

# PDT-11

11 150 SYS EXER  
CVKDAC0

AH F239C MC

COPYRIGHT 78 79

FICHE 1 OF 1

MAY 1979

**digital**

MADE IN USA

This section of the document contains a grid of 100 small, illegible tables or data pages, arranged in 10 rows and 10 columns. The content is too faint to read.

.REM %

IDENTIFICATION

PRODUCT CODE: AC-F238C-MC  
PRODUCT NAME: CVKDACO PDT11/150 SYSTEM EXERCISER  
PRODUCT DATE: APRIL 1979  
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 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.
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.  
-----

VERSION "C" DIFFERS FROM PREVIOUS VERSION BECAUSE VERSION "C":  
A. HAS ADDITIONAL FLOPPY ERROR CODES TO AID PRODUCTION.  
B. SUPPORTS THE "EIS-FIS" INSTRUCTION SET OPTION.  
C. ENHANCED MEMORY TESTING.

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 TFST SUMMARY SEC. 7.0.

NOTE: IF RUNNING UNDER APT, THE SYNC COMM LOGIC WILL BE TESTED ONLY ON THE 1<sup>ST</sup> 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  
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 CVKDAC.  
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 TIMEOUTS)

CVKDAC 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)

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

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 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 (\$DEVN) 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:

\$DEVN (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
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 15-6 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:

1<sup>ST</sup> PASS:

25 - 30 SEC TO STARTUP CLOCK, PRINTER, COMM DEVICE & CLUSTER TERMINALS.  
5 MIN TO WRITE, READ & DATA COMPARE ON TRACKS 0-4, 40-44, 72-76(10) ON BOTH DRIVES.  
5 - 60 SEC TO DO 10 RANDOM SEEKS ON EACH DISK BETWEEN 1<sup>ST</sup> 5 TRACKS.

SUBSEQUENT PASSES:

30 SEC TO STARTUP CLOCK, PRINTER, COMM DEVICE & CLUSTER TERMINALS.  
20 MIN TO WRITE, READ & DATA COMPARE ALL TRACKS ON BOTH DISKS.  
5 MIN TO DO 500 RANDOM SEEKS ON EACH DISK BETWEEN ALL TRACKS.

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'.

3.0 ERROR INFORMATION.  
-----

3.1 ERROR REPORTING PROCEDURE.  
-----

TYPICAL ERROR PRINTOUTS ARE SHOWN BELOW:

TERM #3	DATA COMP	ERR
ERROR #	ERR PC	EXPECT RECVD
000011	006516	000200 000100

DX1 SOFT ERR - DATA COMP							
DXTST #	ERR PC	RXCS	RXES	RXSA	EXPECT	RECVD	# RETRIES
000003	010636	100337	000230	003405	100520	100120	4

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):

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

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
									!CLK! !IE!						
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!						
KB DATA BUFFER																
TPS:								!XMIT! !RDY!		!XMIT! !IE!						
PRINTER DATA BUFFER (WRITE ONLY)																

PRINTER:

VECTOR: 200

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



7.0 SUMMARY OF TESTS (SEE PROGRAM LISTING FOR ADD'L DETAILS ON SPECIFIC TESTS)

PHASE 1

MEMORY TESTS A. LOC 0 THRU END OF PROGRAM IS TESTED FOR TIMEOUT.  
B. END OF PROGRAM TO BOTTOM OF MONITOR/ABS LOADER  
IS TESTED WITH A 125252 & 052525 PATTERN.  
C. IF 28K PRESENT, MEMORY FROM 28K TO 30K IS SIMILARLY TESTED.

PHASE 2

EIS-FIS TESTS A. EIS INSTRUCTIONS "ASH-ASHC-MUL-DIV" ARE EXECUTED.  
B. FIS INSTRUCTIONS "FADD-FSUB-FMUL-FDIV" ARE EXECUTED.

PHASE 3

ALL AVAILABLE PERIPHERALS ARE EXERCISED CONTINUOUSLY IN INTERRUPT MODE.  
SEE SECTION 2.4 FOR DETAILS OF SETTING UP THE DEVICE MAP (\$DEVM) FOR DEVICE TO BE TESTED.  
SEE SECTION 4.1 FOR DETAILS ON PROGRESS REPORTS.

START LINE CLOCK	INTERRUPTS ARE DETECTED.
START PRINTER	CONTINUOUS LINES ARE PRINTED.
START SYNC COMM PORT	CHARS 27 THRU 377 " " " " 1 PASS ONLY
START ASYNC COMM PORT	CHARS 0 THRU 377 TESTED CONTINUOUSLY.
START CLUSTER TERMINAL #1	CHARS 0 THRU 377 ARE XMITTED, REC'D & TESTED.
START CLUSTER TERMINAL #2	SAME
START CLUSTER TERMINAL #3	SAME

PHASE 4

WHILE THE ABOVE DEVICES ARE INTERRUPTING RANDOMLY, DISK SUBSYSTEM TESTS BEGIN:

FILL & EMPTY BUFFER TESTS ARE PERFORMED TO VERIFY THAT NO  
CABLE CROSS-TALK PROBLEMS EXIST BEFORE BEGINNING ACTUAL  
DATA TRANSFERS TO THE DISK SURFACE.

DX0 & DX1 DATA PATTERN: WRITE, READ & DATA COMPARE 1'ST 10 TRACKS ON 1'ST PASS.  
ALL TRACKS ON SUBSEQUENT PASSES.

DX0 & DX1 RANDOM SEEKS: READ & DATA COMPARE.  
20 SEEKS ON 1'ST 10 TRACKS ON 1'ST PASS.  
500 SEEKS ON ALL TRACKS ON SUBSEQUENT PASSES.

DX0 INITIALIZE, RESTORE, WRITE DELETED DATA, READ STATUS, & INVALID ADDRESS TESTS.

NOTE:

1. RECOVERY IS PERFORMED (RESTORE COMMAND) AFTER ANY RNF ERROR ON WRITE OR READ.
2. DATA IS TESTED AFTER A HARD CRC ERROR ON A READ TO TEST WHETHER DATA CRC ERROR OR CRC ERROR.

%

691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746

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

\*MISCELLANEOUS DEFINITIONS

```
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

```
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



747	000000	PR0=	0	::PRIORITY LEVEL 0
748	000040	PR1=	40	::PRIORITY LEVEL 1
749	000100	PR2=	100	::PRIORITY LEVEL 2
750	000140	PR3=	140	::PRIORITY LEVEL 3
751	000200	PR4=	200	::PRIORITY LEVEL 4
752	000240	PR5=	240	::PRIORITY LEVEL 5
753	000300	PR6=	300	::PRIORITY LEVEL 6
754	000340	PR7=	340	::PRIORITY LEVEL 7

755		;*'SWITCH REGISTER' SWITCH DEFINITIONS		
756		SW15=	100000	
757	100000	SW14=	40000	
758	040000	SW13=	20000	
759	020000	SW12=	10000	
760	010000	SW11=	4000	
761	004000	SW10=	2000	
762	002000	SW09=	1000	
763	001000	SW08=	400	
764	000400	SW07=	200	
765	000200	SW06=	100	
766	000100	SW05=	40	
767	000040	SW04=	20	
768	000020	SW03=	10	
769	000010	SW02=	4	
770	000004	SW01=	2	
771	000002	SW00=	1	
772	000001	.EQUIV	SW09,SW9	
773		.EQUIV	SW08,SW8	
774		.EQUIV	SW07,SW7	
775		.EQUIV	SW06,SW6	
776		.EQUIV	SW05,SW5	
777		.EQUIV	SW04,SW4	
778		.EQUIV	SW03,SW3	
779		.EQUIV	SW02,SW2	
780		.EQUIV	SW01,SW1	
781		.EQUIV	SW00,SW0	

782		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)		
783		BIT15=	100000	
784		BIT14=	40000	
785	100000	BIT13=	20000	
786	040000	BIT12=	10000	
787	020000	BIT11=	4000	
788	010000	BIT10=	2000	
789	004000	BIT09=	1000	
790	002000	BIT08=	400	
791	001000	BIT07=	200	
792	000400	BIT06=	100	
793	000200	BIT05=	40	
794	000100	BIT04=	20	
795	000040	BIT03=	10	
796	000020	BIT02=	4	
797	000010	BIT01=	2	
798	000004	BIT00=	1	
799	000002	.EQUIV	BIT09,BIT9	
800	000001	.EQUIV	BIT08,BIT8	
801				
802				

```
803 .EQUIV BIT07,BIT7
804 .EQUIV BIT06,BIT6
805 .EQUIV BIT05,BIT5
806 .EQUIV BIT04,BIT4
807 .EQUIV BIT03,BIT3
808 .EQUIV BIT02,BIT2
809 .EQUIV BIT01,BIT1
810 .EQUIV BIT00,BIT0
811
812 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
813 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
814 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
815 000014 TBITVEC=14 ;: "T" BIT
816 000014 TRTVEC= 14 ;:TRACE TRAP
817 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
818 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
819 000024 PWRVEC= 24 ;:POWER FAIL
820 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
821 000034 TRAPVEC=34 ;:"TRAP" TRAP
822 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
823 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
824 000240 PIQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
825
826
827
828
```

```
829 ;:*****
830 ;* PDT-11/150 DEVICE VECTORS
831 ;:*****
832
833 000200 PRVEC= 200 ;:PRINTER
834
835 000100 CLKVEC= 100 ;:LINE CLOCK
836
837 000330 AMRVEC= 330 ;:ASYNCR MODEM RECVR
838 000334 AMXVEC= 334 ;: XMITR
839
840 000340 SMRVEC= 340 ;:SYNCR MODEM RECVR
841 000344 SMXVEC= 344 ;: XMITR
842
843 000264 RXVEC= 264 ;:DISK
844
845 000300 TK1VEC= 300 ;:CLUSTER TERM #1 VECTORS
846 000304 TP1VEC= 304
847 000310 TK2VEC= 310 ;: #2
848 000314 TP2VEC= 314
849 000320 TK3VEC= 320 ;: #3
850 000324 TP3VEC= 324
```

```
851
852
853      ;*****
854      ;      PDT-11/150 DEVICE REGISTERS
855      ;*****
856      177420      PARAM= 177420      ;PARAMETER REGISTER      (WRITE ONLY)
857
858      177546      CLK= 177546      ;LINE CLOCK
859
860      177514      PRS= 177514      ;PRINTER STATUS REG
861      177516      PRB= 177516      ;      BUFFER      (WRITE ONLY)
862
863      176610      AMRC= 176610      ;ASYNC MODEM RECVR STATUS REG
864      176612      AMRB= 176612      ;      BUFFER
865      176614      AMXC= 176614      ;      XMITR STATUS REG
866      176616      AMXB= 176616      ;      BUFFER      (WRITE ONLY)
867
868      176620      SMRC= 176620      ;SYNC MODEM RECVR STATUS REG
869      176622      SMRB= 176622      ;      BUFFER      (READ ONLY)
870      176622      SMPAR= 176622      ;      PARAMETER REG      (WRITE ONLY)
871      176624      SMXC= 176624      ;      XMITR STATUS REG
872      176626      SMXB= 176626      ;      BUFFER      (WRITE ONLY)
873
874      177170      RXCS= 177170      ;DISK CSR
875      177172      RXDB= 177172      ;      DATA BUFF
876      177172      RXES= 177172      ;      ERROR & STATUS REG
877      177174      RXSA= 177174      ;      TRK & SECTOR ADDR REG
878
879      176500      TK1S= 176500      ;CLUSTER TERMINAL #1
880      176502      TK1B= 176502
881      176504      TP1S= 176504
882      176506      TP1B= 176506
883
884      176510      TK2S= 176510      ;      #2
885      176512      TK2B= 176512
886      176514      TP2S= 176514
887      176516      TP2B= 176516
888
889      176520      TK3S= 176520      ;      #3
890      176522      TK3B= 176522
891      176524      TP3S= 176524
892      176526      TP3B= 176526
```

```
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944
```

```
*****  
* PARAMETER REGISTER EQUATES WRITE ONLY  
*****  
:BIT 14-12 PORT SELECT  
:  
CONSOL= 10000 ;SELECT CONSOLE TERMINAL  
CT1= 0 ; CLUSTER TERMINAL #1  
CT2= 50000 ; #2  
CT3= 60000 ; #3  
PRT= 20000 ; PRINTER PORT  
AMOD= 30000 ; ASYNC 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 & 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
```

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  
974  
975  
976  
977

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
```

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

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)
```

```

1025
1026
1027
1028      100000
1029      010000
1030      004000
1031      000400
1032      000200
1033      000100
1034      000040
1035      000020
1036      000010
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047      100000
1048      000200
1049      000100
1050      000020
1051      000020
1052
1053
1054
1055      000001
1056      000003
1057      000005
1058      000007
1059      000011
1060      000013
1061      000015
1062      000017
1063
1064
1065
1066      000200
1067      000040
1068      000020
1069      000010
1070      000004
1071      000002
1072      000001
1073
1074

;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
HFDUP=   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

```

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  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125

000174  
000174 0000C0  
000176 000000  
  
000100  
000100 000102  
000102 000002  
  
000200  
000200 000137 003334  
  
000240  
000240 000137 003334  
  
000250  
000250 000137 003316  
000260  
000260 000137 003276  
000270  
000270 000137 003304  
  
001000  
  
000024  
000044  
001000  
001000  
000000  
001170  
000024  
000226  
000000  
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  
.WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
```



1126  
1127  
1128  
1129  
1130  
1131  
1132 001100 001100  
1133 001100 000000  
1134 001100 000000  
1135 001102 000  
1136 001103 000  
1137 001104 000000  
1138 001106 000000  
1139 001110 000000  
1140 001112 000000  
1141 001114 000  
1142 001115 001  
1143 001116 000000  
1144 001120 000000  
1145 001122 000000  
1146 001124 000000  
1147 001126 000000  
1148 001130 000000  
1149 001132 000000  
1150 001134 000  
1151 001135 000  
1152 001136 000000  
1153 001140 177570  
1154 001142 177570  
1155 001144 177560  
1156 001146 177562  
1157 001150 177564  
1158 001152 177566  
1159 001154 000  
1160 001155 002  
1161 001156 012  
1162 001157 000  
1163 001160 177607 000377  
1164 001164 077  
1165 001165 015  
1166 001166 000012  
1167  
1168  
1169  
1170  
1171  
1172 001170  
1173 001170 000000  
1174 001172 000000  
1175 001174 000000  
1176 001176 000000  
1177 001200 000000  
1178 001202 000000  
1179 001204 000000  
1180 001206 000000  
1181 001210

```

.SBTTL COMMON TAGS

:*****
:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:*USED IN THE PROGRAM.

      . =1100
SCMTAG:      . =1100      :: START OF COMMON TAGS

$STNM: .WORD 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
$ERTL: .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
      .WORD 0
$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: .ASCIZ /?/    :: QUESTION MARK
$CRLF: .ASCIZ <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

```

1182	001210	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
1183	001211	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
1184	001212	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
1185	001214	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
1186	001216	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
1187			.*			BITS 15-11=CPU TYPE
1188			.*			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
1189			.*			11/70=06,PDQ=07,Q=10
1190			.*			BIT 10=REAL TIME CLOCK
1191			.*			BIT 9=FLOATING POINT PROCESSOR
1192			.*			BIT 8=MEMORY MANAGEMENT
1193	001220	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
1194	001221	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
1195			.*			MEM.TYPE BYTE -- (HIGH BYTE)
1196			.*			900 NSEC CORE=001
1197			.*			300 NSEC BIPOLAR=002
1198			.*			500 NSEC MOS=003
1199	001222	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
1200			.*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
1201	001224	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
1202	001225	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
1203	001226	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
1204	001230	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
1205	001231	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
1206	001232	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
1207	001234	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
1208	001235	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
1209	001236	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
1210	001240	000300	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
1211	001242	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
1212	001244	176500	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
1213	001246	000017	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
1214	001250		\$ETEND:			
1215			.MEXIT			

1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270

001250

001250	021511	024207	024614
001256	025110		
001260	021511	024207	024614
001266	025110		
001270	021530	024233	024624
001276	025110		
001300	021511	024207	024614
001306	025110		
001310	021530	024233	024624
001316	025110		
001320	021552	024170	024606
001326	025110		
001330	021616	024275	024640
001336	025110		
001340	021641	024326	024650
001346	025110		
001350	021667	024326	024662
001356	025110		
001360	021715	024326	024674
001366	025110		
001370	021743	024326	024706
001376	025110		
001400	021774	024326	024706
001406	025110		
001410	022024	024363	024720
001416	025110		

.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

1271					:ERR 16	
1272	001420	022051	024430	024736		EM11,DH7,DT10,DF1
1273	001426	025116				
1274					:ERR 17	
1275	001430	022051	024430	024736		EM11,DH7,DT10,DF1
1276	001436	025116				
1277					:ERR 20	
1278	001440	022067	024430	024736		EM13,DH7,DT10,DF1
1279	001446	025116				
1280					:ERR 21	
1281	001450	022120	024430	024736		EM14,DH7,DT10,DF1
1282	001456	025116				
1283					:ERR 22	
1284	001460	022151	024517	024760		EM15,DH8,DT11,DF2
1285	001466	025126				
1286					:ERR 23	
1287	001470	022202	024517	024760		EM16,DH8,DT11,DF2
1288	001476	025126				
1289					:ERR 24	
1290	001500	022233	024430	025002		EM17,DH7,DT12,DF1
1291	001506	025116				
1292					:ERR 25	
1293	001510	022233	024430	025002		EM17,DH7,DT12,DF1
1294	001516	025116				
1295					:ERR 26	
1296	001520	022251	024430	025002		EM19,DH7,DT12,DF1
1297	001526	025116				
1298					:ERR 27	
1299	001530	022302	024430	025002		EM20,DH7,DT12,DF1
1300	001536	025116				
1301					:ERR 30	
1302	001540	022333	024517	025024		EM21,DH8,DT13,DF2
1303	001546	025126				
1304					:ERR 31	
1305	001550	022364	024517	025024		EM22,DH8,DT13,DF2
1306	001556	025126				
1307					:ERR 32	
1308	001560	022415	024170	024606		EM23,DH1,DT1,DF
1309	001566	025110				
1310					:ERR 33	
1311	001570	022426	024275	024640		EM24,DH4,DT4,DF
1312	001576	025110				
1313					:ERR 34	
1314	001600	022443	024170	024606		EM25,DH1,DT1,DF
1315	001606	025110				
1316					:ERR 35	
1317	001610	022460	024170	024606		EM26,DH1,DT1,DF
1318	001616	025110				
1319					:ERR 36	
1320	001620	022475	024170	024606		EM27,DH1,DT1,DF
1321	001626	025110				

1322					:ERR 37	
1323	001630	022512	024170	024606		EM28,DH1,DT1,DF
1324	001636	025110				
1325					:ERR 40	
1326	001640	022531	024170	024606		EM29,DH1,DT1,DF
1327	001646	025110				
1328					:ERR 41	
1329	001650	022551	024363	025046		EM30,DH6,DT14,DF
1330	001656	025110				
1331					:ERR 42	
1332	001660	022562	024363	025046		EM31,DH6,DT14,DF
1333	001666	025110				
1334					:ERR 43	
1335	001670	022573	024170	024606		EM32,DH1,DT1,DF
1336	001676	025110				
1337					:ERR 44	
1338	001700	022636	024170	024606		EM33,DH1,DT1,DF
1339	001706	025110				
1340					:ERR 45	
1341	001710	022677	024170	024606		EM34,DH1,DT1,DF
1342	001716	025110				
1343					:ERR 46	
1344	001720	022551	024363	025046		EM30,DH6,DT14,DF
1345	001726	025110				
1346					:ERR 47	
1347	001730	022551	024363	025046		EM30,DH6,DT14,DF
1348	001736	025110				
1349					:ERR 50	
1350	001740	022551	024363	025046		EM30,DH6,DT14,DF
1351	001746	025110				
1352					:ERR 51	
1353	001750	022551	024363	025046		EM30,DH6,DT14,DF
1354	001756	025110				
1355					:ERR 52	
1356	001760	022736	024363	024720		EM35,DH6,DT9,DF
1357	001766	025110				
1358					:ERR 53	
1359	001770	023003	024517	024760		EM36,DH8,DT11,DF2
1360	001776	025126				
1361					:ERR 54	
1362	002000	023052	024363	024720		EM37,DH6,DT9,DF
1363	002006	025110				
1364					:ERR 55	
1365	002010	023076	024363	024720		EM38,DH6,DT9,DF
1366	002016	025110				
1367					:ERR 56	
1368	002020	022551	024363	025046		EM30,DH6,DT14,DF
1369	002026	025110				
1370					:ERR 57	
1371	002030	023144	024363	024720		EM39,DH6,DT9,DF
1372	002036	025110				

1373					:ERR 60	
1374	002040	022551	024363	025046		EM30,DH6,DT14,DF
1375	002046	025110				
1376					:ERR 61	
1377	002050	023174	024363	024720		EM40,DH6,DT9,DF
1378	002056	025110				
1379					:ERR 62	
1380	002060	023237	024363	024720		EM41,DH6,DT9,DF
1381	002066	025110				
1382					:ERR 63	
1383	002070	023273	024363	024720		EM42,DH6,DT9,DF
1384	002076	025110				
1385					:ERR 64	
1386	002100	023314	024430	024736		EM43,DH7,DT10,DF1
1387	002106	025116				
1388					:ERR 65	
1389	002110	023314	024430	024736		EM43,DH7,DT10,DF1
1390	002116	025116				
1391					:ERR 66	
1392	002120	023346	024430	024736		EM45,DH7,DT10,DF1
1393	002126	025116				
1394					:ERR 67	
1395	002130	023346	024430	024736		EM45,DH7,DT10,DF1
1396	002136	025116				
1397					:ERR 70	
1398	002140	023377	024430	025002		EM47,DH7,DT12,DF1
1399	002146	025116				
1400					:ERR 71	
1401	002150	023377	024430	025002		EM47,DH7,DT12,DF1
1402	002156	025116				
1403					:ERR 72	
1404	002160	023431	024430	025002		EM49,DH7,DT12,DF1
1405	002166	025116				
1406					:ERR 73	
1407	002170	023431	024430	025002		EM49,DH7,DT12,DF1
1408	002176	025116				
1409					:ERR 74	
1410	002200	023462	024363	024720		EM51,DH6,DT9,DF
1411	002206	025110				
1412					:ERR 75	
1413	002210	023506	024363	025046		EM52,DH6,DT14,DF
1414	002216	025110				
1415					:ERR 76	
1416	002220	023532	024326	025064		EM53,DH5,DT15,DF
1417	002226	025110				
1418					:ERR 77	
1419	002230	023532	024326	025076		EM53,DH5,DT16,DF
1420	002236	025110				

1421					
1422					: ERRORS FOR COPY UTILITY
1423					
1424					:ERR 100
1425	002240	022067	024363	025046	EM13,DH6,DT14,DF
1426	002246	025110			
1427					:ERR 101
1428	002250	022233	024363	025046	EM17,DH6,DT14,DF
1429	002256	025110			
1430					:ERR 102
1431	002260	023601	024363	025046	EM54,DH6,DT14,DF
1432	002266	025110			
1433					:ERR 103
1434	002270	022251	024363	025046	EM19,DH6,DT14,DF
1435	002276	025110			
1436					
1437					:BAD SECTOR/TRACK TABLE FULL ERRORS
1438					
1439					:ERR 104
1440	002300	023616	000000	000000	EM55,0,0,0
1441	002306	000000			
1442					:ERR 105
1443	002310	023653	000000	000000	EM56,0,0,0
1444	002316	000000			
1445					
1446					:CRC ERRORS
1447					
1448					:ERR 106
1449	002320	023710	024430	024736	EM57,DH7,DT10,DF1
1450	002326	025116			
1451					:ERR 107
1452	002330	023731	024517	024760	EM58,DH8,DT11,DF2
1453	002336	025126			
1454					:ERR 110
1455	002340	023752	000000	000000	EM59,0,0,0
1456	002346	000000			
1457					:ERR 111
1458	002350	024010	024430	025002	EM60,DH7,DT12,DF1
1459	002356	025116			
1460					:ERR 112
1461	002360	024031	024517	025024	EM61,DH8,DT13,DF2
1462	002366	025126			
1463					:ERR 113
1464	002370	024052	000000	000000	EM62,0,0,0
1465	002376	000000			
1466					:ERR 114
1467	002400	023506	024363	025046	EM52,DH6,DT14,DF
1468	002406	025110			
1469					:ERR 115
1470	002410	023506	024363	025046	EM52,DH6,DT14,DF
1471	002416	025110			
1472					:ERR 116
1473	002420	023506	024363	025046	EM52,DH6,DT14,DF
1474	002426	025110			
1475					:ERR 117
1476	002430	022551	024363	025046	EM30,DH6,DT14,DF

1477	002436	025110			
1478					
1479	002440	022562	024363	025046	:ERR 120 EM31,DH6,DT14,DF
1480	002446	025110			
1481					
1482	002450	022551	024363	025046	:ERR 121 EM30,DH6,DT14,DF
1483	002456	025110			
1484					
1485	002460	022562	024363	025046	:ERR 122 EM31,DH6,DT14,DF
1486	002466	025110			
1487					
1488	002470	022562	024363	025046	:ERR 123 EM31,DH6,DT14,DF
1489	002476	025110			
1490					
1491	002500	023506	024363	025046	:ERR 124 EM52,DH6,DT14,DF
1492	002506	025110			
1493					
1494	002510	023506	024363	025046	:ERR 125 EM52,DH6,DT14,DF
1495	002516	025110			
1496					
1497					
1498					:EIS FIS ERRORS
1499					
1500					
1501					
1502	002520	024110	024170	024606	:ERR 126 EM63,DH1,DT1,DF
1503	002526	025110			
1504					
1505	002530	024145	024170	024606	:ERR 127 EM64,DH1,DT1,DF
1506	002536	025110			
1507					



1508  
1509  
1510  
1511  
1512 002540 000000  
1513 002542 000000  
1514 002544 000000  
1515 002546 000000  
1516 002550 000000  
1517 002552 000000  
1518 002554 000000  
1519 002556 000000  
1520 002560 000000  
1521 002562 000000  
1522 002564 000100  
1523 002764 000012  
1524  
1525  
1526  
1527  
1528  
1529  
1530 003010 000000  
1531 003012 000000  
1532 003014 000000  
1533 003016 000000  
1534 003020 000000  
1535 003022 000000  
1536 003024 000000  
1537 003026 000000  
1538 003030 000000  
1539 003032 000000  
1540 003034 000100  
1541 003234 000012  
1542  
1543 003260 000000  
1544  
1545  
1546  
1547  
1548  
1549  
1550 003262 000000  
1551 003264 000000  
1552  
1553 003266 000000  
1554  
1555 003270 000000  
1556 003272 000000  
1557 003274 000000  
1558  
1559

\*\*\*\*\*  
: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  
DOEXP: 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

DXTST: 0 ;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.  
MAXSK: 0 ;# OF RANDOM SEEKS TO BE PERFORMED.  
;10 FOR 1'ST PASS, 500 FOR SUBSEQUENT PASSES. SET AT EOP.  
COPFLG: 0 ;0 = NORM TEST 1 = COPY UTILITY ACTIVE  
COMP1: 0 ;1 = COMPAT TESTS, PASS 1  
COMP2: 0 ;1 = PASS 2

