: .WORD ADEV ::DEVICE MAP
1296 001250
1297 .MEXIT
```

1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352

001250

001250 023324 026460 027070
001256 027364
001260 023324 026460 027070
001266 027364
001270 023343 026504 027100
001276 027364
001300 023324 026460 027070
001306 027364
001310 023343 026504 027100
001316 027364
001320 023365 026441 027062
001326 027364
001330 023431 026546 027114
001336 027364
001340 023454 026577 027124
001346 027364
001350 023502 026577 027136
001356 027364
001360 023530 026577 027150
001366 027364
001370 023556 026577 027162
001376 027364
001400 023607 026577 027162
001406 027364
001410 023637 026634 027174
001416 027364

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
;* DH ;;POINTS TO THE DATA HEADER
;* DT ;;POINTS TO THE DATA
;* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB:
;; GLOBAL DATA
;ERR 1 EM1,DH2,DT2,DF
;ERR 2 EM1,DH2,DT2,DF
;ERR 3 EM2,DH3,DT3,DF
;ERR 4 EM1,DH2,DT2,DF
;ERR 5 EM2,DH3,DT3,DF
;ERR 6 EM3,DH1,DT1,DF
;ERR 7 EM4,DH4,DT4,DF
;ERR 10 EM5,DH5,DT5,DF
;ERR 11 EM6,DH5,DT6,DF
;ERR 12 EM7,DH5,DT7,DF
;ERR 13 EM8,DH5,DT8,DF
;ERR 14 EM9,DH5,DT8,DF
;ERR 15 EM10,DH6,DT9,DF

1353					;ERR 16	
1354	001420	023664	026701	027212		EM11,DH7,DT10,DF1
1355	001426	027372				
1356					;ERR 17	
1357	001430	023716	026701	027212		EM12,DH7,DT10,DF1
1358	001436	027372				
1359					;ERR 20	
1360	001440	023750	026701	027212		EM13,DH7,DT10,DF1
1361	001446	027372				
1362					;ERR 21	
1363	001450	024001	026701	027212		EM14,DH7,DT10,DF1
1364	001456	027372				
1365					;ERR 22	
1366	001460	024032	026770	027234		EM15,DH8,DT11,DF2
1367	001466	027402				
1368					;ERR 23	
1369	001470	024066	026770	027234		EM16,DH8,DT11,DF2
1370	001476	027402				
1371					;ERR 24	
1372	001500	024112	026701	027256		EM17,DH7,DT12,DF1
1373	001506	027372				
1374					;ERR 25	
1375	001510	024144	026701	027256		EM18,DH7,DT12,DF1
1376	001516	027372				
1377					;ERR 26	
1378	001520	024176	026701	027256		EM19,DH7,DT12,DF1
1379	001526	027372				
1380					;ERR 27	
1381	001530	024227	026701	027256		EM20,DH7,DT12,DF1
1382	001536	027372				
1383					;ERR 30	
1384	001540	024260	026770	027300		EM21,DH8,DT13,DF2
1385	001546	027402				
1386					;ERR 31	
1387	001550	024314	026770	027300		EM22,DH8,DT13,DF2
1388	001556	027402				
1389					;ERR 32	
1390	001560	024340	026441	027062		EM23,DH1,DT1,DF
1391	001566	027364				
1392					;ERR 33	
1393	001570	024351	026546	027114		EM24,DH4,DT4,DF
1394	001576	027364				
1395					;ERR 34	
1396	001600	024366	026441	027062		EM25,DH1,DT1,DF
1397	001606	027364				
1398					;ERR 35	
1399	001610	024403	026441	027062		EM26,DH1,DT1,DF
1400	001616	027364				
1401					;ERR 36	
1402	001620	024420	026441	027062		EM27,DH1,DT1,DF
1403	001626	027364				

1404					:ERR 37	
1405	001630	024435	026441	027062		EM28,DH1,DT1,DF
1406	001636	027364				
1407					:ERR 40	
1408	001640	024454	026441	027062		EM29,DH1,DT1,DF
1409	001646	027364				
1410					:ERR 41	
1411	001650	024474	026634	027322		EM30,DH6,DT14,DF
1412	001656	027364				
1413					:ERR 42	
1414	001660	024505	026634	027322		EM31,DH6,DT14,DF
1415	001666	027364				
1416					:ERR 43	
1417	001670	024516	026441	027062		EM32,DH1,DT1,DF
1418	001676	027364				
1419					:ERR 44	
1420	001700	024561	026441	027062		EM33,DH1,DT1,DF
1421	001706	027364				
1422					:ERR 45	
1423	001710	024622	026441	027062		EM34,DH1,DT1,DF
1424	001716	027364				
1425					:ERR 46	
1426	001720	024474	026634	027322		EM30,DH6,DT14,DF
1427	001726	027364				
1428					:ERR 47	
1429	001730	024474	026634	027322		EM30,DH6,DT14,DF
1430	001736	027364				
1431					:ERR 50	
1432	001740	024474	026634	027322		EM30,DH6,DT14,DF
1433	001746	027364				
1434					:ERR 51	
1435	001750	024474	026634	027322		EM30,DH6,DT14,DF
1436	001756	027364				
1437					:ERR 52	
1438	001760	024661	026634	027174		EM35,DH6,DT9,DF
1439	001766	027364				
1440					:ERR 53	
1441	001770	024726	026770	027234		EM36,DH8,DT11,DF2
1442	001776	027402				
1443					:ERR 54	
1444	002000	024775	026634	027174		EM37,DH6,DT9,DF
1445	002006	027364				
1446					:ERR 55	
1447	002010	025021	026634	027174		EM38,DH6,DT9,DF
1448	002016	027364				
1449					:ERR 56	
1450	002020	024474	026634	027322		EM30,DH6,DT14,DF
1451	002026	027364				
1452					:ERR 57	
1453	002030	025067	026634	027174		EM39,DH6,DT9,DF
1454	002036	027364				

1455					:ERR 60	
1456	002040	024474	026634	027322		EM30,DH6,DT14,DF
1457	002046	027364				
1458					:ERR 61	
1459	002050	025117	026634	027174		EM40,DH6,DT9,DF
1460	002056	027364				
1461					:ERR 62	
1462	002060	025162	026634	027174		EM41,DH6,DT9,DF
1463	002066	027364				
1464					:ERR 63	
1465	002070	025216	026634	027174		EM42,DH6,DT9,DF
1466	002076	027364				
1467					:ERR 64	
1468	002100	025237	026701	027212		EM43,DH7,DT10,DF1
1469	002106	027372				
1470					:ERR 65	
1471	002110	025301	026701	027212		EM44,DH7,DT10,DF1
1472	002116	027372				
1473					:ERR 66	
1474	002120	025341	026701	027212		EM45,DH7,DT10,DF1
1475	002126	027372				
1476					:ERR 67	
1477	002130	025402	026701	027212		EM46,DH7,DT10,DF1
1478	002136	027372				
1479					:ERR 70	
1480	002140	025441	026701	027256		EM47,DH7,DT12,DF1
1481	002146	027372				
1482					:ERR 71	
1483	002150	025503	026701	027256		EM48,DH7,DT12,DF1
1484	002156	027372				
1485					:ERR 72	
1486	002160	025543	026701	027256		EM49,DH7,DT12,DF1
1487	002166	027372				
1488					:ERR 73	
1489	002170	025604	026701	027256		EM50,DH7,DT12,DF1
1490	002176	027372				
1491					:ERR 74	
1492	002200	025643	026634	027174		EM51,DH6,DT9,DF
1493	002206	027364				
1494					:ERR 75	
1495	002210	025667	026634	027322		EM52,DH6,DT14,DF
1496	002216	027364				
1497					:ERR 76	
1498	002220	025713	026577	027340		EM53,DH5,DT15,DF
1499	002226	027364				
1500					:ERR 77	
1501	002230	025713	026577	027352		EM53,DH5,DT16,DF
1502	002236	027364				

1503						
1504						; ERRORS FOR COPY UTILITY
1505						
1506						;ERR 100
1507	002240	023750	026634	027322		EM13,DH6,DT14,DF
1508	002246	027364				
1509						;ERR 101
1510	002250	024112	026634	027322		EM17,DH6,DT14,DF
1511	002256	027364				
1512						;ERR 102
1513	002260	025762	026634	027322		EM54,DH6,DT14,DF
1514	002266	027364				
1515						;ERR 103
1516	002270	024176	026634	027322		EM19,DH6,DT14,DF
1517	002276	027364				
1518						
1519						;BAD SECTOR/TRACK TABLE FULL ERRORS
1520						
1521						;ERR 104
1522	002300	025777	000000	000000		EM55,0,0,0
1523	002306	000000				
1524						;ERR 105
1525	002310	026034	000000	000000		EM56,0,0,0
1526	002316	000000				
1527						
1528						;CRC ERRORS
1529						
1530						;ERR 106
1531	002320	026071	026701	027212		EM57,DH7,DT10,DF1
1532	002326	027372				
1533						;ERR 107
1534	002330	026126	026770	027234		EM58,DH8,DT11,DF2
1535	002336	027402				
1536						;ERR 110
1537	002340	026167	000000	000000		EM59,0,0,0
1538	002346	000000				
1539						;ERR 111
1540	002350	026225	026701	027256		EM60,DH7,DT12,DF1
1541	002356	027372				
1542						;ERR 112
1543	002360	026262	026770	027300		EM61,DH8,DT13,DF2
1544	002366	027402				
1545						;ERR 113
1546	002370	026323	000000	000000		EM62,0,0,0
1547	002376	000000				
1548						;ERR 114
1549	002400	025667	026634	027322		EM52,DH6,DT14,DF
1550	002406	027364				
1551						;ERR 115
1552	002410	025667	026634	027322		EM52,DH6,DT14,DF
1553	002416	027364				
1554						;ERR 116
1555	002420	025667	026634	027322		EM52,DH6,DT14,DF
1556	002426	027364				
1557						;ERR 117
1558	002430	024474	026634	027322		EM30,DH6,DT14,DF

CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

MACY11 30A(1052) 01-DEC-80^{H 3} 09:47 PAGE 34
ERROR POINTER TABLE

SEQ 0033

1559	002436	027364			
1560					:ERR 120
1561	002440	024505	026634	027322	EM31,DH6,DT14,DF
1562	002446	027364			
1563					:ERR 121
1564	002450	024474	026634	027322	EM30,DH6,DT14,DF
1565	002456	027364			
1566					:ERR 122
1567	002460	024505	026634	027322	EM31,DH6,DT14,DF
1568	002466	027364			
1569					:ERR 123
1570	002470	024505	026634	027322	EM31,DH6,DT14,DF
1571	002476	027364			
1572					:ERR 124
1573	002500	025667	026634	027322	EM52,DH6,DT14,DF
1574	002506	027364			
1575					:ERR 125
1576	002510	025667	026634	027322	EM52,DH6,DT14,DF
1577	002516	027364			
1578					
1579					
1580					:EIS FIS ERRORS
1581					
1582					
1583					:ERR 126
1584	002520	026361	026441	027062	EM63,DH1,DT1,DF
1585	002526	027364			
1586					:ERR 127
1587	002530	026416	026441	027062	EM64,DH1,DT1,DF
1588	002536	027364			
1589					
1590					:NORMAL ERRORS
1591					
1592					:ERR 130
1593	002540	025216	026634	027174	EM42,DH6,DT9,DF
1594	002546	027364			
1595					:ERR 131
1596	002550	025667	026634	027322	EM52,DH6,DT14,DF
1597	002556	027364			
1598					

1599
1600
1601
1602
1603 002560 000000
1604 002562 000000
1605 002564 000000
1606 002566 000000
1607 002570 000000
1608 002572 000000
1609 002574 000000
1610 002576 000000
1611 002600 000000
1612 002602 000000
1613 002604 000100
1614 003004 000012
1615
1616
1617
1618
1619
1620
1621 003030 000000
1622 003032 000000
1623 003034 000000
1624 003036 000000
1625 003040 000000
1626 003042 000000
1627 003044 000000
1628 003046 000000
1629 003050 000000
1630 003052 000000
1631 003054 000100
1632 003254 000012

::*****
:REGISTERS & BUFFERS FOR DX0 TESTS
:*****

DOPAT: 0 ;DX0 PATT TEST DONE FLAG
DOCNT: 0 ;DX0 RANDOM SEEK COUNTER
DOERR: 0 ;DX0 RCSR ERR COUNTER
DOTRK: 0 ;DX0 TRACK
DOSEC: 0 ;DX0 SECTOR
DOCMER: 0 ;DX0 DATA COMPARE ERROR FLAG
DOCERR: 0 ;DX0 TOTAL DATA COMP ERROR CT
C?EXP: 0 ;DX0 EXPECTED DATA
DOREC: 0 ;DX0 RECEIVED DATA
DOCRC: 0 ;DX0 CRC ERROR FLAG
DOBUF: .BLKW 100 ;DX0 DATA BUFFER
DOBAD: .BLKW 10. ;DX0 BAD TRK/SEC TABLE

::*****
:REGISTERS & BUFFERS FOR DX1 TESTS
:*****

D1PAT: 0 ;DX1 PATT TEST DONE FLAG
D1CNT: 0 ;DX1 RANDOM SEEK COUNTER
D1ERR: 0 ;DX1 RCSR ERR COUNTER
D1TRK: 0 ;DX1 TRACK
D1SEC: 0 ;DX1 SECTOR
D1CMER: 0 ;DX1 DATA COMPARE ERROR FLAG
D1CERR: 0 ;DX1 TOTAL DATA COMPARE ERROR CT
D1EXP: 0 ;DX1 EXPECTED DATA
D1REC: 0 ;DX1 RECEIVED DATA
D1CRC: 0 ;DX1 CRC ERROR FLAG
D1BUF: .BLKW 100 ;DX1 DATA BUFFER
D1BAD: .BLKW 10. ;DX1 BAD TRK/SEC TABLE

1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677

003300 000077
003476 000000
003500 000000
003502 000000
003504 000000
003506 000000
003510 000000
003512 000000
003514 000000
003516 000000
003520 007000
003522 007000
003524 007000
003526 005000
003530 005000
003532 005000
003534 000000

```
*****
:PATTERN TABLE
:PATTERN IS "125252" FOR THE 1'ST PASS, RANDOM FOR 2'ND & SUBSEQUENT PASSES.
:PATTERN IS DETERMINED BY "SETPAT" ROUTINE.
*****
WCP: .BLKW 77
CYTST: 0
;0 = FILL & EMPTY BUFFER TEST
;1 = DATA PATT TEST
;2 = RANDOM SEEK TEST
;3 = INITIALIZE TEST
;4 = RESTORE TEST
;5 = WRITE DELETED DATA TEST
;6 = INVALID ADDRESS TEST

FIN: 0 ;COMMON FLG FOR DX0 & DX1 FOR TRK/SEC FINISHED
MAXTRK: 0 ;MAX TRACK #
;0 FOR 1'ST PASS, 114 FOR SUBSEQUENT PASSES. SET AT START.
;# OF RANDOM SEEKS TO BE PERFORMED.
;10 FOR 1'ST PASS, 500 FOR SUBSEQUENT PASSES. SET AT EOP.
MAXSK: 0 ;0 = NORM TEST 1 = COPY UTILITY ACTIVE
;1 = COMPAT TESTS, PASS 1
;1 = PASS 2

COPFLG: 0
COMP1: 0
COMP2: 0

SMFLG: 0 ;SYNC MODEM FLAG
SEKFLG: 0 ;SEEK ERROR FLAG

*****
:DEFAULT BAUD RATE REGISTERS
*****
PRBAUD: B9600 ;PRINTER DEFAULT
SMBAUD: B9600 ;SYNC MODEM DEFAULT
AMBAUD: B9600 ;ASYNC MODEM DEFAULT
C1BAUD: B2400 ;CLUSTER # 1 DEFAULT
C2BAUD: B2400 ;CLUSTER # 2 DEFAULT
C3BAUD: B2400 ;CLUSTER # 3 DEFAULT
HOLD: 0 ;TEMP HOLDING REG
```

```

1678 .SBTTL PROGRAM
1679
1680 003536 005237 003510 START1: INC COMP1 ;COMPAT PASS 1
1681 003542 000424 BR S1
1682
1683 003544 005237 003512 START2: INC COMP2 ;COMPAT PASS 2
1684 003550 005037 003510 CLR COMP1
1685 003554 000421 BR S2
1686
1687 003556 005237 003506 START3: INC COPFLG ;COPY UTILITY
1688 003562 005037 003510 CLR COMP1
1689 003566 005037 003512 CLR COMP2
1690 003572 000414 BR S3
1691
1692 ;NORMAL START AND RESTART
1693 003574 012737 000102 000100 START: MOV #CLKVEC+2,CLKVEC ;DO RTI IF CLOCK INTERR
1694 003602 012737 000002 000102 MOV #RTI,CLKVEC+2
1695
1696 003610 005037 003510 S1: CLR COMP1 ;NORMAL TESTING
1697 003614 005037 003512 S2: CLR COMP2
1698 003620 005037 003506 S3: CLR COPFLG
1699
1700 003624 012706 001100 S3: MOV #STACK,SP ;SETUP STACK POINTER
1701 003630 042777 000100 175306 BIC #100,@$TKS ;DISABLE ANY TTY INTERRUPTS
1702 ;TO RT-11 MONITOR
1703 ;DISABLE INTERRUPTS
1704 003636 106427 ;MTPS #PR4
1705 003640 000200 .WORD 106427
1706 003642 013737 000042 005054 MOV 42,HLD42 ;SAVE
1707 003650 123727 001210 000001 CMPB $ENV,#1 ;APT?
1708 003656 001402 BEQ 2$ ;BR IF YES
1709 003660 005037 000042 CLR 42 ;ELSE CLR SO WE CAN USE SOFTSWR
1710 003664
1711 2$:
1712 .SBTTL INITIALIZE THE COMMON TAGS
1713 003664 012706 001100 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1714 003670 005026 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1715 003672 022706 001140 CLR --(R6)+-- ;;CLEAR MEMORY LOCATION
1716 003676 001374 CMP #SWR,R6 ;;DONE?
1717 003700 012706 001100 BNE -6 ;;LOOP BACK IF NO
1718 MOV #STACK,SP ;;SETUP THE STACK POINTER
1719 003704 012737 031004 000030 ;;INITIALIZE A FEW VECTORS
1720 003712 012737 000340 000032 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1721 003720 012737 032114 000034 MOV #340,@#EMTVEC+2 ;;LEVEL 7
1722 003726 012737 000340 000036 MOV #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1723 MOV #340,@#TRAPVEC+2;LEVEL 7
1724 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1725 003734 013746 000004 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1726 003740 012737 003774 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1727 003746 012737 177570 001140 MOV #64$,@#ERRVEC ;;SET UP ERROR VECTOR
1728 003754 012737 177570 001142 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1729 003762 022777 177777 175150 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1730 003770 001012 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
1731 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1732 BR 65$ ;;AND THE HARDWARE SWR IS NOT = -1
1733 003772 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
1733 003774 012716 004002 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN

```

```

1734 004000 000002
1735 004002 012737 000176 001140 65$: RTI
1736 004010 012737 000174 001142 MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
1737 004016 012637 000004 66$: MOV #DISPREG,DISPLAY
MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
1738
1739 004022 005037 001176 CLR $PASS ;;CLEAR PASS COUNT
1740 004026 132737 000200 001211 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1741 004034 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
1742 004036 012737 001212 001140 MOV #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
1743 004044
1744
1745 .SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1746 004044 005227 177777 INC #-1 ;;FIRST TIME?
1747 004050 001046 BNE 68$ ;;BRANCH IF NO
1748 004052 104401 004120 TYPE ,69$ ;;TYPE ASCIZ STRING
1749 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1750 004056 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
1751 004062 001012 BNE 70$ ;;BRANCH IF YES
1752 004064 123727 001210 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
1753 004072 001406 BEQ 70$ ;;BRANCH IF YES
1754 004074 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
1755 004102 001005 BNE 71$ ;;BRANCH IF NO
1756 004104 104406 GTSWR ;;GET SOFT-SWR SETTINGS
1757 004106 000403 BR 71$
1758 004110 112737 000001 001134 70$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
1759 004116 71$:
1760 004116 000423 BR 68$ ;;GET OVER THE ASCIZ
1761 .:69$: .ASCIZ <CRLF>*CVKDAD PDT-11/150 SYSTEM EXERCISER*<CRLF>
1762 004166 68$:
1763
1764 004166 005227 177777 INC #-1 ;1'ST TIME?
1765 004172 001401 BEQ 3$ ;BR IF YES
1766 004174 104406 GTSWR
1767
1768 004176 012737 000004 003502 3$: MOV #4,MAXTRK ;TRK 0-4, 40-44, 72-76(10) FOR 1'ST PASS
1769 004204 012737 000012 003504 MOV #10.,MAXSK ;10 RANDOM SEEKS FOR 1'ST PASS
1770
1771 004212 005037 031432 CLR TERR ;TOTAL ERR CT
1772 004216 005037 031434 CLR TSERR ;TOTAL SOFT ERR CT FOR EOP TYPEOUT
1773
1774 004222 005737 003506 TST COPFLG ;COPY UTILITY?
1775 004226 001056 BNE LOOP ;BR IF YES
1776

```

CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

M 3
MACY11 30A(1052) 01-DEC-80 09:47 PAGE 39
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0038

```
1777
1778
1779
1780
1781 004230 123727 001210 000001      CMPB  $ENV,#1      :APT?
1782 004236 001413                    BEQ    4$          :BR IF YES
1783 004240 104412                    GTDEVM              :SHOW CURRENT DEVM & GET NEW
1784
1785
1786
1787
1788
1789 004242 105037 030473      CLRB  $TTYIN+7     :MAKE SURE NULL AT END OF INITIAL LOAD
1790 004246 104401 022614      TYPE  ,SERIAL      :SHOW CURRENT SERIAL #
1791 004252 104401 030464      TYPE  ,$TTYIN
1792
1793 004256 104401 030524      TYPE  ,$MNEW       :GET NEW
1794 004262 104411
1795 004264 005726      RDLIN
1796
1797
1798
1799
1800 004266 005737 003510      TST   COMP1        :COMPAT PASS 1?
1801 004272 001034                    BNE   LOOP         :BR IF YES
1802 004274 005737 003512      TST   COMP2        :COMPAT PASS 2?
1803 004300 001031                    BNE   LOOP         :BR IF YES
1804
1805 004302 052737 000006 176610      BIS   #<RTS!DTR>,AMRC
1806 004310 042737 000006 176610      BIC   #<RTS!DTR>,AMRC ;DTR MUST BE TOGGLED.
1807
1808
1809
1810
1811
1812 004316 004737 012106      JSR   PC,SIZE      :SEE WHO IS ON THE SYSTEM
1813 004322 104401 021746      TYPE  ,DEFBD       :SHOW ALL DEFAULT BAUD RATES
1814 004326 123727 001210 000001      CMPB  $ENV,#1      :ARE WE RUNNING UNDER APT?
1815 004334 001413                    BEQ    LOOP        :BR IF YES & DONT HALT
1816 004336 032737 001000 001246      BIT   #1000,$DEV   :DX0 THERE?
1817 004344 001404                    BEQ    1$          :BR IF YES
1818 004346 032737 000400 001246      BIT   #400,$DEV    :DX1 THERE?
1819 004354 001003                    BNE   LOOP        :BR IF NO
1820 004356 104401 022150      1$:  TYPE  ,WARNING  :INSERT SCRATCH DISKS
1821 004362 000000      HALT
1822
```

```
1823      ;:*****
1824      ;:      NOW WE CAN DO SOME SERIOUS TESTING
1825      ;:*****
1826
1827 004364 012706 001100      LOOP:  MOV    #1100,SP      ;LOOP HERE AFTER EOP & RESET STACK
1828
1829 004370 004737 013620      JSR    PC,TIMER1      ;TIMER TO ALLOW 'P' TO PRINT BEFORE RESET
1830 004374 000005      RESET      ;CLEAR ALL INTERRUPT ENABLES ONLY ON 1'ST PASS
1831 004376 042737 000100 177546 3$:  BIC    #IE,CLK      ;RESET SETS CLK IE. DONT ALLOW YET
1832
1833      ;MTPS    #PRO      ;ALLOW INTERRUPTS
1834 004404 106427      .WORD  106427
1835 004406 000000      PRO
1836
1837 004410 052737 000006 176610      BIS    #<RTS!DTR>,AMRC ;TO CONDITION ALL TERMINALS &
1838      ;COMM PORT TO OPERATE IN EXT LOOPBACK MODE.
1839      ;NO HARM IF LOOPBACK NOT USED.
1840 004416 004737 013620      JSR    PC,TIMER1      ;TIMER TO ALLOW PREV XFERS TO FINISH
1841
1842 004422 005037 031436      CLR    PERR      ;PASS ERR COUNT FOR EOP TYPEOUT
1843 004426 005037 031440      CLR    PSERR      ;PASS SOFT ERR COUNT FOR EOP TYPEOUT
1844
1845 004432 004737 013716      JSR    PC,CLRBAD      ;CLEAR BAD SECTOR/TRACK TABLES
1846 004436 004737 012644      JSR    PC,SETPAT      ;LOAD 'WCP' TABLE WITH PATTERN
1847
1848 004442 005737 003510      TST    COMP1      ;COMPAT PASS 1?
1849 004446 001003      BNE    1$      ;BR IF YES
1850 004450 005737 003512      TST    COMP2      ;COMPAT PASS 2?
1851 004454 001405      BEQ    2$      ;BR IF NO
1852
1853 004456 012737 000114 003502 1$:  MOV    #114,MAXTRK
1854 004464 000137 007750      JMP    DXTST1      ;COMPATABILITY TESTING
1855
1856 004470 005737 003506      2$:  TST    COPFLG      ;COPY UTILITY?
1857 004474 001402      BEQ    MEMTST      ;BR IF NO...NORMAL TESTING
1858 004476 000137 020714      JMP    COPY      ;ELSE GO COPY
1859
```


1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894

004502 004737 013104
004506 012737 014174 000004
004514 012737 000200 000006
004522 013703 012622
004526 162703 000000
004532 005004
004534 005037 014202
004540 005714
004542 005737 014202
004546 001403
004550 010437 005052
004554 104001
004556 005724
004560 020427 032276
004564 001363

```

.SBTTL PROGRAM
:*****
:      MEMORY TESTS
:
: 1. MEMORY FROM 0 TO THE LAST LOC OF PROGRAM (LSTAD) IS TESTED FOR TIMEOUT.
: 2. MEMORY FROM 'LSTAD' TO THE BOTTOM OF THE RT-11/XXDP MONITOR
:    IS TESTED BY A PASS OF A 125252 PATT FOLLOWED BY A PASS OF 052525 PATTERN.
: 3. MEMORY FROM 'LSTAD' TO THE BOTTOM OF THE RT-11/XXDP MONITOR
:    IS FIRST WRITTEN WITH ITS ADDRESS AS DATA THEN READ BACK & VERIFIED.
:*****
MEMTST: JSR      PC,EXAMKB
:
:      MOV      #INTSRV,ERRVEC ;SETUP TIMEOUT TRAP ADDR
:      MOV      #200,ERRVEC+2
:
:      MOV      MAXMEM,R3
:      SUB      #0,R3 ;TEST ONLY AS FAR AS XXDP MONITOR IF NOT RT-11
: ;WHEN XXDP AVAIL. SUBSTRACT 1.5K (10)
:
:*****
:TEST FOR TIMEOUT FROM 0 TO 'LSTAD'
:*****
1$:   CLR      R4 ;ADDR POINTER
:   CLR      INTFLG
:   TST      (R4) ;TEST LOCS 0 THRU END OF PGM.
:   TST      INTFLG ;TIMEOUT?
:   BEQ      2$ ;BR IF NO
:   MOV      R4,ADDR ;FOR ERR TYP
:   ERROR    1 ;TIMEOUT ON READ
:   TST      (R4)+ ;BUMP ADDR
:   CMP      R4,#LSTAD ;AT END?
:   BNE      1$ ;BR IF NO

```

```

1895
1896
1897
1898
1899 004566 005005          CLR      R5          ;PASS CTR
1900 004570 012737 125252 005050  MOV      #125252,PAT ;DATA PATT FOR 1'ST PASS
1901 004576 012704 032276          MOV      #LSTAD,R4   ;ADDR POINTER. R/W LOCS LSTAD THRU
1902                                     ;BOT OF RT11 OR BOT OF ABS LOADER
1903 004602 005037 014202          3$:     CLR      INTFLG
1904 004606 013714 005050          MOV      PAT,(R4)   ;WRITE
1905 004612 005737 014202          TST      INTFLG    ;TIMEOUT?
1906 004616 001403          BEQ      4$        ;BR IF NO
1907 004620 010437 005052          MOV      R4,ADDR   ;FOR ERR TYP
1908 004624 104002          ERROR   2         ;TIMEOUT ON WRITE
1909 004626 011437 005046          4$:     MOV      (R4),MEMHLD ;READ
1910 004632 023737 005046 005050  CMP      MEMHLD,PAT ;COMPARE
1911 004640 001403          BEQ      6$        ;FOR ERR TYP
1912 004642 010437 005052          MOV      R4,ADDR   ;COMPARE ERROR
1913 004646 104003          ERROR   3
1914
1915 004650 005724          6$:     TST      (R4)+   ;BUMP POINTER
1916 004652 004737 005014          JSR      PC,BOTMON ;SEE IF AT BOTTOM OF MONITOR
1917 004656 000751          BR       3$        ;RET HERE IF NO
1918
1919          TST      R5          ;ELSE HERE
1920 004662 001005          BNE     9$
1921 004664 005205          INC     R5
1922 004666 012737 052525 005050  MOV      #052525,PAT ;DATA PATT FOR 2'ND PASS
1923 004674 000740          BR       8$        ;REPEAT
1924
1925                                     ;*****
1926                                     ;ADDRESS TEST. WRITE EACH LOC. WITH ITS ADDR. AS DATA. THEN VERIFY.
1927                                     ;*****
1928
1929 004676 012704 032276          9$:     MOV      #LSTAD,R4 ;ADDR REG
1930
1931 004702 010414          10$:    MOV      R4,(R4)   ;LOAD ADDR AS DATA
1932 004704 005724          TST      (R4)+     ;BUMP ADDR
1933 004706 004737 005014          JSR      PC,BOTMON ;SEE IF AT BOT OF MON
1934 004712 000773          BR       10$      ;RET HERE IF NO
1935
1936                                     ;VERIFY PASS
1937 004714 012704 032276          11$:    MOV      #LSTAD,R4
1938 004720 020414          CMP      R4,(R4)   ;READ & CHECK
1939 004722 001407          BEQ      12$      ;BR IF OK
1940 004724 010437 005052          MOV      R4,ADDR
1941 004730 010437 005050          MOV      R4,PAT
1942 004734 011437 005046          MOV      (R4),MEMHLD
1943 004740 104005          ERROR   5         ;MISCOMPARE
1944
1945 004742 005724          12$:    TST      (R4)+   ;BUMP ADDR
1946 004744 004737 005014          JSR      PC,BOTMON ;SEE IF AT BOT OF MON
1947 004750 000763          BR       11$      ;RET HERE IF NO
1948
1949 004752 012737 000006 000004  MOV      #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
1950 004760 005037 000006          CLR      ERRVEC+2

```