```

1560 .SBTTL PROGRAM
1561
1562 003276 005237 003272 START1: INC COMP1 ;COMPAT PASS 1
1563 003302 000424 BR S1
1564
1565 003304 005237 003274 START2: INC COMP2 ;COMPAT PASS 2
1566 003310 005037 003272 CLR COMP1
1567 003314 000421 BR S2
1568
1569 003316 005237 003270 START3: INC COPFLG ;COPY UTILITY
1570 003322 005037 003272 CLR COMP1
1571 003326 005037 003274 CLR COMP2
1572 003332 000414 BR S3
1573
1574 ;NORMAL START AND RESTART
1575 003334 012737 000102 000100 START: MOV #CLKVEC+2,CLKVEC ;DO RTI IF CLOCK INTERR
1576 003342 012737 000002 000102 MOV #RTI,CLKVEC+2
1577
1578 003350 005037 003272 CLR COMP1 ;NORMAL TESTING
1579 003354 005037 003274 S1: CLR COMP2
1580 003360 005037 003270 S2: CLR COPFLG
1581
1582 003364 012706 001100 S3: MOV #STACK,SP ;SETUP STACK POINTER
1583 003370 042777 000100 175546 BIC #100,@$TKS ;DISABLE ANY TTY INTERRUPTS
1584 ;TO RT-11 MONITOR
1585 ;DISABLE INTERRUPTS
1586 ;MTPS #PR4
1587 .WORD 106427
1588 PR4
1589 MOV 42,HLD42 ;SAVE
1590 CMPB $ENV,#1 ;APT?
1591 BEQ 2$ ;BR IF YES
1592 003424 CLR 42 ;ELSE CLR SO WE CAN USE SOFTSWR
1593
1594 .SBTTL INITIALIZE THE COMMON TAGS
1595 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1596 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
1597 CLR (R6)+ ;;CLEAR MEMORY LOCATION
1598 CMP #SWR,R6 ;;DONE?
1599 BNE -6 ;;LOOP BACK IF NO
1600 MOV #STACK,SP ;;SETUP THE STACK POINTER
1601 ;;INITIALIZE A FEW VECTORS
1602 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1603 MOV #340,@#EMTVEC+2 ;;LEVEL 7
1604 MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1605 MOV #340,@#TRAPVEC+2;LEVEL 7
1606 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1607 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1608 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
1609 MOV #64$,@#ERRVEC ;;SET UP ERROR VECTOR
1610 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
1611 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
1612 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
1613 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1614 BR 65$ ;;AND THE HARDWARE SWR IS NOT = -1
1615 003534 012716 003542 64$: MOV #65$,(SP) ;;BRANCH IF NO TIMEOUT
;;SET UP FOR TRAP RETURN

```

```

1616 003540 000002          RTI
1617 003542 012737 000176 001140 65$: MOV #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1618 003550 012737 000174 001142     MOV #DISPREG,DISPLAY
1619 003556 012637 000004          66$: MOV (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1620
1621 003562 005037 001176          CLR $PASS           ;;CLEAR PASS COUNT
1622 003566 132737 000200 001211     BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1623 003574 001403          BEQ 67$           ;;YES,USE NON-APT SWITCH
1624 003576 012737 001212 001140     MOV #$$SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
1625 003604
1626
1627          .SBTTL TYPE PROGRAM NAME
          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1628 003604 005227 177777          INC #-1           ;;FIRST TIME?
1629 003610 001046          BNE 68$           ;;BRANCH IF NO
1630 003612 104401 003660          TYPE ,69$        ;;TYPE ASCIZ STRING
1631          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1632 003616 005737 000042          TST @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
1633 003622 001012          BNE 70$           ;;BRANCH IF YES
1634 003624 123727 001210 000001     CMPB $ENV,#1      ;;ARE WE RUNNING UNDER APT?
1635 003632 001406          BEQ 70$           ;;BRANCH IF YES
1636 003634 023727 001140 000176     CMP SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
1637 003642 001005          BNE 71$           ;;BRANCH IF NO
1638 003644 104406          GTSWR           ;;GET SOFT-SWR SETTINGS
1639 003646 000403          BR 71$
1640 003650 112737 000001 001134 70$: MOVB #1,$AUTOB    ;;SET AUTO-MODE INDICATOR
1641 003656          71$:
1642 003656 000423          BR 68$           ;;GET OVER THE ASCIZ
1643          ;;69$: .ASCIZ <CRLF>*CVKDAC PDT-11/150 SYSTEM EXERCISER*<CRLF>
1644          68$:
1645 003726 005737 003270          TST COPFLG       ;;COPY UTILITY?
1646 003732 001057          BNE LOOP        ;;BR IF YES
1647
1648 003734 005737 003272          TST COMP1        ;;COMPAT PASS 1?
1649 003740 001054          BNE LOOP        ;;BR IF YES
1650 003742 005737 003274          TST COMP2        ;;COMPAT PASS 2?
1651 003746 001051          BNE LOOP        ;;BR IF YES
1652
1653 003750 012737 000004 003264     MOV #4,MAXTRK    ;;TRK 0-4, 40-44, 72-76(10) FOR 1'ST PASS
1654 003756 012737 000012 003266     MOV #10.,MAXSK  ;;10 RANDOM SEEKS FOR 1'ST PASS
1655
1656 003764 005037 027100          CLR TERR        ;;TOTAL ERR CT
1657 003770 005037 027102          CLR TSERR       ;;TOTAL SOFT ERR CT FOR EOP TYPEOUT
1658
1659 003774 052737 000006 176610     BIS #<RTS!DTR> ,AMRC
1660 004002 042737 000006 176610     BIC #<RTS!DTR> ,AMRC ;;DTR MUST BE TOGGLED.

```

```

1661
1662 004010 005227 177777          INC      #-1          ;1'ST TIME?
1663 004014 001005          BNE      4$          ;BR IF NO
1664 004016 123727 001210 000001    CMPB     $ENV,#1     ;APT?
1665 004024 001401          BEQ      4$          ;BR IF YES
1666 004026 104412          GTDEVM          ;SHOW CURRENT DEVM & GET NEW
1667
1668 004030 004737 011340          4$:     JSR      PC,SIZE   ;SEE WHO IS ON THE SYSTEM
1669 004034 123727 001210 000001    CMPB     $ENV,#1     ;ARE WE RUNNING UNDER APT?
1670 004042 001413          BEQ      LOOP        ;BR IF YES & DGNT HALT
1671 004044 032737 001000 001246    BIT      #1000,$DEVM ;DX0 THERE?
1672 004052 001404          BEQ      1$          ;BR IF YES
1673 004054 032737 000400 001246    BIT      #400,$DEVM  ;DX1 THERE?
1674 004062 001003          BNE      LOOP        ;BR IF NO
1675 004064 104401 020365          1$:     TYPE      ,WARNING ;INSERT SCRATCH DISKS
1676 004070 000000          HALT
1677
1678
1679 004072 012706 001100          LOOP:   MOV      #1100,SP ;LOOP HERE AFTER EOP & RESET STACK
1680
1681 004076 004737 012622          JSR      PC,TIMER1   ;TIMER TO ALLOW 'P' TO PRINT BEFORE RESET
1682
1683 004102 000005          RESET
1684 004104 042737 000100 177546    BIC      #IE,CLK     ;CLEAR ALL INTERRUPT ENABLES
1685                                     ;RESET SETS CLK IE. DONT ALLOW YET
1686                                     ;MTPS      #PRO      ;ALLOW INTERRUPTS
1687                                     .WORD     106427
1688                                     PRO
1689
1690 004116 052737 000006 176610    BIS      #<RTS!DTR>,AMRC ;TO CONDITION ALL TERMINALS &
1691                                     ;COMM PORT TO OPERATE IN EXT LOOPBACK MODE.
1692                                     ;NO HARM IF LOOPBACK NOT USED.
1693 004124 004737 012622          JSR      PC,TIMER1   ;TIMER TO ALLOW PREV XFERS TO FINISH
1694
1695 004130 005037 027104          CLR      PERR        ;PASS ERR COUNT FOR EOP TYPEOUT
1696 004134 005037 027106          CLR      PSERR       ;PASS SOFT ERR COUNT FOR EOP TYPEOUT
1697
1698 004140 004737 012720          JSR      PC,CLRBAD   ;CLEAR BAD SECTOR/TRACK TABLES
1699
1700 004144 005737 003272          TST      COMP1       ;COMPAT PASS 1?
1701 004150 001003          BNE      1$          ;BR IF YES
1702 004152 005737 003274          TST      COMP2       ;COMPAT PASS 2?
1703 004156 001405          BEQ      2$          ;BR IF NO
1704
1705 004160 012737 000114 003264    1$:     MOV      #114,MAXTRK
1706 004166 000137 007246          JMP      DX1ST1      ;COMPATABILITY TESTING
1707
1708 004172 005737 003270          2$:     TST      COPFLG     ;COPY UTILITY?
1709 004176 001402          BEQ      MEMTST      ;BR IF NO...NORMAL TESTING
1710 004200 000137 017410          JMP      COPY        ;ELSE GO COPY
1711

```

```

1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723 004204 004737 012136
1724
1725 004210 012737 013176 000004
1726 004216 012737 000200 000006
1727
1728 004224 013703 012022
1729 004230 162703 000000
1730
1731
1732
1733
1734
1735
1736 004234 005004
1737 004236 005037 013204
1738 004242 005714
1739 004244 005737 013204
1740 004250 001403
1741 004252 010437 004546
1742 004256 104001
1743 004260 005724
1744 004262 020427 027744
1745 004266 001363
1746
    
```

```

.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
:      MCV      #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
2$:      TST      (R4)+ ;BUMP ADDR
:      CMP      R4,#LSTAD ;AT END?
:      BNE      1$ ;BR IF NO
    
```

```

1747
1748
1749
1750
1751 004270 005005
1752 004272 012737 125252 004544
1753 004300 012704 027744 8$:
1754
1755 004304 005037 013204 3$:
1756 004310 013714 004544
1757 004314 005737 013204
1758 004320 001403
1759 004322 010437 004546
1760 004326 104002
1761 004330 011437 004542 4$:
1762 004334 023737 004542 004544
1763 004342 001403
1764 004344 010437 004546
1765 004350 104003
1766
1767 004352 005724 6$:
1768 004354 004737 004510
1769 004360 000751
1770
1771 004362 005705
1772 004364 001005
1773 004366 005205
1774 004370 012737 052525 004544
1775 004376 000740
1776
1777
1778
1779
1780
1781 004400 012704 027744 9$:
1782
1783 004404 010414 10$:
1784 004406 005724
1785 004410 004737 004510
1786 004414 000773
1787
1788
1789 004416 012704 027744
1790 004422 020414 11$:
1791 004424 001407
1792 004426 010437 004546
1793 004432 010437 004544
1794 004436 011437 004542
1795 004442 104005
1796
1797 004444 005724 12$:
1798 004446 004737 004510
1799 004452 000763
1800
1801 004454 012737 000006 000004
1802 004462 005037 000006

```

```

:*****
: TEST READ/WRITE PATTERNS FROM 'LSTAD' TO BOTTOM OF MONITOR
:*****

```

```

      CLR      R5           ;PASS CTR
      MOV      #125252,PAT ;DATA PATT FOR 1'ST PASS
      MOV      #LSTAD,R4   ;ADDR POINTER. R/W LOCS LSTAD THRU
                               ;BOT OF RT11 OR BOT OF ABS LOADER
8$:
      CLR      INTFLG
      MOV      PAT,(R4)    ;WRITE
      TST      INTFLG     ;TIMEOUT?
      BEQ      4$         ;BR IF NO
      MOV      R4,ADDR     ;FOR ERR TYP
      ERROR   2           ;TIMEOUT ON WRITE
4$:
      MOV      (R4),MEMHLD ;READ
      CMP      MEMHLD,PAT ;COMPARE
      BEQ      6$
      MOV      R4,ADDR     ;FOR ERR TYP
      ERROR   3           ;COMPARE ERROR
6$:
      TST      (R4)+      ;BUMP POINTER
      JSR      PC,BOTMON  ;SEE IF AT BOTTOM OF MONITOR
      BR      3$         ;RET HERE IF NO
      TST      R5         ;ELSE HERE
      BNE      9$
      INC      R5
      MOV      #052525,PAT ;DATA PATT FOR 2'ND PASS
      BR      8$         ;REPEAT

```

```

:*****
: ADDRESS TEST. WRITE EACH LOC. WITH ITS ADDR. AS DATA. THEN VERIFY.
:*****

```

```

9$:
      MOV      #LSTAD,R4   ;ADDR REG
10$:
      MOV      R4,(R4)    ;LOAD ADDR AS DATA
      TST      (R4)+      ;BUMP ADDR
      JSR      PC,BOTMON  ;SEE IF AT BOT OF MON
      BR      10$        ;RET HERE IF NO
                               ;VERIFY PASS
11$:
      MOV      #LSTAD,R4
      CMP      R4,(R4)    ;READ & CHECK
      BEQ      12$        ;BR IF OK
      MOV      R4,ADDR
      MOV      R4,PAT
      MOV      (R4),MEMHLD
      ERROR   5           ;MISCOMPARE
12$:
      TST      (R4)+      ;BUMP ADDR
      JSR      PC,BOTMON  ;SEE IF AT BOT OF MON
      BR      11$        ;RET HERE IF NO
      MOV      #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
      CLR      ERRVEC+2

```

CVKDAC  
CVKDAC.P11

PDT11-150 EXERCISER  
16-FEB-79 11:12

MACY11 30G(1063) 16-FEB-79<sup>M 3</sup> 11:13 PAGE 38  
PROGRAM

SEQ 0038

1803  
1804 004466 032777 010000 174444  
1805 004474 001426  
1806 004476 104401 020776  
1807 004502 104401 021011  
1808 004506 000421  
1809

BIT #BIT12, @SWR ;SKIP TYPEOUT IF NOT SET  
BEQ EISFIS  
TYPE ,MEMORY  
TYPE ,DUN  
BR EISFIS

```
1810 ;*****  
1811 ;ROUTINE TO DETERMINE IF AT BOTTOM OF RT-11 OR XXDP MONITOR  
1812 ;*****  
1813  
1814 004510 023727 004550 001000 BOTMON: CMP HLD42,#1000 ;RT-11 IF LOC 42 = 1000  
1815 004516 001403 BEQ 1$ ;BR IF RT-11  
1816 004520 0204C3 CMP R4,R3 ;ELSE AT BOT OF XXDP MON?  
1817 004522 001006 BNE 3$ ;BR IF NO  
1818 004524 000403 BR 2$  
1819  
1820 004526 020437 000054 1$: CMP R4,54 ;LOC 54 CONTAINS LOW ADDR OF RT-11 MON.  
1821 004532 001002 BNE 3$ ;BR IF NOT THERE  
1822 004534 062716 000002 2$: ADD #2,(SP) ;RET+2 IF AT END  
1823 004540 000207 3$: RTS PC  
1824  
1825 004542 000000 MEMHLD: 0 ;PUT MEMORY READ HERE  
1826 004544 000000 PAT: 0 ;PATT TO BE WRITTEN & READ  
1827 004546 000000 ADDR: 0 ;ADDR UNDER TEST  
1828 004550 000000 HLD42: 0 ;SAVE LOC 42
```



```

1829
1830
1831
1832 004552 012737 001750 005424
1833 004560 005737 012012
1834 004564 001404
1835 004566 032737 000100 001246
1836 004574 001402
1837 004576 000137 005446
1838
1839
1840
1841 004602 012702 125252
1842 004606 000257
1843 004610 000264
1844 004612 000270
1845 004614 072227 000001
1846
1847 004620 100403
1848 004622 001402
1849 004624 102001
1850 004626 103401
1851 004630 104126
1852 004632 022702 052524
1853 004636 001401
1854 004640 104127
1855
1856
1857
1858 004642 012701 052525
1859 004646 000257
1860 004650 072127 177777
1861 004654 022701 025252
1862 004660 001401
1863 004662 104127
1864
1865
1866
1867 004664 012700 000001
1868 004670 000241
1869 004672 072027 000020
1870 004676 103001
1871 004700 001401
1872 004702 104126
1873
1874
1875
1876 004704 012700 100000
1877 004710 000241
1878 004712 072027 177760
1879 004716 103401
1880 004720 104126
1881 004722 020027 177777
1882 004726 001401
1883 004730 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,$DEVN ;TEST IF EIS/FIS IS ENABLED
          BEQ ASH1 ;BR IF YES
1$: 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 ASH4 ;BR IF OK
       ERROR 127 ;INCORRECT DATA FOR "ASH RIGHT"

```