```

1951
1952 004764 004737 012624          JSR   PC,CHKCON      ;CHECK IF CONSOLE PRESENT
1953 004770 000432                   BR    EISFIS         ;RET HERE IF NOT
1954 004772 032777 010000 174140  BIT   #BIT12,@SWR   ;SKIP TYPEOUT IF NOT SET
1955 005000 001426                   BEQ   EISFIS
1956 005002 104401 022561          TYPE ,MEMORY
1957 005006 104401 022574          TYPE ,DUN
1958 005012 000421                   BR    EISFIS
1959
1960
1961          ::*****
1962          ;ROUTINE TO DETERMINE IF AT BOTTOM OF RT-11 OR XXDP MONITOR
1963          ::*****
1964 005014 023727 005054 001000  BOTMON: CMP   HLD42,#1000  ;RT-11 IF LOC 42 = 1000
1965 005022 001403                   BEQ   1$             ;BR IF RT-11
1966 005024 020403                   CMP   R4,R3         ;ELSE AT BOT OF XXDP MON?
1967 005026 001006                   BNE   3$            ;BR IF NO
1968 005030 000403                   BR    2$
1969
1970 005032 020437 000054          1$:   CMP   R4,54    ;LOC 54 CONTAINS LOW ADDR OF RT-11 MON.
1971 005036 001002                   BNE   3$            ;BR IF NOT THERE
1972 005040 062716 000002          2$:   ADD   #2,(SP)  ;RET+2 IF AT END
1973 005044 000207          3$:   RTS   PC
1974
1975 005046 000000          MEMHLD: 0          ;PUT MEMORY READ HERE
1976 005050 000000          PAT:   0          ;PATT TO BE WRITTEN & READ
1977 005052 000000          ADDR:  0          ;ADDR UNDER TEST
1978 005054 000000          HLD42: 0          ;SAVE LOC 42

```

1979
1980
1981
1982 005056 012737 001750 005730
1983 005064 005737 012612
1984 005070 001404
1985 005072 032737 000100 001246
1986 005100 001402
1987 005102 000137 005760
1988
1989
1990
1991 005106 012702 125252
1992 005112 000257
1993 005114 000264
1994 005116 000270
1995 005120 072227 000001
1996
1997 005124 100403
1998 005126 001402
1999 005130 102001
2000 005132 103401
2001 005134 104126
2002 005136 022702 052524
2003 005142 001401
2004 005144 104127
2005
2006
2007
2008 005146 012701 052525
2009 005152 000257
2010 005154 072127 177777
2011 005160 022701 025252
2012 005164 001401
2013 005166 104127
2014
2015
2016
2017 005170 012700 000001
2018 005174 000241
2019 005176 072027 000020
2020 005202 103001
2021 005204 001401
2022 005206 104126
2023
2024
2025
2026 005210 012700 100000
2027 005214 000241
2028 005216 072027 177760
2029 005222 103401
2030 005224 104126
2031 005226 020027 177777
2032 005232 001401
2033 005234 104127

```

:*****
:      EIS/FIS TEST SECTION
:*****
EISFIS: MOV    #1000.,EISCNT ;LOAD EXECUTION LOOP COUNTER
        TST    EIPRES      ;TEST IF SIZER DETERMINES EIS-FIS PRESENT
        BEQ    1$          ;BR IF NOT PRESENT
        BIT    #BIT6,$DEVM ;TEST IF EIS/FIS IS ENABLED
        BEQ    ASH1        ;BR IF YES
1$:     JMP    NOEIS       ;BYPASS THIS SECTION - NOT SELECTED
:*****
:TEST ASH LEFT INSTRUCTION
:*****
ASH1:   MOV    #125252,R2   ;LOAD VALUE TO BE SHIFTED
        CCC                    ;CLEAR CONDITION CODES
        SEZ                    ;SET Z BIT
        SEN                    ;SET N BIT
        ASH    #1,R2       ;SHIFT LEFT 1, RESULT = 52524
        ;N=0, Z=0, V=1, C=1
        BMI    1$          ;IF N SET, REPORT ERROR
        BEQ    1$          ;IF Z SET, REPORT ERROR
        BVC    1$          ;IF V = 0, REPORT ERROR
        BCS    2$          ;IF C SET, GO ON WITH TEST
1$:     ERROR  126         ;INCORRECT CONDITION CODES FOR "ASH LEFT"
2$:     CMP    #52524,R2   ;RESULT CORRECT ?
        BEQ    ASH2        ;BR IF OK
        ERROR  127         ;INCORRECT DATA FOR "ASH LEFT"
:*****
:TEST ASH RIGHT INSTRUCTION
:*****
ASH2:   MOV    #52525,R1   ;LOAD VALUE TO BE SHIFTED
        CCC                    ;CLEAR CONDITION CODES
        ASH    #-1,R1      ;SHIFT RIGHT 1, RESULT = 25252
        CMP    #25252,R1   ;RESULT CORRECT ?
        BEQ    ASH3        ;BR IF OK
        ERROR  127         ;INCORRECT DATA FOR "ASH RIGHT"
:*****
:TEST ASH LEFT SHIFT 000001 BY 16 SETS C.
:*****
ASH3:   MOV    #1,R0      ;LOAD VALUE TO BE SHIFTED
        CLC                    ;CLEAR C BIT
        ASH    #16.,R0     ;SHIFT LEFT 16 TIMES
        BCC    1$          ;BR IF ERROR
        BEQ    ASH4        ;BR IF DATA CORRECT
1$:     ERROR  126         ;INCORRECT CONDITION CODES FOR "ASH LEFT"
:*****
:TEST ASH RIGHT SHIFT 100000 BY 16 SETS C.
:*****
ASH4:   MOV    #100000,R0 ;LOAD VALUE TO BE SHIFTED
        CLC                    ;CLEAR C BIT
        ASH    #-16.,R0    ;SHIFT RIGHT 16 TIMES
        BCS    1$          ;BR IF NO ERROR
        ERROR  126         ;INCORRECT CONDITION CODE FOR "ASH RIGHT"
1$:     CMP    R0,#177777  ;CHECK DATA
        BEQ    ASHC1       ;BR IF OK
        ERROR  127         ;INCORRECT DATA FOR "ASH RIGHT"

```

2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077

005236 005002
005240 012703 125252
005244 000257
005246 000264
005250 000261
005252 073227 000020
005256 100003
005260 001402
005262 102001
005264 103001
005266 104126
005270 022702 125252
005274 001401
005276 104127
005300 005703
005302 001401
005304 104127
005306 005003
005310 012702 125252
005314 000257
005316 000264
005320 000262
005322 000261
005324 073227 177760
005330 100002
005332 101401
005334 102001
005336 104126
005340 022702 177777
005344 001401
005346 104127
005350 022703 125252
005354 001401
005356 104127

:TEST ASHC LEFT INSTRUCTION

ASHC1: CLR R2 ;CLEAR MSW REGISTER
MOV #125252,R3 ;LOAD VALUE TO BE SHIFTED
CCC ;CLEAR CONDITION CODES
SEZ ;SET Z BIT
SEC ;SET C BIT
ASHC #16.,R2 ;SHIFT LEFT R3 INTO R2
;RESULT: N=1,Z=0,V=1,C=0
BPL 1\$;IF N = 0, REPORT ERROR
BEQ 1\$;IF Z SET, REPORT ERROR
BVC 1\$;IF V = 0, REPORT ERROR
BCC 2\$;IF C = 0, OK
1\$: ERROR 126 ;INCORRECT CONDITION CODE FOR ASHC LEFT
2\$: CMP #125252,R2 ;R2 OK
BEQ 3\$;BR IF CORRECT
ERROR 127 ;INCORRECT DATA FOR "ASHC LEFT"
3\$: TST R3 ;R3 OK
BEQ ASHC2 ;BR IF CORRECT
ERROR 127 ;INCORRECT DATA FOR "ASHC LEFT"

:TEST ASHC RIGHT INSTRUCTION

ASHC2: CLR R3 ;CLEAR MSW REGISTER
MOV #125252,R2 ;LOAD VALUE TO BE SHIFTED
CCC ;CLEAR CONDITION CODES
SEZ ;SET Z BIT
SEV ;SET V BIT
SEC ;SET C BIT
ASHC #-16.,R2 ;SHIFT RIGHT 16 PLACES
;RESULT: N=1,Z=V=C=0
BPL 1\$;IF N =0,REPORT ERROR
BLOS 1\$;IF C OR Z SET, REPORT ERROR
BVC 2\$;IF V = 0, GO ON WITH TEST
1\$: ERROR 126 ;INCORRECT CONDITION CODES FOR "ASHC RIGHT"
2\$: CMP #-1,R2 ;TEST R2
BEQ 3\$;BR IF CORRECT
ERROR 127 ;INCORRECT DATA FOR "ASHC RIGHT"
3\$: CMP #125252,R3 ;TEST R2 TO R3 SHIFT
BEQ ASHC3 ;BR IF CORRECT
ERROR 127 ;INCORRECT DATA FOR "ASHC RIGHT"

2078
2079
2080
2081 005360 005000
2082 005362 012701 000002
2083 005366 073027 000037
2084 005372 103001
2085 005374 001401
2086 005376 104126
2087
2088
2089
2090
2091
2092 005400 012700 100000
2093 005404 005001
2094 005406 073027 000040
2095 005412 103001
2096 005414 100401
2097 005416 104126
2098 005420 022701 177777
2099 005424 001003
2100 005426 022700 177777
2101 005432 001401
2102 005434 104127

```
*****  
:TEST ASHC LEFT SHIFT 00000,0002 BY 31 SETS C.  
*****  
ASHC3: CLR R0 ;CLEAR MSW REGISTER  
MOV #2,R1 ;LOAD VALUE TO BE SHIFTED  
ASHC #31.,R0 ;LEFT SHIFT 0002 BY 31.  
BCC 1$ ;BR IF C BIT CLEARED  
BEQ ASHC4 ;BR IF RESULT =0  
1$: ERROR 126 ;INCORRECT CONDITION CODE FOR "ASHC LEFT"  
  
*****  
:TEST ASHC RIGHT SHIFT 100000,00000 BY 32 SETS C.  
*****  
ASHC4: MOV #100000,R0 ;LOAD VALUE TO BE SHIFTED  
CLR R1 ;CLEAR LSW REGISTER  
ASHC #32.,R0 ;SHIFT RIGHT BY 32.  
BCC 1$ ;BR IF C BIT CLEARED  
BMI 2$ ;BR IF N BIT SET  
1$: ERROR 126 ;INCORRECT CONDITION CODE FOR "ASHC RIGHT"  
2$: CMP #-1,R1 ;CHECK LOW ORDER RESULT  
BNE 3$ ;BR IF IN ERROR  
CMP #-1,R0 ;CHECK HIGH ORDER RESULT  
BEQ MUL0 ;BR IF CORRECT  
3$: ERROR 127 ;INCORRECT DATA FOR "ASHC RIGHT"
```



```

2138
2139
2140
2141
2142
2143
2144
2145
2146 005542 012704 032276
2147 005546 012744 107070
2148 005552 012744 134343
2149 005556 012744 065432
2150 005562 012744 032107
2151 005566 012744 123456
2152 005572 012744 045670
2153 005576 012744 125252
2154 005602 012744 135252
2155 005606 012744 016161
2156 005612 012744 040616
2157 005616 000257
2158 005620 000264
2159 005622 000240
2160 005624 075014
2161 005626 075034
2162 005630 075024
2163 005632 075004
2164 005634 103403
2165 005636 102402
2166 005640 001401
2167 005642 100401
2168 005644 104126
2169 005646 012437 005724
2170 005652 012437 005726
2171 005656 020427 032276
2172 005662 001401
2173 005664 104127
2174 005666 022737 137201 005724
2175 005674 001401
2176 005676 104127
2177 005700 022737 115230 005726
2178 005706 001401
2179 005710 104127
2180 005712 005337 005730
2181 005716 001405
2182 005720 000137 005106
2183 005724 000000
2184 005726 000000
2185 005730 000000
2186
2187 005732 004737 012624
2188 005736 000410
2189 005740 032777 010000 173172
2190 005746 001404
2191 005750 104401 022542
2192 005754 104401 022574
2193 005760

```

```

*****
:TEST ALL FLOATING INSTRUCTIONS TOGETHER
:
:                                045670,123456
:    134343,107070 + 032107,065432 * -----
:                                (135252,125252 - 040616,016161)
:
:    THE RESULT = 137201,115230
*****
FIS:  MOV    #LSTAD,R4                ;LOAD R4 STACK POINTER
      MOV    #107070,-(R4)           ;LOAD DATA ONTO STACK
      MOV    #134343,-(R4)
      MOV    #065432,-(R4)
      MOV    #032107,-(R4)
      MOV    #123456,-(R4)
      MOV    #045670,-(R4)
      MOV    #125252,-(R4)
      MOV    #135252,-(R4)
      MOV    #016161,-(R4)
      MOV    #040616,-(R4)
      CCC
      SEZ                                ;SET Z BIT
      NOP
      FSUB   R4      ;135252,125252-040616,016161=140616,017434
      FDIV   R4      ;045670,123456/140616,017434=145246,047065
      FMUL   R4      ;032107,065432*145246,047065=137201,106137
      FADD   R4      ;134343,107070+137201,106137=137201,115230
      BCS    1$      ;BR IF C SET, REPORT ERROR
      BVS    1$      ;BR IF V SET, REPORT ERROR
      BEQ    1$      ;BR IF Z SET, REPORT ERROR
      BMI    2$      ;BR IF N SET, DO NEXT TEST
1$:  ERROR  126      ;INCORRECT CONDITION CODE AFTER 'FADD,FSUB,FMUL,FDIV'
2$:  MOV    (R4)+,ANS1 ;SAVE FIRST HALF OF THE ANSWER
      MOV    (R4)+,ANS2 ;SAVE 2ND HALF
      CMP    R4,#LSTAD ;CHECK R4 STACK POINTER
      BEQ    3$      ;BR IF CORRECT
      ERROR  127      ;INCORRECT STACK POINTER VALUE
3$:  CMP    #137201,ANS1 ;TEST 1ST HALF OF RESULT
      BEQ    4$      ;BR IF CORRECT
      ERROR  127      ;INCORRECT DATA FROM A 'FIS INSTRUCTION'
4$:  CMP    #115230,ANS2 ;TEST 2ND HALF OF RESULT
      BEQ    5$      ;BR IF CORRECT
      ERROR  127      ;INCORRECT DATA FROM A 'FIS INSTRUCTION'
5$:  DEC    EISCNT    ;FINISHED THE # OF EXECUTION LOOPS ?
      BEQ    EISDON  ;BR IF FINISHED
      JMP    ASH1    ;LOOP UNTIL DONE
      ANS1:  0
      ANS2:  0
      EISCNT: 0
      EISDON: JSR    PC,CHKCON ;CHECK IF CONSOLE PRESENT
              BR    NOEIS      ;RET HERE IF NOT
              BIT    #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
              BEQ    NOEIS      ;BR IF CLEARED
              TYPE   ,FISEIS    ;REPORT EIS/FIS TESTING COMPLETED
              TYPE   ,DUN       ;REPORT 'DONE'
NOEIS: ;TAG TO BYPASS THE EIS/FIS TEST SECTION

```


2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217

005760 032737 000200 001246
005766 001034
005770 012737 014174 000100
005776 012737 000200 000102
006004 005037 014202
006010 052737 000100 177546
006016 004537 013340
006022 014202
006024 000100
006026 104032
006030 000413
006032 004737 012624
006036 000410
006040 032777 010000 173072
006046 001404
006050 104401 021676
006054 104401 022467

```
*****  
: START LINE CLOCK  
:*****  
CLKTST: BIT      #BIT7,$DEV      ;DROP TEST?  
          BNE     PRTST          ;BR IF YES  
          MOV     #INTSRV,CLKVEC ;SETUP VECTOR AREA  
          MOV     #200,CLKVEC+2 ;LOCKOUT OTHER INTER.  
          CLR     INTFLG  
          BIS     #IE,CLK        ;SET CLOCK LOOSE!  
          JSR     R5,TIMER        ;SEE IF INTFLG GOES TO 100  
          INTFLG  
          100  
          ERROR  32              ;CLK NOT RESPONDING  
          BR     PRTST  
          JSR     PC,CHKCON       ;CHECK IF CONSOLE PRESENT  
          BR     PRTST           ;RET HERE IF NOT  
          BIT     #BIT12,@SWR     ;SKIP TYPEOUT IF NOT SET  
          BEQ     PRTST  
          TYPE   ,CLOCK  
          TYPE   ,RUN
```

```

2218
2219
2220          ::*****
2221          :          START PRINTER
2222          :*****
2223 006060 032737 100000 001246 PRTST: BIT    #BIT15,$DEV  ;DROP TEST?
2224 006066 001067          BNE    SMTST      ;BR IF YES
2225 006070 005737 012616      TST    PRPRES     ;PRINTER THERE?
2226 006074 001464          BEQ    SMTST      ;BR IF NO
2227
2228 006076 013737 003520 003534 2$:  MOV    PRBAUD,HOLD
2229 006104 052737 020014 003534      BIS    #<PRT!CHAR8>,HOLD
2230 006112 013737 003534 177420      MOV    HOLD,PARAM ;SETUP PARAM REG
2231
2232 006120 012737 014204 000200      MOV    #PRSRV,PRVEC ;ELSE SETUP PRINTER VECTOR
2233 006126 012737 000200 000202      MOV    #200,PRVEC+2 ;LOCKOUT INTERR
2234
2235 006134 005037 014332          CLR    LINCT      ;CLEAR LINE CTR
2236 006140 012737 000015 014330      MOV    #CR,PRCHR  ;CHAR TO BE PRINTED
2237                                     ;'DEL' (177) CAN BE USED TO CLR PRINTER BUFF
2238 006146 052737 000100 177514      BIS    #IE,PRS    ;SET PRINTER LOOSE!
2239                                     ;WILL PRINT CONTINUOUS 132 COLUMN LINES
2240                                     ;EA BEGINNING WITH ASCII 40 THRU 176
2241 006154 004537 013340          JSR    R5,TIMER   ;SEE IF LINCT GOES TO 5
2242 006160 014332          LINCT
2243 006162 000005          5
2244 006164 104033          ERROR 33
2245 006166 000427          BR     SMTST      ;PRINTER NOT RESPONDING
2246
2247 006170 005737 012620          TST    PRPORT     ;EXT LOOPBACK MODE?
2248 006174 001403          BEQ    3$
2249 006176 042737 000100 177514      BIC    #IE,PRS    ;ELSE SHUT DOWN PRINTER
2250
2251 006204 004737 012624          3$: JSR    PC,CHKCON  ;CHECK IF CONSOLE PRESENT
2252 006210 000416          BR    SMTST      ;RET HERE IF NOT
2253 006212 032777 010000 172720      BIT    #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
2254 006220 001412          BEQ    SMTST
2255 006222 104401 021544          TYPE  ,PRINT
2256 006226 005737 012620          TST    PRPORT     ;EXT LOOPBACK MODE?
2257 006232 001403          BEQ    4$
2258 006234 104401 021555          TYPE  ,PORT
2259 006240 000402          BR    SMTST
2260 006242 104401 022467          4$: TYPE  ,RUN

```

```

2261
2262
2263          ;:*****
2264          ;:          START SYNC COMM PORT
2265          ;:*****
2266 006246 032737 002000 001246 SMTST: BIT    #BIT10,$DEV0    ;DROP TEST?
2267 006254 001402          BEQ     8$          ;BR IF NO
2268 006256 000137 006562          JMP     AMTST          ;ELSE JUMP TEST
2269
2270 006262 123727 001210 000001 8$:   CMPB   $ENV,#1      ;UNDER APT?
2271 006270 001003          BNE     9$          ;BR IF NO
2272 006272 005737 001176          TST     $PASS        ;ELSE 1'ST PASS?
2273 006276 001131          BNE     AMTST        ;EXIT TEST IF NO
2274
2275 006300 005037 015024          9$:   CLR     LSTCHR      ;LAST CHAR FLAG FOR XMIT
2276 006304 005237 003514          INC     SMFLG        ;SET SYNC MODEM FLAG
2277 006310 013737 003522 003534      MOV     SMBAUD,HOLD
2278 006316 052737 030000 003534      BIS     #SMOD,HOLD
2279 006324 013737 003534 177420      MOV     HOLD,PARAM   ;SET PARAM REG
2280 006332 052737 000400 176624      BIS     #MR,SMXC     ;MASTER RESET
2281 006340 052737 004000 176624      BIS     #MCLK,SMXC   ;MAINT CLOCK
2282
2283 006346 032737 000010 001246      BIT     #BIT3,$DEV0  ;EXT LOOPBACK?
2284 006354 001403          BEQ     1$          ;BR IF YES
2285 006356 052737 014000 176624      BIS     #<MM!MCLK>,SMXC ;ELSE SET MAINT MODE FOR INT. LOOPBACK
2286 006364 012737 007026 176622      1$:   MOV     #<CHR8!ODDPAR!026>,SMPAR ;SETUP SYNC PARAMETER REGISTER
2287 006372 052737 000026 176620      BIS     #<RTS!DTR!SRSYN>,SMRC ;SET REQ TO SEND, DATA TERM RDY
2288                                     ;& SEARCH SYNC
2289 006400 052737 000020 176624      BIS     #SEND,SMXC   ;ENABLE XMIT SEND
2290 006406 012737 000026 176626      MOV     #026,SMXB    ;XMIT SYNC CHAR
2291
2292 006414 004537 013632          JSR     R5,TIMER2    ;WAIT FOR DONE
2293 006420 176620          SMRC
2294 006422 000200          DONE
2295 006424 104045          ERROR  45          ;NO DONE
2296 006426 000435          BR     5$          ;EXIT
2297
2298 006430 013737 176622 014734      MOV     SMRB,COMHLD  ;READ WORD
2299 006436 023727 014734 000026      CMP     COMHLD,#026  ;IS IT SYNC CHAR?
2300 006444 001402          BEQ     4$          ;BR IF YES
2301 006446 104006          ERROR  6           ;DID NOT REC SYNC CHAR
2302 006450 000424          BR     5$          ;EXIT
2303
2304 006452 004537 013314          4$:   JSR     R5,SETVEC   ;SETUP VECTOR AREA
2305 006456 000340          SMRVEC
2306 006460 015026          SMRSRV
2307 006462 014736          SMXSRV

```

CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

M 4
MACY11 30A(1052) 01-DEC-80 09:47 PAGE 52
PROGRAM

SEQ 0051

```
2308
2309 006464 012737 000027 014732      MOV    #027,COMCHR      ;1'ST CHAR TO BE XMITTED
2310 006472 052737 000100 176620      BIS    #IE,SMRC        ;SET SYNC MODEM LOOSE!
2311 006500 052737 000100 176624      BIS    #IE,SMXC
2312
2313 006506 004537 013340      JSR    R5,TIMER        ;SEE IF COMCHR GOES TO ALL 1'S (DONE)
2314 006512 014732      COMCHR
2315 006514 177777      177777
2316 006516 104037      ERROR  37              ;SYNC COMM NOT RESPONDING
2317 006520 000240      NOP
2318
2319 006522 004737 012624      5$: JSR    PC,CHKCON    ;CHECK IF CONSOLE PRESENT
2320 006526 000413      BR     2$              ;RET HERE IF NOT
2321 006530 012737 000400 176624      MOV    #MR,SMXC        ;MASTER RESET
2322 006536 032777 010000 172374      BIT    #BIT12,@SWR     ;DO TIMEOUT IF SET
2323 006544 001404      BEQ   2$              ;EXIT IF NOT
2324 006546 104401 021515      TYPE  ,SCOM
2325 006552 104401 022574      TYPE  ,DUN
2326 006556 005037 003514      2$: CLR    SMFLG      ;CLEAR FLAG
```

```
2327  
2328  
2329  
2330  
2331  
2332 006562 032737 004000 001246 AMTST: BIT #BIT11,$DEV  ;DROP TEST?  
2333 006570 001065 BNE CL1TST ;BR IF YES  
2334 006572 013737 003524 003534 MOV AMBAUD,HOLD  
2335  
2336 006600 032737 000010 001246 BIT #BIT3,$DEV ;EXT LOOPBACK?  
2337 006606 001404 BEQ 1$ ;BR IF YES  
2338 006610 052737 030036 003534 BIS #<AMOD!OPAR!CHAR8!MAINT>,HOLD ;INT LOOPBACK  
2339 006616 000403 BR 3$  
2340 006620 052737 030034 003534 1$: BIS #<AMOD!OPAR!CHAR8>,HOLD ;EXT LOOPBACK  
2341 006626 013737 003534 177420 3$: MOV HOLD,PARAM ;SET PARAM REG  
2342 006634 052737 000006 176610 BIS #<RTS!DTR>,AMRC ;SET REQ TO SEND & DATA TERM RDY  
2343  
2344 006642 004537 013314 JSR R5,SETVEC ;SETUP VECTOR AREA  
2345 006646 000330 AMRVEC  
2346 006650 014654 AMRSRV  
2347 006652 014636 AMXSRV  
2348  
2349 006654 005037 014732 CLR COMCHR ;CHAR TO BE XMITTED  
2350 006660 013737 176612 014734 MOV AMRB,COMHLD ;CLR OUT ANY PREVIOUS STORED WORD  
2351 006666 052737 000100 176610 BIS #IE,AMRC ;SET ASYNC MODEM LOOSE!  
2352 006674 052737 000100 176614 BIS #IE,AMXC  
2353  
2354 006702 004537 013340 JSR R5,TIMER ;SEE IF COMCHR GOES TO 377  
2355 006706 014732 COMCHR  
2356 006710 000377 377  
2357 006712 104040 ERROR 40 ;ASYNC COMM NOT RESPONDING  
2358 006714 000413 BR CL1TST  
2359  
2360 006716 004737 012624 JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT  
2361 006722 000410 BR CL1TST ;RET HERE IF NOT  
2362 006724 032777 010000 172206 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET  
2363 006732 001404 BEQ CL1TST  
2364 006734 104401 021530 TYPE ,ACOM  
2365 006740 104401 022467 TYPE ,RUN
```

```

2366
2367
2368
2369
2370
2371 006744 005737 012614          CL1TST: TST      CLPRES      ;OPTION PRESENT?
2372 006750 001002                    BNE      1$          ;BR IF YES
2373 006752 000137 007506          JMP      DXTSTO
2374
2375 006756 032737 004000 001246 1$:  BIT      #4000,$DEV      ;DROP ASYNC?
2376 006764 001406                    BEQ      4$          ;BR IF NO
2377 006766 042737 000006 176610  BIC      #<RTS!DTR> ,AMRC ;MUST BE TOGGLED
2378 006774 052737 000006 176610  BIS      #<RTS!DTR> ,AMRC ;PRIME IT
2379
2380 007002 032737 010000 001246 4$:  BIT      #BIT12,$DEV      ;DROP TEST?
2381 007010 001062                    BNE      CL2TST      ;BR IF YES
2382 007012 013737 003526 003534  MOV      C1BAUD,HOLD
2383
2384 007020 032737 000001 001246  BIT      #BIT0,$DEV      ;TERM #1 SET FOR EXT LOOPBACK?
2385 007026 001404                    BEQ      2$          ;BR IF YES
2386 007030 052737 000016 003534  BIS      #<CT1!CHAR8!MAINT>,HOLD ;INT LOOPBACK
2387 007036 000403                    BR       3$
2388 007040 052737 000014 003534 2$:  BIS      #<CT1!CHAR8>,HOLD ;EXT LOOPBACK
2389
2390 007046 013737 003534 177420 3$:  MOV      HOLD,PARAM    ;SET PARAM REG
2391 007054 004537 013314          JSR      R5,SETVEC    ;SETUP INTERR VECTORS
2392 007060 000300                    TK1VEC
2393 007062 014354                    TK1SRV
2394 007064 014336                    TP1SRV
2395
2396 007066 005037 014432          CLR      T1CHR        ;CHAR TO BE XMITTED
2397 007072 013737 176502 014434  MOV      TK1B,T1HLD    ;READ CHAR (CLEAR STALE DATA)
2398 007100 052737 000100 176500  BIS      #IE,TK1S      ;SET CLUSTER TERM #1 LOOSE!
2399 007106 052737 000100 176504  BIS      #IE,TP1S
2400
2401 007114 004537 013340          JSR      R5,TIMER     ;SEE IF T1CHR GOES TO 377
2402 007120 014432                    T1CHR
2403 007122 000377                    377
2404 007124 104034                    ERROR 34              ;CLUSTER TERM 1 NOT RESPONDING
2405 007126 000413                    BR      CL2TST
2406
2407 007130 004737 012624          JSR      PC,CHKCON    ;CHECK IF CONSOLE PRESENT
2408 007134 000410                    BR      CL2TST      ;RET HERE IF NOT
2409 007136 032777 010000 171774  BIT      #BIT12,@SWR   ;SKIP TYPEOUT IF NOT SET
2410 007144 001404                    BEQ      CL2TST
2411 007146 104401 021462          TYPE    ,CLT1
2412 007152 104401 022467          TYPE    ,RUN

```