```
1884
1885
1886
1887
1888 004732 005002
1889 004734 012703 125252
1890 004740 000257
1891 004742 000264
1892 004744 000261
1893 004746 073227 000020
1894
1895 004752 100003
1896 004754 001402
1897 004756 102001
1898 004760 103001
1899 004762 104126
1900 004764 022702 125252
1901 004770 001401
1902 004772 104127
1903 004774 005703
1904 004776 001401
1905 005000 104127
1906
1907
1908
1909
1910 005002 005003
1911 005004 012702 125252
1912 005010 000257
1913 005012 000264
1914 005014 000262
1915 005016 000261
1916 005020 073227 177760
1917
1918 005024 100002
1919 005026 101401
1920 005030 102001
1921 005032 104126
1922 005034 022702 177777
1923 005040 001401
1924 005042 104127
1925 005044 022703 125252
1926 005050 001401
1927 005052 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"
```

1928  
1929  
1930  
1931 005054 005000  
1932 005056 012701 000002  
1933 005062 073027 000037  
1934 005066 1030C1  
1935 005070 001401  
1936 005072 104126  
1937  
1938  
1939  
1940  
1941  
1942 005074 012700 100000  
1943 005100 005001  
1944 005102 073027 000040  
1945 005106 103001  
1946 005110 100401  
1947 005112 104126  
1948 005114 022701 177777  
1949 005120 001003  
1950 005122 022700 177777  
1951 005126 001401  
1952 005130 104127

```
*****  
:TEST ASHC LEFT SHIFT 0000,0002 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 100000,0000 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"
```

1953  
1954  
1955  
1956  
1957 005132 012700 125252  
1958 005136 070027 040000  
1959 005142 1000C3  
1960 005144 102402  
1961 005146 001401  
1962 005150 103401  
1963 005152 104126  
1964 005154 022700 165252  
1965 005160 001003  
1966 005162 022701 100000  
1967 005166 001401  
1968 005170 104127  
1969  
1970  
1971  
1972  
1973  
1974 005172 012701 125252  
1975 005176 012700 177777  
1976 005202 000261  
1977 005204 071027 000002  
1978 005210 103403  
1979 005212 102402  
1980 005214 001401  
1981 005216 100401  
1982 005220 104126  
1983 005222 005701  
1984 005224 001003  
1985 005226 020027 152525  
1986 005232 001401  
1987 005234 104127

```
*****  
:TEST MUL INSTRUCTION  
*****  
:      125252*40000=165252,100000  
MULO:  MOV    #125252,R0    ;LOAD MULTIPLICAND  
        MUL    #40000,R0    ;R1:R0=40000*125252  
        BPL   1$           ;BR IF N BIT ERROR  
        BVS   1$           ;BR IF V BIT ERROR  
        BEQ   1$           ;BR IF Z BIT ERROR  
        BCS   2$           ;BR IF C BIT CORRECT  
1$:     ERROR 126          ;INCORRECT CONDITION CODE FOR "MUL"  
2$:     CMP    #165252,R0   ;HIGH ORDER RESULT OK ?  
        BNE   3$           ;BR IF NO  
        CMP    #100000,R1   ;LOW ORDER RESULT OK?  
        BEQ   DIV0         ;BR IF YES  
3$:     ERROR 127          ;INCORRECT DATA FOR "MUL"
```

```
*****  
:TEST DIV INSTRUCTION  
*****  
DIV0:  MOV    #125252,R1    ;LOAD LOW ORDER DIVIDEND  
        MOV    #-1,R0      ;HIGH ORDER DIVIDEND  
        SEC    ;SET C BIT TO 1  
        DIV    #2,R0       ;R1:R0=177777,125252/2  
        BCS   1$           ;BR IF C BIT ERROR  
        BVS   1$           ;BR IF V BIT ERROR  
        BEQ   1$           ;BR IF Z BIT ERROR  
        BMI   2$           ;BR IF N BIT OK  
1$:     ERROR 126          ;INCORRECT DONDITION CODE FOR "DIV"  
2$:     TST    R1           ;CHECK LOW ORDER RESULT  
        BNE   3$           ;BR IF ERROR  
        CMP    R0,#152525  ;CHECK HIGH ORDER RESULT  
        BEQ   FIS          ;BR IF OK  
3$:     ERROR 127          ;INCORRECT DATA FOR "DIV"
```

1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996 005236 012704 027744  
1997 005242 012744 107070  
1998 005246 012744 134343  
1999 005252 012744 065432  
2000 005256 012744 032107  
2001 005262 012744 123456  
2002 005266 012744 045670  
2003 005272 012744 125252  
2004 005276 012744 135252  
2005 005302 012744 016161  
2006 005306 012744 040616  
2007 005312 000257  
2008 005314 000264  
2009 005316 000240  
2010 005320 075014  
2011 005322 075034  
2012 005324 075024  
2013 005326 075004  
2014 005330 103403  
2015 005332 102402  
2016 005334 001401  
2017 005336 100401  
2018 005340 104126  
2019 005342 012437 005420  
2020 005346 012437 005422  
2021 005352 020427 027744  
2022 005356 001401  
2023 005360 104127  
2024 005362 022737 137201 005420  
2025 005370 001401  
2026 005372 104127  
2027 005374 022737 115230 005422  
2028 005402 001401  
2029 005404 104127  
2030 005406 005337 005424  
2031 005412 001405  
2032 005414 000137 004602  
2033 005420 000000  
2034 005422 000000  
2035 005424 000000  
2036  
2037 005426 032777 010000 173504  
2038 005434 001404  
2039 005436 104401 020757  
2040 005442 104401 021011  
2041 005446

```

*****
: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 FORM 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: 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

```

```
2042
2043
2044      ;*****
2045      ; START LINE CLOCK
2046      ;*****
2047 005446 032737 000200 001246 CLKTST: BIT      #BIT7,$DEV  ;DROP TEST?
2048 005454 001031          BNE      PRTST    ;BR IF YES
2049 005456 012737 013176 000100 MOV      #INTSRV,CLKVEC ;SETUP VECTOR AREA
2050 005464 012737 000200 000102 MOV      #200,CLKVEC+2 ;LOCKOUT OTHER INTER.
2051 005472 005037 013204          CLR      INTFLG
2052 005476 052737 000100 177546 BIS      #IE,CLK      ;SET CLOCK LOOSE!
2053
2054 005504 004537 012352          JSR      R5,TIMER    ;SEE IF INTFLG GOES TO 100
2055 005510 013204          INTFLG
2056 005512 000100          100
2057 005514 104032          ERROR   32      ;CLK NOT RESPONDING
2058 005516 000410          BR      PRTST
2059
2060 005520 032777 010000 173412 BIT      #BIT12,@SWR   ;SKIP TYPEOUT IF NOT SET
2061 005526 001404          BEQ      PRTST
2062 005530 104401 020300          TYPE   ,CLOCK
2063 005534 104401 020704          TYPE   ,RUN
```

```
2064
2065
2066          ;*****
2067          ;          START PRINTER
2068          ;*****
2069 005540 032737 100000 001246 PRTST: BIT    #BIT15,$DEV  ;DROP TEST?
2070 005546 001056          BNE    SMTST      ;BR IF YES
2071 005550 005737 012016          TST    PRPRES     ;PRINTER THERE?
2072 005554 001453          BEQ    SMTST      ;BR IF NO
2073
2074 005556 012737 027014 177420 2$:  MOV    #<PRT!B9600!CHAR8>,PARAM ;SETUP BAUD RATE & CHAR LENGTH
2075 005564 012737 013206 000200      MOV    #PRSRV,PRVEC ;ELSE SETUP PRINTER VECTOR
2076 005572 012737 000200 000202      MOV    #200,PRVEC+2 ;LOCKOUT INTERR
2077
2078 005600 005037 013334          CLR    LINCT      ;CLEAR LINE CTR
2079 005604 012737 000015 013332      MOV    #CR,PRCHR  ;CHAR TO BE PRINTED
2080                                     ;'DEL' (177) CAN BE USED TO CLR PRINTER BUFF
2081 005612 052737 000100 177514      BIS    #IE,PRS    ;SET PRINTER LOOSE!
2082                                     ;WILL PRINT CONTINUOUS 132 COLUMN LINES
2083                                     ;EA BEGINNING WITH ASCII 40 THRU 176
2084 005620 004537 012352          JSR    R5,TIMER   ;SEE IF LINCT GOES TO 5
2085 005624 013334          LINCT
2086 005626 000005          5
2087 005630 104033          ERROR 33        ;PRINTER NOT RESPONDING
2088 005632 000424          BR     SMTST
2089
2090 005634 005737 012020          TST    PRPORT     ;EXT LOOPBACK MODE?
2091 005640 001403          BEQ    3$        ;BR IF NO
2092 005642 042737 000100 177514      BIC    #IE,PRS    ;ELSE SHUT DOWN PRINTER
2093
2094 005650 032777 010000 173262 3$:  BIT    #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
2095 005656 001412          BEQ    SMTST
2096 005660 104401 020240          TYPE  ,PRINT
2097 005664 005737 012020          TST    PRPORT     ;EXT LOOPBACK MODE?
2098 005670 001403          BEQ    4$        ;BR IF NO
2099 005672 104401 020251          TYPE  ,PORT
2100 005676 000402          BR     SMTST
2101 005700 104401 020704          4$: TYPE  ,RUN
```

```

2102
2103
2104          ;*****
2105          ;          START SYNC COMM PORT
2106          ;*****
2107 005704 032737 002000 001246 SMTST: BIT    #BIT10,$DEV0    ;DROP TEST?
2108 005712 001402          BEQ    8$          ;BR IF NO
2109 005714 000137 006160          JMP    AMTST          ;ELSE JUMP TEST
2110
2111 005720 123727 001210 000001 8$:  CMPB  $ENV,#1      ;UNDER APT?
2112 005726 001003          BNE   9$          ;BR IF NO
2113 005730 005737 001176          TST  $PASS        ;ELSE 1'ST PASS?
2114 005734 001111          BNE   AMTST        ;EXIT TEST IF NO
2115
2116 005736 005037 014026          9$:  CLR  LSTCHR        ;LAST CHAR FLAG FOR XMIT
2117 005742 052737 000400 176624  BIS  #MR,SMXC      ;MASTER RESET
2118 005750 052737 004000 176624  BIS  #MCLK,SMXC    ;MAINT CLOCK
2119
2120 005756 032737 000010 001246  BIT  #BIT3,$DEV0   ;EXT LOOPBACK?
2121 005764 001403          BEQ  1$          ;BR IF YES
2122 005766 052737 014000 176624  BIS  #<MM!MCLK>,SMXC ;ELSE SET MAINT MODE FOR INT. LOOPBACK
2123 005774 012737 007026 176622 1$:  MOV  #<CHR8!ODDPAR!026>,SMPAR ;SETUP SYNC PARAMETER REGISTER
2124 006002 052737 000026 176620  BIS  #<RTS!DTR!SRSYN>,SMRC ;SET REQ TO SEND, DATA TERM RDY
2125                                     ;& SEARCH SYNC
2126 006010 052737 000020 176624  BIS  #SEND,SMXC    ;ENABLE XMIT SEND
2127 006016 012737 000026 176626  MOV  #026,SMXB     ;XMIT SYNC CHAR
2128
2129 006024 004537 012634          JSR  R5,TIMER2     ;WAIT FOR DONE
2130 006030 176620          SMRC
2131 006032 000200          DONE
2132 006034 104045          ERROR 45         ;NO DONE
2133 006036 000435          BR    5$         ;EXIT
2134
2135 006040 013737 176622 013736  MOV  SMRB,COMHLD   ;READ WORD
2136 006046 023727 013736 000026  CMP  CGMHL0,#026   ;IS IT SYNC CHAR?
2137 006054 001402          BEQ  4$         ;BR IF YES
2138 006056 104006          ERROR 6         ;DID NOT REC SYNC CHAR
2139 006060 000424          BR    5$         ;EXIT
2140
2141 006062 004537 012326          4$:  JSR  R5,SETVEC    ;SETUP VECTOR AREA
2142 006066 000340          SMRVEC
2143 006070 014030          SMRSRV
2144 006072 013740          SMXSRV

```



```
2145
2146 006074 012737 000027 013734      MOV    #C27,COMCHR      ;1'ST CHAR TO BE XMITTED
2147 006102 052737 000100 176620      BIS    #IE,SMRC        ;SET SYNC MODEM LOOSE!
2148 006110 052737 000100 176624      BIS    #IE,SMXC
2149
2150 006116 004537 012352                JSR    R5,TIMER        ;SEE IF COMCHR GOES TO ALL 1'S (DONE)
2151 006122 013734                        COMCHR
2152 006124 177777                        177777
2153 006126 104037                        ERROR  37              ;SYNC COMM NOT RESPONDING
2154 006130 000240                        NOP
2155
2156 006132 012737 000400 176624 5$:    MOV    #MR,SMXC        ;MASTER RESET
2157 006140 032777 010000 172772      BIT    #BIT12,@SWR    ;DO TYPEOUT IF SET
2158 006146 001404                        BEQ    AMTST          ;EXIT IF NOT
2159 006150 104401 020211                TYPE  ,SCOM
2160 006154 104401 021011                TYPE  ,DUN
```

```
2161  
2162  
2163  
2164  
2165  
2166 006160 032737 004000 001246 AMTST: BIT #BIT11,$DEV ;DROP TEST?  
2167 006166 001054 BNE CL1TST ;BR IF YES  
2168  
2169 006170 032737 000010 001246 BIT #BIT3,$DEV ;EXT LOOPBACK?  
2170 006176 001404 BEQ 1$ ;BR IF YES  
2171 006200 012737 037036 177420 MOV #<AMOD!B9600!OPAR!CHAR8!MAINT>,PARAM ;ELSE SET FOR INT LOOPBACK  
2172 006206 000403 BR 3$  
2173 006210 012737 037034 177420 1$: MOV #<AMOD!B9600!OPAR!CHAR8>,PARAM ;WAKE UP AMOD IF JUST DID SMOD  
2174 006216 052737 000006 176610 3$: BIS #<RTS!DTR>,AMRC ;SET REQ TO SEND & DATA TERM RDY  
2175  
2176 006224 004537 012326 JSR R5,SETVEC ;SETUP VECTOR AREA  
2177 006230 000330 AMRVEC  
2178 006232 013656 AMRSRV  
2179 006234 013640 AMXSRV  
2180  
2181 006236 005037 013734 CLR COMCHR ;CHAR TO BE XMITTED  
2182 006242 013737 176612 013736 MOV AMRB,COMHLD ;CLR OUT ANY PREVIOUS STORED WORD  
2183 006250 052737 000100 176610 BIS #IE,AMRC ;SET ASYNC MODEM LOOSE!  
2184 006256 052737 000100 176614 BIS #IE,AMXC  
2185  
2186 006264 004537 012352 JSR R5,TIMER ;SEE IF COMCHR GOES TO 377  
2187 006270 013734 COMCHR  
2188 006272 000377 377  
2189 006274 104040 ERROR 40 ;ASYNC COMM NOT RESPONDING  
2190 006276 000410 BR CL1TST  
2191  
2192 006300 032777 010000 172632 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET  
2193 006306 001404 BEQ CL1TST  
2194 006310 104401 020224 TYPE ,ACOM  
2195 006314 104401 020704 TYPE ,RUN
```

```
2196
2197
2198
2199
2200
2201 006320 005737 012014          CL1TST: TSI      CLPRES      ;OPTION PRESENT?
2202 006324 001002                    BNE          1$          ;BR IF YES
2203 006326 000137 006774          JMP          DXTST0
2204
2205 006332 032737 004000 001246 1$:   BIT      #4000,$DEV      ;DROP ASYNC?
2206 006340 001406                    BEQ          4$          ;BR IF NO
2207 006342 042737 000006 176610   BIC      #<RTS!DTR>,$AMRC ;MUST BE TOGGLED
2208 006350 052737 000006 176610   BIS      #<RTS!DTR>,$AMRC ;PRIME IT
2209
2210 006356 032737 010000 001246 4$:   BIT      #BIT12,$DEV      ;DROP TEST?
2211 006364 001051                    BNE          CL2TST      ;BR IF YES
2212
2213 006366 032737 000001 001246   BIT      #BIT0,$DEV      ;TERM #1 SET FOR EXT LOOPBACK?
2214 006374 001404                    BEQ          2$          ;BR IF YES
2215 006376 012737 005016 177420   MOV      #<CT1!B2400!CHAR8!MAINT>,$PARAM ;ELSE SET FOR INT LOOPBACK
2216 006404 000403                    BR          3$
2217 006406 012737 005014 177420 2$:   MOV      #<CT1!B2400!CHAR8>,$PARAM ;SETUP FOR EXT LOOPBACK
2218
2219 006414 004537 012326          3$:   JSR      R5,$SETVEC      ;SETUP INTERR VECTORS
2220 006420 000300                    TK1VEC
2221 006422 013356                    TK1SRV
2222 006424 013340                    TP1SRV
2223
2224 006426 005037 013434          CLR      T1CHR          ;CHAR TO BE XMITTED
2225 006432 013737 176502 013436   MOV      TK1B,T1HLD      ;READ CHAR (CLEAR STALE DATA)
2226 006440 052737 000100 176500   BIS      #IE,TK1S        ;SET CLUSTER TERM #1 LOOSE!
2227 006446 052737 000100 176504   BIS      #IE,TP1S
2228
2229 006454 004537 012352          JSR      R5,$TIMER      ;SEE IF T1CHR GOES TO 377
2230 006460 013434                    T1CHR
2231 006462 000377                    377
2232 006464 104034                    ERROR 34                ;CLUSTER TERM 1 NOT RESPONDING
2233 006466 000410                    BR          CL2TST
2234
2235 006470 032777 010000 172442   BIT      #BIT12,$SWR      ;SKIP TYPEOUT IF NOT SET
2236 006476 001404                    BEQ          CL2TST
2237 006500 104401 020156          TYPE     ,CLT1
2238 006504 104401 020704          TYPE     ,RUN
```

```

2239
2240
2241
2242
2243
2244 006510 032737 020000 001246 CL2TST: BIT    #BIT13,$DEV    ;DROP TEST?
2245 006516 001051                BNE    CL3TST    ;BR IF YES
2246
2247 006520 032737 000002 001246                BIT    #BIT1,$DEV    ;TERM #2 SET FOR EXT LOOPBACK?
2248 006526 001404                BEQ    1$         ;BR IF YES
2249 006530 012737 055016 177420                MOV    #<CT2!B2400!CHAR8!MAINT>,$PARAM ;ELSE SET FOR INT LOOPBACK
2250 006536 000403                BR     2$
2251 006540 012737 055014 177420 1$: MOV    #<CT2!B2400!CHAR8>,$PARAM ;SETUP FOR EXT LOOPBACK
2252
2253 006546 004537 012326                2$: JSR    R5,SETVEC ;SETUP INTERR VECTORS
2254 006552 000310                TK2VEC
2255 006554 013456                TK2SRV
2256 006556 013440                TP2SRV
2257
2258 006560 005037 013534                CLR    T2CHR      ;CHAR TO BE XMITTED
2259 006564 013737 176512 013536                MOV    TK2B,T2HLD ;READ CHAR (CLEAR STALE DATA)
2260 006572 052737 000100 176510                BIS    #IE,TK2S   ;SET CLUSTER TERM #2 LOOSE!
2261 006600 052737 000100 176514                BIS    #IE,TP2S
2262
2263 006606 004537 012352                JSR    R5,TIMER   ;SEE IF T2CHR GOES TO 377
2264 006612 013534                T2CHR
2265 006614 000377                377
2266 006616 104035                ERROR  35         ;CLUSTER TERM 2 NOT RESPONDING
2267 006620 000410                BR     CL3TST
2268
2269 006622 032777 010000 172310                BIT    #BIT12,$SWR ;SKIP TYPEOUT IF NOT SET
2270 006630 001404                BEQ    CL3TST
2271 006632 104401 020167                TYPE  ,CLT2
2272 006636 104401 020704                TYPE  ,RUN

```

```
2273
2274
2275      ;:*****
2276      ;:          START CLUSTER TERMINAL #3
2277      ;:*****
2278 006642 032737 040000 001246 CL3TST: BIT    #BIT14,$DEVN    ;DROP TEST?
2279 006650 001051                BNE    DXTSTO        ;BR IF YES
2280
2281 006652 032737 000004 001246        BIT    #BIT2,$DEVN    ;TERM #3 SET FOR EXT LOOPBACK?
2282 006660 001404                BEQ    1$          ;BR IF YES
2283 006662 012737 065016 177420        MOV    #<CT3!B2400!CHAR8!MAINT>,$PARAM ;ELSE SET FOR INT LOOPBACK
2284 006670 000403                BR     2$
2285 006672 012737 065014 177420 1$:   MOV    #<CT3!B2400!CHAR8>,$PARAM ;SETUP FOR EXT LOOPBACK
2286
2287 006700 004537 012326                2$:   JSR    R5,SETVEC    ;SETUP INTERR VECTORS
2288 006704 000320                TK3VEC
2289 006706 013556                TK3SRV
2290 006710 013540                TP3SRV
2291
2292 006712 005037 013634                CLR    T3CHR        ;CHAR TO BE XMITTED
2293 006716 013737 176522 013636        MOV    TK3B,T3HLD   ;READ CHAR (CLEAR STALE DATA)
2294 006724 052737 000100 176520        BIS    #IE,TK3S     ;SET CLUSTER TERM #3 LOOSE!
2295 006732 052737 000100 176524        BIS    #IE,TP3S
2296
2297 006740 004537 012352                JSR    R5,TIMER     ;SEE IF T3CHR GOES TO 377
2298 006744 013634                T3CHR
2299 006746 000377                377
2300 006750 104036        ERROR 36            ;CLUSTER TERM 3 NOT RESPONDING
2301 006752 000410        BR     DXTSTO
2302
2303 006754 032777 010000 172156        BIT    #BIT12,@SWR  ;SKIP TYPEOUT IF NOT SET
2304 006762 001404                BEQ    DXTSTO
2305 006764 104401 020200                TYPE  ,CLT3
2306 006770 104401 020704                TYPE  ,RUN
```

2307  
2308  
2309  
2310  
2311  
2312  
2313  
2314  
2315  
2316  
2317  
2318  
2319  
2320  
2321  
2322  
2323  
2324  
2325  
2326  
2327  
2328  
2329  
2330  
2331  
2332  
2333  
2334  
2335  
2336  
2337  
2338  
2339  
2340  
2341  
2342  
2343  
2344  
2345  
2346  
2347  
2348  
2349  
2350  
2351  
2352  
2353  
2354  
2355

006774 005037 003260  
007000 032737 000400 001246  
007006 001404  
007010 032737 001000 001246  
007016 001113  
007020 004537 012634  
007024 177170  
007026 000200  
007030 104075  
007032 000500  
007034 012737 000011 177170  
007042 004537 012634  
007046 177170  
007050 000200  
007052 104114  
007054 000467  
007056 005737 177170  
007062 100001  
007064 104140  
007066 012700 000100  
007072 005001  
007074 012737 000001 177170  
007102 010137 177172  
007106 005101  
007110 005300  
007112 001373  
007114 004537 012634  
007120 177170  
007122 000200  
007124 104115  
007126 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.  
:\*\*\*\*\*

DXTST0: CLR DXTST ;0 = DXTST0  
BIT #BIT8,\$DEV ;DROP DX0 TESTS?  
BEQ 7\$ ;BR IF NO  
BIT #BIT9,\$DEV ;DROP DX1 TESTS?  
BNE DXTST1 ;BR IF YES  
7\$: 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\$  
TST RXCS ;ERROR?  
BPL 6\$ ;BR IF NO  
ERROR 140 ;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