```

2413
2414
2415
2416
2417
2418 007156 032737 020000 001246 CL2TST: BIT #BIT13,$DEV  ;DROP TEST?
2419 007164 001062 BNE CL3TST ;BR IF YES
2420 007166 013737 003530 003534 MOV C2BAUD,HOLD
2421
2422 007174 032737 000002 001246 BIT #BIT1,$DEV ;TERM #2 SET FOR EXT LOOPBACK?
2423 007202 001404 BEQ 1$ ;BR IF YES
2424 007204 052737 050016 003534 BIS #<CT2!CHAR8!MAINT>,HOLD ;INT LOOPBACK
2425 007212 000403 BR 2$
2426 007214 052737 050014 003534 1$: BIS #<CT2!CHAR8>,HOLD ;EXT LOOPBACK
2427
2428 007222 013737 003534 177420 2$: MOV HOLD,PARAM ;SET PARAM REG
2429 007230 004537 013314 JSR R5,SETVEC ;SETUP INTERR VECTORS
2430 007234 000310 TK2VEC
2431 007236 014454 TK2SRV
2432 007240 014436 TP2SRV
2433
2434 007242 005037 014532 CLR T2CHR ;CHAR TO BE XMITTED
2435 007246 013737 176512 014534 MOV TK2B,T2HLD ;READ CHAR (CLEAR STALE DATA)
2436 007254 052737 000100 176510 BIS #IE,TK2S ;SET CLUSTER TERM #2 LOOSE!
2437 007262 052737 000100 176514 BIS #IE,TP2S
2438
2439 007270 004537 013340 JSR R5,TIMER ;SEE IF T2CHR GOES TO 377
2440 007274 014532 T2CHR
2441 007276 000377 377
2442 007300 104035 ERROR 35 ;CLUSTER TERM 2 NOT RESPONDING
2443 007302 000413 BR CL3TST
2444
2445 007304 004737 012624 JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT
2446 007310 000410 BR CL3TST ;RET HERE IF NOT
2447 007312 032777 010000 171620 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
2448 007320 001404 BEQ CL3TST
2449 007322 104401 021473 TYPE ,CLT2
2450 007326 104401 022467 TYPE ,RUN

```

```
2451
2452
2453      ;:*****
2454      ;:          START CLUSTER TERMINAL #3
2455      ;:*****
2456 007332 032737 040000 001246 CL3TST: BIT    #BIT14,$DEV    ;DROP TEST?
2457 007340 001062          BNE    DXTSTO      ;BR IF YES
2458 007342 013737 003532 003534      MOV    C3BAUD,HOLD
2459
2460 007350 032737 000004 001246      BIT    #BIT2,$DEV    ;TERM #3 SET FOR EXT LOOPBACK?
2461 007356 001404          BEQ    1$          ;BR IF YES
2462 007360 052737 060016 003534      BIS    #<CT3!CHAR8!MAINT>,HOLD ;INT LOOPBACK
2463 007366 000403          BR     2$
2464 007370 052737 060014 003534 1$:  BIS    #<CT3!CHAR8>,HOLD    ;EXT LOOPBACK
2465
2466 007376 013737 003534 177420 2$:  MOV    HOLD,PARAM    ;SET PARAM REG
2467 007404 004537 013314          JSR    R5,SETVEC    ;SETUP INTERR VECTORS
2468 007410 000320          TK3VEC
2469 007412 014554          TK3SRV
2470 007414 014536          TP3SRV
2471
2472 007416 005037 014632          CLR    T3CHR        ;CHAR TO BE XMITTED
2473 007422 013737 176522 014634      MOV    TK3B,T3HLD  ;READ CHAR (CLEAR STALE DATA)
2474 007430 052737 000100 176520      BIS    #IE,TK3S    ;SET CLUSTER TERM #3 LOOSE!
2475 007436 052737 000100 176524      BIS    #IE,TP3S
2476
2477 007444 004537 013340          JSR    R5,TIMER    ;SEE IF T3CHR GOES TO 377
2478 007450 014632          T3CHR
2479 007452 000377          377
2480 007454 104036          ERROR 36          ;CLUSTER TERM 3 NOT RESPONDING
2481 007456 000413          BR     DXTSTO
2482
2483 007460 004737 012624          JSR    PC,CHKCON   ;CHECK IF CONSOLE PRESENT
2484 007464 000410          BR    DXTSTO      ;RET HERE IF NOT
2485 007466 032777 010000 171444      BIT    #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
2486 007474 001404          BEQ    DXTSTO
2487 007476 104401 021504          TYPE  ,CLT3
2488 007502 104401 022467          TYPE  ,RUN
```


2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499 007506 005037 003476
2500
2501 007512 032737 000040 001246
2502 007520 001113
2503
2504 007522 004537 013632
2505 007526 177170
2506 007530 000200
2507 007532 104075
2508 007534 000500
2509
2510 007536 012737 000011 177170
2511
2512 007544 004537 013632
2513 007550 177170
2514 007552 000200
2515 007554 104114
2516 007556 000467
2517
2518 007560 005737 177170
2519 007564 100001
2520 007566 104130
2521
2522
2523 007570 012700 000100
2524 007574 005001
2525 007576 012737 000001 177170
2526 007604 010137 177172
2527 007610 005101
2528 007612 005300
2529 007614 001373
2530
2531 007616 004537 013632
2532 007622 177170
2533 007624 000200
2534 007626 104115
2535 007630 000442

```

*****
:      FILL & EMPTY BUFFER TEST:  DRIVE 0
*****
:A FILL BUFFER COMMAND IS ISSUED WITH ALTERNATING WORDS OF ALL 0'S & ALL 1'S.
:AN EMPTY BUFFER COMMAND IS ISSUED TO FILL 'DOBUF'.
:UPON COMPLETION, DOBUF IS CHECKED FOR ALTERNATING WORDS OF ALL 0'S & ALL 1'S.
:ERRORS IN THIS TEST MAY INDICATE CABLE CROSS-TALK INTERFERENCE.
*****
DXTSTO: CLR      DXTST      ;0 = DXTSTO
        BIT      #BIT5,$DEV  ;DROP BUFFER TESTS?
        BNE      DXTST1     ;BR IF YES
        JSR      R5,TIMER2   ;WAIT FOR DISK DONE
                RXCS
                DONE
        ERROR    75          ;NO DONE
        BR       8$          ;EXIT
        MOV      #INITAL,RXCS ;DO AN INITIALIZE COMMAND
        JSR      R5,TIMER2   ;WAIT FOR DONE
                RXCS
                DONE
        ERROR    114         ;NO DONE
        BR       8$          ;EXIT
        TST      RXCS        ;ERROR?
        BPL      6$          ;BR IF NO
        ERROR    130         ;INIT CMD ERROR
        6$:      MOV      #100,R0 ;FILL BUFFER
                CLR      R1    ;WORD CT
                MOV      #FBUF,RXCS ;ISSUE FILL BUFF COMMAND
        1$:      MOV      R1,RXDB
                COM      R1    ;COMPLEMENT WORD
                DEC      R0    ;DONE ALL 100 WORDS?
                BNE      1$    ;BR IF NO
        JSR      R5,TIMER2   ;WAIT FOR DISK DONE
                RXCS
                DONE
        ERROR    115         ;NO DONE
        BR       8$          ;EXIT

```

```

2536
2537 007632 012700 000100      MOV      #100,R0          :WORD CT
2538 007636 012701 002604      MOV      #DOBUF,R1       :BUFFER ADDR
2539 007642 012737 000003 177170 2$:  MOV      #EBUF,RXCS      :ISSUE EMPTY BUFF COMMAND
2540 007650 013721 177172      MOV      RXDB,(R1)+      :STORE WORD
2541 007654 005300      DEC      R0              :DONE?
2542 007656 001374      BNE     2$              :BR IF NO
2543
2544 007660 004537 013632      JSR     R5,TIMER2        :WAIT FOR DISK DONE
2545 007664 177170      RXCS
2546 007666 000200      DONE
2547 007670 104116      ERROR   116            :NO DONE
2548 007672 000421      BR      8$              :EXIT
2549
2550
2551 007674 012701 002604      MOV      #DOBUF,R1       :CHECK IT
2552 007700 012137 007742 3$:  MOV      (R1)+,EBHLD      :BUFFER ADDR
2553 007704 001401      BEQ     4$              :STORE IT
2554 007706 104076      ERROR   76              :BR IF ZERO
2555
2556 007710 012137 007742 4$:  MOV      (R1)+,EBHLD      :WORD NOT ALL ZEROS
2557 007714 023727 007742 177777 5$:  CMP      EBHLD,#-1        :ALL 1'S?
2558 007722 001401      BEQ     5$              :BR IF YES
2559 007724 104077      ERROR   77              :WORD NOT ALL ONES
2560
2561 007726 020127 003004 5$:  CMP      R1,#DOBUF+200    :END OF DOBUFF?
2562 007732 001362      BNE     3$              :BR IF NO
2563 007734 000405      BR      DXTST1          :GO TO NXT TST
2564
2565 007736 000137 011706 8$:  JMP     EOP              :EXIT ALL DISK TESTS
2566
2567 007742 000000      EBHLD:  0                :STORAGE
2568 007744 000000      EXPO:   0                :EXPECT 0'S
2569 007746 177777      EXP1:  -1                :FOR ERROR REPORT
                                :EXPECT 1'S
                                :FOR ERROR REPORT

```

2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622

```
*****
: START DX0 & DX1 DATA PATTERN
:
: EACH SECTOR IS FILLED WITH ITS TRACK & SECTOR ADDRESS AS DATA.
: DX1 WILL ALWAYS HAVE ITS DATA BIT 15 SET TO DISTINGUISH IT FROM DX0.
: EACH TRACK IS READ & CHECKED BEFORE GOING TO THE NEXT TRACK.
: DATA IS WRITTEN ON ALTERNATE SECTORS TO AVOID LATENCY TIMES.
: IE: SECTORS 1,3,5...31 FOLLOWED BY 2,4,6...32.
: HARD ERRORS ARE THOSE ERRORS AFTER 10 MORE RETRIES.
: 1'ST PASS PERFORMED ON TRACKS 0-4, 40-44, 72-76(10)
: SUBSEQUENT PASSES ON ALL TRACKS
: *****
```

```
007750 012737 000001 003476 DXTST1: MOV #1,DXTST ;1 = DATA PATT TESTS
007756 005037 002566 CLR D0TRK ;STARTING TRACK
007762 005037 003036 CLR D1TRK
007766 005037 002560 12$: CLR DOPAT ;INIT DX0 FLAGS
007772 005037 002564 CLR DOERR
007776 005037 002574 CLR D0CERR
010002 012737 000001 002570 MOV #1,D0SEC
010010 005037 003030 CLR D1PAT ;INIT DX1 FLAGS
010014 005037 003034 CLR D1ERR
010020 005037 003044 CLR D1CERR
010024 012737 000001 003040 MOV #1,D1SEC
010032 032737 000400 001246 BIT #BIT8,$DEV M ;DROP DX0 TESTS?
010040 001031 BNE 2$ ;BR IF YES
010042 012737 015126 000264 MOV #RXSRV,RXVEC ;SETUP VECTOR AREA
010050 012737 000200 000266 MOV #200,RXVEC+2
010056 005737 003512 1$: TST COMP2 ;DOING COMPAT PASS 2?
010062 001404 BEQ 10$ ;BR IF NO
010064 012737 015632 015226 MOV #DORSEC,DODISP ;ELSE DO READS ONLY
010072 000403 BR 7$
010074 012737 015244 015226 10$: MOV #DOFBUF,DODISP ;DO NORMAL TESTING
010102 005037 003500 7$: CLR FIN ;DO DX1 WHEN SET
010106 052737 000100 177170 BIS #IE,RXCS ;LET INTERRUPTS LOOSE!
010114 004737 013424 JSR PC,TIMDO
010120 104041 ERROR 41 ;DX0 NOT GETTING TRACK DONE FLAG
010122 000240 NOP
010124 032737 001000 001246 2$: BIT #BIT9,$DEV M ;DROP DX1 TESTS?
010132 001031 BNE 4$ ;BR IF YES
010134 012737 015126 000264 MOV #RXSRV,RXVEC ;SETUP VECTOR AREA
010142 012737 000200 000266 MOV #200,RXVEC+2
```

```
2623
2624 010150 005737 003512      3$:   TST   COMP2      ;DOING COMPAT PASS 2?
2625 010154 001404              BEQ   11$         ;BR IF NO
2626 010156 012737 017470 015234  MOV   #D1RSEC,D1DISP ;ELSE DO READ ONLY
2627 010164 000403              BR    8$
2628
2629 010166 012737 017102 015234 11$:   MOV   #D1FBUF,D1DISP ;DO NORMAL TESTING
2630 010174 005037 003500      8$:   CLR   FIN         ;GO BACK TO DX0 WHEN SET
2631 010200 052737 000120 177170  BIS   #<IE!DX1>,RXCS ;LET INTERRUPTS LOOSE!
2632
2633 010206 004737 013524              JSR   PC,TIMD1
2634 010212 104042              ERROR 42         ;DX1 NOT GETTING TRACK DONE FLAG
2635 010214 000240              NOP
2636
2637 010216 032737 000400 001246 4$:   BIT   #BIT8,$DEVM ;DROP DX0?
2638 010224 001007              BNE   5$         ;BR IF YES
2639 010226 005737 002560              TST   DOPAT      ;ELSE IS DX0 ALL DONE?
2640 010232 001004              BNE   5$         ;BR IF YES
2641 010234 005737 013522              TST   DOTMO      ;TIMEOUT?
2642 010240 001706              BEQ   1$         ;BR IF NO
2643 010242 000717              BR    7$         ;ELSE CONT LAST COMMAND
2644
2645 010244 032737 001000 001246 5$:   BIT   #BIT9,$DEVM ;DROP DX1?
2646 010252 001007              BNE   6$         ;BR IF YES
2647 010254 005737 003030              TST   D1PAT      ;ELSE IS DX1 ALL DONE?
2648 010260 001004              BNE   6$         ;BR IF YES
2649 010262 005737 013616              TST   D1TMO      ;TIMEOUT?
2650 010266 001730              BEQ   3$         ;BR IF NO
2651 010270 000741              BR    8$         ;ELSE CONT LAST COMMAND
2652
2653 010272 005737 003510      6$:   TST   COMP1      ;DOING COMPAT PASS 1?
2654 010276 001402              BEQ   13$        ;BR IF NO
2655 010300 000137 011706              JMP   EOP        ;ELSE ALL DONE
2656
2657 010304 005737 001176      13$:  TST   $PASS      ;1'ST PASS?
2658 010310 001035              BNE   DXTST2     ;BR IF NO
2659
2660 010312 023727 003502 000004  CMP   MAXTRK,#4   ;JUST DID 5 OUTER TRACKS?
2661 010320 001012              BNE   14$        ;BR IF NO
2662 010322 012737 000050 002566  MOV   #40.,D0TRK ;ELSE SETUP FOR 5 MIDDLE TRKS
2663 010330 012737 000050 003036  MOV   #40.,D1TRK
2664 010336 012737 000054 003502  MOV   #44.,MAXTRK
2665 010344 000610              BR    12$
2666
2667 010346 023727 003502 000054 14$:  CMP   MAXTRK,#44. ;JUST DID 5 MIDDLE TRACKS?
2668 010354 001013              BNE   DXTST2     ;BR IF NO, MUST BE ALL DONE
2669 010356 012737 000110 002566  MOV   #72.,D0TRK ;ELSE SETUP FOR 5 INNER TRKS
2670 010364 012737 000110 003036  MOV   #72.,D1TRK
2671 010372 012737 000114 003502  MOV   #76.,MAXTRK
2672 010400 000137 007766              JMP   12$
2673
```

2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721

010404 012737 000002 003476
010412 005737 001176
010416 001003
010420 012737 000004 003502
010426 005037 002564
010432 005037 002574
010436 005037 002562
010442 005037 003034
010446 005037 003044
010452 005037 003032
010456 032737 000400 001246
010464 001041
010466 005037 013074
010472 013737 003502 013076
010500 004737 012754
010504 010137 002566
010510 005237 013074
010514 012737 000032 013076
010522 004737 012754
010526 010137 002570
010532 004537 014006
010536 000753
010540 012737 015632 015226
010546 005037 003500
010552 052737 000100 177170
010560 004737 013424
010564 104117
010566 000240

```
*****  
: START DX0 & DX1 RANDOM SEEKS  
:*****  
:RANDOM SEEKS ARE PERFORMED ON EACH DISK.  
:DATA IS READ & VERIFIED TO BE CORRECT FROM THE PREVIOUS TEST.  
:HARD ERRORS ARE THOSE ERRORS AFTER 10 MORE RETRIES.  
:1'ST PASS PERFORMS 20 RANDOM SEEKS ON 1'ST 5 TRACKS.  
:SUBSEQUENT PASSES PERFORMS 500 RANDOM SEEKS BETWEEN ALL TRACKS.  
:*****  
DXTST2: MOV #2,DXTST ;2 = RANDOM SEEK TESTS  
TST $PASS ;1'ST PASS?  
BNE 7$ ;BR IF NO  
MOV #4,MAXTRK ;ELSE JUST DO TRKS 0-4  
7$: CLR DOERR ;INIT DX0 FLAGS  
CLR DOCERR  
CLR DOCNT  
CLR D1ERR ;INIT DX1 FLAGS  
CLR D1CERR  
CLR D1CNT  
BIT #BIT8,$DEV0 ;DROP DX0 TESTS?  
BNE 2$ ;BR IF YES  
1$: CLR LOLIM  
MOV MAXTRK,HILIM  
JSR PC,RAND  
MOV R1,DOTRK ;GET RANDOM TRACK #  
INC LOLIM  
MOV #32,HILIM  
JSR PC,RAND  
MOV R1,DOSEC ;GET RANDOM SECTOR #  
JSR R5,CKOBAD ;SEE IF THIS TRK/SEC FLAGGED BAD  
BR 1$ ;BR IF RETURN HERE  
MOV #DORSEC,DODISP ;SET DISPATCH TABLE TO POINT TO  
;READ SECTOR HANDLER.  
CLR FIN ;DO DX1 WHEN SET  
BIS #IE,RXCS ;LET INTERRUPTS LOOSE!  
JSR PC,TIMDO  
ERROR 117 ;DX0 NOT GETTING DONE FLAG  
NOP
```

```
2722
2723 010570 032737 001000 001246 2$: BIT #BIT9,$DEV  ;DROP DX1 TESTS?
2724 010576 001041 BNE 4$ ;BR IF YES
2725
2726 010600 005037 013074 3$: CLR LOLIM
2727 010604 013737 003502 013076 MOV MAXTRK,HILIM
2728 010612 004737 012754 JSR PC,RAND
2729 010616 010137 003036 MOV R1,D1TRK ;GET RANDOM TRACK #
2730
2731 010622 005237 013074 INC LOLIM
2732 010626 012737 000032 013076 MOV #32,HILIM
2733 010634 004737 012754 JSR PC,RAND
2734 010640 010137 003040 MOV R1,D1SEC ;GET RANDOM SECTOR #
2735
2736 010644 004537 014120 JSR R5,CK1BAD ;SEE IF THIS TRK/SEC FLAGGED BAD
2737 010650 000753 BR 3$ ;BR IF RETURN HERE
2738
2739 010652 012737 017470 015234 MOV #D1RSEC,D1DISP ;SET DISPATCH TABLE TO POINT TO
2740 ;READ SECTOR HANDLER.
2741 010660 005037 003500 CLR FIN ;GO BACK TO DX0 WHEN SET
2742 010664 052737 000120 177170 BIS #<IE!DX1>,RXCS ;LET INTERRUPTS LOOSE!
2743
2744 010672 004737 013524 JSR PC,TIMD1
2745 010676 104120 ERROR 120 ;DX1 NOT GETTING DONE FLAG
2746 010700 000240 NOP
2747
2748 010702 032737 000400 001246 4$: BIT #BIT8,$DEV  ;DROP DX0?
2749 010710 001004 BNE 5$ ;BR IF YES
2750 010712 023737 002562 003504 CMP DOCNT,MAXSK ;DID ALL SEEKS?
2751 010720 001262 BNE 1$ ;BR IF NO
2752
2753 010722 032737 001000 001246 5$: BIT #BIT9,$DEV  ;DROP DX1?
2754 010730 001004 BNE 6$ ;BR IF YES
2755 010732 023737 003032 003504 CMP D1CNT,MAXSK ;DID ALL SEEKS?
2756 010740 001317 BNE 3$ ;BR IF NO
2757
2758 010742 005737 003512 6$: TST COMP2 ;DOING COMPAT PASS 2?
2759 010746 001402 BEQ DXTST3 ;BR IF NO
2760 010750 000137 011706 JMP EOP ;ELSE ALL DONE
2761
```

```

2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772 010754 032737 000400 001246 DXTST3: BIT #BIT8,$DEV  ;DROP DX0 TESTS?
2773 010762 001062 BNE DXTST4 ;BR IF YES
2774
2775 010764 012737 000003 003476 MOV #3,DXTST ;3 = INIT TEST
2776 010772 005037 002564 CLR DOERR ;FLAGS
2777 010776 005037 002574 CLR DOCERR
2778
2779 011002 012737 016736 015226 MOV #DOINIT,DODISP ;SET DISPATCH TABLE TO GO TO INIT HANDLER
2780
2781 011010 005037 003500 CLR FIN
2782 011014 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2783
2784 011022 004737 013424 JSR PC,TIMDO
2785 011026 104046 ERROR 46 ;DX0 HUNG ON INITIALIZE COMMAND
2786 011030 000437 BR DXTST4
2787
2788 011032 005737 177170 TST RXCS ;ERROR?
2789 011036 100001 BPL 1$ ;BR IF NO
2790 011040 104063 ERROR 63 ;INIT COMMAND ERROR
2791
2792 011042 032737 000001 177172 1$: BIT #ID,RXES ;INIT DONE BIT SET?
2793 011050 001001 BNE 2$ ;BR IF YES
2794 011052 104055 ERROR 55 ;INIT DONE NOT SET
2795
2796 011054 005037 002564 2$: CLR DOERR
2797 011060 005037 002574 CLR DOCERR
2798 011064 012737 000001 002566 MOV #1,DOTRK ;EXP TRK & SEC
2799 011072 012737 000001 002570 MOV #1,DOSEC
2800
2801 011100 012737 016136 015226 MOV #DOEBUF,DODISP ;SETUP DISPATCH TABLE TO
2802 ;EMPTY BUFFER & VERIFY DATA
2803 011106 005037 003500 CLR FIN
2804 011112 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2805
2806 011120 004737 013424 JSR PC,TIMDO
2807 011124 104056 ERROR 56 ;DX0 HUNG ON EMPTY BUFF COMMAND
2808 011126 000240 NOP

```

2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860

011130 032737 000400 001246
011136 001050
011140 012737 000004 003476
011146 012737 016762 015226
011154 005037 003500
011160 052737 000100 177170
011166 004737 013424
011172 104047
011174 000431
011176 005737 177170
011202 100001
011204 104054
011206 005037 002566
011212 012737 000001 002570
011220 005037 002564
011224 005037 002574
011230 005037 003500
011234 012737 015632 015226
011242 052737 000100 177170
011250 004737 013424
011254 104121
011256 000240
011260 032737 001000 001246
011266 001050
011270 012737 000004 003476
011276 012737 017000 015234
011304 005037 003500
011310 052737 000120 177170
011316 004737 013524
011322 104122
011324 000431
011326 005737 177170
011332 100001
011334 104074

```

:*****
:      RESTORE COMMAND TEST
:THIS TEST VERIFIES THAT THE RESTORE COMMAND CAN BE ISSUED
:TO DX0 & DX1 & THAT NO ERRORS RESULT FROM IT.
:*****
DXTST4: BIT    #BIT8,$DEV    ;DROP DX0 TESTS?
        BNE    2$           ;BR IF YES
        MOV    #4,DXTST     ;4 = RESTORE TEST
        MOV    #D0REST,D0DISP ;SETUP DISPATCH TABLE
        CLR    FIN
        BIS    #IE,RXCS
        JSR    PC,TIMD0
        ERROR  47           ;DX0 HUNG ON RESTORE COMMAND
        BR     2$
        TST    RXCS        ;ERROR?
        BPL    1$         ;BR IF NO
        ERROR  54         ;DX0 RESTORE COMMAND ERROR
1$:    CLR    D0TRK        ;CHK FOR RECORD NOT FOUND (RNF)
        MOV    #1,D0SEC    ;SET TRK/SEC. SHOULD NOT MOVE
                        ;LOGIC THINKS ITS ALREADY THERE.
        CLR    D0ERR
        CLR    D0CERR
        CLR    FIN
        MOV    #D0RSEC,D0DISP ;SETUP TO READ SECTOR
        BIS    #IE,RXCS    ;LET INTERR LOOSE
        JSR    PC,TIMD0
        ERROR  121        ;DX0 HUNG ON READ CMD
        NOP
2$:    BIT    #BIT9,$DEV    ;DROP DX1 TESTS?
        BNE    DXTST5     ;BR IF YES
        MOV    #4,DXTST     ;4 = RESTORE TEST
        MOV    #D1REST,D1DISP ;REPEAT FOR DX1
        CLR    FIN
        BIS    #<IE!DX1>,RXCS ;LET INTERR LOOSE
        JSR    PC,TIMD1
        ERROR  122        ;DX1 HUNG ON RESTORE CMD
        BR     DXTST5
        TST    RXCS        ;ERROR?
        BPL    3$         ;BR IF NO
        ERROR  74         ;DX1 RESTORE CMD ERROR
    
```



```

2861
2862 011336 005037 003036 3$: CLR D1TRK ;READ TO CHECK FOR RNF
2863 011342 012737 000001 003040 MOV #1,D1SEC
2864 011350 005037 003034 CLR D1ERR
2865 011354 005037 003044 CLR D1CERR
2866 011360 005037 003500 CLR FIN
2867 011364 012737 017470 015234 MOV #D1RSEC,D1DISP ;SETUP TO READ SECTOR
2868 011372 052737 000120 177170 BIS #<IE!DX1>,RXCS ;LET INTERR LOOSE
2869
2870 011400 004737 013524 JSR PC,TIMD1
2871 011404 104123 ERROR 123 ;DX1 HUNG ON READ CMD
2872 011406 000240 NOP
2873
2874
2875
2876
2877 :*****
2878 : WRITE SECTOR WITH DELETED DATA MARK TEST
2879 :
2880 : THIS TEST VERIFIES THAT THE WRITE DELETED DATA COMMAND CAN BE ISSUED
2881 : & THAT THE 'DELETED DATA' BIT 5 IS SET IN RXCS AFTER READY IS RECV'D.
2882 : HARD ERRORS ARE THOSE ERRORS AFTER 10 MORE RETRIES.
2883 :*****
2884 011410 032737 000400 001246 DXTST5: BIT #BIT8,$DEVN ;DROP DX0 TESTS?
2885 011416 001043 BNE DXTST6 ;BR IF YES
2886
2887 011420 012737 000005 003476 MOV #5,DXTST ;5 = WR DEL DATA TEST
2888 011426 005037 002564 CLR DOERR
2889 011432 005037 C02574 CLR DOCERR
2890
2891 011436 012737 015244 015226 MOV #DOFBUF,DODISP ;SETUP DISPATCH TABLE
2892
2893 011444 005037 003500 CLR FIN
2894 011450 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2895
2896 011456 004737 013424 JSR PC,TIMD0
2897 011462 104050 ERROR 50 ;DX0 HUNG ON WRITE DEL DATA CMD
2898 011464 000420 BR DXTST6
2899
2900 011466 012737 017016 015226 MOV #DOSTAT,DODISP ;SETUP DISPATCH TABLE
2901
2902 011474 005037 003500 CLR FIN
2903 011500 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2904
2905 011506 004737 013424 JSR PC,TIMD0
2906 011512 104051 ERROR 51 ;DX0 HUNG ON READ STATUS
2907 011514 000404 BR DXTST6
2908
2909 011516 005737 177170 TST RXCS ;ERROR?
2910 011522 100001 BPL .+4 ;BR IF NO
2911 011524 104057 ERROR 57 ;READ STATUS ERROR

```


2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010

011706 005237 001176
011712 005037 001200
011716 004737 012624
011722 000437
011724 104401 022633
011730 013746 001176
011734 104405

011736 104401 022650
011742 013746 031432
011746 104405

011750 104401 022677
011754 013746 031434
011760 104405

011762 104401 022614
011766 104401 030464

011772 104401 022733
011776 013746 031436
012002 104405

012004 104401 022767
012010 013746 031440
012014 104405
012016 104401 012102
012022 012737 000114 003502 3\$:
012030 005737 003510
012034 001006
012036 005737 003512
012042 001003
012044 012737 000764 003504

012052 005737 003510 2\$:
012056 001407
012060 104401 023216
012064 000000
012066 005037 003510
012072 005237 003512
012076 000137 004364 1\$:

012102 377 377 000 NULL:
012106 .377 .377 .EVEN

```
::*****  
: END OF PASS  
:*****  
EOP: INC $PASS ;PASS CTR  
CLR $DEVCT ;CLEAR SO DONT GET NEG NOS OR APT WILL BARF  
JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT  
BR 3$ ;RET HERE IF NOT  
TYPE ,ENDPAS  
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
  
TYPE ,MSG1  
MOV TERR,-(SP) ;;SAVE TERR FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
  
TYPE ,MSG2  
MOV TSERR,-(SP) ;;SAVE TSERR FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
  
TYPE ,SERIAL ;SHOW SERIAL #  
TYPE ,STTYIN  
  
TYPE ,MSG3  
MOV PERR,-(SP) ;;SAVE PERR FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
  
TYPE ,MSG4  
MOV PSERR,-(SP) ;;SAVE PSERR FOR TYPEOUT  
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN  
TYPE ,NULL ;NULL CHAR  
MOV #114,MAXTRK ;FOR ALL SUBSEQUENT PASSES  
TST COMP1  
BNE 2$  
TST COMP2  
BNE 2$  
MOV #500.,MAXSK ;SET ONLY IF NOT COMPATABILTY TESTS  
  
TST COMP1 ;COMPAT PASS 1?  
BEQ 1$ ;BR IF NO  
TYPE ,COMPAT ;DONE MSG  
HALT  
CLR COMP1  
INC COMP2  
JMP LOOP ;REPEAT  
  
.BYTE -1,-1,0 ;NULL CHAR STRING  
.EVEN
```

```

3011
3012
3013
3014
3015
3016 012106 012737 014174 000004 SIZE:  MOV #INTSRV,ERRVEC ;SETUP TIMEOUT TRAP ADDR
3017 012114 012737 000200 000006      MOV #200,ERRVEC+2 ;LOCKOUT INTERR
3018 012122 012737 014174 000010      MOV #INTSRV,RESVEC ;SETUP RESERVED INST TRAP ADDR
3019 012130 012737 000200 000012      MOV #200,RESVEC+2 ;LOCKOUT INTERR
3020
3021
3022
3023
3024 012136 005037 014202
3025 012142 012700 003776
3026 012146 005001
3027 012150 005710
3028 012152 005737 014202
3029 012156 001007
3030 012160 005201
3031 012162 020027 167776
3032 012166 001405
3033 012170 062700 004000
3034 012174 000765
3035
3036 012176 162700 004000 2$:  SUB #4000,R0 ;GO BACK TO LAST VALID 1K
3037 012202 010037 012622 3$:  MOV RO,MAXMEM ;SAVE
3038 012206 010146      MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
3039
3040 012210 104405      TYPDS ;TYPE MEM SIZE
3041 012212 104401 021725      TYPE ,MEM ;GO TYPE--DECIMAL ASCII WITH SIGN
3042
3043
3044
3045
3046 012216 005037 012612
3047 012222 005037 014202
3048 012226 072100
3049 012230 000240
3050 012232 005737 014202
3051 012236 001003
3052 012240 005237 012612
3053 012244 000404
3054 012246 104401 021705 60$:  TYPE ,EISTXT ;EIS-FIS TEXT
3055 012252 104401 022133      TYPE ,NOTPRES
3056 012256

```

:SYSTEM SIZER SUBROUTINE

:ROUTINE TO SIZE THE AMOUNT OF MEMORY

:ROUTINE TO CHECK IF "EIS-FIS" OPTION IS PRESENT

```
3057  
3058  
3059  
3060  
3061 012256 005000  
3062 012260 005037 014202  
3063 012264 012700 176500  
3064 012270 005710 4$:  
3065 012272 005737 014202  
3066 012276 001011  
3067 012300 020027 176520  
3068 012304 001403  
3069 012306 062700 000010  
3070 012312 000766  
3071  
3072 012314 005237 012614 5$:  
3073 012320 000406  
3074  
3075 012322 005037 012614 6$:  
3076 012326 104401 021442  
3077 012332 104401 022133  
3078  
3079 012336 012737 000006 000004 7$:  
3080 012344 005037 000006  
3081 012350 012737 000012 000010  
3082 012356 005037 000012  
3083
```

```
*****  
:ROUTINE TO CHECK IF CLUSTER PORT IS PRESENT  
*****  
CLR R0  
CLR INTFLG  
MOV #TK1S,R0 ;SETUP CLUSTER CSR ADDR  
4$: TST (R0) ;DARE IT TO TIMEOUT  
TST INTFLG ;DID IT?  
BNE 6$ ;BR IF YES  
CMP R0,#TK3S ;DID ALL 3?  
BEQ 5$ ;BR IF YES  
ADD #10,R0 ;ELSE DO NEXT  
BR 4$  
5$: INC CLPRES ;OPTION PRESENT  
BR 7$ ;ALL 3 MUST RESPOND  
6$: CLR CLPRES ;OPTION NOT PRESENT  
TYPE ,CLOPT  
TYPE ,NOTPRES  
7$: MOV #ERRVEC+2,ERRVEC ;RESET TIME-OUT VECTOR  
CLR ERRVEC+2  
MOV #RESVEC+2,RESVEC ;RESET RESERVED INST VECTOR  
CLR RESVEC+2
```

```
3084  
3085  
3086  
3087  
3088 012362 005000  
3089 012364 012701 000040  
3090  
3091 012370 030137 001246 8$: BIT R1,$DEV ;DEVICE DROPPED?  
3092 012374 001006 BNE 10$ ;BR IF YES  
3093 012376 000241 9$: CLC  
3094 012400 006301 ASL R1  
3095 012402 103426 BCS 13$ ;BR IF ALL BITS TESTED  
3096 012404 062700 000002 ADD #2,R0 ;ELSE BUMP INDEX  
3097 012410 000767 BR 8$  
3098  
3099 012412 016037 012432 012422 10$: MOV 12$(R0),11$  
3100 012420 104401 TYPE  
3101 012422 000000 11$: .WORD 0 ;DEVICE TO BE DROPPED  
3102 012424 104401 022446 TYPE ,DROP  
3103 012430 000762 BR 9$  
3104  
3105 012432 021604 12$: BUFTST  
3106 012434 021705 EISTXT  
3107 012436 021676 CLOCK  
3108 012440 021572 DRVO  
3109 012442 021577 DRV1  
3110 012444 021515 SCOM  
3111 012446 021530 ACOM  
3112 012450 021462 CLT1  
3113 012452 021473 CLT2  
3114 012454 021504 CLT3  
3115 012456 021544 PRINT
```

```

3116
3117
3118
3119
3120 012460 005037 012616      13$: CLR      PRPRES
3121 012464 005037 012620      CLR      PRPORT
3122 012470 005737 177514      TST      PRS          ;SEE IF PRINTER ERR
3123 012474 100035      BPL      16$          ;BR IF NO
3124
3125 012476 032737 000010 001246  BIT      #BIT3,$DEVM  ;COMM PORT IN EXT LOOPBACK?
3126 012504 001405      BEQ      15$          ;BR IF YES
3127
3128 012506 104401 021544      14$: TYPE      ,PRINT
3129 012512 104401 022133      TYPE      ,NOTPRES
3130 012516 000426      BR       17$
3131
3132 012520 013737 003524 003534  15$: MOV      AMBAUD,HOLD
3133 012526 052737 030034 003534  BIS      #<AMOD!OPAR!CHAR8>,HOLD
3134 012534 013737 003534 177420  MOV      HOLD,PARAM  ;SET PARAM REG
3135
3136 012542 042737 000006 176610  BIC      #<RTS!DTR>,AMRC ;MUST BE TOGGLED
3137 012550 052737 000006 176610  BIS      #<RTS!DTR>,AMRC
3138 012556 005737 177514      TST      PRS          ;NOW ANY ERRORS?
3139 012562 100751      BMI      14$          ;BR IF YES
3140 012564 005237 012620      INC      PRPORT
3141 012570 005237 012616      16$: INC      PRPRES  ;PRINTER PRESENT
3142
3143
3144
3145
3146
3147
3148
3149 012574 032737 000020 001246  17$: BIT      #BIT4,$DEVM
3150 012602 001402      BEQ      18$          ;BR IF CONSOLE IS TO BE USED
3151 012604 104401 021631      TYPE      ,NOCON     ;ELSE TELL OPERATOR
3152 012610 000207      18$: RTS      PC
3153
3154 012612 000000      EIPRES: 0           ;EIS-FIS PRESENT = NON-ZERO
3155 012614 000000      CLPRES: 0          ;CLUSTER PRESENT = NON-ZERO
3156 012616 000000      PRPRES: 0          ;PRINTER PRESENT = NON-ZERO
3157 012620 000000      PRPORT: 0          ;1 = PRINTER ENABLED THRU LOOPBACK
3158
3159 012622 000000      MAXMEM: 0          ;MAX MEMORY
3160

```

```

3161
3162
3163
3164
3165
3166
3167 012624 004737 013620
3168 012630 105777 166314
3169 012634 100002
3170 012636 062716 000002
3171 012642 000207
3172
3173
3174
3175
3176
3177
3178
3179 012644 012700 000077
3180 012650 012702 003300
3181
3182 012654 005737 003510
3183 012660 001007
3184 012662 005737 003512
3185 012666 001004
3186
3187 012670 032737 000001 001176
3188 012676 001006
3189
3190 012700 012701 125252
3191 012704 010122
3192 012706 005300
3193 012710 001375
3194 012712 000414
3195
3196 012714 005237 012752
3197 012720 005037 013074
3198 012724 012737 177777 013076
3199 012732 004737 012754
3200 012736 010122
3201 012740 005300
3202 012742 001373
3203
3204 012744 005037 012752
3205 012750 000207
3206
3207 012752 000000

```

```

*****
:ROUTINE TO SEE IF CONSOLE PRESENT
:RETURN IF NO
:RETURN + 2 IF YES
*****
CHKCON: JSR      PC,TIMER1      ;SPEND SOME TIME
          TSTB   @STPS          ;XMIT RDY?
          BPL    1$             ;BR IF NO (NO CONSOLE)
          ADD    #2,(SP)        ;BUMP RETURN
1$:      RTS      PC

*****
:ROUTINE TO FILL THE "WCP" TABLE WITH 125252 ON THE 1'ST PASS
:& RANDOM NUMBERS ON 2'ND & SUBSEQUENT PASSES.
*****
SETPAT: MOV     #77,R0          ;WORD COUNT
          MOV     #WCP,R2       ;TABLE ADDR
          TST     COMP1         ;COMPATABILITY TESTS DO WCP ONLY
          BNE    5$
          TST     COMP2
          BNE    5$
          BIT     #BIT0,$PASS   ;CHECK ODD-EVEN PASS
          BNE    2$             ;BR IF EVEN
          5$:    MOV     #125252,R1 ;ELSE LOAD WORST CASE PATT
          1$:    MOV     R1,(R2)+
          DEC     R0
          BNE    1$
          BR     4$
          2$:    INC     PATFLG
          CLR     LOLIM         ;SET LIMITS
          MOV     #-1,HILIM
          3$:    JSR     PC,RAND   ;GET RANDOM NUMBER, RET WITH IT IN R1
          MOV     R1,(R2)+
          DEC     R0
          BNE    3$
          4$:    CLR     PATFLG
          RTS     PC
PATFLG: .WORD 0                ;FLAG USED BY "RAND" ROUTINE

```


3208
3209
3210
3211
3212
3213 012754 010046
3214 012756 010246
3215 012760 013700 013102
3216 012764 013701 013100
3217 012770 012702 177771
3218 012774 006300
3219 012776 006101
3220 013000 005202
3221 013002 001374
3222 013004 063700 013102
3223 013010 005501
3224 013012 063701 013100
3225 013016 062700 001057
3226 013022 005501
3227 013024 062701 047401
3228 013030 010037 013102
3229 013034 010137 013100
3230
3231 013040 005737 012752
3232 013044 001002
3233
3234 013046 042701 177400
3235 013052 020137 013074
3236 013056 103740
3237 013060 020137 013076
3238 013064 101335
3239 013066 012602
3240 013070 012600
3241 013072 000207
3242
3243 013074 000000
3244 013076 000000
3245
3246 013100 176543
3247 013102 123456
3248
3249
3250 013104 123727 001210 000001 EXAMKB: CMPB \$ENV,#1
3251 013112 001404 BEQ 1\$
3252 013114 005737 003514 TST SMFLG
3253 013120 001001 BNE 1\$
3254 013122 104407 CKSWR
3255 013124 000207 1\$: RTS PC

```

*****
: RANDOM NUMBER GENERATOR. RETURN WITH NUMBER IN R1
*****
RAND:  MOV    R0,-(SP)      ;SAVE
        MOV    R2,-(SP)
3$:    MOV    LONUM,R0
        MOV    HINUM,R1
        MOV    #-7,R2
1$:    ASL    R0
        ROL    R1          ;ROTATE CARRY TO R1
        INC    R2          ;DONE?
        BNE    1$         ;BR IN NO
        ADD    LONUM,R0    ;ADD NUMBER TO MAKE X 129
        ADC    R1
        ADD    HINUM,R1    ;ADD NUMBER TO MAKE X 129
        ADD    #1057,R0    ;ADD LO CONSTANT
        ADC    R1
        ADD    #47401,R1   ;ADD HI CONSTANT
        MOV    R0,LONUM
        MOV    R1,HINUM

        TST    PATFLG     ;FILLING DATA TABLE?
        BNE    2$         ;BR IF YES
                               ;ELSE SCALE TO BE WITHIN LIMITS
                               ;SAVE LO BYTE ONLY
2$:    BIC    #177400,R1
        CMP    R1,LOLIM
        BLO    3$         ;BR IF N < LOLIM
        CMP    R1,HILIM
        BHI    3$         ;BR IF N > HILIM
        MOV    (SP)+,R2    ;RESTORE
        MOV    (SP)+,R0
        RTS    PC         ;ELSE EXIT WITH R1 = N

LOLIM:  0
HILIM:  0

HINUM:  .WORD 176543
LONUM:  .WORD 123456

EXAMKB: CMPB $ENV,#1
        BEQ  1$
        TST  SMFLG
        BNE  1$
1$:     CKSWR
        RTS  PC

```

```

3256
3257
3258
3259
3260
3261 013126 104401 023313
3262 013132 013746 001246
3263 013136 104402
3264 013140 104401 030524
3265
3266 013144 005046
3267 013146 005046
3268
3269 013150 105777 165770
3270 013154 100375
3271 013156 117746 165764
3272 013162 042716 177600
3273
3274 013166 021627 000015
3275 013172 001013
3276 013174 005766 000004
3277 013200 001403
3278 013202 016637 000002 001246
3279 013210 062706 000006
3280 013214 104401 001165
3281 013220 000002
3282
3283 013222 004737 027124
3284 013226 021627 000060
3285 013232 002420
3286 013234 021627 000067
3287 013240 003015
3288 013242 042726 000060
3289 013246 005766 000002
3290 013252 001403
3291 013254 006316
3292 013256 006316
3293 013260 006316
3294
3295 013262 005266 000002
3296 013266 056616 177776
3297 013272 000726
3298
3299 013274 021627 000025
3300 013300 001402
3301 013302 104401 001164
3302 013306 062706 000006
3303 013312 000705
3304

```

```

*****
:ROUTINE TO DISPLAY CURRENT DEVM & ALLOW OPERATOR TO INPUT NEW VALUE.
*****
GT$DEV: TYPE      ,DEVM      ;ELSE SHOW CURRENT
          MOV      $DEVM,-(SP) ;:SAVE $DEVM FOR TYPEOUT
          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
          TYPE     , $MNEW     ;:GET NEW

1$:      CLR      -(SP)      ;CLR CTR
          CLR      -(SP)      ;CLR NEW DEVM

2$:      TSTB    @ $TKS      ;CHAR THERE?
          BPL     2$          ;BR IF NO
          MOVB   @ $TKB,-(SP) ;ELSE GET CHAR
          BIC    #^C177,(SP) ;MAKE IT 7 BIT ASCII

3$:      CMP     (SP),#15     ;<CR>?
          BNE    7$          ;BR IF NO
          TST    4(SP)       ;ELSE IS IT 1'ST CHAR?
          BEQ    4$          ;BR IF YES
          MOV    2(SP), $DEVM ;ELSE SAVE NEW DEVM
          ADD    #6, SP      ;RESTORE STACK

4$:      ADD    #6, SP
5$:      TYPE    , $CRLF
6$:      RTI

7$:      JSR    PC, $TYPEC   ;ECHO CHAR
          CMP    (SP),#60    ;CHAR < 0?
          BLT    9$          ;BR IF YES
          CMP    (SP),#67    ;CHAR > 7?
          BGT    9$          ;BR IF YES
          BIC    #60,(SP)+   ;STRIP OFF ASCII
          TST    2(SP)       ;IS IT 1'ST CHAR?
          BEQ    8$          ;BR IF YES
          ASL    (SP)        ;ELSE SHIFT PRESENT CHAR
          ASL    (SP)        ;TO MAKE ROOM
          ASL    (SP)        ;FOR NEW ONE

8$:      INC    2(SP)        ;KEEP CHAR CT
          BIS    -2(SP),(SP) ;SET IN NEW CHAR
          BR     2$          ;GET NEW ONE

9$:      CMP    (SP),#25     ;CONTROL-U?
          BEQ    10$         ;BR IF YES
          TYPE    , $QUES    ;ELSE TYPE ?<CRLF>
          ADD    #6, SP      ;RESTORE STACK
          BR     GT$DEV      ;TRY AGAIN

10$:     ADD    #6, SP
          BR     GT$DEV

```

3305
3306
3307
3308
3309
3310
3311
3312
3313
3314 013314 010046
3315 013316 012500
3316 013320 012520
3317 013322 012720 000200
3318 013326 012520
3319 013330 012710 000200
3320 013334 012600
3321 013336 000205

```

:*****
:* ROUTINE TO SET 2 CONSECUTIVE INTERRUPT VECTORS
:* 3 ARGUMENTS ARE PASSED THRU R5:
:*   VECTOR ADDRESS
:*   INTERRUPT SERVICE ROUTINE ADDRESS FOR RECVR
:*   INTERRUPT SERVICE ROUTINE ADDRESS FOR XMITTER.
:* PRIORITY WILL BE SET TO DISABLE FURTHUR INTERR.
:*****

```

```

SETVEC: MOV    R0, -(SP)           ;SAVE
        MOV    (R5)+, R0         ;GET VECTOR ADDR
        MOV    (R5)+, (R0)+      ;PUT SERV ROUTINE ADDR THERE
        MOV    #200, (R0)+      ;DISABLE INTERR
        MOV    (R5)+, (R0)+      ;GET XMIT SERV ADDR NEXT
        MOV    #200, (R0)
        MOV    (SP)+, R0         ;RESTORE
        RTS    R5

```

3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332

```

:*****
:WATCHDOG TIMER TO PREVENT DEVICES FROM ENTERING AN ENDLESS WAIT LOOP
:RETURN IF TIMED OUT
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR
:*****

```

3333 013340 012537 013420
3334 013344 012537 013422
3335
3336 013350 012701 000006
3337 013354 012700 177777
3338 013360 027737 000034 013422
3339 013366 001411
3340 013370 004737 013104
3341 013374 005300
3342 013376 001370
3343 013400 005237 001200
3344 013404 005301
3345 013406 001362
3346 013410 000402
3347
3348 013412 062705 000004
3349 013416 000205

```

TIMER:  MOV    (R5)+, FLGHLD     ;GET FLAG -
        MOV    (R5)+, CTHLD     ;GET COUNT THAT FLAG SHOULD GO TO
        MOV    #6, R1
        MOV    #-1, R0
        CMP    @FLGHLD, CTHLD   ;RESPONDING?
        BEQ    2$              ;BR IF YES
        JSR    PC, EXAMKB
        DEC    R0               ;ELSE DEC COUNTER
        BNE    1$              ;BR IF NOT TIMED OUT
        INC    $DEVCT          ;FOR APT
        DEC    R1
        BNE    4$
        BR     .+6             ;ELSE TIMED OUT
        2$:  ADD    #4, R5       ;JUMP OVER ERROR ON RETURN
        RTS    R5

```

3350
3351 013420 000000
3352 013422 000000
3353

```

FLGHLD: 0 ;FLAG
CTHL D: 0 ;COUNT THAT FLAG MUST REACH

```

```

3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388

```

```

*****
:WATCHDOG TIMER TO PREVENT DX0 FROM ENTERING AN ENDLESS WAIT LOOP
:RETURN IF TIMED OUT
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR
*****

```

```

013424 005037 013522 TIMD0: CLR DOTMO
013430 012737 000002 013520 MOV #2,CTREG1
013436 012737 177777 013516 4$: MOV #-1,CTREG
013444 005737 003500 1$: TST FIN ;FINISHED?
013450 001015 BNE 2$ ;BR IF YES
013452 004737 013104 JSR PC,EXAMKB
013456 005337 013516 DEC CTREG ;ELSE DEC CTR
013462 001370 BNE 1$ ;BR IF NOT TIMED OUT
013464 005237 001200 INC $DEVCT ;FOR APT
013470 005337 013520 DEC CTREG1
013474 001360 BNE 4$
013476 005237 013522 INC DOTMO ;SET FLAG
013502 000402 BR .+6 ;TIMED OUT

013504 005237 001200 2$: INC $DEVCT ;FOR APT
013510 062716 000004 ADD #4,(SP) ;BUMP RET
013514 000207 RTS PC

CTREG: 0
CTREG1: 0
DOTMO: 0 ;1 = TIMEOUT OCCURED

```

```

3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408

```

```

*****
:WATCHDOG TIMER TO PREVENT DX1 FROM ENTERING AN ENDLESS WAIT LOOP
:RETURN IF TIMED OUT
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR
*****

```

```

013524 005037 013616 TIMD1: CLR D1TMO
013530 012737 000002 013520 MOV #2,CTREG1
013536 012737 177777 013516 4$: MOV #-1,CTREG
013544 005737 003500 1$: TST FIN ;FINISHED?
013550 001015 BNE 2$ ;BR IF YES
013552 004737 013104 JSR PC,EXAMKB
013556 005337 013516 DEC CTREG ;ELSE DEC CTR
013562 001370 BNE 1$ ;BR IF NOT TIMED OUT
013564 005237 001200 INC $DEVCT ;FOR APT
013570 005337 013520 DEC CTREG1
013574 001360 BNE 4$
013576 005237 013616 INC D1TMO ;SET FLAG
013602 000402 BR .+6 ;TIMED OUT

013604 005237 001200 2$: INC $DEVCT ;FOR APT
013610 062716 000004 ADD #4,(SP) ;BUMP RET
013614 000207 RTS PC

D1TMO: 0 ;1 = TIMEOUT OCCURED

```

3409
3410
3411
3412
3413 013620 012700 177777
3414 013624 005300
3415 013626 001376
3416 013630 000207
3417
3418
3419
3420
3421
3422
3423
3424 013632 012537 013712
3425 013636 012537 013714
3426
3427 013642 012701 000001
3428 013646 012700 177777
3429 013652 033777 013714 000032
3430 013660 001011
3431 013662 004737 013104
3432 013666 005300
3433 013670 001370
3434 013672 005237 001200
3435 013676 005301
3436 013700 001362
3437 013702 000402
3438
3439 013704 062705 000004
3440 013710 000205
3441
3442 013712 000000
3443 013714 000000
3444
3445
3446
3447
3448
3449
3450 013716 012700 003004
3451 013722 005020
3452 013724 020027 003030
3453 013730 001374
3454
3455 013732 012700 003254
3456 013736 005020
3457 013740 020027 003300
3458 013744 001374
3459
3460 013746 000207
3461

: GENERAL TIMER

TIMER1: MOV #-1,R0
DEC R0
BNE -2
RTS PC

:WATCHDOG TIMER TO PREVENT DEVICES FROM ENTERING AN ENDLESS WAIT LOOP
:RETURN IF TIMED OUT
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR

TIMER2: MOV (R5)+,REGHLD ;GET REG
MOV (R5)+,BITHLD ;GET BIT TO BE TESTED

MOV #1,R1
4\$: MOV #-1,R0
1\$: BIT BITHLD,@REGHLD ;BIT THERE?
BNE 2\$;BR IF YES
JSR PC,EXAMKB
DEC R0 ;ELSE DEC COUNTER
BNE 1\$;BR IF NOT TIMED OUT
INC \$DEVCT ;FOR APT
DEC R1
BNE 4\$
BR .+6 ;ELSE TIMED OUT
2\$: ADD #4,R5 ;JUMP OVER ERROR ON RETURN
RTS R5

REGHLD: 0 ;DEVICE REG
BITHLD: 0 ;DEV REG BIT TO BE TESTED

:ROUTINE TO CLEAR BOTH BAD TRACK/SECTOR TABLES

CLRBAD: MOV #DOBAD,R0
1\$: CLR (R0)+
CMP R0,#DOBAD+20.
BNE 1\$

2\$: MOV #D1BAD,R0
CLR (R0)+
CMP R0,#D1BAD+20.
BNE 2\$

RTS PC

3462
3463
3464
3465
3466 013750 010046
3467
3468 013752 012700 003004
3469 013756 005710
3470 013760 001003
3471 013762 013710 177174
3472 013766 000405
3473
3474 013770 005720
3475 013772 020027 003030
3476 013776 001367
3477 014000 104104
3478 014002 012600
3479 014004 000207
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490 014006 010046
3491 014010 010146
3492
3493 014012 013701 002566
3494 014016 000301
3495 014020 053701 002570
3496
3497 014024 012700 003004
3498 014030 020027 003030
3499 014034 001405
3500 014036 005710
3501 014040 001403
3502 014042 022001
3503 014044 001371
3504 014046 000402
3505
3506 014050 062705 000002
3507 014054 012601
3508 014056 012600
3509 014060 000205

:LOAD DX0 BAD TRACK/SECTOR TABLE

LDOBAD: MOV R0,-(SP) ;SAVE
1\$: MOV #DOBAD,R0
TST (R0) ;ENTRY PRESENT?
BNE 2\$;BR IF YES
MOV RXSA,(R0) ;ELSE STORE BAD RXSA IN TABLE
BR 3\$;EXIT
2\$: TST (R0)+ ;BUMP PTR
CMP R0,#DOBAD+20. ;AT END OF TABLE?
BNE 1\$;BR IF NO
ERROR 104 ;10 BAD SECTORS...REPLACE
3\$: MOV (SP)+,R0 ;RESTORE
RTS PC

:CHECK DX0 BAD TRACK/SECTOR TABLE BEFORE DOING RANDOM SEEKS.
:IF TRK/SEC IN TABLE, RETURN TO RE-CALCULATE NEW TRACK & SECTOR.

CKOBAD: MOV R0,-(SP) ;SAVE
MOV R1,-(SP)
MOV D0TRK,R1
SWAB R1
BIS D0SEC,R1
1\$: MOV #DOBAD,R0
CMP R0,#DOBAD+20. ;END OF TABLE?
BEQ 2\$;BR IF YES
TST (R0) ;ELSE ANY ENTRY?
BEQ 2\$;BR IF NO
CMP (R0)+,R1 ;ELSE COMPARE?
BNE 1\$;BR IF NO
BR 3\$;ELSE DONT BUMP RET ADDR
2\$: ADD #2,R5 ;BUMP RET ADDR
3\$: MOV (SP)+,R1 ;RESTORE
MOV (SP)+,R0
RTS R5

```

3510
3511
3512
3513
3514 014062 010046
3515
3516 014064 012700 003254
3517 014070 005710
3518 014072 001003
3519 014074 013710 177174
3520 014100 000405
3521
3522 014102 005720
3523 014104 020027 003300
3524 014110 001367
3525 014112 104105
3526 014114 012600
3527 014116 000207
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538 014120 010046
3539 014122 010146
3540
3541 014124 013701 003036
3542 014130 000301
3543 014132 053701 003040
3544
3545 014136 012700 003254
3546 014142 020027 003300
3547 014146 001405
3548 014150 005710
3549 014152 001403
3550 014154 022001
3551 014156 001371
3552 014160 000402
3553
3554 014162 062705 000002
3555 014166 012601
3556 014170 012600
3557 014172 000205
3558

:*****
:LOAD DX1 BAD TRACK/SECTOR TABLE
:*****
LD1BAD: MOV R0,-(SP) ;SAVE
          MOV #D1BAD,R0
1$:      TST (R0) ;ENTRY PRESENT?
          BNE 2$ ;BR IF YES
          MOV RXSA,(R0) ;ELSE STORE BAD RXSA IN TABLE
          BR 3$ ;EXIT
          TST (R0)+ ;BUMP PTR
          CMP R0,#D1BAD+20. ;AT END OF TABLE?
          BNE 1$ ;BR IF NO
          ERROR 105 ;10 BAD SECTORS...REPLACE
3$:      MOV (SP)+,R0 ;RESTORE
          RTS PC

:*****
:CHECK DX1 BAD TRACK/SECTOR TABLE BEFORE DOING RANDOM SEEKS.
:IF TRK/SEC IN TABLE, RETURN TO RE-CALCULATE NEW TRACK & SECTOR.
:*****
CK1BAD: MOV R0,-(SP) ;SAVE
          MOV R1,-(SP)
          MOV D1TRK,R1
          SWAB R1
          BIS D1SEC,R1
          MOV #D1BAD,R0
1$:      CMP R0,#D1BAD+20. ;END OF TABLE?
          BEQ 2$ ;BR IF YES
          TST (R0) ;ELSE ANY ENTRY?
          BEQ 2$ ;BR IF NO
          CMP (R0)+,R1 ;ELSE COMPARE?
          BNE 1$ ;BR IF NO
          BR 3$ ;ELSE DONT BUMP RET ADDR
          ADD #2,R5 ;BUMP RET ADDR
2$:      ADD #2,R5
3$:      MOV (SP)+,R1 ;RESTORE
          MOV (SP)+,R0
          RTS R5

```

3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601

014174 005237 014202
014200 000002
014202 000000

014204 013737 177514 014334
014212 005737 014334
014216 100002
014220 104007
014222 000002

014224 013737 014330 177516
014232 023727 014330 000015
014240 001413
014242 023727 014330 000012
014250 001413
014252 023727 014330 000176
014260 001413
014262 005237 014330
014266 000417

014270 012737 000012 014330
014276 000413
014300 012737 000040 014330
014306 000407
014310 012737 000015 014330
014316 005237 014332
014322 005237 001200
014326 000002

014330 000000
014332 000000
014334 000000

```
.SBTTL INTERRUPT HANDLERS
:*****
: GENERAL SERVICE ROUTINE TO BUMP 'INTFLG'
: CALLING ROUTINE WILL KNOW WHAT TO DO
:*****

INTSRV: INC      INTFLG
        RTI
INTFLG: 0

:*****
:PRINTER INTERRUPT HANDLER: VALUES FROM 40 THRU 176.
:*****

PRSRV:  MOV      PRS,SPRS      ;STORE
        TST      SPRS          ;ERROR?
        BPL      1$           ;BR IF NO
        ERROR    7            ;PRINTER STATUS ERROR
        RTI

1$:     MOV      PRCHR,PRB     ;ELSE PRINT CHAR
        CMP      PRCHR,#CR    ;JUST DID CR?
        BEQ      2$           ;BR IF YES
        CMP      PRCHR,#LF    ;JUST DID LF?
        BEQ      3$           ;BR IF YES
        CMP      PRCHR,#176   ;JUST DID LAST CHAR?
        BEQ      4$           ;BR IF YES
        INC      PRCHR        ;ELSE BUMP CHAR FOR NEXT INTERRUPT
        BR       5$          ;EXIT

2$:     MOV      #LF,PRCHR    ;DO LF NEXT
        BR       5$

3$:     MOV      #40,PRCHR    ;DO SPACE NEXT
        BR       5$

4$:     MOV      #CR,PRCHR    ;DO CR NEXT & START OVER
        INC      LINCT
        INC      $DEVCT      ;FOR APT
        RTI