```

2356
2357 007130 012700 000100      MOV      #100,R0      ;WORD CT
2358 007134 012701 002564      MOV      #DOBUF,R1   ;BUFFER ADDR
2359 007140 012737 000003 177170  MOV      #EBUF,RXCS  ;ISSUE EMPTY BUFF COMMAND
2360 007146 013721 177172      MOV      RXDB,(R1)+  ;STORE WORD
2361 007152 005300      DEC      R0          ;DONE?
2362 007154 001374      BNE     2$          ;BR IF NO
2363
2364 007156 004537 012634      JSR     R5,TIMER2   ;WAIT FOR DISK DONE
2365 007162 177170      RXCS
2366 007164 000200      DONE
2367 007166 104116      ERROR   116        ;NO DONE
2368 007170 000421      BR     8$          ;EXIT
2369
2370
2371 007172 012701 002564      MOV      #DOBUF,R1   ;CHECK IT
2372 007176 012137 007240 3$:  MOV      (R1)+,EBHLD ;BUFFER ADDR
2373 007202 001401      BEQ     4$          ;STORE IT
2374 007204 104076      ERROR   76        ;BR IF ZERO
2375
2376 007206 012137 007240 4$:  MOV      (R1)+,EBHLD ;WORD NOT ALL ZEROS
2377 007212 023727 007240 177777  CMP     EBHLD,#-1   ;ALL 1'S?
2378 007220 001401      BEQ     5$          ;BR IF YES
2379 007222 104077      ERROR   77        ;WORD NOT ALL ONES
2380
2381 007224 020127 002764 5$:  CMP     R1,#DOBUF+200 ;END OF DOBUFF?
2382 007230 001362      BNE     3$          ;BR IF NO
2383 007232 000405      BR     DXTST1      ;GO TO NXT TST
2384
2385 007234 000137 011176 8$:  JMP     EOP        ;EXIT ALL DISK TESTS
2386
2387 007240 000000      EBHLD:  0          ;STORAGE      FOR ERROR REPORT
2388 007242 000000      EXPO:   0          ;EXPECT 0'S   FOR ERROR REPORT
2389 007244 177777      EXP1:  -1         ;EXPECT 1'S   FOR ERROR REPORT

```

2390  
2391  
2392  
2393  
2394  
2395  
2396  
2397  
2398  
2399  
2400  
2401  
2402  
2403  
2404 007246 012737 000001 003260  
2405  
2406 007254 005037 002546  
2407 007260 005037 003016  
2408  
2409 007264 005037 002540  
2410 007270 005037 002544  
2411 007274 005037 002554  
2412 007300 012737 000001 002550  
2413  
2414 007306 005037 003010  
2415 007312 005037 003014  
2416 007316 005037 003024  
2417 007322 012737 000001 003020  
2418  
2419 007330 032737 000400 001246  
2420 007336 001031  
2421  
2422 007340 012737 014130 000264  
2423 007346 012737 000200 000266  
2424  
2425 007354 005737 003274  
2426 007360 001404  
2427 007362 012737 014622 014230  
2428 007370 000403  
2429  
2430 007372 012737 014246 014230  
2431 007400 005037 003262  
2432 007404 052737 000100 177170  
2433  
2434 007412 004737 012436  
2435 007416 104041  
2436 007420 000240  
2437  
2438 007422 032737 001000 001246  
2439 007430 001031  
2440  
2441 007432 012737 014130 000264  
2442 007440 012737 000200 000266

```

*****
: 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.
: 1ST PASS PERFORMED ON TRACKS 0-4, 40-44, 72-76(10)
: SUBSEQUENT PASSES ON ALL TRACKS
*****

```

```

DXTST1: MOV #1,DXTST ;1 = DATA PATT TESTS
CLR D0TRK ;STARTING TRACK
CLR D1TRK
12$: CLR DOPAT ;INIT DX0 FLAGS
CLR DOERR
CLR DOCERR
MOV #1,DOSEC
CLR D1PAT ;INIT DX1 FLAGS
CLR D1ERR
CLR D1CERR
MOV #1,D1SEC
BIT #BIT8,$DEVM ;DROP DX0 TESTS?
BNE 2$ ;BR IF YES
MOV #RXSRV,RXVEC ;SETUP VECTOR AREA
MOV #200,RXVEC+2
1$: TST COMP2 ;DOING COMPAT PASS 2?
BEQ 10$ ;BR IF NO
MOV #DORSEC,DODISP ;ELSE DO READS ONLY
BR 7$
10$: MOV #DOFBUF,DODISP ;DO NORMAL TESTING
7$: CLR FIN ;DO DX1 WHEN SET
BIS #IE,RXCS ;LET INTERRUPTS LOOSE!
JSR PC,TIMDO
ERROR 41 ;DX0 NOT GETTING TRACK DONE FLAG
NOP
2$: BIT #BIT9,$DEVM ;DROP DX1 TESTS?
BNE 4$ ;BR IF YES
MOV #RXSRV,RXVEC ;SETUP VECTOR AREA
MOV #200,RXVEC+2

```



```
2443
2444 007446 005737 003274      3$:  TST    COMP2      ;DOING COMPAT PASS 2?
2445 007452 001404              BEQ    11$          ;BR IF NO
2446 007454 012737 016406 014236  MOV    #D1RSEC,D1DISP ;ELSE DO READ ONLY
2447 007462 000403              BR     8$
2448
2449 007464 012737 016032 014236 11$:  MOV    #D1FBUF,D1DISP ;DO NORMAL TESTING
2450 007472 005037 003262      8$:  CLR    FIN          ;GO BACK TO DX0 WHEN SET
2451 007476 052737 000120 177170  BIS    #<IE!DX1>,RXCS ;LET INTERRUPTS LOOSE!
2452
2453 007504 004737 012532              JSR    PC,TIMD1
2454 007510 104042              ERROR  42          ;DX1 NOT GETTING TRACK DONE FLAG
2455 007512 000240              NOP
2456
2457 007514 032737 000400 001246 4$:  BIT    #BIT8,$DEVMM ;DROP DX0?
2458 007522 001007              BNE    5$          ;BR IF YES
2459 007524 005737 002540              TST    DOPAT       ;ELSE IS DX0 ALL DONE?
2460 007530 001004              BNE    5$          ;BR IF YES
2461 007532 005737 012530              TST    DOTMO       ;TIMEOUT?
2462 007536 001706              BEQ    1$          ;BR IF NO
2463 007540 000717              BR     7$          ;ELSE CONT LAST COMMAND
2464
2465 007542 032737 001000 001246 5$:  BIT    #BIT9,$DEVMM ;DROP DX1?
2466 007550 001007              BNE    6$          ;BR IF YES
2467 007552 005737 003010              TST    D1PAT       ;ELSE IS DX1 ALL DONE?
2468 007556 001004              BNE    6$          ;BR IF YES
2469 007560 005737 012620              TST    D1TMO       ;TIMEOUT?
2470 007564 001730              BEQ    3$          ;BR IF NO
2471 007566 000741              BR     8$          ;ELSE CONT LAST COMMAND
2472
2473 007570 005737 003272      6$:  TST    COMP1      ;DOING COMPAT PASS 1?
2474 007574 001402              BEQ    13$         ;BR IF NO
2475 007576 000137 011176              JMP    EOP         ;ELSE ALL DONE
2476
2477 007602 005737 001176      13$: TST    $PASS       ;1'ST PASS?
2478 007606 001035              BNE    DXTST2     ;BR IF NO
2479
2480 007610 023727 003264 000004      CMP    MAXTRK,#4   ;JUST DID 5 OUTER TRACKS?
2481 007616 001012              BNE    14$         ;BR IF NO
2482 007620 012737 000050 002546      MOV    #40.,D0TRK ;ELSE SETUP FOR 5 MIDDLE TRKS
2483 007626 012737 000050 003016      MOV    #40.,D1TRK
2484 007634 012737 000054 003264      MOV    #44.,MAXTRK
2485 007642 000610              BR     12$
2486
2487 007644 023727 003264 000054 14$:  CMP    MAXTRK,#44. ;JUST DID 5 MIDDLE TRACKS?
2488 007652 001013              BNE    DXTST2     ;BR IF NO, MUST BE ALL DONE
2489 007654 012737 000110 002546      MOV    #72.,D0TRK ;ELSE SETUP FOR 5 INNER TRKS
2490 007662 012737 000110 003016      MOV    #72.,D1TRK
2491 007670 012737 000114 003264      MOV    #76.,MAXTRK
2492 007676 000137 007264              JMP    12$
2493
```

```

2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505 007702 012737 000002 003260 DXTST2: MOV #2,DX1ST ;2 = RANDOM SEEK TESTS
2506 007710 005737 001176 TST $PASS ;1'ST PASS?
2507 007714 001003 BNE 7$ ;BR IF NO
2508 007716 012737 000004 003264 MOV #4,MAXTRK ;ELSE JUST DO TRKS 0-4
2509
2510 007724 005037 002544 7$: CLR DOERR ;INIT DX0 FLAGS
2511 007730 005037 002554 CLR DOCERR
2512 007734 005037 002542 CLR DOCNT
2513
2514 007740 005037 003014 CLR DIERR ;INIT DX1 FLAGS
2515 007744 005037 003024 CLR DICERR
2516 007750 005037 003012 CLR DICNT
2517
2518 007754 032737 000400 001246 BIT #BIT8,$DEVMS ;DROP DX0 TESTS?
2519 007762 001041 BNE 2$ ;BR IF YES
2520
2521 007764 005037 012126 1$: CLR LOLIM
2522 007770 013737 003264 012130 MOV MAXTRK,HILIM
2523 007776 004737 012024 JSR PC,RAND
2524 010002 010137 002546 MOV R1,DOTRK ;GET RANDOM TRACK #
2525
2526 010006 005237 012126 INC LOLIM
2527 010012 012737 000032 012130 MOV #32,HILIM
2528 010020 004737 012024 JSR PC,RAND
2529 010024 010137 002550 MOV R1,DOSEC ;GET RANDOM SECTOR #
2530
2531 010030 004537 013010 JSR R5,CKOBAD ;SEE IF THIS TRK/SEC FLAGGED BAD
2532 010034 000753 BR 1$ ;BR IF RETURN HERE
2533
2534 010036 012737 014622 014230 MOV #DORSEC,DODISP ;SET DISPATCH TABLE TO POINT TO
2535 ;READ SECTOR HANDLER.
2536 010044 005037 003262 CLR FIN ;DO DX1 WHEN SET
2537 010050 052737 000100 177170 BIS #IE,RXCS ;LET INTERRUPTS LOOSE!
2538
2539 010056 004737 012436 JSR PC,TIMDO
2540 010062 104117 ERROR 117 ;DX0 NOT GETTING DONE FLAG
2541 010064 000240 NOP

```

```

2542
2543 010066 032737 001000 001246 2$: BIT #BIT9,$DEV  ;DROP DX1 TESTS?
2544 010074 001041 BNE 4$ ;BR IF YES
2545
2546 010076 005037 012126 3$: CLR LOLIM
2547 010102 013737 003264 012130 MOV MAXTRK,HILIM
2548 010110 004737 012024 JSR PC,RAND
2549 010114 010137 003016 MOV R1,D1TRK ;GET RANDOM TRACK #
2550
2551 010120 005237 012126 INC LOLIM
2552 010124 012737 000032 012130 MOV #32,HILIM
2553 010132 004737 012024 JSR PC,RAND
2554 010136 010137 003020 MOV R1,D1SEC ;GET RANDOM SECTOR #
2555
2556 010142 004537 013122 JSR R5,CK1BAD ;SEE IF THIS TRK/SEC FLAGGED BAD
2557 010146 000753 BR 3$ ;BR IF RETURN HERE
2558
2559 010150 012737 016406 014236 MOV #D1RSEC,D1DISP ;SET DISPATCH TABLE TO POINT TO
2560 ;READ SECTOR HANDLER.
2561 010156 005037 003262 CLR FIN ;GO BACK TO DX0 WHEN SET
2562 010162 052737 000120 177170 BIS #<IE!DX1>,RXCS ;LET INTERRUPTS LOOSE!
2563
2564 010170 004737 012532 JSR PC,TIMD1
2565 010174 104120 ERROR 120 ;DX1 NOT GETTING DONE FLAG
2566 010176 000240 NOP
2567
2568 010200 032737 000400 001246 4$: BIT #BIT8,$DEV ;DROP DX0?
2569 010206 001004 BNE 5$ ;BR IF YES
2570 010210 023737 002542 003266 CMP DOCNT,MAXSK ;DID ALL SEEKS?
2571 010216 001262 BNE 1$ ;BR IF NO
2572
2573 010220 032737 001000 001246 5$: BIT #BIT9,$DEV ;DROP DX1?
2574 010226 001004 BNE 6$ ;BR IF YES
2575 010230 023737 003012 003266 CMP D1CNT,MAXSK ;DID ALL SEEKS?
2576 010236 001317 BNE 3$ ;BR IF NO
2577
2578 010240 005737 003274 6$: TST COMP2 ;DOING COMPAT PASS 2?
2579 010244 001402 BEQ DXTST3 ;BR IF NO
2580 010246 000137 011176 JMP EOP ;ELSE ALL DONE
2581

```

```

2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592 010252 032737 000400 001246 DXTST3: BIT #BIT8,$DEV  ;DROP DX0 TESTS?
2593 010260 001062 BNE DXTST4 ;BR IF YES
2594
2595 010262 012737 000003 003260 MOV #3,DXTST ;3 = INIT TEST
2596 010270 005037 002544 CLR DOERR ;FLAGS
2597 010274 005037 002554 CLR DOCERR
2598
2599 010300 012737 015666 014230 MOV #DOINIT,DODISP ;SET DISPATCH TABLE TO GO TO INIT HANDLER
2600
2601 010306 005037 003262 CLR FIN
2602 010312 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2603
2604 010320 004737 012436 JSR PC,TIMDO
2605 010324 104046 ERROR 46 ;DX0 HUNG ON INITIALIZE COMMAND
2606 010326 000437 BR DXTST4
2607
2608 010330 005737 177170 TST RXCS ;ERROR?
2609 010334 100001 BPL 1$ ;BR IF NO
2610 010336 104063 ERFOR 63 ;INIT COMMAND ERROR
2611
2612 010340 032737 000001 177172 1$: BIT #ID,RXES ;INIT DONE BIT SET?
2613 010346 001001 BNE 2$ ;BR IF YES
2614 010350 104055 ERROR 55 ;INIT DONE NOT SET
2615
2616 010352 005037 002544 2$: CLR DOERR
2617 010356 005037 002554 CLR DOCERR
2618 010362 012737 000001 002546 MOV #1,DOTRK ;EXP TRK & SEC
2619 010370 012737 000001 002550 MOV #1,DOSEC
2620
2621 010376 012737 015102 014230 MOV #DOEBUF,DODISP ;SETUP DISPATCH TABLE TO
2622 ;EMPTY BUFFER & VERIFY DATA
2623 010404 005037 003262 CLR FIN
2624 010410 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2625
2626 010416 004737 012436 JSR PC,TIMDO
2627 010422 104056 ERROR 56 ;DX0 HUNG ON EMPTY BUFF COMMAND
2628 010424 000240 NOP

```

2629  
2630  
2631  
2632  
2633  
2634  
2635  
2636  
2637  
2638  
2639  
2640  
2641  
2642  
2643  
2644  
2645  
2646  
2647  
2648  
2649  
2650  
2651  
2652  
2653  
2654  
2655  
2656  
2657  
2658  
2659  
2660  
2661  
2662  
2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679

010426 032737 000400 001246  
010434 001121  
010436 012737 000004 003260  
010444 012737 015712 014230  
010452 005037 003262  
010456 052737 000100 177170  
010464 004737 012436  
010470 104047  
010472 000431  
010474 005737 177170  
010500 100001  
010502 104054  
010504 005037 002546  
010510 012737 000001 002550  
010516 005037 002544  
010522 005037 002554  
010526 005037 003262  
010532 012737 014622 014230  
010540 052737 000100 177170  
010546 004737 012436  
010552 104121  
010554 000240  
010556 032737 001000 001246  
010564 001045  
010566 012737 015730 014236  
010574 005037 003262  
010600 052737 000120 177170  
010606 004737 012532  
010612 104122  
010614 000431  
010616 005737 177170  
010622 100001  
010624 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,$DEV  ;DROP DX0 TESTS?  
BNE DXTST5 ;BR IF YES  
MOV #4,DXTST ;4 = RESTORE TEST  
MOV #DORREST,DODISP ;SETUP DISPATCH TABLE  
CLR FIN  
BIS #IE,RXCS  
JSR PC,TIMD0  
ERROR 47 ;DX0 HUNG ON RESTORE COMMAND  
BR 2$  
TST RXCS ;ERROR?  
BPL 1$ ;BR IF NO  
ERROR 54 ;DX0 RESTORE COMMAND ERROR  
1$: CLR DTRK ;CHK FOR RECORD NOT FOUND (RNF)  
MOV #1,DOSEC ;SET TRK/SEC. SHOULD NOT MOVE  
;LOGIC THINKS ITS ALREADY THERE.  
CLR DOERR  
CLR DOCERR  
CLR FIN  
MOV #DORSEC,DODISP ;SETUP TO READ SECTOR  
BIS #IE,RXCS ;LET INTERR LOOSE  
JSR PC,TIMD0  
ERROR 121 ;DX0 HUNG ON READ CMD  
NOP  
2$: BIT #BIT9,$DEV ;DROP DX1 TESTS?  
BNE DXTST5 ;BR IF YES  
MOV #D1REST,D1DISP ;REPEAT FOR DX1  
CLR FIN  
BIS #<IE!DX1>,RXCS ;LET INTERR LOOSE  
JSR PC,TIMD1  
ERROR 122 ;DX1 HUNG ON RESTORE CMD  
BR DXTST5  
TST RXCS ;ERROR?  
BPL 3$ ;BR IF NO  
ERROR 74 ;DX1 RESTORE CMD ERROR
```

```

2680
2681 010626 005037 003016 3$: CLR D1TRK ;READ TO CHECK FOR RNF
2682 010632 012737 000001 003020 MOV #1,D1SEC
2683 010640 005037 003014 CLR D1ERR
2684 010644 005037 003024 CLR D1CERR
2685 010650 005037 003262 CLR FIN
2686 010654 012737 016406 014236 MOV #D1RSEC,D1DISP ;SETUP TO READ SECTOR
2687 010662 052737 000120 177170 BIS #<IE!DX1>,RXCS ;LET INTERR LOOSE
2688
2689 010670 004737 012532 JSR PC,TIMD1
2690 010674 104123 ERROR 123 ;DX1 HUNG ON READ CMD
2691 010676 000240 NOP
2692
2693
2694
2695
2696 ;*****
2697 ; WRITE SECTOR WITH DELETED DATA MARK TEST
2698 ;
2699 ; THIS TEST VERIFIES THAT THE WRITE DELETED DATA COMMAND CAN BE ISSUED
2700 ; & THAT THE 'DELETED DATA' BIT 5 IS SET IN RXCS AFTER READY IS RECV'D.
2701 ; HARD ERRORS ARE THOSE ERRORS AFTER 10 MORE RETRIES.
2702 ;*****
2703 010700 032737 000400 001246 DXTST5: BIT #BIT8,$DEVM ;DROP DX0 TESTS?
2704 010706 001043 BNE DXTST6 ;BR IF YES
2705
2706 010710 012737 000005 003260 MOV #5,DXTST ;5 = WR DEL DATA TEST
2707 010716 005037 002544 CLR DOERR
2708 010722 005037 002554 CLR DOCERR
2709
2710 010726 012737 014246 014230 MOV #DOFBUF,DODISP ;SETUP DISPATCH TABLE
2711
2712 010734 005037 003262 CLR FIN
2713 010740 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2714
2715 010746 004737 012436 JSR PC,TIMD0
2716 010752 104050 ERROR 50 ;DX0 HUNG ON WRITE DEL DATA CMD
2717 010754 000420 BR DXTST6
2718
2719 010756 012737 015746 014230 MOV #DOSTAT,DODISP ;SETUP DISPATCH TABLE
2720
2721 010764 005037 003262 CLR FIN
2722 010770 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2723
2724 010776 004737 012436 JSR PC,TIMD0
2725 011002 104051 ERROR 51 ;DX0 HUNG ON READ STATUS
2726 011004 000404 BR DXTST6
2727
2728 011006 005737 177170 TST RXCS ;ERROR?
2729 011012 100001 BPL .+4 ;BR IF NO
2730 011014 104057 ERROR 57 ;READ STATUS ERROR