5$:     RTI

PRCHR:  0      ;CHAR TO BE PRINTED
LINCT:  0      ;LINE CTR
SPRS:   0      ;STORE PRINTER CSR
```



```
3602
3603
3604
3605
3606
3607 014336 042737 000100 176504 TP1SRV: BIC #IE,TP1S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3608 014344 013737 014432 176506 MOV T1CHR,TP1B ;PRINT CHAR
3609 014352 000002 RTI
3610
3611 014354 013737 176502 014434 TK1SRV: MOV TK1B,T1HLD ;STORE CHAR
3612 014362 023737 014434 014432 CMP T1HLD,T1CHR ;OK?
3613 014370 001401 BEQ 1$ ;BR IF YES
3614 014372 104010 ERROR 10 ;CHAR COMP ERROR
3615
3616 014374 023727 014432 000377 1$: CMP T1CHR,#377 ;LAST CHAR?
3617 014402 001403 BEQ 2$ ;BR IF YES
3618 014404 005237 014432 INC T1CHR ;ELSE BUMP
3619 014410 000404 BR 3$
3620
3621 014412 005037 014432 2$: CLR T1CHR ;START OVER
3622 014416 005237 001200 INC $DEVCT ;FOR APT
3623 014422 052737 000100 176504 3$: BIS #IE,TP1S ;ALLOW PRINTER INTERR
3624 014430 000002 RTI
3625
3626 014432 000000 T1CHR: 0 ;CHAR TO XMITT
3627 014434 000000 T1HLD: 0 ;REC'D CHAR
3628
3629
3630
```

```
::*****
:CLUSTER TERMINAL #2 INTERRUPT HANDLER: LOOP VALUES 0 THRU 377.
:*****
```

```
3631
3632
3633
3634 014436 042737 000100 176514 TP2SRV: BIC #IE,TP2S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3635 014444 013737 014532 176516 MOV T2CHR,TP2B ;PRINT CHAR
3636 014452 000002 RTI
3637
3638 014454 013737 176512 014534 TK2SRV: MOV TK2B,T2HLD ;STORE CHAR
3639 014462 023737 014534 014532 CMP T2HLD,T2CHR ;OK?
3640 014470 001401 BEQ 1$ ;BR IF YES
3641 014472 104011 ERROR 11 ;CHAR COMP ERROR
3642
3643 014474 023727 014532 000377 1$: CMP T2CHR,#377 ;LAST CHAR?
3644 014502 001403 BEQ 2$ ;BR IF YES
3645 014504 005237 014532 INC T2CHR ;ELSE BUMP
3646 014510 000404 BR 3$
3647
3648 014512 005037 014532 2$: CLR T2CHR ;START OVER
3649 014516 005237 001200 INC $DEVCT ;FOR APT
3650 014522 052737 000100 176514 3$: BIS #IE,TP2S ;ALLOW PRINTER INTERR
3651 014530 000002 RTI
3652
3653 014532 000000 T2CHR: 0 ;CHAR TO XMITT
3654 014534 000000 T2HLD: 0 ;REC'D CHAR
```

```

3655
3656
3657
3658
3659
3660 014536 042737 000100 176524 TP3SRV: BIC #IE,TP3S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3661 014544 013737 014632 176526 MOV T3CHR,TP3B ;PRINT CHAR
3662 014552 000002 RTI
3663
3664 014554 013737 176522 014634 TK3SRV: MOV TK3B,T3HLD ;STORE CHAR
3665 014562 023737 014634 014632 CMP T3HLD,T3CHR ;OK?
3666 014570 001401 BEQ 1$ ;BR IF YES
3667 014572 104012 ERROR 12 ;CHAR COMP ERROR
3668
3669 014574 023727 014632 000377 1$: CMP T3CHR,#377 ;LAST CHAR?
3670 014602 001403 BEQ 2$ ;BR IF YES
3671 014604 005237 014632 INC T3CHR ;ELSE BUMP
3672 014610 000404 BR 3$
3673
3674 014612 005037 014632 2$: CLR T3CHR ;START OVER
3675 014616 005237 001200 INC $DEVCT ;FOR APT
3676 014622 052737 000100 176524 3$: BIS #IE,TP3S ;ALLOW PRINTER INTERR
3677 014630 000002 RTI
3678
3679 014632 000000 T3CHR: 0 ;CHAR TO XMITT
3680 014634 000000 T3HLD: 0 ;REC'D CHAR
3681
3682
3683
3684
3685
3686

```

```

;*****
;ASYNC COMM PORT INTERRUPT HANDLER: LOOP VALUES 0 THRU 377.
;*****

```

```

3687 014636 042737 000100 176614 AMXSRV: BIC #IE,AMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
3688 014644 013737 014732 176616 MOV COMCHR,AMXB ;XMITT CHAR
3689 014652 000002 RTI
3690
3691 014654 013737 176612 014734 AMRSRV: MOV AMRB,COMHLD ;STORE CHAR
3692 014662 023737 014734 014732 CMP COMHLD,COMCHR ;OK?
3693 014670 001401 BEQ 1$ ;BR IF YES
3694 014672 104013 ERROR 13 ;CHAR COMPARE ERROR
3695
3696 014674 023727 014732 000377 1$: CMP COMCHR,#377 ;LAST CHAR?
3697 014702 001403 BEQ 2$ ;BR IF YES
3698 014704 005237 014732 INC COMCHR ;ELSE BUMP
3699 014710 000404 BR 3$
3700
3701 014712 005037 014732 2$: CLR COMCHR ;START OVER
3702 014716 005237 001200 INC $DEVCT ;FOR APT
3703 014722 052737 000100 176614 3$: BIS #IE,AMXC ;ALLOW XMIT INTERR
3704 014730 000002 RTI
3705
3706 014732 000000 COMCHR: 0 ;XMIT CHAR
3707 014734 000000 COMHLD: 0 ;REC'D CHAR

```

```

3708
3709
3710
3711
3712
3713 014736 042737 000100 176624 SMXSRV: BIC #IE,SMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
3714 014744 013737 014732 176626 MOV COMCHR,SMXB ;XMITT CHAR
3715 014752 023727 014732 000377 CMP COMCHR,#377 ;LAST CHAR?
3716 014760 001020 BNE 2$ ;BR IF NO
3717
3718 014762 005737 015024 TST LSTCHR
3719 014766 001010 BNE 1$
3720 014770 005237 015024 INC LSTCHR
3721 014774 052737 000100 176624 BIS #IE,SMXC ;ALLOW RDY INTR SO CAN SHUT DOWN FAST
3722 015002 005237 001200 INC $DEVCT ;FOR APT
3723 015006 000002 RTI
3724
3725 015010 042737 000020 176624 1$: BIC #SEND,SMXC ;NO MORE TO SEND
3726 015016 005237 001200 INC $DEVCT ;FOR APT
3727 015022 000002 2$: RTI
3728
3729 015024 000000 LSTCHR: 0 ;SET WHEN LAST CHAR XMIT
3730
3731
3732
3733
3734
3735 015026 013737 176622 014734 SMRSRV: MOV SMRB,COMHLD ;STORE CHAR
3736 015034 123727 014734 000026 CMPB COMHLD,#026 ;IGNORE SYNC CHARS
3737 015042 001430 BEQ 4$
3738 015044 023737 014734 014732 CMP COMHLD,COMCHR ;OK?
3739 015052 001401 BEQ 1$ ;BR IF YES
3740 015054 104014 ERROR 14 ;CHAR COMPARE ERROR
3741
3742 015056 023727 014732 000377 1$: CMP COMCHR,#377 ;LAST CHAR?
3743 015064 001403 BEQ 2$ ;BR IF YES
3744 015066 005237 014732 INC COMCHR ;ELSE BUMP
3745 015072 000411 BR 3$
3746
3747 015074 012737 177777 014732 2$: MOV #-1,COMCHR ;INDICATE DONE
3748 015102 042737 000100 176620 BIC #IE,SMRC ;ALL DONE
3749 015110 005237 001200 INC $DEVCT ;FOR APT
3750 015114 000403 BR 4$
3751
3752 015116 052737 000100 176624 3$: BIS #IE,SMXC ;ALLOW XMIT INTERR
3753 015124 000002 4$: RTI
3754

```

3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793

015126 105737 177170
015132 100413
015134 013737 177170 015236
015142 013737 177172 015240
015150 013737 177174 015242
015156 104015
015160 000002
015162 005737 177170
015166 100011
015170 013737 177170 015236
015176 013737 177172 015240
015204 013737 177174 015242
015212 032737 000020 177170
015220 001003
015222 000177 000000
015226 000000
015230 000177 000000
015234 000000
015236 000000
015240 000000
015242 000000

```
*****
:          DISK INTERRUPT HANDLERS
: ALL DISK INTERRUPTS ENTER THRU RXSRV & DISPATCH TO THE
: CURRENT SERVICE ROUTINE POINTED TO BY:
:          D0DISP FOR DX0 INTERRUPTS
:          D1DISP FOR DX1 INTERRUPTS
: WILL GENERALLY BE IN THE ORDER OF SERVICE ROUTINES BELOW.
*****
```

```
RXSRV:  TSTB   RXCS      ;CONTR RDY?
        BMI    1$      ;BR IF YES
        MOV    RXCS,SRXCS ;ELSE SAVE FOR ERROR TYPEOUTS
        MOV    RXES,SRXES
        MOV    RXSA,SRXSA
        ERROR  15      ;UNEXPECTED INTERRUPT
        RTI                ;GO BACK

1$:     TST    RXCS      ;ERROR?
        BPL    2$      ;BR IF NO

        MOV    RXCS,SRXCS ;ELSE SAVE FOR ERROR TYPEOUTS
        MOV    RXES,SRXES
        MOV    RXSA,SRXSA

2$:     BIT    #USEL,RXCS ;CHECK UNIT SELECT BIT
        BNE    D1INT    ;BR IF INTER FROM DX1

        JMP    @D0DISP   ;ELSE DISPATCH TO DX0 SERVICE ROUTINE
D0DISP: 0

D1INT:  JMP    @D1DISP   ;DX1 INTERRUPT DISPATCH
D1DISP: 0

SRXCS:  0      ;SAVE RXCS
SRXES:  0      ;SAVE RXES
SRXSA:  0      ;SAVE RXSA
```

3794
3795
3796
3797
3798
3799
3800 015244 010246
3801 015246 012703 000100
3802 015252 013701 002566
3803 015256 000301
3804 015260 053701 002570
3805 015264 012737 015326 015226
3806 015272 012737 000101 177170
3807 015300 010137 177172
3808 015304 005303
3809 015306 012702 003300
3810 015312 012237 177172
3811 015316 005303
3812 015320 001374
3813 015322 012602
3814 015324 000002
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824 015326 013701 002566
3825 015332 000301
3826 015334 053701 002570
3827 015340 010137 177174
3828 015344 012737 015402 015226
3829 015352 023727 003476 000005
3830 015360 001004
3831 015362 012737 000115 177170
3832 015370 000403
3833 015372 012737 000105 177170
3834 015400 000002
3835

```

*****
: HANDLER TO FILL THE DRIVE BUFFER FOR DX0
: & POINT TO THE WRITE SECTOR HANDLER.
*****

```

```

DOFBUF: MOV R2, -(SP) ;SAVE
MOV #100, R3 ;WORD CT
MOV DOTRK, R1
SWAB R1
BIS DOSEC, R1
MOV #DOWSEC, DODISP ;POINT TO WRITE SECTOR AFTER RTI
MOV #<FBUF!IE>, RXCS ;ISSUE FILL BUFFER CMD
MOV R1, RXDB ;1'ST WORD = ADDRESS
DEC R3
MOV #WCP, R2 ;GET TABLE ADDR
1$: MOV (R2)+, RXDB ;ALL REST IS FROM TABLE
DEC R3
BNE 1$
MOV (SP)+, R2 ;RESTORE
RTI

```

```

*****
: HANDLER TO WRITE THE SECTOR FOR DX0 & POINT TO WRITE CHECK.
*****

```

```

DOWSEC: MOV DOTRK, R1
SWAB R1
BIS DOSEC, R1
MOV R1, RXSA ;LOAD TRACK & SECTOR ADDR
MOV #DOWCHK, DODISP ;POINT TO WRITE CHECK HANDLER
CMP DXTST, #5
BNE 1$
MOV #<WDDSEC!IE>, RXCS ;ONLY FOR WRITE DELETED DATA TEST
BR 2$
1$: MOV #<WSEC!IE>, RXCS ;ISSUE WRITE SECTOR COMMAND
2$: RTI

```

```

3836
3837
3838
3839
3840
3841
3842
3843
3844 015402 005737 177170
3845 015406 100033
3846 015410 023727 002564 000012
3847 015416 001066
3848 015420 004737 013750
3849 015424 032737 000020 015240
3850 015432 001417
3851 015434 004737 020612
3852 015440 005737 003516
3853 015444 001412
3854 015446 005037 003516
3855 015452 104064
3856 015454 012737 015534 015226
3857 015462 012737 000117 177170
3858 015470 000002
3859
3860 015472 104016
3861 015474 000417
3862
3863 015476 005737 002564
3864 015502 001414
3865 015504 005237 031434
3866 015510 005237 031440
3867 015514 005737 003516
3868 015520 001404
3869 015522 005037 003516
3870 015526 104065
3871 015530 000401
3872 015532 104017
3873
3874 015534 005037 002564
3875 015540 004537 020532
3876 015544 002570
3877 015546 000137 015244
3878
3879 015552 012737 000001 002570
3880 015560 005037 002564
3881 015564 005037 002572
3882 015570 000137 015632
3883
3884 015574 005237 002564
3885 015600 032737 000020 015240
3886 015606 001757
3887 015610 004737 020612
3888 015614 012737 015244 015226
3889 015622 012737 000117 177170
3890 015630 000002
3891

```

```

*****
;HANDLER TO WRITE CHECK ON DXO.
;WILL GO TO FILL BUFFER TO DO SAME SECTOR ON SOFT ERROR
;OR NEXT SECTOR ON NO ERROR OR HARD ERROR.
;WILL GO TO READ SECTOR AFTER ALL SECTORS ON TRACK ARE WRITTEN.
;A RESTORE IS PERFORMED AFTER ANY SEEK ERROR (RNF).
*****
DOWCHK: TST      RXCS      ;ERROR?
        BPL      1$        ;BR IN NO
        CMP      DOERR,#10. ;10 ERRORS?
        BNE      6$        ;BR IF NO & TRY AGAIN
        JSR      PC,LDOBAD  ;LOAD BAD TRK/SEC TABLE
        BIT      #RNF,SRXES ;SEEK ERROR?
        BEQ      7$        ;BR IF NO
        JSR      R7,SEKTST ;SEE IF REALLY ON RIGHT TRACK
        TST      SEKFLG
        BEQ      7$        ;BR IF YES
        CLR      SEKFLG    ;ELSE CLR FLG
        ERROR    64        ;HARD SEEK ERROR WRITE SECTOR
        MOV      #2$,DODISP ;GO TO 2$ AFTER RESTORE
        MOV      #<RESTOR!IE>,RXCS
        RTI
7$:     ERROR    16        ;HARD ERROR WRITE SECTOR
        BR       2$        ;GO TO NEXT SECTOR
1$:     TST      DOERR      ;ANY PREV ERRORS?
        BEQ      2$        ;BR IF NO
        INC      TSERR      ;TOTAL SOFT ERR CT
        INC      PSERR      ;PASS SOFT ERR CT
        TST      SEKFLG    ;SEEK ERROR?
        BEQ      8$        ;BR IF NO
        CLR      SEKFLG
        ERROR    65        ;SOFT SEEK ERROR WRITE SECTOR
        BR       2$
8$:     ERROR    17        ;SOFT ERROR WRITE SECTOR
2$:     CLR      DOERR
        JSR      R5,NXTSEC ;CALC NEXT SECTOR
        DOSEC
4$:     JMP      DOFBUF    ;GO TO FILL BUFFER & DO ANOTHER
5$:     MOV      #1,DOSEC  ;ALL SECTORS DONE...CHECK DATA
        CLR      DOERR
        CLR      DOCMER
        JMP      DORSEC    ;GO TO READ SECTOR HANDLER
6$:     INC      DOERR
        BIT      #RNF,SRXES ;SEEK ERROR?
        BEQ      4$        ;BR IN NO
        JSR      R7,SEKTST ;SEE IF ON TRACK
        MOV      #DOFBUF,DODISP ;GOTO FILL BUFF AFTER RESTORE
        MOV      #<RESTOR!IE>,RXCS
        RTI

```

3892
3893
3894
3895
3896
3897 015632 013701 002566
3898 015636 000301
3899 015640 053701 002570
3900 015644 010137 177174
3901
3902 015650 012737 015666 015226
3903 015656 012737 000107 177170
3904 015664 000002
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914 015666 005737 177170
3915 015672 100065
3916 015674 023727 002564 000012
3917 015702 001417
3918 015704 005237 002564
3919 015710 032737 000020 015240
3920 015716 001745
3921 015720 004737 020612
3922 015724 012737 015632 015226
3923 015732 012737 000117 177170
3924 015740 000002
3925
3926 015742 004737 013750 015240
3927 015746 032737 000020 015240
3928 015754 001417
3929 015756 004737 020612
3930 015762 005737 003516
3931 015766 001422
3932 015770 005037 003516
3933 015774 104066
3934
3935 015776 012737 016714 015226
3936 016004 012737 000117 177170
3937 016012 000002
3938
3939 016014 032737 000010 015240
3940 016022 001404
3941 016024 005237 002602
3942 016030 104106
3943 016032 000424
3944
3945 016034 005037 002602
3946 016040 104020
3947 016042 000137 016714

```

:*****
:HANDLER TO READ A SECTOR ON DX0 & POINT TO CHECK.
:*****

```

```

DORSEC: MOV    D0TRK,R1
        SWAB   R1
        BIS    D0SEC,R1
        MOV    R1,RXSA ;LOAD TRK & SECTOR ADDR
        MOV    #DORCHK,DODISP ;POINT TO READ CHECK HANDLER
        MOV    #<RSEC!IE>,RXCS ;ISSUE READ SECTOR COMMAND
        RTI

```

```

:*****
:HANDLER TO READ CHECK ON DX0.
:WILL GO TO NEXT SECTOR/TRK IF HARD ERROR.
:WILL GO TO EMPTY BUFFER IF NO ERROR OR SOFT ERROR.
:WILL GO TO READ SECTOR (SAME) ON ERROR.
:A RESTORE IS PERFORMED AFTER ANY SEEK ERROR (RNF).
:*****

```

```

DORCHK: TST    RXCS          ;ERROR?
        BPL    1$           ;BR IF NO
        CMP    DOERR,#10.   ;10 ERRORS?
        BEQ    3$           ;BR IF YES
        INC    DOERR        ;ELSE DO AGAIN
        BIT    #RNF,SRXES   ;SEEK ERROR?
        BEQ    DORSEC       ;BR IF NO
        JSR    R7,SEKTST    ;SEE IF ON TRACK
        MOV    #DORSEC,DODISP ;ELSE DO RESTORE & RET TO RD SEC
        MOV    #<RESTOR!IE>,RXCS
        RTI

```

```

3$: JSR    PC,LDOBAD        ;LOAD BAD TRK/SEC TABLE
    BIT    #RNF,SRXES      ;SEEK ERROR?
    BEQ    7$             ;BR IF NO
    JSR    R7,SEKTST      ;SEE IF REALLY ON RIGHT TRACK
    TST    SEKFLG
    BEQ    10$            ;BR IF YES
    CLR    SEKFLG         ;ELSE CLR FLG
    ERROR  66             ;SEEK ERROR READ SECTOR

```

```

MOV    #DONEXT,DODISP ;DO RESTORE
MOV    #<RESTOR!IE>,RXCS
RTI

```

```

7$: BIT    #CRC,SRXES     ;CRC ERROR?
    BEQ    10$            ;BR IF NO
    INC    D0CRC         ;ELSE SET FLAG
    ERROR  106           ;HARD CRC ERROR
    BR     2$            ;& GO TO EMP BUFF TO CHK DATA

```

```

10$: CLR    D0CRC
    ERROR  20            ;HARD READ ERROR
    JMP    DONEXT

```

```

3948
3949 016046 005737 002564 1$: TST DOERR ;ANY PREV ERRORS?
3950 016052 001414 BEQ 2$ ;BR IF NO
3951 016054 005237 031434 INC TSERR ;TOTAL SOFT ERR CT
3952 016060 005237 031440 INC PSERR ;PASS SOFT ERR CT
3953 016064 005737 003516 TST SEKFLG ;SEEK ERROR?
3954 016070 001404 BEQ 9$ ;BR IF NO
3955 016072 005037 003516 CLR SEKFLG
3956 016076 104067 ERROR 67 ;SEEK ERROR READ SECTOR
3957 016100 000401 BR 2$
3958 016102 104021 9$: ERROR 21 ;SOFT ERROR READ SECTOR
3959
3960 016104 005037 002564 2$: CLR DOERR
3961 016110 023727 003476 000005 CMP DXTST,#5 ;DOING WRITE DELETED DATA TESTS?
3962 016116 001005 BNE 6$ ;BR IF NO
3963 016120 032737 000040 177172 BIT #DD,RXES ;'DELETED DATA' SET?
3964 016126 001001 BNE 6$ ;BR IF YES
3965 016130 104052 ERROR 52 ;DD NOT SET
3966
3967 016132 000137 016136 6$: JMP DOEBUF ;GO TO EMPTY BUFFER
3968
3969
3970
3971
3972
3973
3974
3975 ;:*****
3976 ;: HANDLER TO EMPTY THE DRIVE BUFFER FOR DX0
3977 ;:& POINT TO THE CHECK DATA HANDLER.
3978 ;:*****
3979 016136 012703 000100 DOEBUF: MOV #100,R3 ;WORD CT
3980 016142 012701 002604 MOV #DOBUF,R1 ;BUFFER ADDRESS
3981 016146 012737 016174 015226 MOV #DOCDAT,DODISP ;POINT TO CHK DATA AFT INTER.
3982 016154 012737 000103 177170 MOV #<EBUF!IE>,RXCS ;ISSUE EMPTY BUFFER CMD
3983 016162 013721 177172 1$: MOV RXDB,(R1)+ ;GET WORD & STORE IT
3984 016166 005303 DEC R3
3985 016170 001374 BNE 1$
3986 016172 000002 RTI
3987
3988

```



```

3989
3990
3991
3992
3993
3994
3995
3996 016174 005037 002572
3997 016200 012703 000100
3998 016204 012704 003276
3999 016210 013701 002566
4000 016214 000301
4001 016216 053701 002570
4002 016222 012702 002604
4003 016226 020112
4004 016230 001414
4005 016232 010137 002576
4006 016236 011237 002600
4007 016242 005237 002572
4008 016246 000414
4009
4010 016250 021412
4011 016252 001403
4012 016254 011437 002576
4013 016260 000766
4014
4015 016262 005303
4016 016264 001405
4017 016266 062702 000002
4018 016272 062704 000002
4019 016276 000764
4020
4021 016300 005737 002572
4022 016304 001421
4023 016306 005737 002602
4024 016312 001402
4025 016314 104107
4026 016316 000431
4027
4028 016320 023727 002574 000012
4029 016326 001033
4030 016330 023727 003476 000003
4031 016336 001002
4032 016340 104053
4033 016342 000417
4034
4035 016344 104022
4036 016346 000415
4037
4038 016350 005737 002602
4039 016354 001402
4040 016356 104110
4041 016360 000410

```

```

:*****
: HANDLER TO CHECK DATA AFTER EMPTY BUFFER CMD ON DX0.
: WILL GO TO NEXT SECTOR HANDLER WHEN NO ERR OR HARD ERR.
: OR TO READ SECTOR (SAME) WHEN SOFT ERR.
: THIS ROUTINE IS ALSO ENTERED FROM A CRC ERROR FROM A READ COMMAND.
:*****
DOCDAT: CLR      DOCMER
        MOV      #100,R3      :WORD CTR
        MOV      #WCP-2,R4    :GET TABLE ADDR
        MOV      D0TRK,R1
        SWAB     R1
        BIS      DOSEC,R1     :R1 NOW HAS EXPECTED DATA (1'ST WORD)
        MOV      #DOBUF,R2    :GET BUFFER ADDR
        CMP      R1,(R2)      :1'ST WORD COMPARE OK?
        BEQ      2$           :BR IF YES
        MOV      R1,DOEXP     :ELSE SAVE
6$:     MOV      (R2),DOREC
        INC      DOCMER
        BR       3$

1$:     CMP      (R4),(R2)    :COMPARE ALL REST
        BEQ      2$           :BR IF OK
        MOV      (R4),DOEXP   :ELSE SAVE
        BR       6$

2$:     DEC      R3
        BEQ      3$
        ADD     #2,R2
        ADD     #2,R4
        BR       1$

3$:     TST      DOCMER      :ANY COMP ERR?
        BEQ      4$          :BR IF NO
        TST      DOCRC      :HERE FROM CRC ERROR?
        BEQ      14$         :BR IF NO
        ERROR   107         :DATA CRC ERROR
        BR       5$

14$:    CMP      DOCERR,#10. :ELSE, 10 ERRS YET?
        BNE     7$           :BR IF NO
        CMP     DXTST,#3     :DOING INIT TEST?
        BNE     8$           :BR IN NO
        ERROR   53          :INITIALIZE ERROR
        BR       5$

8$:     ERROR   22
        BR       5$

4$:     TST      DOCRC      :HERE FROM CRC ERROR?
        BEQ      15$         :BR IF NO
        ERROR   110         :CRC ERR WITH DATA OK
        BR       5$

```

```

4042
4043 016362 005737 002574 15$: TST DOCERR ;ANY PREV ERR?
4044 016366 001405 BEQ 5$ ;BR IF NO
4045 016370 005237 031434 INC TSERR ;TOTAL SOFT ERR CT
4046 016374 005237 031440 INC PSERR ;PASS SOFT ERR CT
4047 016400 104023 ERROR 23 ;SOFT ERROR DATA COMP
4048 016402 005037 002574 5$: CLR DOCERR
4049 016406 005037 002602 CLR DOCRC
4050 016412 000137 016714 JMP DONEXT
4051
4052 016416 005237 002574 7$: INC DOCERR
4053 016422 023727 003476 000003 CMP DXTST,#3 ;DOING INIT TEST?
4054 016430 001402 BEQ 12$ ;BR IF YES
4055 016432 000137 015632 JMP DORSEC ;ELSE READ SECTOR OVER AGAIN
4056 016436 000137 017052 12$: JMP DODUN ;ALL DONE
4057
4058
4059
4060 ;:*****
4061 ;:HANDLER TO SETUP NEXT SECTOR/TRACK FOR DX0.
4062 ;:WILL GO TO READ SECTOR (NEXT) IF ALL SECTORS ON TRACK NOT DONE.
4063 ;:WILL GO TO FILL BUFFER (NEXT TRACK) IF ALL SECTORS ON TRACK DONE.
4064 ;:WILL SHUT OFF DRV INTERRUPTS IF ALL TRACKS DONE, WITH A MESSAGE.
4065 ;:*****
4066 016442 005037 002564 DONXT1: CLR DOERR
4067 016446 005037 002572 CLR DOCMER
4068 016452 004537 020532 JSR R5,NXTSEC ;CALC NEXT SECTOR
4069 016456 002570 DOSEC
4070 016460 000137 015632 2$: JMP DORSEC ;GO READ SECTOR
4071
4072 016464 004737 012624 3$: JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT
4073 016470 000415 BR 6$ ;RET HERE IF NOT
4074 016472 032777 010000 162440 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4075 016500 001411 BEQ 6$
4076 016502 104401 021572 TYPE ,DRVO
4077 016506 104401 022504 TYPE ,TRK
4078 016512 013746 002566 MOV DOTRK,-(SP) ;:SAVE DOTRK FOR TYPEOUT
4079 016516 104405 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
4080 016520 104401 022574 TYPE ,DUN
4081 016524 005237 003500 6$: INC FIN ;SETUP TO GO TO DX1
4082 016530 023737 002566 003502 CMP DOTRK,MAXTRK ;LAST TRACK?
4083 016536 001406 BEQ 4$ ;BR IF YES
4084 016540 005237 002566 INC DOTRK ;BUMP TRACK
4085 016544 012737 000001 002570 MOV #1,DOSEC ;INIT SECTOR
4086 016552 000417 BR 5$
4087
4088 016554 004737 012624 4$: JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT
4089 016560 000412 BR 7$ ;RET HERE IF NOT
4090 016562 032777 010000 162350 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4091 016570 001406 BEQ 7$
4092 016572 104401 021572 TYPE ,DRVO
4093 016576 104401 022511 TYPE ,PATT
4094 016602 104401 022574 TYPE ,DUN
4095 016606 005237 002560 7$: INC DOPAT ;PATT DONE FLAG
4096 016612 042737 000100 177170 5$: BIC #IE,RXCS ;DISABLE DX0 INTERRUPTS
4097 016620 000002 RTI

```

```

4098
4099
4100
4101
4102
4103 016622 005037 002564 DONXT2: CLR DOERR
4104 016626 005037 002572 CLR DOCMER
4105 016632 005237 003500 INC FIN ;SETUP TO GO TO DX1
4106 016636 005237 002562 INC DOCNT
4107 016642 023737 002562 003504 CMP DOCNT,MAXSK ;DID ALL SEEKS?
4108 016650 001015 BNE 1$ ;BR IF NO
4109 016652 004737 012624 JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT
4110 016656 000412 BR 1$ ;RET HERE IF NOT
4111 016660 032777 010000 162252 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4112 016666 001406 BEQ 1$
4113 016670 104401 021572 TYPE ,DRVO
4114 016674 104401 022524 TYPE ,RANDOM
4115 016700 104401 022574 TYPE ,DUN
4116 016704 042737 000100 177170 1$: BIC #IE,RXCS ;DISABLE DX0 INTERRUPTS
4117 016712 000002 RTI
4118
4119
4120
4121
4122
4123 016714 023727 003476 000001 DONEXT: CMP DXTST,#1
4124 016722 001647 BEQ DONXT1
4125 016724 023727 003476 000002 CMP DXTST,#2
4126 016732 001733 BEQ DONXT2
4127 016734 000446 BR DODUN
4128

```

```

4129
4130
4131
4132
4133
4134 016736 012737 046032 177174 DOINIT: MOV #46032,RXSA ;TRY TO FAKE DRIVE TO GO TO TRK 76 SEC 26
4135 ;INIT SHOULD GO TO TRK 1, SEC 1.
4136 016744 012737 017052 015226 MOV #DODUN,DODISP ;POINT TO DONE HANDLER
4137 016752 012737 000111 177170 MOV #<INITAL!IE>,RXCS ;DO INIT COMMAND
4138 016760 000002 RTI
4139
4140
4141
4142
4143
4144
4145 016762 012737 017052 015226 DOREST: MOV #DODUN,DODISP ;POINT TO DONE HANDLER
4146 016770 012737 000117 177170 MOV #<RESTOR!IE>,RXCS
4147 016776 000002 RTI
4148
4149 017000 012737 017066 015234 D1REST: MOV #D1DUN,D1DISP
4150 017006 012737 000137 177170 MOV #<RESTOR!IE!DX1>,RXCS
4151 017014 000002 RTI
4152
4153
4154
4155
4156
4157 017016 012737 017052 015226 DOSTAT: MOV #DODUN,DODISP ;POINT TO DONE
4158 017024 012737 000113 177170 MOV #<RSTAT!IE>,RXCS ;DO READ STATUS COMMAND
4159 017032 000002 RTI
4160
4161
4162
4163
4164
4165 017034 012737 017052 015226 DOINV: MOV #DODUN,DODISP ;POINT TO DONE
4166 017042 012737 000105 177170 MOV #<WSEC!IE>,RXCS ;DO WRITE SECTOR COMMAND
4167 017050 000002 RTI
4168
4169
4170
4171
4172
4173
4174
4175 017052 005237 003500 DODUN: INC FIN ;SET DONE FLAG
4176 017056 042737 000100 177170 BIC #IE,RXCS ;DISABLE FURTHUR DX0 INTERRUPTS
4177 017064 000002 RTI
4178
4179 017066 005237 003500 D1DUN: INC FIN
4180 017072 042737 000120 177170 BIC #<IE!DX1>,RXCS
4181 017100 000002 RTI
4182

```

4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225

017102 010246
017104 012703 000100
017110 013701 003036
017114 000301
017116 053701 003040
017122 005737 003510
017126 001005
017130 005737 003512
017134 001002
017136 052701 100000
017142 012737 017204 015234 2\$:
017150 012737 000121 177170
017156 010137 177172
017162 005303
017164 012702 003300
017170 012237 177172 1\$:
017174 005303
017176 001374
017200 012602
017202 000002
017204 013701 003036
017210 000301
017212 053701 003040
017216 010137 177174
017222 012737 017240 015234
017230 012737 000125 177170
017236 000002

```
*****  
: HANDLER TO FILL THE DRIVE BUFFER FOR DX1  
: & POINT TO THE WRITE SECTOR HANDLER.  
*****  
D1FBUF: MOV R2, -(SP) ;SAVE  
MOV #100, R3 ;WORD CT  
MOV D1TRK, R1  
SWAB R1  
BIS D1SEC, R1  
  
TST COMP1 ;DONT SET BIT15 IN COMPATABILITY TESTS  
BNE 2$  
TST COMP2  
BNE 2$  
BIS #BIT15, R1 ;TO DISTINGUISH DX1 DATA FROM DX0 DATA  
  
2$: MOV #D1WSEC, D1DISP ;POINT TO WRITE SECTOR AFTER RTI  
MOV #<FBUF!IE!DX1>, RXCS ;ISSUE FILL BUFFER CMD  
MOV R1, RXDB ;1'ST WORD = ADDRESS  
DEC R3  
MOV #WCP, R2 ;GET TABLE ADDR  
1$: MOV (R2)+, RXDB  
DEC R3  
BNE 1$  
MOV (SP)+, R2 ;RESTORE  
RTI  
  
*****  
: HANDLER TO WRITE THE SECTOR FOR DX1 & POINT TO WRITE CHECK.  
*****  
D1WSEC: MOV D1TRK, R1  
SWAB R1  
BIS D1SEC, R1  
MOV R1, RXSA ;LOAD TRACK & SECTOR ADDR  
MOV #D1WCHK, D1DISP ;POINT TO WRITE CHECK HANDLER  
MOV #<WSEC!IE!DX1>, RXCS ;ISSUE WRITE SECTOR COMMAND  
RTI
```

```

4226
4227
4228
4229
4230
4231
4232
4233
4234 017240 005737 177170
4235 017244 100033
4236 017246 023727 003034 000012
4237 017254 001066
4238 017256 004737 014062
4239 017262 032737 000020 015240
4240 017270 001417
4241 017272 004737 020612
4242 017276 005737 003516
4243 017302 001412
4244 017304 005037 003516
4245 017310 104070
4246 017312 012737 017372 015234
4247 017320 012737 000137 177170
4248 017326 000002
4249
4250 017330 104024
4251 017332 000417
4252
4253 017334 005737 003034
4254 017340 001414
4255 017342 005237 031434
4256 017346 005237 031440
4257 017352 005737 003516
4258 017356 001404
4259 017360 005037 003516
4260 017364 104071
4261 017366 000401
4262 017370 104025
4263
4264 017372 005037 003034
4265 017376 004537 020532
4266 017402 003040
4267 017404 000137 017102
4268
4269 017410 012737 000001 003040
4270 017416 005037 003034
4271 017422 005037 003042
4272 017426 000137 017470
4273
4274 017432 005237 003034
4275 017436 032737 000020 015240
4276 017444 001757
4277 017446 004737 020612
4278 017452 012737 017102 015234
4279 017460 012737 000137 177170
4280 017466 000002

*****
:HANDLER TO WRITE CHECK ON DX1.
:WILL GO TO FILL BUFFER TO DO SAME SECTOR ON SOFT ERROR
:OR NEXT SECTOR ON NO ERROR OR HARD ERROR.
:WILL GO TO READ SECTOR AFTER ALL SECTORS ON TRACK ARE WRITTEN.
:A RESTORE IS PERFORMED-AFTER ANY SEEK ERROR (RNF).
*****
D1WCHK: TST      RXCS      ;ERROR?
        BPL      1$        ;BR IN NO
        CMP      D1ERR,#10. ;10 ERRORS?
        BNE      6$        ;BR IF NO & TRY AGAIN
        JSR      PC,LD1BAD  ;LOAD BAD TRK/SEC TABLE
        BIT      #RNF,SRXES ;SEEK ERROR?
        BEQ      7$        ;BR IF NO
        JSR      R7,SEKTST ;SEE IF REALLY ON RIGHT TRACK
        TST      SEKFLG
        BEQ      7$        ;BR IF YES
        CLR      SEKFLG    ;ELSE CLR FLG
        ERROR    70        ;SEEK ERROR WRITE SECTOR
        MOV      #2$,D1DISP ;RET TO 2$ AFT RESTORE
        MOV      #<RESTOR!IE!DX1>,RXCS
        RTI

7$:  ERROR 24      ;ERROR WRITE SECTOR
     BR    2$      ;GO TO NEXT SECTOR

1$:  TST    D1ERR  ;ANY PREV ERRORS?
     BEQ    2$    ;BR IF NO
     INC    TSERR  ;TOTAL SOFT ERR CT
     INC    PSERR  ;PASS SOFT ERR CT
     TST    SEKFLG ;SEEK ERROR?
     BEQ    8$    ;BR IF NO
     CLR    SEKFLG
     ERROR  71    ;SEEK ERROR WRITE SECTOR
     BR    2$

8$:  ERROR 25      ;ERROR WRITE SECTOR

2$:  CLR    D1ERR
     JSR    R5,NXTSEC ;CALC NEXT SECTOR
        D1SEC
4$:  JMP    D1FBUF  ;GO TO FILL BUFFER & DO ANOTHER

5$:  MOV    #1,D1SEC ;ALL SECTORS DONE...CHECK DATA
     CLR    D1ERR
     CLR    D1CMER
     JMP    D1RSEC  ;GO TO READ SECTOR HANDLER

6$:  INC    D1ERR
     BIT    #RNF,SRXES ;SEEK ERROR?
     BEQ    4$        ;BR IF NO
     JSR    R7,SEKTST ;SEE IF ON TRACK
     MOV    #D1FBUF,D1DISP ;RET TO FILL BUFF AFT RESTORE
     MOV    #<RESTOR!IE!DX1>,RXCS
     RTI

```

```
4281
4282
4283
4284
4285
4286 017470 013701 003036
4287 017474 000301
4288 017476 053701 003040
4289 017502 010137 177174
4290
4291 017506 012737 017524 015234
4292 017514 012737 000127 177170
4293 017522 000002
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303 017524 005737 177170
4304 017530 100065
4305 017532 023727 003034 000012
4306 017540 001417
4307 017542 005237 003034
4308 017546 032737 000020 015240
4309 017554 001745
4310 017556 004737 020612
4311 017562 012737 017470 015234
4312 017570 012737 000137 177170
4313 017576 000002
4314
4315 017600 004737 014062
4316 017604 032737 000020 015240
4317 017612 001417
4318 017614 004737 020612
4319 017620 005737 003516
4320 017624 001422
4321 017626 005037 003516
4322 017632 104072
4323
4324 017634 012737 020520 015234
4325 017642 012737 000137 177170
4326 017650 000002
4327
4328 017652 032737 000010 015240
4329 017660 001404
4330 017662 005237 003052
4331 017666 104111
4332 017670 000424
4333
4334 017672 005037 003052
4335 017676 104026
4336 017700 000137 020520

;*****
;HANDLER TO READ A SECTOR ON DX1 & POINT TO CHECK.
;*****

D1RSEC: MOV D1TRK,R1
        SWAB R1
        BIS D1SEC,R1
        MOV R1,RXSA ;LOAD TRK & SECTOR ADDR
        MOV #D1RCHK,D1DISP ;POINT TO READ CHECK HANDLER
        MOV #<RSEC!IE!DX1>,RXCS ;ISSUE READ SECTOR COMMAND
        RTI

;*****
;HANDLER TO READ CHECK ON DX1.
;WILL GO TO NEXT SECTOR/TRK IF HARD ERROR.
;WILL GO TO EMPTY BUFFER IF NO ERROR OR SOFT ERROR.
;WILL GO TO READ SECTOR (SAME) ON ERROR.
;A RESTORE IS PERFORMED AFTER ANY SEEK ERROR (RNF).
;*****

D1RCHK: TST RXCS ;ERROR?
        BPL 1$ ;BR IF NO
        CMP D1ERR,#10. ;10 ERRORS?
        BEQ 3$ ;BR IF YES
        INC D1ERR ;ELSE TRY AGAIN
        BIT #RNF,SRXES ;SEEK ERROR?
        BEQ D1RSEC ;BR IF NO
        JSR R7,SEKTST ;SEE IF ON TRACK
        MOV #D1RSEC,D1DISP ;ELSE DO RESTORE & RET TO RD SEC
        MOV #<RESTOR!IE!DX1>,RXCS
        RTI

3$: JSR PC,LD1BAD ;LOAD BAD TRK/SEC TABLE
    BIT #RNF,SRXES ;SEEK ERROR?
    BEQ 7$ ;BR IF NO
    JSR R7,SEKTST ;SEE IF REALLY ON RIGHT TRACK
    TST SEKFLG
    BEQ 10$ ;BR IF YES
    CLR SEKFLG ;ELSE CLR FLG
    ERROR 72 ;SEEK ERROR READ SECTOR

MOV #D1NEXT,D1DISP ;DO RESTORE
MOV #<RESTOR!IE!DX1>,RXCS
RTI

7$: BIT #CRC,SRXES ;CRC ERROR?
    BEQ 10$ ;BR IF NO
    INC D1CRC ;ELSE SET FLAG
    ERROR 111 ;HARD CRC ERROR
    BR 2$ ;& GO TO EMP BUFF TO CHK. DATA

10$: CLR D1CRC
     ERROR 26 ;HARD READ ERROR
     JMP D1NEXT
```

```

4337
4338 017704 005737 003034      1$:   TST      D1ERR      ;ANY PREV ERRORS?
4339 017710 001414              BEQ      2$           ;BR IF NO
4340 017712 005237 031434      INC      TSERR       ;TOTAL SOFT ERR CT
4341 017716 005237 031440      INC      PSERR       ;PASS SOFT ERR CT
4342 017722 005737 003516      TST      SEKFLG     ;SEEK ERROR?
4343 017726 001404              BEQ      9$           ;BR IF NO
4344 017730 005037 003516      CLR      SEKFLG     ;SEEK ERROR READ SECTOR
4345 017734 104073              ERROR   73
4346 017736 000401              BR       2$
4347 017740 104027      9$:   ERROR   27           ;SOFT ERROR READ SECTOR
4348
4349 017742 005037 003034      2$:   CLR      D1ERR
4350 017746 000137 017752      JMP     D1EBUF      ;GO TO EMPTY BUFFER
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362 017752 012703 000100      D1EBUF: MOV     #100,R3      ;WORD CT
4363 017756 012701 003054      MOV     #D1BUF,R1       ;BUFFER ADDRESS
4364 017762 012737 020010 015234  MOV     #D1CDAT,D1DISP  ;POINT TO CHK DATA AFT INTER.
4365 017770 012737 000123 177170  MOV     #<EBUF!IE!DX1>,RXCS ;ISSUE EMPTY BUFFER CMD
4366 017776 013721 177172      1$:   MOV     RXDB,(R1)+    ;GET WORD & STORE IT
4367 020002 005303              DEC     R3
4368 020004 001374              BNE    1$
4369 020006 000002              RTI
4370

```

```

:*****
: HANDLER TO EMPTY THE DRIVE BUFFER FOR DX1
: & POINT TO THE CHECK DATA HANDLER.
:*****

```



```

4371
4372
4373
4374
4375
4376
4377
4378 020010 005037 003042
4379 020014 012703 000100
4380 020020 012704 003276
4381 020024 013701 003036
4382 020030 000301
4383 020032 053701 003040
4384
4385 020036 005737 003510
4386 020042 001005
4387 020044 005737 003512
4388 020050 001002
4389 020052 052701 100000
4390
4391 020056 012702 003054
4392 020062 020112
4393 020064 001414
4394 020066 010137 003046
4395 020072 011237 003050
4396 020076 005237 003042
4397 020102 000414
4398
4399 020104 021412
4400 020106 001403
4401 020110 011437 003046
4402 020114 000766
4403
4404 020116 005303
4405 020120 001405
4406 020122 062702 000002
4407 020126 062704 000002
4408 020132 000764
4409
4410 020134 005737 003042
4411 020140 001413
4412 020142 005737 003052
4413 020146 001402
4414 020150 104112
4415 020152 000423
4416 020154 023727 003044 000012 14$:
4417 020162 001025
4418 020164 104030
4419 020166 000415

```

```

*****
: HANDLER TO CHECK DATA AFTER EMPTY BUFFER CMD ON DX1.
: WILL GO TO NEXT SECTOR HANDLER WHEN NO ERR OR HARD ERR.
: OR TO READ SECTOR (SAME) WHEN SOFT ERR.
: THIS ROUTINE IS ALSO ENTERED FROM A CRC ERROR FROM A READ COMMAND.
*****
D1CDAT: CLR      D1CMER
        MOV      #100,R3      ;WORD CTR
        MOV      #WCP-2,R4   ;GET TABLE ADDR
        MOV      D1TRK,R1
        SWAB     R1
        BIS      D1SEC,R1

        TST      COMP1      ;DONT SET BIT15 IN COMPATABILITY TESTS
        BNE     6$
        TST      COMP2
        BNE     6$
        BIS      #BIT15,R1   ;TO DISTINGUISH DX1 DATA FROM DX0 DATA

6$:     MOV      #D1BUF,R2   ;GET BUFFER ADDR
        CMP      R1,(R2)    ;1'ST WORD COMPARE OK?
        BEQ     2$          ;BR IF YES
        MOV      R1,D1EXP   ;ELSE SAVE
        MOV      (R2),D1REC
8$:     INC      D1CMER
        BR      3$

1$:     CMP      (R4),(R2)   ;COMPARE ALL REST
        BEQ     2$          ;BR IF OK
        MOV      (R4),D1EXP ;ELSE SAVE
        BR      8$

2$:     DEC      R3
        BEQ     3$
        ADD     #2,R2
        ADD     #2,R4
        BR      1$

3$:     TST      D1CMER     ;ANY COMP ERR?
        BEQ     4$          ;BR IF NO
        TST      D1CRC     ;HERE FROM CRC ERROR?
        BEQ     14$        ;BR IF NO
        ERROR   112        ;DATA CRC ERROR
        BR      5$
        CMP      D1CERR,#10. ;ELSE, 10 ERRS YET?
        BNE     7$          ;BR IF NO
        ERROR   30         ;HARD ERROR DATA COMP
        BR      5$

```

```

4420
4421 020170 005737 003052      4$:   TST      D1CRC      ;HERE FROM CRC ERROR?
4422 020174 001402              BEQ      15$          ;BR IF NO
4423 020176 104113              ERROR   113          ;CRC ERR WITH DATA OK
4424 020200 000410              BR      5$
4425 020202 005737 003044      15$:  TST      D1CERR     ;ANY ERR?
4426 020206 001405              BEQ      5$
4427 020210 005237 031434      INC      TSERR       ;TOTAL SOFT ERR CT
4428 020214 005237 031440      INC      PSERR       ;PASS SOFT ERR CT
4429 020220 104031              ERROR   31          ;SOFT ERROR DATA COMP
4430
4431 020222 005037 003044      5$:   CLR      D1CERR     ;
4432 020226 005037 003052      CLR      D1CRC
4433 020232 000137 020520      JMP      D1NEXT
4434
4435 020236 005237 003044      7$:   INC      D1CERR     ;
4436 020242 000137 017470      JMP      D1RSEC     ;READ SECTOR OVER AGAIN
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446 020246 005037 003034      D1NXT1: CLR      D1ERR
4447 020252 005037 003042      CLR      D1CMER
4448 020256 004537 020532      JSR      R5,NXTSEC  ;CALC NEXT SECTOR
4449 020262 003040              D1SEC
4450 020264 000137 017470      2$:   JMP      D1RSEC     ;GO READ SECTOR
4451
4452 020270 004737 012624      3$:   JSR      PC,CHKCON ;CHECK IF CONSOLE PRESENT
4453 020274 000415              BR      6$          ;RET HERE IF NOT
4454 020276 032777 010000 160634  BIT      #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4455 020304 001411              BEQ      6$
4456 020306 104401 021577      TYPE    ,DRV1
4457 020312 104401 022504      TYPE    ,TRK
4458 020316 013746 003036      MOV      D1TRK,-(SP) ;:SAVE D1TRK FOR TYPEOUT
4459 020322 104405              TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
4460 020324 104401 022574      TYPE    ,DUN
4461 020330 005237 003500      6$:   INC      FIN
4462 020334 023737 003036 003502  CMP      D1TRK,MAXTRK ;:SETUP TO GO TO DX0
4463 020342 001406              BEQ      4$          ;:LAST TRACK?
4464 020344 005237 003036      INC      D1TRK       ;:BR IF YES
4465 020350 012737 000001 003040  MOV      #1,D1SEC    ;:BUMP TRACK
4466 020356 000417              BR      5$          ;:INIT SECTOR
4467
4468 020360 004737 012624      4$:   JSR      PC,CHKCON ;CHECK IF CONSOLE PRESENT
4469 020364 000412              BR      7$          ;RET HERE IF NOT
4470 020366 032777 010000 160544  BIT      #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4471 020374 001406              BEQ      7$
4472 020376 104401 021577      TYPE    ,DRV1
4473 020402 104401 022511      TYPE    ,PATT
4474 020406 104401 022574      TYPE    ,DUN
4475 020412 005237 003030      7$:   INC      D1PAT     ;PATT DONE FLAG

```

```

4476 020416 042737 000120 177170 5$: BIC #<IE!DX1>,RXCS ;DISABLE DX1 INTERRUPTS
4477 020424 000002 RTI
4478
4479
4480
4481 ::*****
4482 :HANDLER TO SETUP TO GO TO NEXT RANDOM TRACK/SECTOR FOR DX1.
4483 :*****
4484 020426 005037 003034 D1NXT2: CLR D1ERR
4485 020432 005037 003042 CLR D1CMER
4486 020436 005237 003500 INC FIN ;SETUP TO GO TO DX0
4487 020442 005237 003032 INC D1CNT
4488 020446 023737 003032 003504 CMP D1CNT,MAXSK ;DID ALL SEEKS?
4489 020454 001015 BNE 1$ ;BR IF NO
4490 020456 004737 012624 JSR PC,CHKCON ;CHECK IF CONSOLE PRESENT
4491 020462 000412 BR 1$ ;RET HERE IF NOT
4492 020464 032777 010000 160446 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
4493 020472 001406 BEQ 1$
4494 020474 104401 021577 TYPE ,DRV1
4495 020500 104401 022524 TYPE ,RANDOM
4496 020504 104401 022574 TYPE ,DUN
4497 020510 042737 000120 177170 1$: BIC #<IE!DX1>,RXCS ;DISABLE DX1 INTERRUPTS
4498 020516 000002 RTI
4499
4500
4501 020520 023727 003476 000001 D1NEXT: CMP DXTST,#1
4502 020526 001647 BEQ D1NXT1
4503 020530 000736 BR D1NXT2
4504
4505
4506
4507
4508
4509 ::*****
4510 :ROUTINE TO DETERMINE NEXT SECTOR FOR INTERLEAVING
4511 :*****
4512 020532 027527 000000 000031 NXTSEC: CMP @0(R5),#31 ;ARG IS DOSEC/D1SEC
4513 020540 001003 BNE 1$
4514 020542 012735 000002 MOV #2,@(R5)+ ;BUMP RET ADDR
4515 020546 000420 BR 4$
4516
4517 020550 027527 000000 000032 1$: CMP @0(R5),#32
4518 020556 001003 BNE 2$
4519 020560 012735 000003 MOV #3,@(R5)+
4520 020564 000411 BR 4$
4521
4522 020566 027527 000000 000030 2$: CMP @0(R5),#30
4523 020574 001403 BEQ 3$
4524 020576 062735 000003 ADD #3,@(R5)+
4525 020602 000402 BR 4$
4526
4527 020604 062705 000006 3$: ADD #6,R5 ;SKIP ARG, JMP DOFBUF/D1FBUF
4528 020610 000205 4$: RTS R5

```

```

4529
4530
4531
4532
4533
4534
4535
4536
4537
4538 020612 005037 003516
4539 020616 042737 000100 177170
4540 020624 105037 177174
4541 020630 005237 177174
4542 020634 052737 000007 177170
4543 020642 004537 013632
4544 020646 177170
4545 020650 000200
4546 020652 104131
4547 020654 000240
4548
4549 020656 032737 000020 177172
4550 020664 001412
4551 020666 122737 000031 177174
4552 020674 003404
4553 020676 062737 000004 177174
4554 020704 000753
4555
4556 020706 005237 003516
4557 020712 000207

```

```

*****
:ROUTINE TO READ MANY SECTORS ON A GIVEN TRACK.
:IF GET RNF ON ALL READS, IT'S ASSUMED THE HEAD IS ON THE WRONG TRACK & SEKFLG SETS.
:IF ANY SECTOR READS CORRECTLY, IT'S ASSUMED THE INITIAL ERROR WAS
:DUE TO READ HEADER ERROR & SEKFLG CLEARS.
:THIS ROUTINE PERFORMS IN FLAG MODE.
*****
SEKTST: CLR      SEKFLG
        BIC      #IE,RXCS      ;TURN OFF INTR ENABLE
        CLRB     RXSA          ;SET UP CURRENT TRK WITH SECTOR 1
        INC      RXSA
4$:     BIS      #RSEC,RXCS    ;ISSUE READ SECTOR
        JSR      R5,TIMER2    ;WAIT FOR DISK DONE
        RXCS
        DONE
        ERROR    131          ;NO DONE
        NOP
        BIT      #RNF,RXES    ;RNF ERROR?
        BEQ      2$           ;BR IF NO
        CMPB     #31,RXSA     ;LAST SECTOR?
        BLE      3$           ;BR IF YES
        ADD      #4,RXSA      ;ELSE TRY NEXT SECTOR
        BR       4$
3$:     INC      SEKFLG      ;SET SEEK ERROR FLAG
2$:     RTS      R7

```

```

4558
4559
4560
4561
4562
4563
4564
4565
4566
4567 020714 005037 003506
4568 020720 104401 023023
4569 020724 000000
4570
4571 020726 004737 021234
4572
4573 020732 005037 021226
4574 020736 012737 000001 021230
4575 020744 012737 000001 021232
4576 020752 105737 177170
4577 020756 100375
4578
4579 020760 004537 021304
4580 020764 000007
4581
4582 020766 005737 177170
4583 020772 100011
4584 020774 005237 021226
4585 021000 023727 021226 000012
4586 021006 001364
4587 021010 104100
4588 021012 000000
4589 021014 000776
4590
4591 021016 005037 021226
4592 021022 004537 021344
4593 021026 002604
4594
4595 021030 004537 021304
4596 021034 000025
4597
4598 021036 005737 177170
4599 021042 100011
4600 021044 005237 021226
4601 021050 023727 021226 000012
4602 021056 001364
4603 021060 104101
4604 021062 000000
4605 021064 000776

```

```

*****
:THE FOLLOWING CODE, UP TO THE PROGRAM MESSAGES, IS A UTILITY TO COPY
:THE DISKETTE CONTAINING ONLY THE BOOT & THE EXERCISER FROM DX0 TO DX1.
:THE OPERATOR CAN TYPE 'P' TO DO ANOTHER COPY OR
:TYPE '240G' TO ENTER NORMAL TESTING.
:ENTIRE CODE, INCLUDING TEXT & ERROR TABLE, IS 400(10) WORDS.
*****
COPY:  CLR      COPFLG      ;DONT NEED FLAG ANY MORE
      TYPE     ,COPMSG      ;INSERT SCRATCH IN DX1 & TYPE P
      HALT                                ;WAIT FOR PROCEED
      JSR      PC,GETSEC      ;CALCULATE MAX SECTORS TO BE COPIED
      CLR      ERFLG          ;INITIALIZE
      MOV      #1,TRACK
      MOV      #1,SECT
      TSTB     RXCS           ;DONE? (CTL RDY)
      BPL      .-4           ;BR IF NO & TRY AGAIN
1$:   JSR      R5,RDWR        ;READ DX0
      RSEC
      TST      RXCS          ;ERROR?
      BPL      2$           ;BR IF NO
      INC      ERFLG
      CMP      ERFLG,#10.   ;10 ERRORS?
      BNE      1$           ;BR IF NO & TRY AGAIN
      ERROR    100         ;HARD ERR READ SECTOR DX0
      HALT
      BR       .-2         ;FORCE A RESTART FROM 250
2$:   CLR      ERFLG
      JSR      R5,EBUFF     ;EMPTY BUFFER INTO DOBUF
      DOBUF
3$:   JSR      R5,RDWR        ;WRITE DX1
      <WSEC!DX1>
      TST      RXCS          ;ERROR?
      BPL      4$           ;BR IF NO
      INC      ERFLG
      CMP      ERFLG,#10.   ;10 ERRORS?
      BNE      3$           ;BR IF NO & TRY AGAIN
      ERROR    101         ;ERROR WRITE SECTOR
      HALT
      BR       .-2         ;FORCE RESTART AT 250

```

```

4606
4607 021066 005037 021226      4$: CLR ERFLG
4608 021072 004537 021304      JSR R5,RDWR ;READ DX1
4609 021076 000027              <RSEC!DX1>
4610
4611 021100 005737 177170      TST RXCS ;ERROR?
4612 021104 100011      BPL 5$ ;BR IF NO
4613 021106 005237 021226      INC ERFLG
4614 021112 023727 021226 000012  CMP ERFLG,#10. ;10 ERRORS?
4615 021120 001362      BNE 4$ ;BR IF NO & TRY AGAIN
4616 021122 104103      ERROR 103 ;HARD ERR READ SECTOR DX1
4617 021124 000000      HALT
4618 021126 000776      BR .-2 ;FORCE 250 RESTART
4619
4620 021130 004537 021344      5$: JSR R5,EBUFF ;EMPTY BUFFER INTO D1BUF
4621 021134 003054              D1BUF
4622
4623 021136 004737 021406      JSR PC,DATCHK ;COMPARE DOBUF WITH D1BUF
4624
4625 021142 005237 021224      INC SECCNT ;TOTAL DONE
4626
4627 021146 023737 021224 021222  CMP SECCNT,MAXSEC ;DID TOTAL # SECTORS?
4628 021154 001415      BEQ 8$ ;BR IF YES
4629
4630 021156 023727 021232 000032  CMP SECT,#32 ;ELSE DID WE DO LAST SECTOR ON TRACK?
4631 021164 001403      BEQ 6$ ;BR IF YES
4632 021166 005237 021232      INC SECT ;ELSE BUMP & DO ANOTHER
4633 021172 000672      BR 1$
4634
4635 021174 005237 021230      6$: INC TRACK ;BUMP TRK
4636 021200 012737 000001 021232  MOV #1,SECT ;INIT SECTOR
4637 021206 000664      BR 1$ ;& DO ANOTHER
4638
4639 021210 104401 023124      8$: TYPE ,COPDUN ;DONE MSG
4640 021214 000000      HALT
4641 021216 000137 020714      JMP COPY ;DO AGAIN IF 'P' TYPED
4642
4643 021222 000000      MAXSEC: 0 ;TOTAL # SECTORS TO READ FROM DX0
4644 021224 000000      SECCNT: 0 ;TOTAL # SECTORS WRITTEN TO DX1
4645 021226 000000      ERFLG: 0
4646 021230 000000      TRACK: 0
4647 021232 000000      SECT: 0

```

4648
4649
4650
4651
4652
4653 021234 012700 032276
4654 021240 006200
4655 021242 005500
4656 021244 062700 000377
4657 021250 042700 000377
4658
4659 021254 000300
4660 021256 006300
4661 021260 006300
4662
4663 021262 062700 000074
4664 021266 062700 000032
4665
4666 021272 010037 021222
4667
4668
4669 021276 005037 021224
4670 021302 000207
4671
4672
4673
4674
4675
4676
4677
4678 021304 013700 021230
4679 021310 000300
4680 021312 053700 021232
4681 021316 010037 177174
4682 021322 012537 177170
4683 021326 004537 013632
4684 021332 177170
4685 021334 000200
4686 021336 104124
4687 021340 000000
4688 021342 000205
4689

```

:*****
:ROUTINE TO CALCULATE THE MAX # OF SECTORS TO READ FROM DX0
:*****

```

```

GETSEC: MOV     #LSTAD,RO      ;GET MAX ADDR
        ASR     RO            ;GET WORD CT
        ADC     RO            ;DONT LOOSE ANY
        ADD     #377,RO       ;ROUND OFF TO NEAREST WHOLE WORD
        BIC     #377,RO
        SWAB    RO
        ASL     RO
        ASL     RO            ;DIVIDE-BY 100(8) FOR SECTOR CT.
        ADD     #<15.*4>,RO   ;ADD SECTOR CT FOR BLOCKS 0 - 14 OF DX0
        ADD     #26.,RO      ;FACTOR TO TAKE CARE OF RT-11 SECTOR MAPPING
        MOV     RO,MAXSEC     ;SAVE
                                ;THIS IS THE MAX SECTORS WE WILL READ OFF DX0
                                ;& WRITE TO DX1 STARTING AT TRK 1, SECT 1
                                ;COUNT OF TOTAL SECTORS WRITTEN TO DX1
        CLR     SECCNT
        RTS     PC

```