```

```

2731
2732
2733
2734
2735
2736
2737
2738
2739
2740 011016 032737 000400 001246 DXTST6: BIT #BIT8,$DEV  ;DROP DXC TESTS?
2741 011024 001062 BNE 4$ ;BR IF YES
2742
2743 011026 005037 011174 CLR INVCT
2744 011032 012737 000006 003260 MOV #6,DXTST ;6 = INV ADDR TESTS
2745 011040 012737 000115 002546 MOV #115,DOTRK
2746 011046 012737 000001 002550 MOV #1,DOSEC
2747 011054 012737 046401 177174 MOV #46401,RXSA ;INVALID TRACK
2748
2749 011062 012737 015764 014230 1$: MOV #DOINV,DODISP ;SETUP DISPATCH TABLE
2750
2751 011070 005037 003262 CLR FIN
2752 011074 052737 000100 177170 BIS #IE,RXCS ;LET LOOSE
2753
2754 011102 004737 012436 JSR PC,TIMDO
2755 011106 104060 ERROR 60 ;DXC HUNG ON INVALID ADDR
2756 011110 000432 BR EOP
2757
2758 011112 032737 000002 177172 BIT #INVADR,RXES ;INV ADDR BIT SET?
2759 011120 001002 BNE 2$ ;BR IF YES
2760 011122 104061 ERROR 61 ;INVALID ADDR BIT NOT SET
2761 011124 000404 BR 3$
2762
2763 011126 005737 177170 2$: TST RXCS ;ERROR BIT SET?
2764 011132 100401 BMI 3$ ;BR IF YES
2765 011134 104062 ERROR 62 ;ERROR BIT NOT SET
2766
2767 011136 005737 011174 3$: TST INVCT ;DID WE JUST DO INV SECTOR?
2768 011142 001013 BNE 4$ ;BR IF YES...DONE
2769
2770 011144 005237 011174 INC INVCT
2771 011150 005037 002546 CLR DOTRK
2772 011154 012737 000033 002550 MOV #33,DOSEC
2773 011162 012737 000033 177174 MOV #33,RXSA ;INVALID SECTOR
2774 011170 000734 BR 1$ ;REPEAT FOR INVALID SECTOR
2775
2776 011172 000401 4$: BR .+4
2777
2778 011174 000000 INVCT: 0 ;0 = DOING INVALID TRACK
2779 ;1 = DOING INVALID SECTOR
2780

```

```

2781
2782
2783
2784
2785 011176 005237 001176
2786 011202 104401 021017
2787 011206 013746 001176
2788 011212 104405
2789 011214 104401 021034
2790 011220 013746 027100
2791 011224 104405
2792 011226 104401 021063
2793 011232 013746 027102
2794 011236 104405
2795 011240 104401 021117
2796 011244 013746 027104
2797 011250 104405
2798 011252 104401 021154
2799 011256 013746 027106
2800 011262 104405
2801 011264 104401 011334
2802 011270 012737 000114 003264
2803 011276 012737 000764 003266
2804
2805 011304 005737 003272
2806 011310 001407
2807 011312 104401 021403
2808 011316 000000
2809 011320 005037 003272
2810 011324 005237 003274
2811 011330 000137 004072
2812
2813 011334 377 377 000 NULL:
2814 011340
2815

```

```

*****
: END OF PASS
*****
EOP:  INC  $PASS      ;PASS CTR
      TYPE ,ENDPAS
      MOV  $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
      TYPDS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,MSG1
      MOV  TERR,-(SP)   ;;SAVE TERR FOR TYPEOUT
      TYPDS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,MSG2
      MOV  TSERR,-(SP)  ;;SAVE TSERR FOR TYPEOUT
      TYPDS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,MSG3
      MOV  PERR,-(SP)   ;;SAVE PERR FOR TYPEOUT
      TYPDS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,MSG4
      MOV  PSERR,-(SP)  ;;SAVE PSERR FOR TYPEOUT
      TYPDS                                ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,NULL        ;NULL CHAR
      MOV  #114,MAXTRK  ;FOR ALL SUBSEQUENT PASSES
      MOV  #500.,MAXSK
      TST  COMP1        ;COMPAT PASS 1?
      BEQ  1$           ;BR IF NO
      TYPE ,COMPAT     ;DONE MSG
      HALT
      CLR  COMP1
      INC  COMP2
      JMP  LOOP         ;REPEAT
1$:   .BYTE -1,-1,0    ;NULL CHAR STRING
      .EVEN

```



```

2816
2817
2818
2819
2820
2821 011340 012737 013176 000004 SIZE:  MOV    #INTSRV,ERRVEC ;SETUP TIMEOUT TRAP ADDR
2822 011346 012737 000200 000006      MOV    #200,ERRVEC+2 ;LOCKOUT INTERR
2823 011354 012737 013176 000010      MOV    #INTSRV,RESVEC ;SETUP RESERVED INST TRAP ADDR
2824 011362 012737 000200 000012      MOV    #200,RESVEC+2 ;LOCKOUT INTERR
2825
2826
2827
2828
2829 011370 005037 013204
2830 011374 012700 003776
2831 011400 005001
2832 011402 005710
2833 011404 005737 013204
2834 011410 001007
2835 011412 005201
2836 011414 020027 167776
2837 011420 001405
2838 011422 062700 004000
2839 011426 000765
2840
2841 011430 162700 004000
2842 011434 010037 012022
2843 011440 010146
2844
2845 011442 104405
2846 011444 104401 020327
2847
2848
2849
2850
2851 011450 005037 012012
2852 011454 005037 013204
2853 011460 072100
2854 011462 000240
2855 011464 005737 013204
2856 011470 001003
2857 011472 005237 012012
2858 011476 000404
2859 011500 104401 020307
2860 011504 104401 020350
2861 011510

```

```

*****
:SYSTEM SIZER SUBROUTINE
*****

:ROUTINE TO SIZE THE AMOUNT OF MEMORY
*****
      CLR    INTFLG
      MOV    #3776,R0 ;SIZE MEM. START WITH 1K
      CLR    R1 ;CTR, 1=1K, 2=2K, ETC
1$:   TST    (R0)
      TST    INTFLG ;GOT TIMEOUT INTERR?
      BNE    2$ ;BR IF YES
      INC    R1
      CMP    R0,#167776 ;DONE 32K MEM?
      BEQ    3$ ;BR IF YES, ALL DONE
      ADD    #4000,R0 ;GO TO NEXT 1K
      BR     1$

2$:   SUB    #4000,R0 ;GO BACK TO LAST VALID 1K
3$:   MOV    R0,MAXMEM ;SAVE
      MOV    R1,-(SP) ;;SAVE R1 FOR TYPEOUT
      TYPDS ;:TYPE MEM SIZE
      TYPE  ,MEM ;:GO TYPE--DECIMAL ASCII WITH SIGN

*****
:ROUTINE TO CHECK IF 'EIS-FIS' OPTION IS PRESENT
*****
      CLR    EIPRES ;CLEAR EIS-FIS PRESET FLAG
      CLR    INTFLG ;CLEAR TRAP FLAG
      ASH    R0,R1 ;EXECUTE EIS INSTRUCTION
      NOP
      TST    INTFLG ;TEST IF TRAP OCCURRED
      BNE    60$ ;BR IF YES
      INC    EIPRES ;NO TRAP - SET EIS-FIS PRESENT FLAG
      BR     61$ ;BR OVER TYPE-OUT
60$:  TYPE  ,EISTXT ;EIS-FIS TEXT
61$:  TYPE  ,NOTPRES

```

```
2862  
2863  
2864  
2865  
2866 011510 005000  
2867 011512 005037 013204  
2868 011516 0127C0 176500  
2869 011522 005710  
2870 011524 005737 013204  
2871 011530 001011  
2872 011532 020027 176520  
2873 011536 001403  
2874 011540 062700 000010  
2875 011544 000766  
2876  
2877 011546 005237 012014  
2878 011552 000406  
2879  
2880 011554 005037 012014  
2881 011560 104401 020136  
2882 011564 104401 020350  
2883  
2884 011570 012737 000006 000004  
2885 011576 005037 000006  
2886 011602 012737 000012 000010  
2887 011610 005037 000012  
:*****  
:ROUTINE TO CHECK IF CLUSTER PORT IS PRESENT  
:*****  
CLR R0  
CLR INTFLG  
MOV #TK15,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
```

```

2888
2889
2890
2891
2892 011614 005000
2893 011616 012701 000100
2894
2895 011622 030137 001246
2896 011626 001006
2897 011630 000241
2898 011632 006301
2899 011634 103425
2900 011636 062700 000002
2901 011642 000767
2902
2903 011644 016037 011664 011654
2904 011652 104401
2905 011654 000000
2906 011656 104401 020663
2907 011662 000762
2908
2909 011664 020307
2910 011666 020300
2911 011670 020266
2912 011672 020273
2913 011674 020211
2914 011676 020224
2915 011700 020156
2916 011702 020167
2917 011704 020200
2918 011706 020240
2919
2920
2921
2922
2923 011710 005037 012016
2924 011714 005037 012020
2925 011720 005737 177514
2926 011724 100027
2927
2928 011726 032737 000010 001246
2929 011734 001405
2930
2931 011736 104401 020240
2932 011742 104401 020350
2933 011746 000420
2934
2935 011750 012737 037034 177420
2936 011756 042737 000006 176610
2937 011764 052737 000006 176610
2938 011772 005737 177514
2939 011776 100757
2940 012000 005237 012020
2941 012004 005237 012016
2942
2943 012010 000207

```

```

:*****
:ROUTINE TO TYPEOUT DROPPED ITEMS
:*****
      CLR      R0
      MOV      #BIT6,R1
8$:   BIT      R1,$DEV0      ;DEVICE DROPPED?
      BNE     10$           ;BR IF YES
9$:   CLC
      ASL     R1
      BCS     13$           ;BR IF ALL BITS TESTED
      ADD     #2,R0         ;ELSE BUMP INDEX
      BR      8$
10$:  MOV      12$(R0),11$
      TYPE
11$:  .WORD   0             ;DEVICE TO BE DROPPED
      TYPE   ,DROP
      BR      9$
12$:  EISTXT
      CLOCK
      DRVO
      DRV1
      SCOM
      ACOM
      CLT1
      CLT2
      CLT3
      PRINT
:*****
:ROUTINE TO CHECK IF PRINTER EXISTS
:*****
13$:  CLR      PRPRES
      CLR     PRPORT
      TST    PRS           ;SEE IF PRINTER ERR
      BPL    16$           ;BR IF NO
      BIT    #BIT3,$DEV0  ;COMM PORT IN EXT LOOPBACK?
      BEQ   15$           ;BR IF YES
14$:  TYPE    ,PRINT
      TYPE   ,NOTPRES
      BR     17$
15$:  MOV      #<AMOD!B9600!OPAR!CHAR8>,PARAM
      BIC    #<RTS!DTR>,AMRC ;MUST BE TOGGLED
      BIS    #<RTS!DTR>,AMRC
      TST    PRS           ;NOW ANY ERRORS?
      BMI    14$           ;BR IF YES
16$:  INC     PRPORT
      INC    PRPRES       ;PRINTER PRESENT
17$:  RTS     PC

```

2944  
2945 012012 000000  
2946 012014 000000  
2947 012016 000000  
2948 012020 000000  
2949  
2950 012022 000000  
2951  
2952  
2953  
2954  
2955

EIPRES: 0 ;EIS-FIS PRESENT = NON-ZERO  
CLPRES: 0 ;CLUSTER PRESENT = NON-ZERO  
PRPRES: 0 ;PRINTER PRESENT = NON-ZERO  
PRPORT: 0 ;1 = PRINTER ENABLED THRU LOOPBACK  
  
MAXMEM: 0 ;MAX MEMORY

\*\*\*\*\*  
: RANDOM NUMBER GENERATOR  
\*\*\*\*\*

2956 012024 013700 012134  
2957 012030 013701 012132  
2958 012034 012702 177771  
2959 012040 006300  
2960 012042 006101  
2961 012044 005202  
2962 012046 001374  
2963 012050 063700 012134  
2964 012054 005501  
2965 012056 063701 012132  
2966 012062 062700 001057  
2967 012066 005501  
2968 012070 062701 047401  
2969 012074 010037 012134  
2970 012100 010137 012132  
2971  
2972 012104 042701 177400  
2973 012110 020137 012126  
2974 012114 103743  
2975 012116 020137 012130  
2976 012122 101340  
2977 012124 000207

RAND: 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  
  
;SCALE TO BE WITHIN LIMITS  
;SAVE LO BYTE ONLY  
BIC #177400,R1  
CMP R1,LOLIM  
BLO RAND ;BR IF N < LOLIM  
CMP R1,HILIM  
BHI RAND ;BR IF N > HILIM  
RTS PC ;ELSE EXIT WITH R1 = N

2978  
2979 012126 000000  
2980 012130 000000  
2981  
2982 012132 176543  
2983 012134 123456  
2984  
2985

LOLIM: 0  
HILIM: 0  
  
HINUM: .WORD 176543  
LONUM: .WORD 123456

2986 012136 123727 001210 000001  
2987 012144 001401  
2988 012146 104407  
2989 012150 000207

EXAMKB: CMPB \$ENV,#1 ;APT?  
BEQ 1\$ ;BR IF YES  
CKSWR  
1\$: RTS PC

```

2990
2991
2992
2993
2994
2995 012152 104401 021500
2996 012156 013746 001246
2997 012162 104402
2998 012164 104401 026250
2999
3000 012170 005046
3001 012172 005046
3002
3003 012174 105777 166744
3004 012200 100375
3005 012202 117746 166740
3006 012206 042716 177600
3007
3008 012212 021627 000015
3009 012216 001013
3010 012220 005766 000004
3011 012224 001403
3012 012226 016637 000002 001246
3013 012234 062706 000006
3014 012240 104401 001165
3015 012244 000002
3016
3017 012246 004737 025350
3018 012252 021627 000060
3019 012256 002420
3020 012260 021627 000067
3021 012264 003015
3022 012266 042726 000060
3023 012272 005766 000002
3024 012276 001403
3025 012300 006316
3026 012302 006316
3027 012304 006316
3028
3029 012306 005266 000002
3030 012312 056616 177776
3031 012316 000726
3032
3033 012320 104401 001164
3034 012324 000743
3035

:*****
: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),$DEVM ;ELSE SAVE NEW DEVM
   ADD #6,SP ;RESTORE STACK
4$:
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$: TYPE , $QUES ;TYPE ?<CRLF>
   BR 4$ ;DO ALL OVER

```

```
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046  
3047  
3048  
3049  
3050  
3051  
3052  
3053  
3054  
3055  
3056  
3057  
3058  
3059  
3060  
3061  
3062  
3063
```

\*\*\*\*\*  
\* 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.  
\*\*\*\*\*

```
012326 010046  
012330 012500  
012332 012520  
012334 012720 000200  
012340 012520  
012342 012710 000200  
012346 012600  
012350 000205
```

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

```
3058  
3059  
3060  
3061  
3062  
3063
```

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

```
012352 012537 012432  
012356 012537 012434  
012362 012701 000010  
012366 012700 177777  
012372 027737 000034 012434  
012400 001411  
012402 004737 012136  
012406 005300  
012410 001370  
012412 005237 001200  
012416 005301  
012420 001362  
012422 000402  
012424 062705 000004  
012430 000205  
012432 000000  
012434 000000
```

TIMER: MOV (R5)+, FLGHLD ;GET FLAG  
MOV (R5)+, CTHLD ;GET COUNT THAT FLAG SHOULD GO TO  
MOV #10, R1  
4\$: MOV #-1, R0  
1\$: 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  
FLGHLD: 0 ;FLAG  
CTHLD: 0 ;COUNT THAT FLAG MUST REACH

```
3085  
3086  
3087  
3088  
3089  
3090  
3091 012436 005037 012530  
3092 012442 012737 000010 012526  
3093 012450 012737 177777 012524  
3094 012456 005737 003262  
3095 012462 001015  
3096 012464 004737 012136  
3097 012470 005337 012524  
3098 012474 001370  
3099 012476 005237 001200  
3100 012502 005337 012526  
3101 012506 001360  
3102 012510 005237 012530  
3103 012514 000402  
3104  
3105 012516 062716 000004  
3106 012522 000207  
3107  
3108 012524 000000  
3109 012526 000000  
3110 012530 000000  
3111  
3112  
3113  
3114  
3115  
3116  
3117  
3118  
3119 012532 005037 012620  
3120 012536 012737 000010 012526  
3121 012544 012737 177777 012524  
3122 012552 005737 003262  
3123 012556 001015  
3124 012560 004737 012136  
3125 012564 005337 012524  
3126 012570 001370  
3127 012572 005237 001200  
3128 012576 005337 012526  
3129 012602 001360  
3130 012604 005237 012620  
3131 012610 000402  
3132  
3133 012612 062716 000004  
3134 012616 000207  
3135  
3136 012620 000000  
3137
```

```
*****  
:WATCHDOG TIMER TO PREVENT DX0 FROM ENTERING AN ENDLESS WAIT LOOP  
:RETURN IF TIMED OUT  
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR  
*****  
TIMD0: CLR DOTMO  
MOV #10,CTREG1  
4$: MOV #-1,CTREG  
1$: TST FIN ;FINISHED?  
BNE 2$ ;BR IF YES  
JSR PC,EXAMKB  
DEC CTREG ;ELSE DEC CTR  
BNE 1$ ;BR IF NOT TIMED OUT  
INC $DEVCT ;FOR APT  
DEC CTREG1  
BNE 4$  
INC DOTMO ;SET FLAG  
BR .+6 ;TIMED OUT  
2$: ADD #4,(SP) ;BUMP RET  
RTS PC  
CTREG: 0  
CTREG1: 0  
DOTMO: 0 ;! = TIMEOUT OCCURED
```

```
*****  
:WATCHDOG TIMER TO PREVENT DX1 FROM ENTERING AN ENDLESS WAIT LOOP  
:RETURN IF TIMED OUT  
:RETURN +4 IF NOT TIMED OUT & JUMP OVER ERROR  
*****  
TIMD1: CLR DITMO  
MOV #10,CTREG1  
4$: MOV #-1,CTREG  
1$: TST FIN ;FINISHED?  
BNE 2$ ;BR IF YES  
JSR PC,EXAMKB  
DEC CTREG ;ELSE DEC CTR  
BNE 1$ ;BR IF NOT TIMED OUT  
INC $DEVCT ;FOR APT  
DEC CTREG1  
BNE 4$  
INC DITMO ;SET FLAG  
BR .+6 ;TIMED OUT  
2$: ADD #4,(SP) ;BUMP RET  
RTS PC  
DITMO: 0 ;! = TIMEOUT OCCURED
```

```
3138  
3139  
3140  
3141  
3142 012622 012700 177777  
3143 012626 005300  
3144 012630 001376  
3145 012632 000207  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153 012634 012537 012714  
3154 012640 012537 012716  
3155  
3156 012644 012701 000010  
3157 012650 012700 177777  
3158 012654 033777 012716 000032  
3159 012662 001011  
3160 012664 004737 012136  
3161 012670 005300  
3162 012672 001370  
3163 012674 005237 001200  
3164 012700 005301  
3165 012702 001362  
3166 012704 000402  
3167  
3168 012706 062705 000004  
3169 012712 000205  
3170  
3171 012714 000000  
3172 012716 000000  
3173  
3174  
3175  
3176  
3177  
3178  
3179 012720 012700 002764  
3180 012724 005020  
3181 012726 020027 003010  
3182 012732 001374  
3183  
3184 012734 012700 003234  
3185 012740 005020  
3186 012742 020027 003260  
3187 012746 001374  
3188  
3189 012750 000207  
3190
```

```
*****  
: GENERAL TIMER  
*****  
TIMER1: MOV # -1,RO  
: DEC RO  
: 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 #10,R1  
4$: MOV # -1,RO  
1$: BIT BITHLD,@REGHLD ;BIT THERE?  
: BNE 2$ ;BR IF YES  
: JSR PC,EXAMKB  
: DEC RO ;ELSE DEC COUNTER  
: BNE 1$ ;BR IF NOT TIMED OUT  
: INC $DEVCT ;FOR APT  
: DEC R1  
: BNE 4$  
: BR .+6 ;ELSE TIMED OUT  
2$: ADD #4,R5 ;JUMP OVER ERROR ON RETURN  
: RTS R5  
REGHLD: 0 ;DEVICE REG  
BITHLD: 0 ;DEV REG BIT TO BE TESTED  
*****  
:ROUTINE TO CLEAR BOTH BAD TRACK/SECTOR TABLES  
*****  
CLRBAD: MOV #DOBAD,RO  
1$: CLR (RO)+  
: CMP RO,#DOBAD+20.  
: BNE 1$  
2$: MOV #D1BAD,RO  
: CLR (RO)+  
: CMP RO,#D1BAD+20.  
: BNE 2$  
: RTS PC
```