```

:*****
:ROUTINE TO READ OR WRITE DX0 OR DX1
:R5 POINTS TO THE COMMAND & DISK #
:*****

```

```

RDWR:  MOV     TRACK,RO
        SWAB    RO
        BIS     SECT,RO
        MOV     RO,RXSA      ;LOAD TRK/SEC
        MOV     (R5)+,RXCS   ;ISSUE COMMAND
        JSR     R5,TIMER2    ;WAIT FOR DONE
                                RXCS
                                DONE
        ERROR   124          ;NO DONE
        HALT
        RTS     R5

```

```
4690  
4691  
4692  
4693  
4694  
4695  
4696 021344 012700 000100  
4697 021350 012501  
4698 021352 012737 000003 177170  
4699 021360 013721 177172  
4700 021364 005300  
4701 021366 001374  
4702 021370 004537 013632  
4703 021374 177170  
4704 021376 000200  
4705 021400 104125  
4706 021402 000000  
4707 021404 000205  
4708  
4709  
4710  
4711  
4712  
4713  
4714 021406 012700 000100  
4715 021412 012701 002604  
4716 021416 012702 003054  
4717 021422 022122  
4718 021424 001403  
4719 021426 104102  
4720 021430 000000  
4721 021432 000776  
4722  
4723 021434 005300  
4724 021436 001371  
4725 021440 000207  
4726
```

```
*****  
:ROUTINE TO EMPTY BUFFER INTO DOBUF OR D1BUF  
:R5 POINTS TO IT  
*****  
EBUFF: MOV #100,R0 ;WD CT  
MOV (R5)+,R1 ;BUFFER ADDR  
MOV #EBUF,RXCS ;ISSUE EMPTY BUFFER COMMAND  
1$: MOV RXDB,(R1)+ ;PUT IT IN TABLE  
DEC R0  
BNE 1$  
JSR R5,TIMER2 ;WAIT FOR DONE  
RXCS  
DONE  
ERROR 125 ;NO DONE  
HALT  
RTS R5  
*****  
:ROUTINE TO COMPARE DOBUF WITH D1BUF  
*****  
DATCHK: MOV #100,R0 ;WD CT  
MOV #DOBUF,R1  
MOV #D1BUF,R2  
1$: CMP (R1)+,(R2)+ ;COMPARE?  
BEQ 2$ ;BR IF YES  
ERROR 102 ;MISCOMPARE  
HALT  
BR .-2  
2$: DEC R0  
BNE 1$  
RTS PC
```


4727
4728
4729
4730
4731
4732
4733
4734

.SBTTL PROGRAM MESSAGES

::*****
: MESSAGES & ERROR FORMATS
::*****

.NLIST BEX

021442	041600	052514	052123	CLOPT:	.ASCIZ	<CRLF>/CLUSTER OPTION/	
021462	052200	051105	020115	CLT1:	.ASCIZ	<CRLF>/TERM #1/	
021473	200	042524	046522	CLT2:	.ASCIZ	<CRLF>/TERM #2/	
021504	052200	051105	020115	CLT3:	.ASCIZ	<CRLF>/TERM #3/	
021515	200	054523	041516	SCOM:	.ASCIZ	<CRLF>/SYNC COMM/	
021530	040600	054523	041516	ACOM:	.ASCIZ	<CRLF>/ASYN COMM/	
021544	050200	044522	052116	PRINT:	.ASCIZ	<CRLF>/PRINTER/	
021555	040	047520	052122	PORT:	.ASCIZ	/ PORT TESTED/	
021572	042200	030130	000	DRVO:	.ASCIZ	<CRLF>/DX0/	
021577	200	054104	000061	DRV1:	.ASCIZ	<CRLF>/DX1/	
021604	043200	046111	020114	BUFTST:	.ASCIZ	<CRLF>/FILL & EMPTY BUFFER/	
021631	200	052522	047116	NOCON:	.ASCIZ	<CRLF>/RUNNING WITH NO CONSOLE IS SELECTED/	
021676	041600	047514	045503	CLOCK:	.ASCIZ	<CRLF>/CLOCK/	
021705	200	044505	026523	EISTXT:	.ASCIZ	<CRLF>/EIS-FIS OPTION/	
021725	113	046440	046505	MEM:	.ASCIZ	/K MEMORY PRESENT/	
021746	005200	051120	043517	DEFBD:	.ASCII	<CRLF><LF>/PROGRAM DEFAULTS (UNLESS MODIFIED):/	
022013	200	041411	052514		.ASCII	<CRLF>/ CLUSTER @ 2400 BAUD/	
022040	004600	051120	047111		.ASCII	<CRLF>/ PRINTER @ 9600 BAUD/	
022065	200	051411	047131		.ASCIZ	<CRLF>/ SYNC & ASYN COMM PORTS @ 9600 BAUD/	
022133	040	047516	020124	NOTPRES:	.ASCIZ	/ NOT PRESENT/	
022150	005200	047111	042523	WARNING:	.ASCII	<CRLF><LF>/INSERT SCRATCH DISKS, TYPE 'P' FOR NORMAL TESTING/	
022233	200	031047	030064		.ASCII	<CRLF>/'240G' FOR NORMAL RESTARTS/	
022266	023600	032462	043460		.ASCII	<CRLF>/'250G' TO COPY SYS EXERCISER DISK/	
022330	023600	033062	043460		.ASCII	<CRLF>/'260G' FOR COMPATABILITY PASS 1: WRITE/	
022377	200	031047	030067		.ASCIZ	<CRLF>/'270G' FOR COMPATABILITY PASS 2: READ/	
022446	052040	051505	044524	DROP:	.ASCIZ	/ TESTING DROPPED/	
022467	040	052522	047116	RUN:	.ASCIZ	/ RUNNING/	
022500	047440	000113		OK:	.ASCIZ	/ OK/	
022504	052040	045522	000	TRK:	.ASCIZ	/ TRK/	
022511	040	040504	040524	PATT:	.ASCIZ	/ DATA PATT/	
022524	051040	047101	047504	RANDOM:	.ASCIZ	/ RANDOM SEEKS/	
022542	042600	051511	043055	FISEIS:	.ASCIZ	<CRLF>/EIS-FIS TESTS/	
022561	200	042515	020115	MEMORY:	.ASCIZ	<CRLF>/MEM TESTS/	
022574	042040	047117	000105	DUN:	.ASCIZ	/ DONE/	
022602	005200	040520	051523	PASS:	.ASCIZ	<CRLF><LF>/PASS # /	
022614	051600	051105	040511	SERIAL:	.ASCIZ	<CRLF>/SERIAL NO. = /	
022633	200	042412	042116	ENDPAS:	.ASCIZ	<CRLF><LF>/END PASS #/	
022650	052011	052117	046101	MSG1:	.ASCIZ	/ TOTAL ERRORS: /	
022677	200	004411	052011	MSG2:	.ASCIZ	<CRLF>/	TOTAL SOFT ERRORS: /
022733	200	004411	052011	MSG3:	.ASCIZ	<CRLF>/	TOTAL ERRORS THIS PASS:/
022767	200	004411	051411	MSG4:	.ASCIZ	<CRLF>/	SOFT ERRORS THIS PASS:/
023023	200	047503	054520	COPMSG:	.ASCII	<CRLF>/COPY DX0 TO DX1/	
023043	200	047111	042523		.ASCIZ	<CRLF>/INSERT SCRATCH DISK IN DXi, TYPE 'P' TO PROCEED/	
023124	042200	047117	027105	COPDUN:	.ASCIZ	<CRLF>/DONE...TYPE 'P' TO DO AGAIN OR '240G' FOR NORMAL TESTING/	
023216	005200	047503	050115	COMPAT:	.ASCII	<CRLF><LF>/COMPATABILITY PASS 1 (WRITE) DONE/	
023261	200	047504	023440		.ASCIZ	<CRLF>/DO 'P' FOR PASS 2 (READ)/	

B 9

023313	200	042504	046526	DEVM:	.ASCIZ	<CRLF>/DEVM = /
023324	042515	047515	054522	EM1:	.ASCIZ	/MEMORY TIMEOUT/
023343	115	046505	042040	EM2:	.ASCIZ	/MEM DATA COMP ERR/
023365	123	047131	020103	EM3:	.ASCIZ	/SYNC COMM DID NOT RECEIVE SYNC CHAR/
023431	120	044522	052116	EM4:	.ASCIZ	/PRINTER STATUS ERR/
023454	042524	046522	021440	EM5:	.ASCIZ	/TERM #1 DATA COMP ERR/
023502	042524	046522	021440	EM6:	.ASCIZ	/TERM #2 DATA COMP ERR/
023530	042524	046522	021440	EM7:	.ASCIZ	/TERM #3 DATA COMP ERR/
023556	051501	047131	020103	EM8:	.ASCIZ	/ASync COMM DATA COMP ERR/
023607	123	047131	020103	EM9:	.ASCIZ	/SYNC COMM DATA COMP ERR/
023637	125	042516	050130	EM10:	.ASCIZ	/UNEXP DISK INTERRUPT/
023664	054104	020060	040510	EM11:	.ASCIZ	/DX0 HARD ERR - WRITE SECT/
023716	054104	020060	047523	EM12:	.ASCIZ	/DX0 SOFT ERR - WRITE SECT/
023750	054104	020060	040510	EM13:	.ASCIZ	/DX0 HARD ERR - READ SECT/
024001	104	030130	051440	EM14:	.ASCIZ	/DX0 SOFT ERR - READ SECT/
024032	054104	020060	047125	EM15:	.ASCIZ	/DX0 UNRECOV ERR - DATA COMP/
024066	054104	020060	051105	EM16:	.ASCIZ	/DX0 ERR - DATA COMP/
024112	054104	020061	040510	EM17:	.ASCIZ	/DX1 HARD ERR - WRITE SECT/
024144	054104	020061	047523	EM18:	.ASCIZ	/DX1 SOFT ERR - WRITE SECT/
024176	054104	020061	040510	EM19:	.ASCIZ	/DX1 HARD ERR - READ SECT/
024227	104	030530	051440	EM20:	.ASCIZ	/DX1 SOFT ERR - READ SECT/
024260	054104	020061	047125	EM21:	.ASCIZ	/DX1 UNRECOV ERR - DATA COMP/
024314	054104	020061	051105	EM22:	.ASCIZ	/DX1 ERR - DATA COMP/
024340	046103	020113	052510	EM23:	.ASCIZ	/CLK HUNG/
024351	120	044522	052116	EM24:	.ASCIZ	/PRINTER HUNG/
024366	042524	046522	021440	EM25:	.ASCIZ	/TERM #1 HUNG/
024403	124	051105	020115	EM26:	.ASCIZ	/TERM #2 HUNG/
024420	042524	046522	021440	EM27:	.ASCIZ	/TERM #3 HUNG/
024435	123	047131	020103	EM28:	.ASCIZ	/SYNC COMM HUNG/
024454	051501	047131	020103	EM29:	.ASCIZ	/ASync COMM HUNG/
024474	054104	020060	052510	EM30:	.ASCIZ	/DX0 HUNG/
024505	104	030530	044040	EM31:	.ASCIZ	/DX1 HUNG/
024516	054523	041516	041440	EM32:	.ASCIZ	/SYNC COMM - DATA NOT AVAIL NOT SET/
024561	123	047131	020103	EM33:	.ASCIZ	/SYNC COMM - RECVR ACTIVE NOT SET/
024622	054523	041516	041440	EM34:	.ASCIZ	/SYNC COMM - RECVR DONE NOT SET/
024661	104	030130	023440	EM35:	.ASCIZ	/DX0 'DEL DATA' BIT 5 NOT SET IN RXES/
024726	054104	020060	047111	EM36:	.ASCIZ	/DX0 INIT CMD DID NOT READ TRK 1, SEC 1/
024775	104	030130	051040	EM37:	.ASCIZ	/DX0 RESTORE CMD ERR/
025021	104	030130	023440	EM38:	.ASCIZ	/DX0 'INIT DONE' BIT 0 NOT SET IN RXES/
025067	104	030130	051040	EM39:	.ASCIZ	/DX0 READ STATUS CMD ERR/
025117	104	030130	044440	EM40:	.ASCIZ	/DX0 INV ADDR BIT 1 NOT SET IN RXES/
025162	054104	020060	051105	EM41:	.ASCIZ	/DX0 ERR BIT NOT SET IN RXCS/
025216	054104	020060	047111	EM42:	.ASCIZ	/DX0 INIT CMD ERR/
025237	104	030130	052440	EM43:	.ASCIZ	/DX0 UNRECOV SEEK ERR - WRITE SECT/
025301	104	030130	051040	EM44:	.ASCIZ	/DX0 RECOV SEEK ERR - WRITE SECT/
025341	104	030130	052440	EM45:	.ASCIZ	/DX0 UNRECOV SEEK ERR - READ SECT/
025402	054104	020060	042522	EM46:	.ASCIZ	/DX0 RECOV SEEK ERR - READ SECT/
025441	104	030530	052440	EM47:	.ASCIZ	/DX1 UNRECOV SEEK ERR - WRITE SECT/
025503	104	030530	051040	EM48:	.ASCIZ	/DX1 RECOV SEEK ERR - WRITE SECT/
025543	104	030530	052440	EM49:	.ASCIZ	/DX1 UNRECOV SEEK ERR - READ SECT/
025604	054104	020061	042522	EM50:	.ASCIZ	/DX1 RECOV SEEK ERR - READ SECT/
025643	104	030530	051040	EM51:	.ASCIZ	/DX1 RESTORE CMD ERR/
025667	104	051511	020113	EM52:	.ASCIZ	/DISK 'DONE' NOT SET/
025713	106	046111	020114	EM53:	.ASCIZ	/FILL & EMPTY BUFF TEST... DATA MISCOMP/
025762	040504	040524	046440	EM54:	.ASCIZ	/DATA MISCOMP/
025777	122	050105	040514	EM55:	.ASCIZ	/REPLACE DX0...10 BAD SECTORS/

026034	042522	046120	041501	EM56:	.ASCIZ	/REPLACE DX1...10 BAD SECTORS/													
026071	104	030130	044040	EM57:	.ASCIZ	/DX0 HARD CRC ERR - READ SECT/													
026126	054104	020060	051120	EM58:	.ASCIZ	/DX0 PREV CRC ERR WITH DATA WRONG/													
026167	104	030130	050040	EM59:	.ASCIZ	/DX0 PREV CRC ERR WITH DATA OK/													
026225	104	030530	044040	EM60:	.ASCIZ	/DX1 HARD CRC ERR - READ SECT/													
026262	054104	020061	051120	EM61:	.ASCIZ	/DX1 PREV CRC ERR WITH DATA WRONG/													
026323	104	030530	050040	EM62:	.ASCIZ	/DX1 PREV CRC ERR WITH DATA OK/													
026361	105	051511	043055	EM63:	.ASCIZ	/EIS-FIS CONDITION CODE ERROR/													
026416	044505	026523	044506	EM64:	.ASCIZ	/EIS-FIS DATA ERROR/													

026441	105	051122	051117	DH1:	.ASCIZ	/ERROR #	ERR PC/												
026460	051105	047522	020122	DH2:	.ASCIZ	/ERROR #	ERR PC ADDR/												
026504	051105	047522	020122	DH3:	.ASCIZ	/ERROR #	ERR PC ADDR EXPECT RECVD/												
026546	051105	047522	020122	DH4:	.ASCIZ	/ERROR #	ERR PC PR STATUS/												
026577	105	051122	051117	DH5:	.ASCIZ	/ERROR #	ERR PC EXPECT RECVD/												
026634	051105	047522	020122	DH6:	.ASCIZ	/ERROR #	DXTST# ERR PC RXCS RXES RXSA/												
026701	105	051122	051117	DH7:	.ASCIZ	/ERROR #	DXTST# ERR PC RXCS RXES TRACK SECTOR # RETRIE												
026770	051105	047522	020122	DH8:	.ASCIZ	/ERROR #	DXTST# ERR PC RXCS RXSA EXPECT RECVD # RETRIES												

027062 .EVEN

027062	031430	001116	000000	DT1:	.WORD	ERRNUM,\$ERRPC,0													
027070	031430	001116	005052	DT2:	.WORD	ERRNUM,\$ERRPC,ADDR,0													
027100	031430	001116	005052	DT3:	.WORD	ERRNUM,\$ERRPC,ADDR,PAT,MEMHLD,0													
027114	031430	001116	014334	DT4:	.WORD	ERRNUM,\$ERRPC,SPRS,0													
027124	031430	001116	014432	DT5:	.WORD	ERRNUM,\$ERRPC,T1CHR,T1HLD,0													
027136	031430	001116	014532	DT6:	.WORD	ERRNUM,\$ERRPC,T2CHR,T2HLD,0													
027150	031430	001116	014632	DT7:	.WORD	ERRNUM,\$ERRPC,T3CHR,T3HLD,0													
027162	031430	001116	014732	DT8:	.WORD	ERRNUM,\$ERRPC,COMCHR,COMHLD,0													
027174	031430	003476	001116	DT9:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,SRXSA,0													
027212	031430	003476	001116	DT10:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,DOTRK,DOSEC,DOERR,0													
027234	031430	003476	001116	DT11:	.WORD	ERRNUM,DXTST,\$ERRPC,RXCS,RXSA,DOEXP,DOREC,DOCERR,0													
027256	031430	003476	001116	DT12:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,D1TRK,D1SEC,D1ERR,0													
027300	031430	003476	001116	DT13:	.WORD	ERRNUM,DXTST,\$ERRPC,RXCS,RXSA,D1EXP,D1REC,D1CERR,0													
027322	031430	003476	001116	DT14:	.WORD	ERRNUM,DXTST,\$ERRPC,RXCS,RXES,RXSA,0													
027340	031430	001116	007744	DT15:	.WORD	ERRNUM,\$ERRPC,EXPO,EBHLD,0													
027352	031430	001116	007746	DT16:	.WORD	ERRNUM,\$ERRPC,EXP1,EBHLD,0													

027364	000000			DF:	.WORD	0													
027366	000000				.WORD	0													
027370	000000				.WORD	0													

027372	000000			DF1:	.WORD	0													
027374	000000				.WORD	0													
027376	000400				.WORD	400													
027400	000401				.WORD	401													

027402	000000			DF2:	.WORD	0													
027404	000000				.WORD	0													
027406	000000				.WORD	0													
027410	000400				.WORD	400													

.LIST BEX

4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790

027412 105737 001157
027416 100002
027420 000000
027422 000430
027424 010046
027426 017600 000002
027432 122737 000001 001210
027440 001011
027442 132737 000100 001211
027450 001405
027452 010037 027462
027456 004737 030544
027462 000000
027464 132737 000040 001211
027472 001003
027474 112046
027476 001005
027500 005726
027502 012600
027504 062716 000002
027510 000002
027512 122716 000011
027516 001430
027520 122716 000200
027524 001006
027526 005726
027530 104401
027532 001165
027534 105037 027670
027540 000755
027542 004737 027624
027546 123726 001156
027552 001350
027554 013746 001154
027560 105366 000001
027564 002770
027566 004737 027624
027572 105337 027670

```
.SBTTL TYPE ROUTINE
*****
*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*
*CALL:
*1) USING A TRAP INSTRUCTION
* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
*OR
* TYPE
* MESADR
*
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES
HALT ;;HALT HERE IF NO TERMINAL
BR 3$ ;;LEAVE
1$: MOV R0,-(SP) ;;SAVE R0
MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
61$: .WORD 0 ;;MESSAGE ADDRESS
62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
BNE 60$ ;;YES,SKIP TYPE OUT
MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,R0 ;;RESTORE R0
3$: ADD #2,(SP) ;;ADJUST RETURN PC
RTI ;;RETURN
4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
4780 CLR B $CHARCNT ;;CLEAR CHARACTER COUNT
BR 2$ ;;GET NEXT CHARACTER
4782 JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
4783 CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;;IF NO GO GET NEXT CHAR.
4785 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
4787 DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
4788 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
4789 JSR PC,$TYPEC ;;GO TYPE A NULL
4790 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
```

```

4791 027576 000770          BR      7$          ;;LOOP
4792
4793          ;HORIZONTAL TAB PROCESSOR
4794
4795 027600 112716 000040    8$:   MOVB   #' (SP)          ;;REPLACE TAB WITH SPACE
4796 027604 004737 027624    9$:   JSR    PC,$TYPEC        ;;TYPE A SPACE
4797 027610 132737 000007 027670    BITB  #7,$CHARCNT          ;;BRANCH IF NOT AT
4798 027616 001372          BNE    9$                  ;;TAB STOP
4799 027620 005726          TST    (SP)+              ;;POP SPACE OFF STACK
4800 027622 000724          BR     2$                  ;;GET NEXT CHARACTER
4801 027624 105777 151320    $TYPEC: TSTB  @$TPS          ;;WAIT UNTIL PRINTER IS READY
4802 027630 100375          BPL   $TYPEC
4803 027632 116677 000002 151312    MOVB  2(SP),@$TPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4804 027640 122766 000015 000002    CMPB  #CR,2(SP)           ;;IS CHARACTER A CARRIAGE RETURN?
4805 027646 001003          BNE   1$                  ;;BRANCH IF NO
4806 027650 105037 027670    CLRB  $CHARCNT           ;;YES--CLEAR CHARACTER COUNT
4807 027654 000406          BR    $TYPEX             ;;EXIT
4808 027656 122766 000012 000002    1$:   CMPB  #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
4809 027664 001402          BEQ  $TYPEX             ;;BRANCH IF YES
4810 027666 105227          INCB  (PC)+             ;;COUNT THE CHARACTER
4811 027670 000000    $CHARCNT: .WORD 0          ;;CHARACTER COUNT STORAGE
4812 027672 000207    $TYPEX: RTS    PC
4813

```

```

4814      .SBTTL  TTY INPUT ROUTINE
4815
4816      ::*****
4817      .ENABL  LSB
4818
4819      ::*****
4820      *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4821      *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4822      *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4823      *WHEN OPERATING IN TTY FLAG MODE.
4824 027674 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
4825 027702 001114          BNE      15$          ;;BRANCH IF NO
4826 027704 105777 151234          TSTB     @STKS          ;;CHAR THERE?
4827 027710 100111          BPL      15$          ;;IF NO, DON'T WAIT AROUND
4828 027712 117746 151230          MOVB     @STKB,-(SP)      ;;SAVE THE CHAR
4829 027716 042716 177600          BIC     #^C177,(SP)    ;;STRIP-OFF THE ASCII
4830 027722 022726 000007          CMP     #7,(SP)+      ;;IS IT A CONTROL G?
4831 027726 001102          BNE     15$          ;;NO, RETURN TO USER
4832 027730 123727 001134 000001          CMPB    $AUTOB,#1     ;;ARE WE RUNNING IN AUTO-MODE?
4833 027736 001476          BEQ     15$          ;;BRANCH IF YES
4834
4835 027740 104401 030506          TYPE    ,%CNTLG      ;;ECHO THE CONTROL-G (^G)
4836 027744 104401 030513          $GTSWR: TYPE    ,%MSWR      ;;TYPE CURRENT CONTENTS
4837 027750 013746 000176          MOV     SWREG,-(SP)   ;;SAVE SWREG FOR TYPEOUT
4838 027754 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4839 027756 104401 030524          TYPE    ,%MNEW      ;;PROMPT FOR NEW SWR
4840 027762 005046          19$:  CLR     -(SP)    ;;CLEAR COUNTER
4841 027764 005046          CLR     -(SP)    ;;THE NEW SWR
4842 027766 105777 151152          7$:  TSTB     @STKS          ;;CHAR THERE?
4843 027772 100375          BPL     7$        ;;IF NOT TRY AGAIN
4844
4845 027774 117746 151146          MOVB     @STKB,-(SP)  ;;PICK UP CHAR
4846 030000 042716 177600          BIC     #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
4847
4848 030004 021627 000003          CMP     (SP),#3      ;;IS IT A CONTROL-C?
4849 030010 001015          BNE     9$          ;;BRANCH IF NOT
4850 030012 104401 030474          TYPE    ,%CNTLC     ;;YES, ECHO CONTROL-C (^C)
4851 030016 062706 000006          ADD     #6,SP        ;;CLEAN UP STACK
4852 030022 123727 001135 000001          CMPB    $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
4853 030030 001003          BNE     8$          ;;BRANCH IF NO
4854 030032 012777 000100 151104          MOV     #100,@STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
4855 030040 000137 013126          8$:  JMP     GT$DEV     ;;CONTROL-C RESTART
4856
4857
4858 030044 021627 000025          9$:  CMP     (SP),#25   ;;IS IT A CONTROL-U?
4859 030050 001005          BNE     10$         ;;BRANCH IF NOT
4860 030052 104401 030501          TYPE    ,%CNTLU     ;;YES, ECHO CONTROL-U (^U)
4861 030056 062706 000006          20$:  ADD     #6,SP        ;;IGNORE PREVIOUS INPUT
4862 030062 000737          BR      19$         ;;LET'S TRY IT AGAIN
4863
4864
4865 030064 021627 000015          10$:  CMP     (SP),#15   ;;IS IT A <CR>?
4866 030070 001022          BNE     16$         ;;BRANCH IF NO
4867 030072 005766 000004          TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
4868 030076 001403          BEQ     11$         ;;BRANCH IF YES
4869 030100 016677 000002 151032          MOV     2(SP),@SWR  ;;SAVE NEW SWR

```

```

4870 030106 062706 000006      11$:  ADD      #6,SP      ;;CLEAR UP STACK
4871 030112 104401 001165      14$:  TYPE     $CRLF     ;;ECHO <CR> AND <LF>
4872 030116 123727 001135 000001  CMPB    $INTAG,#1     ;;RE-ENABLE TTY KBD INTERRUPTS?
4873 030124 001003                BNE     15$           ;;BRANCH IF NOT
4874 030126 012777 000100 151010  MOV     #100,@$TKS    ;;RE-ENABLE TTY KBD INTERRUPTS
4875 030134 000002                RTI                    ;;RETURN
4876 030136 004737 027624      15$:  JSR     PC,$TYPEC    ;;ECHO CHAR
4877 030142 021627 000060      16$:  CMP     (SP),#60     ;;CHAR < 0?
4878 030146 002420                BLT     18$           ;;BRANCH IF YES
4879 030150 021627 000067      CMP     (SP),#67     ;;CHAR > 7?
4880 030154 003015                BGT     18$           ;;BRANCH IF YES
4881 030156 042726 000060      BIC     #60,(SP)+    ;;STRIP-OFF ASCII
4882 030162 005766 000002      TST     2(SP)        ;;IS THIS THE FIRST CHAR
4883 030166 001403                BEQ     17$           ;;BRANCH IF YES
4884 030170 006316                ASL     (SP)         ;;NO, SHIFT PRESENT
4885 030172 006316                ASL     (SP)         ;;CHAR OVER TO MAKE
4886 030174 006316                ASL     (SP)         ;;ROOM FOR NEW ONE.
4887 030176 005266 000002      17$:  INC     2(SP)        ;;KEEP COUNT OF CHAR
4888 030202 056616 177776      BIS     -2(SP),(SP)  ;;SET IN NEW CHAR
4889 030206 000667                BR      7$           ;;GET THE NEXT ONE
4890 030210 104401 001164      18$:  TYPE     $QUES     ;;TYPE ?<CR><LF>
4891 030214 000720                BR      20$         ;;SIMULATE CONTROL-U
4892                .DSABL  LSB

```

```

4893
4894
4895
4896                ;;*****
4897                ;;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4898                ;;CALL:
4899                ;;*      RDCHR                ;;INPUT A SINGLE CHARACTER FROM THE TTY
4900                ;;*      RETURN HERE         ;;CHARACTER IS ON THE STACK
4901                ;;*                          ;;WITH PARITY BIT STRIPPED OFF
4902

```

```

4903 030216 011646                $RDCHR: MOV     (SP),-(SP)  ;;PUSH DOWN THE PC
4904 030220 016666 000004 000002  MOV     4(SP),2(SP)  ;;SAVE THE PS
4905 030226 105777 150712      1$:  TSTB    @$TKS       ;;WAIT FOR
4906 030232 100375                BPL     1$           ;;A CHARACTER
4907 030234 117766 150706 000004  MOVB    @$TKB,4(SP)  ;;READ THE TTY
4908 030242 042766 177600 000004  BIC     #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
4909 030250 026627 000004 000023  CMP     4(SP),#23   ;;IS IT A CONTROL-S?
4910 030256 001013                BNE     3$           ;;BRANCH IF NO
4911 030260 105777 150660      2$:  TSTB    @$TKS       ;;WAIT FOR A CHARACTER
4912 030264 100375                BPL     2$           ;;LOOP UNTIL ITS THERE
4913 030266 117746 150654  MOVB    @$TKB,-(SP)  ;;GET CHARACTER
4914 030272 042716 177600      BIC     #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
4915 030276 022627 000021  CMP     (SP)+,#21   ;;IS IT A CONTROL-Q?
4916 030302 001366                BNE     2$           ;;IF NOT DISCARD IT
4917 030304 000750                BR      1$           ;;YES, RESUME
4918 030306 026627 000004 000140  3$:  CMP     4(SP),#140  ;;IS IT UPPER CASE?
4919 030314 002407                BLT     4$           ;;BRANCH IF YES
4920 030316 026627 000004 000175  CMP     4(SP),#175  ;;IS IT A SPECIAL CHAR?
4921 030324 003003                BGT     4$           ;;BRANCH IF YES
4922 030326 042766 000040 000004  BIC     #40,4(SP)   ;;MAKE IT UPPER CASE
4923 030334 000002      4$:  RTI                    ;;GO BACK TO USER

```