3191  
3192  
3193  
3194  
3195 012752 010046  
3196  
3197 012754 012700 002764  
3198 012760 005710  
3199 012762 001003  
3200 012764 013710 177174  
3201 012770 000405  
3202  
3203 012772 005720  
3204 012774 020027 003010  
3205 013000 001367  
3206 013002 104104  
3207 013004 012600  
3208 013006 000207  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219 013010 010046  
3220 013012 010146  
3221  
3222 013014 013701 002546  
3223 013020 000301  
3224 013022 053701 002550  
3225  
3226 013026 012700 002764  
3227 013032 020027 003010  
3228 013036 001405  
3229 013040 005710  
3230 013042 001403  
3231 013044 022001  
3232 013046 001371  
3233 013050 000402  
3234  
3235 013052 062705 000002  
3236 013056 012601  
3237 013060 012600  
3238 013062 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

```

3239  
3240  
3241  
3242  
3243 013064 010046  
3244  
3245 013066 012700 003234  
3246 013072 005710  
3247 013074 001003  
3248 013076 013710 177174  
3249 013102 000405  
3250  
3251 013104 005720  
3252 013106 020027 003260  
3253 013112 001367  
3254 013114 104105  
3255 013116 012600  
3256 013120 000207  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267 013122 010046  
3268 013124 010146  
3269  
3270 013126 013701 003016  
3271 013132 000301  
3272 013134 053701 003020  
3273  
3274 013140 012700 003234  
3275 013144 020027 003260  
3276 013150 001405  
3277 013152 005710  
3278 013154 001403  
3279 013156 022001  
3280 013160 001371  
3281 013162 000402  
3282  
3283 013164 062705 000002  
3284 013170 012601  
3285 013172 012600  
3286 013174 000205  
3287

\*\*\*\*\*  
: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$:      MOV     (SP)+,R1     ;RESTORE
          MOV     (SP)+,R0
          RTS     R5
```

```
3288 .SBTTL INTERRUPT HANDLERS
3289 :*****
3290 : GENERAL SERVICE ROUTINE TO BUMP 'INTFLG'
3291 : CALLING ROUTINE WILL KNOW WHAT TO DO
3292 :*****
3293
3294 013176 005237 013204 INTSRV: INC INTFLG
3295 013202 000002 RTI
3296 013204 000000 INTFLG: 0
3297
3298
3299 :*****
3300 :PRINTER INTERRUPT HANDLER: VALUES FROM 40 THRU 176.
3301 :*****
3302
3303 013206 013737 177514 013336 PRSRV: MOV PRS,SPRS ;STORE
3304 013214 005737 013336 TST SPRS ;ERROR?
3305 013220 100002 BPL 1$ ;BR IF NO
3306 013222 104007 ERROR 7 ;PRINTER STATUS ERROR
3307 013224 000002 RTI
3308
3309 013226 013737 013332 177516 1$: MOV PRCHR,PRB ;ELSE PRINT CHAR
3310 013234 023727 013332 000015 CMP PRCHR,#CR ;JUST DID CR?
3311 013242 001413 BEQ 2$ ;BR IF YES
3312 013244 023727 013332 000012 CMP PRCHR,#LF ;JUST DID LF?
3313 013252 001413 BEQ 3$ ;BR IF YES
3314 013254 023727 013332 000176 CMP PRCHR,#176 ;JUST DID LAST CHAR?
3315 013262 001413 BEQ 4$ ;BR IF YES
3316 013264 005237 013332 INC PRCHR ;ELSE BUMP CHAR FOR NEXT INTERRUPT
3317 013270 000417 BR 5$ ;EXIT
3318
3319 013272 012737 000012 013332 2$: MOV #LF,PRCHR ;DO LF NEXT
3320 013300 000413 BR 5$
3321 013302 012737 000040 013332 3$: MOV #40,PRCHR ;DO SPACE NEXT
3322 013310 000407 BR 5$
3323 013312 012737 000015 013332 4$: MOV #CR,PRCHR ;DO CR NEXT & START OVER
3324 013320 005237 013334 INC LINCT
3325 013324 005237 001200 INC $DEVCT ;FOR APT
3326 013330 000002 5$: RTI
3327
3328 013332 000000 PRCHR: 0 ;CHAR TO BE PRINTED
3329 013334 000000 LINCT: 0 ;LINE CTR
3330 013336 000000 SPRS: 0 ;STORE PRINTER CSR
```

```
3331
3332
3333
3334
3335
3336 013340 042737 000100 176504 TP1SRV: BIC #IE,TP1S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3337 013346 013737 013434 176506 MOV T1CHR,TP1B ;PRINT CHAR
3338 013354 000002 RTI
3339
3340 013356 013737 176502 013436 TK1SRV: MOV TK1B,T1HLD ;STORE CHAR
3341 013364 023737 013436 013434 CMP T1HLD,T1CHR ;OK?
3342 013372 001401 BEQ 1$ ;BR IF YES
3343 013374 104010 ERROR 10 ;CHAR COMP ERROR
3344
3345 013376 023727 013434 000377 1$: CMP T1CHR,#377 ;LAST CHAR?
3346 013404 001403 BEQ 2$ ;BR IF YES
3347 013406 005237 013434 INC T1CHR ;ELSE BUMP
3348 013412 000404 BR 3$
3349
3350 013414 005037 013434 2$: CLR T1CHR ;START OVER
3351 013420 005237 001200 INC $DEVCT ;FOR APT
3352 013424 052737 000100 176504 3$: BIS #IE,TP1S ;ALLOW PRINTER INTERR
3353 013432 000002 RTI
3354
3355 013434 000000 T1CHR: 0 ;CHAR TO XMITT
3356 013436 000000 T1HLD: 0 ;REC'D CHAR
3357
3358
3359
3360
3361
3362
3363 013440 042737 000100 176514 TP2SRV: BIC #IE,TP2S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3364 013446 013737 013534 176516 MOV T2CHR,TP2B ;PRINT CHAR
3365 013454 000002 RTI
3366
3367 013456 013737 176512 013536 TK2SRV: MOV TK2B,T2HLD ;STORE CHAR
3368 013464 023737 013536 013534 CMP T2HLD,T2CHR ;OK?
3369 013472 001401 BEQ 1$ ;BR IF YES
3370 013474 104011 ERROR 11 ;CHAR COMP ERROR
3371
3372 013476 023727 013534 000377 1$: CMP T2CHR,#377 ;LAST CHAR?
3373 013504 001403 BEQ 2$ ;BR IF YES
3374 013506 005237 013534 INC T2CHR ;ELSE BUMP
3375 013512 000404 BR 3$
3376
3377 013514 005037 013534 2$: CLR T2CHR ;START OVER
3378 013520 005237 001200 INC $DEVCT ;FOR APT
3379 013524 052737 000100 176514 3$: BIS #IE,TP2S ;ALLOW PRINTER INTERR
3380 013532 000002 RTI
3381
3382 013534 000000 T2CHR: 0 ;CHAR TO XMITT
3383 013536 000000 T2HLD: 0 ;REC'D CHAR
```

```

3384
3385
3386
3387
3388
3389 013540 042737 000100 176524 TP3SRV: BIC #IE,TP3S ;DISABLE INTER, RECV'R WILL TURN BACK ON
3390 013546 013737 013634 176526      MOV T3CHR,TP3B ;PRINT CHAR
3391 013554 000002
3392
3393 013556 013737 176522 013636 TK3SRV: MOV TK3B,T3HLD ;STORE CHAR
3394 013564 023737 013636 013634      CMP T3HLD,T3CHR ;OK?
3395 013572 001401          BEQ 1$ ;BR IF YES
3396 013574 104012          ERROR 12 ;CHAR COMP ERROR
3397
3398 013576 023727 013634 000377 1$: CMP T3CHR,#377 ;LAST CHAR?
3399 013604 001403          BEQ 2$ ;BR IF YES
3400 013606 005237 013634          INC T3CHR ;ELSE BUMP
3401 013612 000404          BR 3$
3402
3403 013614 005037 013634          2$: CLR T3CHR ;START OVER
3404 013620 005237 001200          INC $DEVCT ;FOR APT
3405 013624 052737 000100 176524 3$: BIS #IE,TP3S ;ALLOW PRINTER INTERR
3406 013632 000002
3407
3408 013634 000000          T3CHR: 0 ;CHAR TO XMITT
3409 013636 000000          T3HLD: 0 ;REC'D CHAR
3410
3411
3412
3413
3414
3415
3416 013640 042737 000100 176614 AMXSRV: BIC #IE,AMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
3417 013646 013737 013734 176616      MOV COMCHR,AMXB ;XMITT CHAR
3418 013654 000002
3419
3420 013656 013737 176612 013736 AMRSRV: MOV AMRB,COMHLD ;STORE CHAR
3421 013664 023737 013736 013734      CMP COMHLD,COMCHR ;OK?
3422 013672 001401          BEQ 1$ ;BR IF YES
3423 013674 104013          ERROR 13 ;CHAR COMPARE ERROR
3424
3425 013676 023727 013734 000377 1$: CMP COMCHR,#377 ;LAST CHAR?
3426 013704 001403          BEQ 2$ ;BR IF YES
3427 013706 005237 013734          INC COMCHR ;ELSE BUMP
3428 013712 000404          BR 3$
3429
3430 013714 005037 013734          2$: CLR COMCHR ;START OVER
3431 013720 005237 001200          INC $DEVCT ;FOR APT
3432 013724 052737 000100 176614 3$: BIS #IE,AMXC ;ALLOW XMIT INTERR
3433 013732 000002
3434
3435 013734 000000          COMCHR: 0 ;XMIT CHAR
3436 013736 000000          COMHLD: 0 ;REC'D CHAR

```

```

3437
3438
3439
3440
3441
3442 013740 042737 000100 176624 SMXSRV: BIC #IE,SMXC ;DISABLE INTER, RECV'R WILL TURN BACK ON
3443 013746 013737 013734 176626 MOV COMCHR,SMXB ;XMITT CHAR
3444 013754 023727 013734 000377 CMP COMCHR,#377 ;LAST CHAR?
3445 013762 001020 BNE 2$ ;BR IF NO
3446
3447 013764 005737 014026 TST LSTCHR
3448 013770 001010 BNE 1$
3449 013772 005237 014026 INC LSTCHR
3450 013776 052737 000100 176624 BIS #IE,SMXC ;ALLOW RDY INTR SO CAN SHUT DOWN FAST
3451 014004 005237 001200 INC $DEVCT ;FOR APT
3452 014010 000002 RTI
3453
3454 014012 042737 000020 176624 1$: BIC #SEND,SMXC ;NO MORE TO SEND
3455 014020 005237 001200 INC $DEVCT ;FOR APT
3456 014024 000002 2$: RTI
3457
3458 014026 000000 LSTCHR: 0 ;SET WHEN LAST CHAR XMIT
3459
3460
3461
3462
3463
3464 014030 013737 176622 013736 SMRSRV: MOV SMRB,COMHLD ;STORE CHAR
3465 014036 123727 013736 000026 CMPB COMHLD,#026 ;IGNORE SYNC CHARS
3466 014044 001430 BEQ 4$
3467 014046 023737 013736 013734 CMP COMHLD,COMCHR ;OK?
3468 014054 001401 BEQ 1$ ;BR IF YES
3469 014056 104014 ERROR 14 ;CHAR COMPARE ERROR
3470
3471 014060 023727 013734 000377 1$: CMP COMCHR,#377 ;LAST CHAR?
3472 014066 001403 BEQ 2$ ;BR IF YES
3473 014070 005237 013734 INC COMCHR ;ELSE BUMP
3474 014074 000411 BR 3$
3475
3476 014076 012737 177777 013734 2$: MOV #-1,COMCHR ;INDICATE DONE
3477 014104 042737 000100 176620 BIC #IE,SMRC ;ALL DONE
3478 014112 005237 001200 INC $DEVCT ;FOR APT
3479 014116 000403 BR 4$
3480
3481 014120 052737 000100 176624 3$: BIS #IE,SMXC ;ALLOW XMIT INTERR
3482 014126 000002 4$: RTI
3483

```

3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522

014130 105737 177170  
014134 100413  
014136 013737 177170 014240  
014144 013737 177172 014242  
014152 013737 177174 014244  
014160 104015  
014162 000002  
  
014164 005737 177170  
014170 100011  
  
014172 013737 177170 014240  
014200 013737 177172 014242  
014206 013737 177174 014244  
  
014214 032737 000020 177170  
014222 001003  
  
014224 000177 000000  
014230 000000  
  
014232 000177 000000  
014236 000000  
  
014240 000000  
014242 000000  
014244 000000

```

:*****
:                               DISK INTERRUPT HANDLERS
: ALL DISK INTERRUPTS ENTER THRU RXSRV & DISPATCH TO THE
: CURRENT SERVICE ROUTINE POINTED TO BY:
:                               DODISP 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    @DODISP      ;ELSE DISPATCH TO DX0 SERVICE ROUTINE
DODISP: 0

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

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

```

```

3523
3524
3525
3526
3527
3528
3529 014246 0127C3 000100
3530 014252 013701 002546
3531 014256 000301
3532 014260 053701 002550
3533 014264 012737 014312 014230
3534 014272 012737 000101 177170
3535 014300 010137 177172
3536 014304 005303
3537 014306 001374
3538 014310 000002
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548 014312 013701 002546
3549 014316 000301
3550 014320 053701 002550
3551 014324 010137 177174
3552 014330 012737 014366 014230
3553 014336 023727 003260 000005
3554 014344 001004
3555 014346 012737 000115 177170
3556 014354 000403
3557 014356 012737 000105 177170
3558 014364 000002
3559

:*****
: HANDLER TO FILL THE DRIVE BUFFER FOR DX0
: & POINT TO THE WRITE SECTOR HANDLER.
:*****
DOFBUF: 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
1$: MOV R1,RXDB ;FILL ENTIRE SECTOR WITH ITS ADDRESS
    DEC R3
    BNE 1$
    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

```



```

3560
3561
3562
3563
3564
3565
3566
3567
3568 014366 005737 177170
3569 014372 100024
3570 014374 023727 002544 000012
3571 014402 001072
3572 014404 004737 012752
3573 014410 032737 000020 014242
3574 014416 001410
3575 014420 104064
3576 014422 012737 014500 014230
3577 014430 012737 000117 177170
3578 014436 000002
3579
3580 014440 104016
3581 014442 000416
3582
3583 014444 005737 002544
3584 014450 001413
3585 014452 005237 027102
3586 014456 005237 027106
3587 014462 032737 000020 014242
3588 014470 001402
3589 014472 104065
3590 014474 000401
3591 014476 104017
3592
3593 014500 005037 002544
3594 014504 023727 002550 000031
3595 014512 001004
3596 014514 012737 000002 002550
3597 014522 000407
3598 014524 023727 002550 000032
3599 014532 001405
3600 014534 062737 000002 002550
3601 014542 000137 014246
3602
3603 014546 012737 000001 002550
3604 014554 005037 002544
3605 014560 005037 002552
3606 014564 000137 014622
3607
3608 014570 005237 002544
3609 014574 032737 000020 014242
3610 014602 001757
3611 014604 012737 014246 014230
3612 014612 012737 000117 177170
3613 014620 000002
3614
3615

```

```

*****
;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
        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
        BIT      #RNF,SRXES ;SEEK ERROR?
        BEQ      8$        ;BR IF NO
        ERROR    65        ;SOFT SEEK ERROR WRITE SECTOR
        BR       2$
8$:     ERROR    17        ;SOFT ERROR WRITE SECTOR
2$:     CLR      DOERR     ;LAST ODD SECTOR?
        CMP      DOSEC,#31 ;BR IF NO
        BNE      3$        ;ELSE DO EVEN SECTORS NOW
        MOV      #2$,DOSEC
        BR       4$
3$:     CMP      DOSEC,#32 ;LAST EVEN SECTOR?
        BEQ      5$        ;BR IF YES
        ADD      #2$,DOSEC  ;ELSE BUMP SECTOR
        JMP      DOFBUF    ;GO TO FILL BUFFER & DO ANOTHER
4$:     JMP      DOFBUF
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
        MOV      #DOFBUF,DODISP ;GOTO FILL BUFF AFTER RESTORE
        MOV      #<RESTOR!IE>,RXCS
        RTI

```

```

3616
3617
3618
3619
3620 014622 013701 002546
3621 014626 000301
3622 014630 0537C1 002550
3623 014634 010137 177174
3624
3625 014640 012737 014656 014230
3626 014646 012737 000107 177170
3627 014654 000002
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638 014656 005737 177170
3639 014662 100054
3640 014664 023727 002544 000012
3641 014672 001415
3642 014674 005237 002544
3643 014700 032737 000020 014242
3644 014706 001745
3645 014710 012737 014622 014230
3646 014716 012737 000117 177170
3647 014724 000002
3648
3649 014726 004737 012752
3650 014732 032737 000020 014242
3651 014740 001410
3652 014742 104066
3653
3654 014744 012737 015644 014230
3655 014752 012737 000117 177170
3656 014760 000002
3657
3658 014762 032737 000010 014242
3659 014770 001404
3660 014772 005237 002562
3661 014776 104106
3662 015000 000423
3663
3664 015002 005037 002562
3665 015006 104020
3666 015010 000137 015644

```

```

:*****
: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
        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
    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    DOCRC         ;ELSE SET FLAG
    ERROR  106          ;HARD CRC ERROR
    BR     2$           ;& GO TO EMP BUFF TO CHK DATA

```

```

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

```

```

3667
3668 015014 005737 002544 1$: TST DOERR ;ANY PREV ERRORS?
3669 015020 001413 BEQ 2$ ;BR IF NO
3670 015022 005237 027102 INC TSERR ;TOTAL SOFT ERR CT
3671 015026 005237 027106 INC PSERR ;PASS SOFT ERR CT
3672 015032 032737 000020 014242 BIT #RNF,SRXES ;SEEK ERROR?
3673 015040 0014C2 BEQ 9$ ;BR IF NO
3674 015042 104067 ERROR 67 ;SEEK ERROR READ SECTOR
3675 015044 000401 BR 2$
3676 015046 104021 9$: ERROR 21 ;SOFT ERROR READ SECTOR
3677
3678 015050 005037 002544 2$: CLR DOERR
3679 015054 023727 003260 000005 CMP DXTST,#5 ;DOING WRITE DELETED DATA TESTS?
3680 015062 001005 BNE 6$ ;BR IF NO
3681 015064 032737 000040 177172 BIT #DD,RXES ;'DELETED DATA' SET?
3682 015072 001001 BNE 6$ ;BR IF YES
3683 015074 104052 ERROR 52 ;DD NOT SET
3684
3685 015076 000137 015102 6$: JMP DOEBUF ;GO TO EMPTY BUFFER
3686
3687
3688
3689
3690
3691
3692
3693 ;:*****
3694 ; HANDLER TO EMPTY THE DRIVE BUFFER FOR DX0
3695 ;& POINT TO THE CHECK DATA HANDLER.
3696 ;:*****
3697 015102 012703 000100 DOEBUF: MOV #100,R3 ;WORD CT
3698 015106 012701 002564 MOV #DOBUF,R1 ;BUFFER ADDRESS
3699 015112 012737 015140 014230 MOV #DOCDAT,DODISP ;POINT TO CHK DATA AFT INTER.
3700 015120 012737 000103 177170 MOV #<EBUF!IE>,RXCS ;ISSUE EMPTY BUFFER CMD
3701 015126 013721 177172 1$: MOV RXDB,(R1)+ ;GET WORD & STORE IT
3702 015132 005303 DEC R3
3703 015134 001374 BNE 1$
3704 015136 000002 RTI
3705
3706

```

```
3707
3708
3709
3710
3711
3712
3713
3714 015140 005037 002552
3715 015144 012703 000100
3716 015150 013701 002546
3717 015154 000301
3718 015156 053701 002550
3719 015162 012702 002564
3720 015166 020112
3721 015170 001407
3722 015172 010137 002556
3723 015176 011237 002560
3724 015202 005237 002552
3725 015206 000405
3726
3727 015210 005303
3728 015212 001403
3729 015214 062702 000002
3730 015220 000762
3731
3732 015222 005737 002552
3733 015226 001421
3734 015230 005737 002562
3735 015234 001402
3736 015236 104107
3737 015240 000431
3738
3739 015242 023727 002554 000012 14$:
3740 015250 001033
3741 015252 023727 003260 000003
3742 015260 001002
3743 015262 104053
3744 015264 000417
3745
3746 015266 104022
3747 015270 000415
3748
3749 015272 005737 002562
3750 015276 001402
3751 015300 104110
3752 015302 000410
3753
3754 015304 005737 002554
3755 015310 001405
3756 015312 005237 027102
3757 015316 005237 027106
3758 015322 104023
```

```
*****
: 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     DOTRK,R1
          SWAB    R1
          BIS     DOSEC,R1    ;R1 NOW HAS EXPECTED DATA
          MOV     #DOBUF,R2   ;GET BUFFER ADDR
1$:      CMP     R1,(R2)      ;COMPARE OK?
          BEQ     2$          ;BR IF YES
          MOV     R1,DOEXP    ;ELSE SAVE
          MOV     (R2),DOREC
          INC     DOCMER
          BR      3$

2$:      DEC     R3
          BEQ     3$
          ADD     #2,R2
          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$:      ERROR  22          ;HARD ERROR DATA COMPARE
          BR      5$

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

15$:     TST     DOCERR      ;ANY PREV ERR?
          BEQ     5$          ;BR IF NO
          INC     TSERR      ;TOTAL SOFT ERR CT
          INC     PSERR      ;PASS SOFT ERR CT
          ERROR  23          ;SOFT ERROR DATA COMP
```

```

3759
3760 015324 005037 002554      5$:   CLR   DOCERR
3761 015330 005037 002562      CLR   DOCRC
3762 015334 000137 015644      JMP   DONEXT
3763
3764 015340 005237 002554      7$:   INC   DOCERR
3765 015344 023727 003260 000003  CMP   DXTST,#3      ;DOING INIT TEST?
3766 015352 001402          BEQ   12$           ;BR IF YES
3767 015354 000137 014622          JMP   DORSEC       ;ELSE READ SECTOR OVER AGAIN
3768
3769 015360 000137 016002      12$:  JMP   DODUN        ;ALL DONE
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780 015364 005037 002544      DONXT1: CLR   DOERR
3781 015370 005037 002552      CLR   DOCMER
3782 015374 023727 002550 000031  CMP   DOSEC,#31     ;LAST ODD SECTOR?
3783 015402 001004          BNE   1$           ;BR IF NO
3784 015404 012737 000002 002550  MOV   #2,DOSEC      ;ELSE DO EVEN SECTORS NOW
3785 015412 000407          BR    2$
3786 015414 023727 002550 000032  1$:  CMP   DOSEC,#32     ;LAST EVEN SECTOR?
3787 015422 001405          BEQ   3$           ;BR IF YES
3788 015424 062737 000002 002550  ADD   #2,DOSEC      ;ELSE BUMP SECTOR
3789 015432 000137 014622      2$:  JMP   DORSEC       ;GO READ SECTOR
3790
3791 015436 032777 010000 163474  3$:  BIT   #BIT12,@SWR   ;SKIP TYPEOUT IF NOT SET
3792 015444 001411          BEQ   6$
3793 015446 104401 020266          TYPE ,DRVO
3794 015452 104401 020721          TYPE ,TRK
3795 015456 013746 002546          MOV   DOTRK,-(SP)  ;;SAVE DOTRK FOR TYPEOUT
3796 015462 104405          TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
3797 015464 104401 021011          TYPE ,DUN
3798 015470 005237 003262          6$:  INC   FIN           ;SETUP TO GO TO DX1
3799 015474 023737 002546 003264  CMP   DOTRK,MAXTRK ;LAST TRACK?
3800 015502 001406          BEQ   4$           ;BR IF YES
3801 015504 005237 002546          INC   DOTRK        ;BUMP TRACK
3802 015510 012737 000001 002550  MOV   #1,DOSEC      ;INIT SECTOR
3803 015516 000414          BR    5$
3804
3805 015520 032777 010000 163412  4$:  BIT   #BIT12,@SWR   ;SKIP TYPEOUT IF NOT SET
3806 015526 001406          BEQ   7$
3807 015530 104401 020266          TYPE ,DRVO
3808 015534 104401 020726          TYPE ,PATT
3809 015540 104401 021011          TYPE ,DUN
3810 015544 005237 002540          7$:  INC   DOPAT        ;PATT DONE FLAG
3811 015550 042737 000100 177170  5$:  BIC   #IE,RXCS     ;DISABLE DX0 INTERRUPTS
3812 015556 000002          RTI