```

4924                ;;*****
4925                ;;THIS ROUTINE WILL INPUT A STRING FROM THE TTY

```

```

4926                                     ;*CALL:
4927                                     ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
4928                                     ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4929                                     ;*                                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4930
4931 030336 010346 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
4932 030340 012703 030464 1$:      MOV      #$TTYIN,R3    ;;GET ADDRESS
4933 030344 022703 030474 2$:      CMP      #$TTYIN+8.,R3  ;;BUFFER FULL?
4934 030350 101415      BLOS     4$          ;;BR IF YES
4935 030352 104410      RDCHR     ;;GO READ ONE CHARACTER FROM THE TTY
4936 030354 112613      MOVB     (SP)+,(R3)  ;;GET CHARACTER
4937 030356 122713 000003      CMPB     #3,(R3)    ;;IS IT A CONTROL-C?
4938 030362 001005      BNE     10$         ;;BRANCH IF NO
4939 030364 104401 030474      TYPE     ,%CNTLC    ;;TYPE A CONTROL-C (^C)
4940 030370 012603      MOV      (SP)+,R3    ;;RESTORE R3
4941 030372 000137 013126      JMP      GT$DEV     ;;GOTO CONTROL-C RESTART
4942 030376 122713 000177 10$:    CMPB     #177,(R3)  ;;IS IT A RUBOUT
4943 030402 001003      BNE     3$          ;;SKIP IF NOT
4944 030404 104401 001164 4$:      TYPE     ,%QUES    ;;TYPE A '?'
4945 030410 000753      BR      1$         ;;CLEAR THE BUFFER AND LOOP
4946 030412 111337 030462 3$:      MOVB     (R3),9$    ;;ECHO THE CHARACTER
4947 030416 104401 030462      TYPE     ,9$
4948 030422 122723 000015      CMPB     #15,(R3)+ ;;CHECK FOR RETURN
4949 030426 001346      BNE     2$         ;;LOOP IF NOT RETURN
4950 030430 105063 177777      CLRB     -1(R3)    ;;CLEAR RETURN (THE 15)
4951 030434 104401 001166      TYPE     ,%LF     ;;TYPE A LINE FEED
4952 030440 012603      MOV      (SP)+,R3  ;;RESTORE R3
4953 030442 011646      MOV      (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
4954 030444 016666 000004 000002  MOV      4(SP),2(SP) ;;FIRST ASCII CHARACTER ON IT
4955 030452 012766 030464 000004  MOV      #$TTYIN,4(SP)
4956 030460 000002      RTI              ;;RETURN
4957 030462 000      9$:      .BYTE   0          ;;STORAGE FOR ASCII CHAR. TO TYPE
4958 030463 000      .BYTE   0          ;;TERMINATOR
4959 030464 000010 $TTYIN: .BLKB   8.    ;;RESERVE 8 BYTES FOR TTY INPUT
4960 030474 041536 005015 000 $CNTLC: .ASCIZ  / ^C / <15> <12> ;;CONTROL 'C'
4961 030501 0136 006525 000012 $CNTLU: .ASCIZ  / ^U / <15> <12> ;;CONTROL 'U'
4962 030506 043536 005015 000 $CNTLG: .ASCIZ  / ^G / <15> <12> ;;CONTROL 'G'
4963 030513 015 051412 051127 $MSWR: .ASCIZ  <15> <12> / SWR = /
4964 030520 036440 000040 $MNEW: .ASCIZ  / NEW = /
4965 030524 020040 042516 020127
4966 030532 020075 000
4967 030536 .EVEN

```



```

4968 .SBTTL APT COMMUNICATIONS ROUTINE
4969
4970 *****
4971 030536 112737 000001 031002 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
4972 030544 112737 000001 031000 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
4973 030552 000403                BR      $ATYC
4974 030554 112737 000001 031002 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
4975 030562 $ATYC:
4976 030562 010046                MOV     RO,-(SP)      ;;PUSH RO ON STACK
4977 030564 010146                MOV     R1,-(SP)      ;;PUSH R1 ON STACK
4978 030566 105737 031000                TSTB   $MFLG         ;;SHOULD TYPE A MESSAGE?
4979 030572 001450                BEQ    5$            ;;IF NOT: BR
4980 030574 122737 000001 001210                CMPB   #APTENV,$ENV  ;;OPERATING UNDER APT?
4981 030602 001031                BNE    3$            ;;IF NOT: BR
4982 030604 132737 000100 001211                BITB   #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4983 030612 001425                BEQ    3$            ;;IF NOT: BR
4984 030614 017600 000004                MOV     @4(SP),RO     ;;GET MESSAGE ADDR.
4985 030620 062766 000002 000004                ADD     #2,4(SP)      ;;BUMP RETURN ADDR.
4986 030626 005737 001170                1$:    TST     $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
4987 030632 001375                BNE    1$            ;;IF NOT: WAIT
4988 030634 010037 001204                MOV     RO,$MSGAD     ;;PUT ADDR IN MAILBOX
4989 030640 105720                2$:    TSTB   (RO)+      ;;FIND END OF MESSAGE
4990 030642 001376                BNE    2$
4991 030644 163700 001204                SUB     $MSGAD,RO     ;;SUB START OF MESSAGE
4992 030650 006200                ASR    RO            ;;GET MESSAGE LNTH IN WORDS
4993 030652 010037 001206                MOV     RO,$MSGGLT    ;;PUT LENGTH IN MAILBOX
4994 030656 012737 000004 001170                MOV     #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
4995 030664 000413                BR     5$
4996 030666 017637 000004 030712 3$:    MOV     @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
4997 030674 062766 000002 000004                ADD     #2,4(SP)      ;;BUMP RETURN ADDRESS
4998 030702 013746 177776                MOV     177776,-(SP) ;;PUSH 177776 ON STACK
4999 030706 004737 027412                JSR    PC,$TYPE      ;;CALL TYPE MACRO
5000 030712 000000                4$:    .WORD  0
5001 030714                5$:
5002 030714 105737 031002                10$:   TSTB   $FFLG         ;;SHOULD REPORT FATAL ERROR?
5003 030720 001416                BEQ    12$          ;;IF NOT: BR
5004 030722 005737 001210                TST    $ENV         ;;RUNNING UNDER APT?
5005 030726 001413                BEQ    12$          ;;IF NOT: BR
5006 030730 005737 001170                11$:   TST     $MSGTYPE   ;;FINISHED LAST MESSAGE?
5007 030734 001375                BNE    11$          ;;IF NOT: WAIT
5008 030736 017637 000004 001172                MOV     @4(SP),$FATAL ;;GET ERROR #
5009 030744 062766 000002 000004                ADD     #2,4(SP)      ;;BUMP RETURN ADDR.
5010 030752 005237 001170                INC     $MSGTYPE     ;;TELL APT TO TAKE ERROR
5011 030756 105037 031002                12$:   CLRB   $FFLG         ;;CLEAR FATAL FLAG
5012 030762 105037 031001                CLRB   $LFLG        ;;CLEAR LOG FLAG
5013 030766 105037 031000                CLRB   $MFLG        ;;CLEAR MESSAGE FLAG
5014 030772 012601                MOV     (SP)+,R1     ;;POP STACK INTO R1
5015 030774 012600                MOV     (SP)+,RO     ;;POP STACK INTO RO
5016 030776 000207                RTS     PC           ;;RETURN
5017 031000 000                $MFLG: .BYTE 0      ;;MESSG. FLAG
5018 031001 000                $LFLG: .BYTE 0      ;;LOG FLAG
5019 031002 000                $FFLG: .BYTE 0      ;;FATAL FLAG
5020 031004                .EVEN
5021 000200                APTSIZE=200
5022 000001                APTENV=001
5023 000100                APTSPOOL=100

```

```

5024      000040
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038 031004
5039 031004 104407
5040 031006 105237 001103
5041 031012 001775
5042 031014 013777 001102 150120
5043 031022 032777 002000 150110
5044 031030 001402
5045 031032 104401 001160
5046 031036 005237 001112
5047 031042 011637 001116
5048 031046 162737 000002 001116
5049 031054 117737 150036 001114
5050 031062 032777 020000 150050
5051 031070 001004
5052 031072 004737 031320
5053 031076 104401 001165
5054 031102
5055 031102 122737 000001 001210
5056 031110 001007
5057 031112 113737 001114 031124
5058 031120 004737 030554
5059 031124 000
5060 031125 000
5061 031126 000777
5062 031130 005777 150004
5063 031134 100002
5064 031136 000000
5065 031140 104407
5066 031142
5067 031142 000002

```

```

APTCSUP=040
.SBTTL  ERROR HANDLER ROUTINE

;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;AND GO TO MYTYPE ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1      HALT ON ERROR
;*SW13=1      INHIBIT ERROR TYPEOUTS
;*SW10=1      BELL ON ERROR
;*CALL
;*          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
7$:      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
        INCB          $ERFLG      ;;SET THE ERROR FLAG
        BEQ          7$          ;;DON'T LET THE FLAG GO TO ZERO
        MOV          $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
        BIT          #BIT10,@SWR      ;;BELL ON ERROR?
        BEQ          1$          ;;NO - SKIP
        TYPE        ,SBELL          ;;RING BELL
1$:      INC          $ERTTL          ;;COUNT THE NUMBER OF ERRORS
        MOV          (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
        SUB          #2,$ERRPC
        MOVB         @ $ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
        BIT          #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
        BNE          20$          ;;SKIP TYPEOUTS
        JSR          PC,MYTYPE        ;;GO TO USER ERROR ROUTINE
        TYPE        ,SRLF

20$:     CMPB         #APTENV,$ENV      ;;RUNNING IN APT MODE
        BNE          2$          ;;NO,SKIP APT ERROR REPORT
        MOVB         $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
        JSR          PC,$ATY4        ;;REPORT FATAL ERROR TO APT
21$:     .BYTE        0
        .BYTE        0
22$:     BR          22$          ;;APT ERROR LOOP
2$:      TST          @SWR          ;;HALT ON ERROR
        BPL          3$          ;;SKIP IF CONTINUE
        HALT         ;;HALT ON ERROR!
        CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
3$:      RTI          ;;RETURN

```

```

5068
5069
5070
5071
5072
5073
5074
5075 031144
5076 031144 104401 001165
5077 031150 010046
5078 031152 005000
5079 031154 153700 001114
5080 031160 001004
5081
5082 031162 013746 001116
5083
5084 031166 104402
5085 031170 000445
5086 031172 005300
5087 031174 006300
5088 031176 006300
5089 031200 006300
5090 031202 062700 001250
5091 031206 012037 031216
5092 031212 001404
5093 031214 104401
5094 031216 000000
5095 031220 104401 001165
5096 031224 012037 031234
5097 031230 001404
5098 031232 104401
5099 031234 000000
5100 031236 104401 001165
5101 031242 010146
5102 031244 012001
5103 031246 001415
5104 031250 012000
5105 031252 105720
5106 031254 001003
5107 031256 013146
5108 031260 104402
5109 031262 000402
5110 031264
5111 031264 013146
5112 031266 104405
5113 031270 005711
5114 031272 001403
5115 031274 104401 031314
5116 031300 000764
5117
5118 031302 012601
5119 031304 012600
5120 031306 104401 001165
5121 031312 000207
5122 031314 020040 000
5123 031320

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

*****
: *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
: *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
: *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

```

$ERRTYP:
      TYPE      , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
      MOV      R0, -(SP)    ;; SAVE R0
      CLR      R0          ;; PICKUP THE ITEM INDEX
      BISB     @#$ITEMB, R0
      BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
                          ;; TYPE THE PC OF THE ERROR
      MOV     $LRRPC, -(SP) ;; SAVE $ERRPC FOR TYPEOUT
                          ;; ERROR ADDRESS
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      TYPDC   10$         ;; GET OUT
      BR     1$          ;; ADJUST THE INDEX SO THAT IT WILL
                          ;; WORK FOR THE ERROR TABLE
      ASL     R0
      ASL     R0
      ASL     R0
      ADD     #$ERRTB, R0  ;; FORM TABLE POINTER
      MOV     (R0)+, 2$   ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
      TYPE    'ERROR MESSAGE'
                          ;; TYPE THE "ERROR MESSAGE"
      TYPE    'ERROR MESSAGE' POINTER GOES HERE
      TYPE    , $CRLF     ;; 'CARRIAGE RETURN' & 'LINE FEED'
      MOV     (R0)+, 4$   ;; PICKUP "DATA HEADER" POINTER
      BEQ     5$          ;; SKIP TYPEOUT IF 0
      TYPE    'DATA HEADER'
                          ;; TYPE THE "DATA HEADER"
      TYPE    'DATA HEADER' POINTER GOES HERE
      MOV     R1, -(SP)   ;; 'CARRIAGE RETURN' & 'LINE FEED'
      MOV     (R0)+, R1  ;; SAVE R1
      BEQ     9$          ;; PICKUP "DATA TABLE" POINTER
      MOV     (R0)+, R0  ;; BR IF NO DATA TO BE TYPED
      TSTB   (R0)+      ;; PICKUP "DATA FORMAT" POINTER
      BNE     7$          ;; "OCTAL" OR "DECIMAL"
      MOV     @ (R1)+, -(SP) ;; BR IF DECIMAL
      TYPDC   8$          ;; SAVE @ (R1)+ FOR TYPEOUT
                          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      MOV     @ (R1)+, -(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
      TYPDC   9$          ;; GO TYPE--DECIMAL ASCII WITH SIGN
      TST     (R1)       ;; IS THERE ANOTHER NUMBER?
      BEQ     9$          ;; BR IF NO
      TYPE    , 11$      ;; TYPE TWO(2) SPACES
      BR     6$          ;; LOOP
      MOV     (SP)+, R1   ;; RESTORE R1
      MOV     (SP)+, R0   ;; RESTORE R0
      TYPE    , $CRLF     ;; 'CARRIAGE RETURN' & 'LINE FEED'
      RTS     PC          ;; RETURN
      .ASCIZ  / /        ;; TWO(2) SPACES
      .EVEN

```

```

5124
5125
5126
5127 031320 113737 001114 031430 MYTYPE: MOVB $ITEMB,ERRNUM ;FOR ERROR TYPEOUT
5128 031326 013737 015242 001174 MOV SRXSA,$TESTN ;FOR APT
5129 ;APT PRINTS $TESTN & $FATAL IN THAT ORDER.
5130 ;$TESTN WILL BE FLOPPY RXSA
5131 ;$FATAL IS ERR #
5132 031334 032737 000020 001246 BIT #BIT4,$DEVN ;RUNNING WITHOUT CONSOLE?
5133 031342 001407 BEQ 12$ ;BR IF NO
5134 031344 012737 040300 177420 MOV #<ULITE!300>,PARAM ;ELSE SET USER LIGHT 1 & 2
5135 031352 000000 HALT ;AND HALT....WAIT TO PROCEED
5136 031354 012737 040000 177420 MOV #<ULITE>,PARAM ;CLEAR LIGHTS
5137
5138 031362 104401 022602 12$: TYPE ,PASS ;TYPE PASS # BEFORE ERROR
5139 031366 013746 001176 MOV $PASS,-(SP) ;LOAD PASS #
5140 031372 005216 INC (SP) ;NO SUCH THING AS PASS 0 FOR OPERATOR
5141 031374 104405 TYPDS ;TYPE IT
5142
5143 031376 104401 022614 TYPE ,SERIAL ;TYPE SERIAL # BEFORE ERROR
5144 031402 104401 030464 TYPE ,STTYIN
5145 031406 104401 012102 TYPE ,NULL
5146
5147 031412 004737 031144 JSR PC,$ERRTYP ;TYPE ERROR MSG
5148 031416 005237 031432 INC TERR
5149 031422 005237 031436 INC PERR
5150 031426 000207 RTS PC
5151
5152 031430 000000 ERRNUM: 0 ;FOR ERR TYPEOUT
5153
5154 031432 000000 TERR: 0 ;TOTAL ERROR COUNT
5155 031434 000000 TSERR: 0 ;TOTAL SOFT ERR COUNT
5156 031436 000000 PERR: 0 ;PASS ERR COUNT
5157 031440 000000 PSERR: 0 ;PASS SOFT ERR COUNT... ALL FOR EOP TYPEOUT

```

```

5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170 031442
5171 031442 010046
5172 031444 010146
5173 031446 010246
5174 031450 010346
5175 031452 010546
5176 031454 012746 020200
5177 031460 016605 000020
5178 031464 100004
5179 031466 005405
5180 031470 112766 000055 000001
5181 031476 005000
5182 031500 012703 031656
5183 031504 112723 000040
5184 031510 005002
5185 031512 016001 031646
5186 031516 160105
5187 031520 002402
5188 031522 005202
5189 031524 000774
5190 031526 060105
5191 031530 005702
5192 031532 001002
5193 031534 105716
5194 031536 100407
5195 031540 106316
5196 031542 103003
5197 031544 116663 000001 177777
5198 031552 052702 000060
5199 031556 052702 000040
5200 031562 110223
5201 031564 005720
5202 031566 020027 000010
5203 031572 002746
5204 031574 003002
5205 031576 010502
5206 031600 000764
5207 031602 105726
5208 031604 100003
5209 031606 116663 177777 177776
5210 031614 105013
5211 031616 012605
5212 031620 012603
5213 031622 012602
    
```

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:
*
*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*      TYPDS    ;;GO TO THE ROUTINE

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2            ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
3$:      SUB      R1,R5        ;;FORM THIS BCD DIGIT
BLT      4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      ADD      R1,R5        ;;ADD BACK THE CONSTANT
TST      R2            ;;CHECK IF BCD DIGIT=0
BNE      5$            ;;FALL THROUGH IF 0
TSTB     (SP)          ;;STILL DOING LEADING 0'S?
BMI      7$            ;;BR IF YES
5$:      ASLB     (SP)        ;;MSD?
BCC      6$            ;;BR IF NO
MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
6$:      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+        ;;JUST INCREMENTING
CMP      R0,#10        ;;CHECK THE TABLE INDEX
BLT      2$            ;;GO DO THE NEXT DIGIT
BGT      8$            ;;GO TO EXIT
MOV      R5,R2        ;;GET THE LSD
BR       6$            ;;GO CHANGE TO ASCII
8$:      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
BPL      9$            ;;BR IF NO
MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB     (R3)        ;;SET THE TERMINATOR
MOV      (SP)+,R5      ;;POP STACK INTO R5
MOV      (SP)+,R3      ;;POP STACK INTO R3
MOV      (SP)+,R2      ;;POP STACK INTO R2
    
```

5214	031624	012601		
5215	031626	012600		
5216	031630	104401	031656	
5217	031634	016666	000002	000004
5218	031642	012616		
5219	031644	000002		
5220	031646	023420		
5221	031650	001750		
5222	031652	000144		
5223	031654	000012		
5224	031656	000004		
5225				
5226				
5227				
5228				
5229				
5230				
5231				
5232				
5233				
5234				
5235				
5236				
5237				
5238				
5239				
5240				
5241				
5242				
5243				
5244				
5245				
5246				
5247				
5248				
5249				
5250	031666	017646	000000	
5251	031672	116637	000001	032111
5252	031700	112637	032113	
5253	031704	062716	000002	
5254	031710	000406		
5255	031712	112737	000001	032111
5256	031720	112737	000006	032113
5257	031726	112737	000005	032110
5258	031734	010346		
5259	031736	010446		
5260	031740	010546		
5261	031742	113704	032113	
5262	031746	005404		
5263	031750	062704	000006	
5264	031754	110437	032112	
5265	031760	113704	032111	
5266	031764	016605	000012	
5267	031770	005003		
5268	031772	006105		
5269	031774	000404		

```

MOV      (SP)+,R1      ;;POP STACK INTO R1
MOV      (SP)+,R0      ;;POP STACK INTO R0
TYPE     $DBLK         ;;NOW TYPE THE NUMBER
MOV      2(SP),4(SP)   ;;ADJUST THE STACK
MOV      (SP)+,(SP)
RTI                      ;;RETURN TO USER

$DTBL:   10000.
         1000.
         100.
         10.

$DBLK:   .BLKW 4
$.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

:*****
:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
:OCTAL (ASCII) NUMBER AND TYPE IT.
:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPOS                    ;;CALL FOR TYPEOUT
:*      .BYTE   N                ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
:*      .BYTE   M                ;;M=1 OR 0
:*                                  ;;1=TYPE LEADING ZEROS
:*                                  ;;0=SUPPRESS LEADING ZEROS
:$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
:$TYPOS OR $TYPOC
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPON                    ;;CALL FOR TYPEOUT
:$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
:CALL:
:*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
:*      TYPOC                    ;;CALL FOR TYPEOUT

$TYPOS:  MOV      @ (SP),-(SP)   ;;PICKUP THE MODE
         MOV      1(SP),$OFILL  ;;LOAD ZERO FILL SWITCH
         MOV      (SP)+,$SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
         ADD      #2,(SP)       ;;ADJUST RETURN ADDRESS
         BR      $TYPON

$TYPOC:  MOV      #1,$OFILL     ;;SET THE ZERO FILL SWITCH
         MOV      #6,$SOMODE+1 ;;SET FOR SIX(6) DIGITS
$TYPON:  MOV      #5,$SOCNT     ;;SET THE ITERATION COUNT
         MOV      R3,-(SP)      ;;SAVE R3
         MOV      R4,-(SP)      ;;SAVE R4
         MOV      R5,-(SP)      ;;SAVE R5
         MOV      $SOMODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
         NEG      R4
         ADD      #6,R4        ;;SUBTRACT IT FOR MAX. ALLOWED
         MOV      R4,$SOMODE    ;;SAVE IT FOR USE
         MOV      $OFILL,R4    ;;GET THE ZERO FILL SWITCH
         MOV      12(SP),R5    ;;PICKUP THE INPUT NUMBER
         CLR      R3           ;;CLEAR THE OUTPUT WORD
         ROL      R5           ;;ROTATE MSB INTO "C"
         BR      3$           ;;GO DO MSB

```

5270	031776	006105		2\$:	ROL	R5	::FORM THIS DIGIT
5271	032000	006105			ROL	R5	
5272	032002	006105			ROL	R5	
5273	032004	010503			MOV	R5,R3	
5274	032006	006103		3\$:	ROL	R3	::GET LSB OF THIS DIGIT
5275	032010	105337	032112		DECB	\$OMODE	::TYPE THIS DIGIT?
5276	032014	100016			BPL	7\$::BR IF NO
5277	032016	042703	177770		BIC	#177770,R3	::GET RID OF JUNK
5278	032022	001002			BNE	4\$::TEST FOR 0
5279	032024	005704			TST	R4	::SUPPRESS THIS 0?
5280	032026	001403			BEQ	5\$::BR IF YES
5281	032030	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
5282	032032	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
5283	032036	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
5284	032042	110337	032106		MOV	R3,8\$::SAVE FOR TYPING
5285	032046	104401	032106		TYPE	8\$::GO TYPE THIS DIGIT
5286	032052	105337	032110	7\$:	DECB	\$OCNT	::COUNT BY 1
5287	032056	003347			BGT	2\$::BR IF MORE TO DO
5288	032060	002402			BLT	6\$::BR IF DONE
5289	032062	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
5290	032064	000744			BR	2\$::GO DO THE LAST DIGIT
5291	032066	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
5292	032070	012604			MOV	(SP)+,R4	::RESTORE R4
5293	032072	012603			MOV	(SP)+,R3	::RESTORE R3
5294	032074	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
5295	032102	012616			MOV	(SP)+,(SP)	
5296	032104	000002			RTI		::RETURN
5297	032106	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
5298	032107	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
5299	032110	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
5300	032111	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
5301	032112	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

5302
5303
5304
5305
5306
5307
5308
5309
5310 032114 010046
5311 032116 016600 000002
5312 032122 005740
5313 032124 111000
5314 032126 006300
5315 032130 016000 032150
5316 032134 000200
5317
5318
5319
5320
5321 032136 011646
5322 032140 016666 000004 000002
5323 032146 000002
5324
5325
5326
5327
5328
5329
5330
5331
5332 032150 032136
5333 032152 027412
5334 032154 031712
5335 032156 031666
5336 032160 031726
5337 032162 031442
5338
5339 032164 027744
5340
5341 032166 027674
5342 032170 030216
5343 032172 030336
5344 032174 013126
5345
5346 032176 000040
5347
5348 032276
5349
5350
5351 003574

.SBTTL TRAP DECODER

```

;*****
;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.

```

```

$TRAP:  MOV    R0,-(SP)      ;;SAVE R0
        MOV    2(SP),R0    ;;GET TRAP ADDRESS
        TST    -(R0)       ;;BACKUP BY 2
        MOVB   (R0),R0     ;;GET RIGHT BYTE OF TRAP
        ASL    R0          ;;POSITION FOR INDEXING
        MOV    $TRPAD(R0),R0 ;;INDEX TO TABLE
        RTS    R0          ;;GO TO ROUTINE

```

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

```

$TRAP2: MOV    (SP),-(SP)  ;;MOVE THE PC DOWN
        MOV    4(SP),2(SP) ;;MOVE THE PSW DOWN
        RTI                    ;;PESTORE THE PSW

```

.SBTTL TRAP TABLE

```

;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.

```

```

:      ROUTINE
:      -----
$TRPAD: .WORD   $TRAP2
        $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
        $TDEV   ;;CALL=GTDEVM   TRAP+12(104412) ^C WILL OPEN UP $DEV

        .BLKW   32.           ;STACK FOR "EIS-FIS" TESTS

LSTAD:  ;MAX ADDR FOR 8K WITH APT = 37400
        ;MAX ADDR FOR 8K NO APT BUT TO SAVE ABS LOADER = 37500

.END    START

```


CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

MACY11 30A(1052) 01-DEC-80 09:47 PAGE 130
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0128

MSG2	022677	2978	4734#																	
MSG3	022733	2985	4734#																	
MSG4	022767	2989	4734#																	
MULO	005436	2101	2107#																	
MYTYPE	031320	5052	5127#																	
NOCON	021631	3151	4734#																	
NOEIS	005760	1987	2188	2190	2193#															
NOTPRE	022133	3055	3077	3129	4734#															
NULL	012102	2992	3008#	5145																
NXTSEC	020532	3875	4068	4265	4448	4512#														
ODDPAR=	001000	1106#	2286																	
OK	022500	4734#																		
OPAR =	000020	1014#	2338	2340	3133															
ORERR =	040000	1050#	1086#																	
PARAM =	177420	937#	2230*	2279*	2341*	2390*	2428*	2466*	3134*	5134*	5136*									
PARERR=	010000	1052#	1087#																	
PASS	022602	4734#	5138																	
PAT	005050	1900*	1904	1910	1922*	1941*	1976#	4734												
PATFLG	012752	3196*	3204*	3207#	3231															
PATT	022511	4093	4473	4734#																
PERR	031436	1842*	2986	5149*	5156#															
PIRQ =	177772	811#																		
PIRQVE=	000240	905#																		
PORT	021555	2258	4734#																	
PRB =	177516	942#	3580*																	
PRBAUD	003520	1669#	2228																	
PRCHR	014330	2236*	3580	3581	3583	3585	3587*	3590*	3592*	3594*	3599#									
PRINT	021544	2255	3115	3128	4734#															
PRPORT	012620	2247	2256	3121*	3140*	3157#														
PRPRES	012616	2225	3120*	3141*	3156#															
PRS =	177514	941#	2238*	2249*	3122	3138	3574													
PRSRV	014204	2232	3574#																	
PRT =	020000	985#	2229																	
PRTST	006060	2200	2210	2213	2215	2223#														
PRVEC =	000200	914#	2232*	2233*																
PRO =	000000	828#	1835																	
PR1 =	000040	829#																		
PR2 =	000100	830#																		
PR3 =	000140	831#																		
PR4 =	000200	832#	1705																	
PR5 =	000240	833#																		
PR6 =	000300	834#																		
PR7 =	000340	835#																		
PS =	177776	808#	809																	
PSERR	031440	1843*	2990	3866*	3952*	4046*	4256*	4341*	4428*	5157#										
PSW =	177776	809#																		
PWRVEC=	000024	900#																		
RACT =	004000	1071#																		
RAND	012754	2703	2708	2728	2733	3199	3213#													
RANDOM	022524	4114	4495	4734#																
RDCHR =	104410	4935	5342#																	
RDLIN =	104411	1794	5343#																	
RDWR	021304	4579	4595	4608	4678#															
RDY =	000200	1056#	1114#	1148#																
REGHLD	013712	3424*	3429	3442#																
RESTOR=	000017	1144#	3857	3889	3923	3936	4146	4150	4247	4279	4312	4325								

CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

MACY11 30A(1052) 01-DEC-80 09:47 PAGE 136
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0134

. = 032276

1162#	1166#	1170#	1173#	1176#	1178#	1180#	1185#	1191	1192#	1194#	1196#	1214#
1249	1613#	1614#	1631#	1632#	1642#	1716	2910	2957	3009#	3346	3372	3401
3415	3437	4577	4589	4605	4618	4721	4734#	4814	4817	4959#	4960	4967#
5020#	5068	5123#	5224#	5346#								
4972	4975											
1191#	1196											

.SASTA= ***** U

.SX = 001000

CVKDAD
CVKDAD.P11

PDT11-150 EXERCISER
14-JUL-80 16:36

MACY11 30A(1052) 01-DEC-80 09:47 PAGE 139
CROSS REFERENCE TABLE -- MACRO NAMES

G 11

SEQ 0136

SSSKIP	1#	906#	
.EQUAT	1#	772#	796
.HEADE	1#	772#	773
.KT11	1#		
.SETUP	1#	772#	1678
.SWRHI	1#	772#	
.SACT1	1#	772#	
.SAPT8	1#	772#	1250#
.SAPTH	1#	772#	1186
.SAPTY	1#	772#	4968
.SASTA	1#		
.SCATC	1#	772#	
.SCMTA	1#	772#	1208
.SDB2D	1#		
.SDB2O	1#		
.SDIV	1#		
.SEOP	1#	772#	
.SERRO	1#	772#	5025
.SERRT	1#	772#	5068
.SMULT	1#		
.SPOWE	1#		
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#		
.SREAD	1#	772#	4814
.SR2AZ	1#		
.SSAVE	1#		
.SSB2D	1#		
.SSB2O	1#		
.SSCOP	1#	772#	
.SSIZE	1#		
.SSUPR	1#		
.STRAP	1#	772#	5302
.STYPB	1#		
.STYPD	1#	772#	5158
.STYPE	1#	772#	4735
.STYPO	1#	772#	5225
.S4OCA	1#		
.1170	1#		

. ABS. 032276 000

ERRORS DETECTED: 0

CVKDAD, CVKDAD.SEQ/CRF/SOL=SYSMAC.SML, CVKDAD.P11
RUN-TIME: 55 62 5 SECONDS
RUN-TIME RATIO: 229/123=1.8
CORE USED: 33K (65 PAGES)