```

```
3813
3814
3815
3816
3817
3818 015560 005037 002544 DONXT2: CLR DOERR
3819 015564 005037 002552 CLR DOCMER
3820 015570 005237 003262 INC FIN ;SETUP TO GO TO DX1
3821 015574 005237 002542 INC DOCNT
3822 015600 023737 002542 003266 CMP DOCNT,MAXSK ;DID ALL SEEKS?
3823 015606 001012 BNE 1$ ;BR IF NO
3824 015610 032777 010000 163322 BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
3825 015616 001406 BEQ 1$
3826 015620 104401 020266 TYPE ,DRVO
3827 015624 104401 020741 TYPE ,RANDOM
3828 015630 104401 021011 TYPE ,DUN
3829 015634 042737 000100 177170 1$: BIC #IE,RXCS ;DISABLE DX0 INTERRUPTS
3830 015642 000002 RTI
3831
3832
3833
3834
3835
3836 015644 023727 003260 000001 DONEXT: CMP DXTST,#1
3837 015652 001644 BEQ DONXT1
3838 015654 023727 003260 000002 CMP DXTST,#2
3839 015662 001736 BEQ DONXT2
3840 015664 000446 BR DODUN
3841
```

```
3842
3843
3844      ;*****
3845      ;HANDLER TO SETUP INITIALIZE COMMAND TEST
3846      ;*****
3847 015666 012737 046032 177174 DOINIT: MOV    #46032,RXSA    ;TRY TO FAKE DRIVE TO GO TO TRK 76 SEC 26
3848                                     ;INIT SHOULD GO TO TRK 1, SEC 1.
3849 015674 012737 016002 014230      MOV    #DODUN,DODISP ;POINT TO DONE HANDLER
3850 015702 012737 000111 177170      MOV    #<INITAL!IE>,RXCS ;DO INIT COMMAND
3851 015710 000002
3852      RTI
3853
3854      ;*****
3855      ;HANDLERS TO SETUP RESTORE COMMAND TEST
3856      ;*****
3857
3858 015712 012737 016002 014230 DOREST: MOV    #DODUN,DODISP ;POINT TO DONE HANDLER
3859 015720 012737 000117 177170      MOV    #<RESTOR!IE>,RXCS
3860 015726 000002      RTI
3861
3862 015730 012737 016016 014236 DIREST: MOV    #D1DUN,D1DISP
3863 015736 012737 000137 177170      MOV    #<RESTOR!IE!DX1>,RXCS
3864 015744 000002      RTI
3865
3866      ;*****
3867      ;HANDLER TO SETUP READ STATUS COMMAND TEST
3868      ;*****
3869
3870 015746 012737 016002 014230 DOSTAT: MOV    #DODUN,DODISP ;POINT TO DONE
3871 015754 012737 000113 177170      MOV    #<RSTAT!IE>,RXCS ;DO READ STATUS COMMAND
3872 015762 000002      RTI
3873
3874      ;*****
3875      ;HANDLER TO START INVALID ADDRESS TESTS
3876      ;*****
3877
3878 015764 012737 016002 014230 DOINV:  MOV    #DODUN,DODISP ;POINT TO DONE
3879 015772 012737 000105 177170      MOV    #<WSEC!IE>,RXCS ;DO WRITE SECTOR COMMAND
3880 016000 000002      RTI
3881
3882      ;*****
3883      ;HANDLERS TO FINISH INITIALIZE, RESTORE, WRITE DELETED DATA,
3884      ;READ STATUS & INVALID ADDRESS TESTS.
3885      ;*****
3886
3887
3888 016002 005237 003262 DODUN:  INC    FIN          ;SET DONE FLAG
3889 016006 042737 000100 177170      BIC    #!IE,RXCS      ;DISABLE FURTHER DX0 INTERRUPTS
3890 016014 000002      RTI
3891
3892 016016 005237 003262 D1DUN:  INC    FIN
3893 016022 042737 000120 177170      BIC    #!IE!DX1>,RXCS
3894 016030 000002      RTI
3895
```

3896  
3897  
3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905  
3906  
3907  
3908  
3909  
3910  
3911  
3912  
3913  
3914  
3915  
3916  
3917  
3918  
3919  
3920  
3921  
3922  
3923  
3924  
3925  
3926  
3927  
3928  
3929  
3930  
3931  
3932  
3933

016032 012703 000100  
016036 013701 003016  
016042 000301  
016044 053701 003020  
016050 005737 003272  
016054 001005  
016056 005737 003274  
016062 001002  
016064 052701 100000  
016070 012737 016116 014236  
016076 012737 000121 177170  
016104 010137 177172  
016110 005303  
016112 001374  
016114 000002  
013701 003016  
016122 000301  
016124 053701 003020  
016130 010137 177174  
016134 012737 016152 014236  
016142 012737 000125 177170  
016150 000002

\*\*\*\*\*  
: HANDLER TO FILL THE DRIVE BUFFER FOR DX1  
: & POINT TO THE WRITE SECTOR HANDLER.  
\*\*\*\*\*

D1FBUF: MOV #100,R3 ;WORD CT  
MOV D1TRK,R1  
SWAB R1  
BIS D1SEC,R1  
TST COMP1 ;DONT SET BIT15 IN COMPATABILTIY 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  
1\$: MCV R1,RXDB ;FILL ENTIRE SECTOR WITH ITS ADDRESS  
DEC R3 ;& BIT 15 SET  
BNE 1\$  
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



```

3934
3935
3936
3937
3938
3939
3940
3941
3942 016152 005737 177170
3943 016156 100024
3944 016160 023727 003014 000012
3945 016166 001072
3946 016170 004737 013064
3947 016174 032737 000020 014242
3948 016202 001410
3949 016204 104070
3950 016206 012737 016264 014236
3951 016214 012737 000137 177170
3952 016222 000002
3953
3954 016224 104024
3955 016226 000416
3956
3957 016230 005737 003014
3958 016234 001413
3959 016236 005237 027102
3960 016242 005237 027106
3961 016246 032737 000020 014242
3962 016254 001402
3963 016256 104071
3964 016260 000401
3965 016262 104025
3966
3967 016264 005037 003014
3968 016270 023727 003020 000031
3969 016276 001004
3970 016300 012737 000002 003020
3971 016306 000407
3972 016310 023727 003020 000032
3973 016316 001405
3974 016320 062737 000002 003020
3975 016326 000137 016032
3976
3977 016332 012737 000001 003020
3978 016340 005037 003014
3979 016344 005037 003022
3980 016350 000137 016406
3981
3982 016354 005237 003014
3983 016360 032737 000020 014242
3984 016366 001757
3985 016370 012737 016032 014236
3986 016376 012737 000137 177170
3987 016404 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
        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
        BIT      #RNF,SRXES ;SEEK ERROR?
        BEQ      8$        ;BR IF NO
        ERROR    71        ;SEEK ERROR WRITE SECTOR
        BR      2$
8$:     ERROR    25        ;ERROR WRITE SECTOR
2$:     CLR      D1ERR      ;LAST ODD SECTOR?
        CMP      D1SEC,#31  ;BR IF NO
        BNE      3$        ;ELSE DO EVEN SECTORS NOW
        MOV      #2,D1SEC
        BR      4$
3$:     CMP      D1SEC,#32  ;LAST EVEN SECTOR?
        BEQ      5$        ;BR IF YES
        ADD      #2,D1SEC   ;ELSE BUMP SECTOR
        JMP      D1FBUF     ;GO TO FILL BUFFER & DO ANOTHER
4$:
5$:     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
        MOV      #D1FBUF,D1DISP ;RET TO FILL BUFF AFT RESTORE
        MOV      #<RESTOR!IE!DX1>,RXCS
        RTI

```

```

3988
3989
3990
3991
3992
3993 016406 013701 003016
3994 016412 0003C1
3995 016414 053701 003020
3996 016420 010137 177174
3997
3998 016424 012737 016442 014236
3999 016432 012737 000127 177170
4000 016440 000002
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012 016442 005737 177170
4013 016446 100054
4014 016450 023727 003014 000012
4015 016456 001415
4016 016460 005237 003014
4017 016464 032737 000020 014242
4018 016472 001745
4019 016474 012737 016406 014236
4020 016502 012737 000137 177170
4021 016510 000002
4022
4023 016512 004737 013064
4024 016516 032737 000020 014242
4025 016524 001410
4026 016526 104072
4027
4028 016530 012737 017376 014236
4029 016536 012737 000137 177170
4030 016544 000002
4031
4032 016546 032737 000010 014242
4033 016554 001404
4034 016556 005237 003032
4035 016562 104111
4036 016564 000423
4037
4038 016566 005037 003032
4039 016572 104026
4040 016574 000137 017376

```

```

;*****
;HANDLER TO READ A SECTOR ON DX1 & POINT TO CHECK.
;*****
DIRSEC: 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).
;*****
DIRCHK: 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
        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
    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

```

```

4041
4042 016600 005737 003014      1$:   TST      D1ERR      ;ANY PREV ERRORS?
4043 016604 001413              BEQ      2$           ;BR IF NO
4044 016606 005237 027102      INC      TSERR       ;TOTAL SOFT ERR CT
4045 016612 005237 027106      INC      PSERR       ;PASS SOFT ERR CT
4046 016616 032737 000020 014242 BIT      #RNF,SRXES  ;SEEK ERROR?
4047 016624 001402              BEQ      9$           ;BR IF NO
4048 016626 104073              ERROR    73          ;SEEK ERROR READ SECTOR
4049 016630 000401              BR       2$
4050 016632 104027      9$:   ERROR    27          ;SOFT ERROR READ SECTOR
4051
4052 016634 005037 003014      2$:   CLR      D1ERR
4053 016640 000137 016644      JMP      D1EBUF      ;GO TO EMPTY BUFFER
4054
4055
4056
4057
4058
4059

```

```

4060
4061      ;*****
4062      ; HANDLER TO EMPTY THE DRIVE BUFFER FOR DX1
4063      ; & POINT TO THE CHECK DATA HANDLER.
4064      ;*****

```

```

4065 016644 012703 000100      D1EBUF: MOV      #100,R3      ;WORD CT
4066 016650 012701 003034      MOV      #D1BUF,R1      ;BUFFER ADDRESS
4067 016654 012737 016702 014236 MOV      #D1CDAT,D1DISP ;POINT TO CHK DATA AFT INTER.
4068 016662 012737 000123 177170 MOV      #<EBUF!IE!DX1>,RXCS ;ISSUE EMPTY BUFFER CMD
4069 016670 013721 177172      1$:   MOV      RXDB,(R1)+  ;GET WORD & STORE IT
4070 016674 005303      DEC      R3
4071 016676 001374      BNE     1$
4072 016700 000002      RTI
4073

```

4074  
4075  
4076  
4077  
4078  
4079  
4080  
4081 016702 005037 003022  
4082 016706 012703 000100  
4083 016712 013701 003016  
4084 016716 000301  
4085 016720 053701 003020  
4086  
4087 016724 005737 003272  
4088 016730 001005  
4089 016732 005737 003274  
4090 016736 001002  
4091 016740 052701 100000  
4092  
4093 016744 012702 003034  
4094 016750 020112  
4095 016752 001407  
4096 016754 010137 003026  
4097 016760 011237 003030  
4098 016764 005237 003022  
4099 016770 000405  
4100 016772 005303  
4101 016774 001403  
4102 016776 062702 000002  
4103 017002 000762  
4104  
4105 017004 005737 003022  
4106 017010 001413  
4107 017012 005737 003032  
4108 017016 001402  
4109 017020 104112  
4110 017022 000423  
4111 017024 023727 003024 000012  
4112 017032 001025  
4113 017034 104030  
4114 017036 000415  
4115  
4116 017040 005737 003032  
4117 017044 001402  
4118 017046 104113  
4119 017050 000410  
4120 017052 005737 003024  
4121 017056 001405  
4122 017060 005237 027102  
4123 017064 005237 027106  
4124 017070 104031

```

:*****
: 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      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$:      MOV      #D1BUF,R2    ;GET BUFFER ADDR
1$:      CMP      R1,(R2)      ;COMPARE OK?
          BEQ      2$          ;BR IF YES
          MOV      R1,D1EXP    ;ELSE SAVE
          MOV      (R2),D1REC
          INC      D1CMER
          BR       3$
2$:      DEC      R3
          BEQ      3$
          ADD     #2,R2
          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$
14$:     CMP      D1CERR,#10.  ;ELSE, 10 ERRS YET?
          BNE      7$          ;BR IF NO
          ERROR   30          ;HARD ERROR DATA COMP
          BR       5$

4$:      TST      D1CRC      ;HERE FROM CRC ERROR?
          BEQ      15$        ;BR IF NO
          ERROR   113        ;CRC ERR WITH DATA OK
          BR       5$
15$:     TST      D1CERR      ;ANY ERR?
          BEQ      5$
          INC     TSERR      ;TOTAL SOFT ERR CT
          INC     PSERR      ;PASS SOFT ERR CT
          ERROR   31          ;SOFT ERROR DATA COMP

```

```

4125
4126 017072 005037 003024
4127 017076 005037 003032
4128 017102 000137 017376
4129
4130 017106 005237 003024
4131 017112 000137 016406
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141 017116 005037 003014
4142 017122 005037 003022
4143 017126 023727 003020 000031
4144 017134 001004
4145 017136 012737 000002 003020
4146 017144 000407
4147 017146 023727 003020 000032
4148 017154 001405
4149 017156 062737 000002 003020
4150 017164 000137 016406
4151
4152 017170 032777 010000 161742
4153 017176 001411
4154 017200 104401 020273
4155 017204 104401 020721
4156 017210 013746 003016
4157 017214 104405
4158 017216 104401 021011
4159 017222 005237 003262
4160 017226 023737 003016 003264
4161 017234 001406
4162 017236 005237 003016
4163 017242 012737 000001 003020
4164 017250 000414
4165
4166 017252 032777 010000 161660
4167 017260 001406
4168 017262 104401 020273
4169 017266 104401 020726
4170 017272 104401 021011
4171 017276 005237 003010
4172 017302 042737 000120 177170
4173 017310 000002
4174
5$: CLR D1CERR
   CLR D1CRC
   JMP D1NEXT
7$: INC D1CERR
   JMP D1RSEC ;READ SECTOR OVER AGAIN

*****
:HANDLER TO SETUP NEXT SECTOR/TRACK FOR DX1.
:WILL GO TO READ SECTOR (NEXT) IF ALL SECTORS ON TRACK NOT DONE.
:WILL GO TO FILL BUFFER (NEXT TRACK) IF ALL SECTORS ON TRACK DONE.
:WILL SHUT OFF DRV INTERRUPTS IF ALL TRACKS DONE, WITH A MESSAGE.
*****
D1NXT1: CLR D1ERR
        CLR D1CMER
        CMP D1SEC,#31 ;LAST ODD SECTOR?
        BNE 1$ ;BR IF NO
        MOV #2,D1SEC ;ELSE DO EVEN SECTORS NOW
        BR 2$
1$: CMP D1SEC,#32 ;LAST EVEN SECTOR?
   BEQ 3$ ;BR IF YES
   ADD #2,D1SEC ;ELSE BUMP SECTOR
   JMP D1RSEC ;GO READ SECTOR
2$:
3$: BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
   BEQ 6$
   TYPE ,DRV1
   TYPE ,TRK
   MOV D1TRK,-(SP) ;;SAVE D1TRK FOR TYPEOUT
   TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
   TYPE ,DUN
6$: INC FIN ;SETUP TO GO TO DX0
   CMP D1TRK,MAXTRK ;LAST TRACK?
   BEQ 4$ ;BR IF YES
   INC D1TRK ;BUMP TRACK
   MOV #1,D1SEC ;INIT SECTOR
   BR 5$
4$: BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET
   BEQ 7$
   TYPE ,DRV1
   TYPE ,PATT
   TYPE ,DUN
7$: INC D1PAT ;PATT DONE FLAG
5$: BIC #<IE!DX1>,RXCS ;DISABLE DX1 INTERRUPTS
   RTI

```

4175  
4176  
4177  
4178  
4179  
4180  
4181  
4182  
4183  
4184  
4185  
4186  
4187  
4188  
4189  
4190  
4191  
4192  
4193  
4194  
4195  
4196  
4197  
4198  
4199  
4200

017312 005037 003014  
017316 005037 003022  
017322 005237 003262  
017326 005237 003012  
017332 023737 003012 003266  
017340 001012  
017342 032777 010000 161570  
017350 001406  
017352 104401 020273  
017356 104401 020741  
017362 104401 021011  
017366 042737 000120 177170 1\$:  
017374 000002  
  
017376 023727 003260 000001  
017404 001644  
017406 000741

\*\*\*\*\*  
:HANDLER TO SETUP TO GO TO NEXT RANDOM TRACK/SECTOR FOR DX1.  
\*\*\*\*\*

D1NXT2: CLR D1ERR  
CLR D1CMER  
INC FIN ;SETUP TO GO TO DX0  
INC D1CNT  
CMP D1CNT,MAXSK ;DID ALL SEEKS?  
BNE 1\$ ;BR IF NO  
BIT #BIT12,@SWR ;SKIP TYPEOUT IF NOT SET  
BEQ 1\$  
TYPE ,DRV1  
TYPE ,RANDOM  
TYPE ,DUN  
BIC #<IE!DX1> ,RXCS ;DISABLE DXi INTERRUPTS  
RTI

D1NEXT: CMP DXTST,#1  
BEQ D1NXT1  
BR D1NXT2

```

4201
4202
4203
4204
4205
4206
4207
4208
4209
4210 017410 005037 003270
4211 017414 104401 021210
4212 017420 000000
4213
4214 017422 004737 017730
4215
4216 017426 005037 017722
4217 017432 012737 000001 017724
4218 017440 012737 000001 017726
4219 017446 105737 177170
4220 017452 100375
4221
4222 017454 004537 020000
4223 017460 000007
4224
4225 017462 005737 177170
4226 017466 100011
4227 017470 005237 017722
4228 017474 023727 017722 000012
4229 017502 001364
4230 017504 104100
4231 017506 000000
4232 017510 000776
4233
4234 017512 005037 017722
4235 017516 004537 020040
4236 017522 002564
4237
4238 017524 004537 020000
4239 017530 000025
4240
4241 017532 005737 177170
4242 017536 100011
4243 017540 005237 017722
4244 017544 023727 017722 000012
4245 017552 001364
4246 017554 104101
4247 017556 000000
4248 017560 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

```

```

4249
4250 017562 005037 017722      4$: CLR      ERFLG
4251 017566 004537 020000      JSR      R5,RDWR      ;READ DX1
4252 017572 000027      <RSEC!DX1>
4253
4254 017574 005737 177170      TST      RXCS      ;ERROR?
4255 017600 1000i1      BPL      5$      ;BR IF NO
4256 017602 005237 017722      INC      ERFLG
4257 017606 023727 017722 000012      CMP      ERFLG,#10.  ;10 ERRORS?
4258 017614 001362      BNE      4$      ;BR IF NO & TRY AGAIN
4259 017616 104103      ERROR   103      ;HARD ERR READ SECTOR DX1
4260 017620 000000      HALT
4261 017622 000776      BR      .-2      ;FORCE 250 RESTART
4262
4263 017624 004537 020040      5$: JSR      R5,EBUFF  ;EMPTY BUFFER INTO D1BUF
4264 017630 003034      D1BUF
4265
4266 017632 004737 020102      JSR      PC,DATCHK ;COMPARE DOBUF WITH D1BUF
4267
4268 017636 005237 017720      INC      SECCNT      ;TOTAL DONE
4269
4270 017642 023737 017720 017716      CMP      SECCNT,MAXSEC ;DID TOTAL # SECTORS?
4271 017650 001415      BEQ      8$      ;BR IF YES
4272
4273 017652 023727 017726 000032      CMP      SECT,#32   ;ELSE DID WE DO LAST SECTOR ON TRACK?
4274 017660 001403      BEQ      6$      ;BR IF YES
4275 017662 005237 017726      INC      SECT      ;ELSE BUMP & DO ANOTHER
4276 017666 000672      BR      1$
4277
4278 017670 005237 017724      6$: INC      TRACK      ;BUMP TRK
4279 017674 012737 000001 017726      MOV      #1,SECT   ;INIT SECTOR
4280 017702 000664      BR      1$      ;& DO ANOTHER
4281
4282 017704 104401 021311      8$: TYPE      ,COPDUN ;DONE MSG
4283 017710 000000      HALT
4284 017712 000137 017410      JMP      COPY      ;DO AGAIN IF 'P' TYPED
4285
4286 017716 000000      MAXSEC: 0      ;TOTAL # SECTORS TO READ FROM DX0
4287 017720 000000      SECCNT: 0      ;TOTAL # SECTORS WRITTEN TO DX1
4288 017722 000000      ERFLG: 0
4289 017724 000000      TRACK: 0
4290 017726 000000      SECT: 0

```



4291  
4292  
4293  
4294  
4295  
4296 017730 012700 027744  
4297 017734 0062C0  
4298 017736 005500  
4299 017740 062700 000377  
4300 017744 042700 000377  
4301  
4302 017750 000300  
4303 017752 006300  
4304 017754 006300  
4305  
4306 017756 062700 000074  
4307 017762 062700 000010  
4308  
4309 017766 010037 017716  
4310  
4311  
4312 017772 005037 017720  
4313 017776 000207  
4314  
4315  
4316  
4317  
4318  
4319  
4320  
4321 020000 013700 017724  
4322 020004 000300  
4323 020006 053700 017726  
4324 020012 010037 177174  
4325 020016 012537 177170  
4326 020022 004537 012634  
4327 020026 177170  
4328 020030 000200  
4329 020032 104124  
4330 020034 000000  
4331 020036 000205  
4332

\*\*\*\*\*  
: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
        BIC     #377,RO      ;ROUND OFF TO NEAREST WHOLE WORD

        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     #8.,RO       ;2 MORE BLOCKS (NECESSARY FUDGE FACTOR)

        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
    
```

```
4333  
4334  
4335  
4336  
4337  
4338  
4339 020040 0127C0 000100          EBUFF:  MOV    #100,R0          ;WD CT  
4340 020044 012501                MOV    (R5)+,R1          ;BUFFER ADDR  
4341 020046 012737 000003 177170  MOV    #EBUF,RXCS       ;ISSUE EMPTY BUFFER COMMAND  
4342 020054 013721 177172          1$:   MOV    RXDB,(R1)+      ;PUT IT IN TABLE  
4343 020060 005300                DEC    R0  
4344 020062 001374                BNE    1$  
4345 020064 004537 012634          JSR    R5,TIMER2        ;WAIT FOR DONE  
4346 020070 177170                RXCS  
4347 020072 000200                DONE  
4348 020074 104125                ERROR 125              ;NO DONE  
4349 020076 000000                HALT  
4350 020100 000205                RTS    R5  
4351  
4352  
4353  
4354  
4355  
4356  
4357 020102 012700 000100          DATCHK: MOV    #100,R0          ;WD CT  
4358 020106 012701 002564          MOV    #DOBUF,R1  
4359 020112 012702 003034          MOV    #D1BUF,P2  
4360 020116 022122          1$:   CMP    (R1)+,(R2)+      ;COMPARE?  
4361 020120 001403                BEQ    2$              ;BR IF YES  
4362 020122 104102                ERROR 102              ;MISCOMPARE  
4363 020124 000000                HALT  
4364 020126 000776                BR    .-2  
4365  
4366 020130 005300          2$:   DEC    R0  
4367 020132 001371                BNE    1$  
4368 020134 000207                RTS    PC  
4369
```

4370  
4371  
4372  
4373  
4374  
4375  
4376  
4377

.SBTTL PROGRAM MESSAGES

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

.NLIST BEX

020136	041600	052514	052123	CLOPT:	.ASCIZ	<CRLF>/CLUSTER OPTION/			
020156	052200	051105	020115	CLT1:	.ASCIZ	<CRLF>/TERM #1/			
020167	200	042524	046522	CLT2:	.ASCIZ	<CRLF>/TERM #2/			
020200	052200	051105	020115	CLT3:	.ASCIZ	<CRLF>/TERM #3/			
020211	200	054523	041516	SCOM:	.ASCIZ	<CRLF>/SYNC COMM/			
020224	040600	054523	041516	ACOM:	.ASCIZ	<CRLF>/ASYN COMM/			
020240	050200	044522	052116	PRINT:	.ASCIZ	<CRLF>/PRINTER/			
020251	040	047520	052122	PORT:	.ASCIZ	/ PORT TESTED/			
020266	042200	030130	000	DRV0:	.ASCIZ	<CRLF>/DX0/			
020273	200	054104	000061	DRV1:	.ASCIZ	<CRLF>/DX1/			
020300	041600	047514	045503	CLOCK:	.ASCIZ	<CRLF>/CLOCK/			
020307	200	044505	026523	EISTXT:	.ASCIZ	<CRLF>/EIS-FIS OPTION/			
020327	113	046440	046505	MEM:	.ASCIZ	/K MEMORY PRESENT/			
020350	047040	052117	050040	NOTPRES:	.ASCIZ	/ NOT PRESENT/			
020365	200	044412	051516	WARNING:	.ASCII	<CRLF><LF>/INSERT SCRATCH DISKS, TYPE 'P' FOR NORMAL TESTING/			
020450	023600	032062	043460		.ASCII	<CRLF>/'240G' FOR NORMAL RESTARTS/			
020503	200	031047	030065		.ASCII	<CRLF>/'250G' TO COPY SYS EXERCISER DISK/			
020545	200	031047	030066		.ASCII	<CRLF>/'260G' FOR COMPATABILITY PASS 1: WRITE/			
020614	023600	033462	043460		.ASCII	<CRLF>/'270G' FOR COMPATABILITY PASS 2: READ/			
020663	040	042524	052123	DROP:	.ASCIZ	/ TESTING DROPPED/			
020704	051040	047125	044516	RUN:	.ASCIZ	/ RUNNING/			
020715	040	045517	000	OK:	.ASCIZ	/ OK/			
020721	040	051124	000113	TRK:	.ASCIZ	/ TRK/			
020726	042040	052101	020101	PATT:	.ASCIZ	/ DATA PATT/			
020741	040	040522	042116	RANDOM:	.ASCIZ	/ RANDOM SEEKS/			
020757	200	044505	026523	FISEIS:	.ASCIZ	<CRLF>/EIS-FIS TESTS/			
020776	046600	046505	052040	MEMORY:	.ASCIZ	<CRLF>/MEM TESTS/			
021011	040	047504	042516	DUN:	.ASCIZ	/ DONE/			
021017	200	042412	042116	ENDPAS:	.ASCIZ	<CRLF><LF>/END PASS #/			
021034	052011	052117	046101	MSG1:	.ASCIZ	/ TOTAL ERRORS: /			
021063	200	004411	052011	MSG2:	.ASCIZ	<CRLF>/ TOTAL SOFT ERRORS: /			
021117	200	004412	004411	MSG3:	.ASCIZ	<CRLF><LF>/ TOTAL ERRORS THIS PASS:/			
021154	004600	004411	047523	MSG4:	.ASCIZ	<CRLF>/ SOFT ERRORS THIS PASS:/			
021210	041600	050117	020131	COPMSG:	.ASCII	<CRLF>/COPY DX0 TO DX1/			
021230	044600	051516	051105		.ASCIZ	<CRLF>/INSERT SCRATCH DISK IN DX1, TYPE 'P' TO PROCEED/			
021311	200	047504	042516	COPDUN:	.ASCIZ	<CRLF>/DONE...TYPE 'P' TO DO AGAIN OR '240G' FOR NORMAL TESTING/			
021403	200	041412	046517	COMPAT:	.ASCII	<CRLF><LF>/COMPATABILITY PASS 1 (WRITE) DONE/			
021446	042200	020117	050047		.ASCIZ	<CRLF>/DO 'P' FOR PASS 2 (READ)/			
021500	042200	053105	020115	DEVM:	.ASCIZ	<CRLF>/DEVM = /			
021511	115	046505	051117	EM1:	.ASCIZ	/MEMORY TIMEOUT/			
021530	042515	020115	040504	EM2:	.ASCIZ	/MEM DATA COMP ERR/			
021552	054523	041516	041440	EM3:	.ASCIZ	/SYNC COMM DID NOT RECEIVE SYNC CHAR/			
021616	051120	047111	042524	EM4:	.ASCIZ	/PRINTER STATUS ERR/			
021641	124	051105	020115	EM5:	.ASCIZ	/TERM #1 DATA COMP ERR/			
021667	124	051105	020115	EM6:	.ASCIZ	/TERM #2 DATA COMP ERR/			

```
021715 124 051105 020115 EM7: .ASCIZ /TERM #3 DATA COMP ERR/
021743 101 054523 041516 EM8: .ASCIZ /ASYN COMM DATA COMP ERR/
021774 054523 041516 041440 EM9: .ASCIZ /SYNC COMM DATA COMP ERR/
022024 047125 054105 020120 EM10: .ASCIZ /UNEXP DISK INTERRUPT/
022051 104 030130 053440 EM11: .ASCIZ /DX0 WRITE ERR/
022067 104 030130 044040 EM13: .ASCIZ /DX0 HARD ERR - READ SECT/
022120 054104 020060 047523 EM14: .ASCIZ /DX0 SOFT ERR - READ SECT/
022151 104 030130 044040 EM15: .ASCIZ /DX0 HARD ERR - DATA COMP/
022202 054104 020060 047523 EM16: .ASCIZ /DX0 SOFT ERR - DATA COMP/
022233 104 030530 053440 EM17: .ASCIZ /DX1 WRITE ERR/
022251 104 030530 044040 EM19: .ASCIZ /DX1 HARD ERR - READ SECT/
022302 054104 020061 047523 EM20: .ASCIZ /DX1 SOFT ERR - READ SECT/
022333 104 030530 044040 EM21: .ASCIZ /DX1 HARD ERR - DATA COMP/
022364 054104 020061 047523 EM22: .ASCIZ /DX1 SOFT ERR - DATA COMP/
022415 103 045514 044040 EM23: .ASCIZ /CLK HUNG/
022426 051120 047111 042524 EM24: .ASCIZ /PRINTER HUNG/
022443 124 051105 020115 EM25: .ASCIZ /TERM #1 HUNG/
022460 042524 046522 021440 EM26: .ASCIZ /TERM #2 HUNG/
022475 124 051105 020115 EM27: .ASCIZ /TERM #3 HUNG/
022512 054523 041516 041440 EM28: .ASCIZ /SYNC COMM HUNG/
022531 101 054523 041516 EM29: .ASCIZ /ASYN COMM HUNG/
022551 104 030130 044040 EM30: .ASCIZ /DX0 HUNG/
022562 054104 020061 052510 EM31: .ASCIZ /DX1 HUNG/
022573 123 047131 020103 EM32: .ASCIZ /SYNC COMM - DATA NOT AVAIL NOT SET/
022636 054523 041516 041440 EM33: .ASCIZ /SYNC COMM - RECVR ACTIVE NOT SET/
022677 123 047131 020103 EM34: .ASCIZ /SYNC COMM - RECVR DONE NOT SET/
022736 054104 020060 042047 EM35: .ASCIZ /DX0 'DEL DATA' BIT 5 NOT SET IN RXES/
023003 104 030130 044440 EM36: .ASCIZ /DX0 INIT CMD DID NOT READ TRK 1, SEC 1/
023052 054104 020060 042522 EM37: .ASCIZ /DX0 RESTORE CMD ERR/
023076 054104 020060 044447 EM38: .ASCIZ /DX0 'INIT DONE' BIT 0 NOT SET IN RXES/
023144 054104 020060 042522 EM39: .ASCIZ /DX0 READ STATUS CMD ERR/
023174 054104 020060 047111 EM40: .ASCIZ /DX0 INV ADDR BIT 1 NOT SET IN RXES/
023237 104 030130 042440 EM41: .ASCIZ /DX0 ERR BIT NOT SET IN RXCS/
023273 104 030130 044440 EM42: .ASCIZ /DX0 INIT CMD ERR/
023314 054104 020060 042523 EM43: .ASCIZ /DX0 SEEK ERR - WRITE SECT/
023346 054104 020060 042523 EM45: .ASCIZ /DX0 SEEK ERR - READ SECT/
023377 104 030530 051440 EM47: .ASCIZ /DX1 SEEK ERR - WRITE SECT/
023431 104 030530 051440 EM49: .ASCIZ /DX1 SEEK ERR - READ SECT/
023462 054104 020061 042522 EM51: .ASCIZ /DX1 RESTORE CMD ERR/
023506 044504 045523 023440 EM52: .ASCIZ /DISK 'DONE' NOT SET/
023532 044506 046114 023040 EM53: .ASCIZ /FILL & EMPTY BUFF TEST... DATA MISCOMP/
023601 104 052101 020101 EM54: .ASCIZ /DATA MISCOMP/
023616 042522 046120 041501 EM55: .ASCIZ /REPLACE DX0...10 BAD SECTORS/
023653 122 050105 040514 EM56: .ASCIZ /REPLACE DX1...10 BAD SECTORS/
023710 054104 020060 040510 EM57: .ASCIZ /DX0 HARD CRC ERR/
023731 104 030130 042040 EM58: .ASCIZ /DX0 DATA CRC ERR/
023752 054104 020060 051120 EM59: .ASCIZ /DX0 PREV CRC ERR WITH DATA OK/
024010 054104 020061 040510 EM60: .ASCIZ /DX1 HARD CRC ERR/
024031 104 030530 042040 EM61: .ASCIZ /DX1 DATA CRC ERR/
024052 054104 020061 051120 EM62: .ASCIZ /DX1 PREV CRC ERR WITH DATA OK/
024110 044505 026523 044506 EM63: .ASCIZ /EIS-FIS CONDITION CODE ERROR/
024145 105 051511 043055 EM64: .ASCIZ /EIS-FIS DATA ERROR/

024170 051105 047522 020122 DH1: .ASCIZ /ERROR # ERR PC/
024207 105 051122 051117 DH2: .ASCIZ /ERROR # ERR PC ADDR/
024233 105 051122 051117 DH3: .ASCIZ /ERROR # ERR PC ADDR EXPECT RECVD/
```

024275	105	051122	051117	DH4:	.ASCIZ	/ERROR #	ERR PC	PR STATUS/							
024326	051105	047522	020122	DH5:	.ASCIZ	/ERROR #	ERR PC	EXPECT	RECV						
024363	105	051122	051117	DH6:	.ASCIZ	/ERRJR #	DXTST#	ERR PC	RXCS	RXES	RXSA/				
024430	051105	047522	020122	DH7:	.ASCIZ	/ERROR #	DXTST#	ERR PC	RXCS	RXES	TRACK	SECTOR	#	RETRIE	
024517	104	052130	052123	DH8:	.ASCIZ	/DXTST #	ERR PC	RXCS	RXES	RXSA	EXPECT	RECVD	#	RETRIES	

.EVEN

024606	027076	001116	000000	DT1:	.WORD	ERRNUM,\$ERRPC,0									
024614	027076	001116	004546	DT2:	.WORD	ERRNUM,\$ERRPC,ADDR,0									
024624	027076	001116	004546	DT3:	.WORD	ERRNUM,\$ERRPC,ADDR,PAT,MEMHLD,0									
024640	027076	001116	013336	DT4:	.WORD	ERRNUM,\$ERRPC,SPRS,0									
024650	027076	001116	013434	DT5:	.WORD	ERRNUM,\$ERRPC,T1CHR,T1HLD,0									
024662	027076	001116	013534	DT6:	.WORD	ERRNUM,\$ERRPC,T2CHR,T2HLD,0									
024674	027076	001116	013634	DT7:	.WORD	ERRNUM,\$ERRPC,T3CHR,T3HLD,0									
024706	027076	001116	013734	DT8:	.WORD	ERRNUM,\$ERRPC,COMCHR,COMHLD,0									
024720	027076	003260	001116	DT9:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,SRXSA,0									
024736	027076	003260	001116	DT10:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,DOTRK,DOSEC,DOERR,0									
024760	003260	001116	177170	DT11:	.WORD	DXTST,\$ERRPC,RXCS,RXES,RXSA,DOEXP,DOREC,DOCERR,0									
025002	027076	003260	001116	DT12:	.WORD	ERRNUM,DXTST,\$ERRPC,SRXCS,SRXES,D1TRK,D1SEC,D1ERR,0									
025024	003260	001116	177170	DT13:	.WORD	DXTST,\$ERRPC,RXCS,RXES,RXSA,D1EXP,D1REC,D1CERR,0									
025046	027076	003260	001116	DT14:	.WORD	ERRNUM,DXTST,\$ERRPC,RXCS,RXES,RXSA,0									
025064	027076	001116	007242	DT15:	.WORD	ERRNUM,\$ERRPC,EXPO,EBHLD,0									
025076	027076	001116	007244	DT16:	.WORD	ERRNUM,\$ERRPC,EXP1,EBHLD,0									

025110	000000			DF:	.WORD	0									
025112	000000				.WORD	0									
025114	000000				.WORD	0									

025116	000000			DF1:	.WORD	0									
025120	000000				.WORD	0									
025122	000400				.WORD	400									
025124	000401				.WORD	401									

025126	000000			DF2:	.WORD	0									
025130	000000				.WORD	0									
025132	000000				.WORD	0									
025134	000400				.WORD	400									

.LIST BEX

```

4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395 025136 105737 001157
4396 025142 100002
4397 025144 000000
4398 025146 000430
4399 025150 010046
4400 025152 017600 000002
4401 025156 122737 000001 001210
4402 025164 001011
4403 025166 132737 000100 001211
4404 025174 001405
4405 025176 010037 025206
4406 025202 004737 026270
4407 025206 000000
4408 025210 132737 000040 001211
4409 025216 001003
4410 025220 112046
4411 025222 001005
4412 025224 005726
4413 025226 012600
4414 025230 062716 000002
4415 025234 000002
4416 025236 122716 000011
4417 025242 001430
4418 025244 122716 000200
4419 025250 001006
4420 025252 005726
4421 025254 104401
4422 025256 001165
4423 025260 105037 025414
4424 025264 000755
4425 025266 004737 025350
4426 025272 123726 001156
4427 025276 001350
4428 025300 013746 001154
4429
4430 025304 105366 000001
4431 025310 002770
4432 025312 004737 025350
4433 025316 105337 025414

```

```

.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$: MOV RO,-(SP) ;;SAVE RO
MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
CMPB #APTENV,$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 RO,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
MOV (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
TST (SP)+ ;;BR IF IT ISN'T THE TERMINATOR
60$: MOV (SP)+,RO ;;IF TERMINATOR POP IT OFF THE STACK
3$: ADD #2,(SP) ;;RESTORE RO
RTI ;;ADJUST RETURN PC
4$: CMPB #HT,(SP) ;;RETURN
BEQ 8$ ;;BRANCH IF <HT>
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
BR 2$ ;;GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;;IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;;IF NO GO GET NEXT CHAR.
;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;;GO TYPE A NULL
DECB $CHARCNT ;;DO NOT COUNT AS A COUNT

```

```

4434 025322 000770          BR      7$          ;:LOOP
4435
4436          ;HORIZONTAL TAB PROCESSOR
4437
4438 025324 112716 000040      8$:      MOVB    #' ,(SP)          ;:REPLACE TAB WITH SPACE
4439 025330 004737 025350      9$:      JSR     PC,$TYPEC          ;:TYPE A SPACE
4440 025334 132737 000007 025414  BITB    #7,$CHARCNT          ;:BRANCH IF NOT AT
4441 025342 001372          BNE     9$          ;:TAB STOP
4442 025344 005726          TST     (SP)+          ;:POP SPACE OFF STACK
4443 025346 000724          BR      2$          ;:GET NEXT CHARACTER
4444 025350 105777 153574      $TYPEC: TSTB   @$TPS          ;:WAIT UNTIL PRINTER IS READY
4445 025354 100375          BPL     $TYPEC
4446 025356 116677 000002 153566  MOVB    2(SP),@$TPB          ;:LOAD CHAR TO BE TYPED INTO DATA REG.
4447 025364 122766 000015 000002  CMPB    #CR,2(SP)          ;:IS CHARACTER A CARRIAGE RETURN?
4448 025372 001003          BNE     1$          ;:BRANCH IF NO
4449 025374 105037 025414      CLRB    $CHARCNT          ;:YES--CLEAR CHARACTER COUNT
4450 025400 000406          BR      $TYPEX          ;:EXIT
4451 025402 122766 000012 000002  1$:     CMPB    #LF,2(SP)          ;:IS CHARACTER A LINE FEED?
4452 025410 001402          BEQ     $TYPEX          ;:BRANCH IF YES
4453 025412 105227          INCB    (PC)+          ;:COUNT THE CHARACTER
4454 025414 000000      $CHARCNT: .WORD 0          ;:CHARACTER COUNT STORAGE
4455 025416 000207      $TYPEX: RTS     PC
4456

```

```

4457 .SBTTL TTY INPUT ROUTINE
4458
4459 ::*****
4460 .ENABL LSB
4461
4462 ::*****
4463 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4464 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4465 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4466 *WHEN OPERATING IN TTY FLAG MODE.
4467 025420 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
4468 025426 001114 BNE 15$ ;;BRANCH IF NO
4469 025430 105777 153510 TSTB @STKS ;;CHAR THERE?
4470 025434 100111 BPL 15$ ;;IF NO, DON'T WAIT AROUND
4471 025436 117746 153504 MOVB @STKB,-(SP) ;;SAVE THE CHAR
4472 025442 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
4473 025446 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
4474 025452 001102 BNE 15$ ;;NO, RETURN TO USER
4475 025454 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
4476 025462 001476 BEQ 15$ ;;BRANCH IF YES
4477
4478 025464 104401 026232 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
4479 025470 104401 026237 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
4480 025474 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
4481 025500 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4482 025502 104401 026250 TYPE , $MNEW ;;PROMPT FOR NEW SWR
4483 025506 005046 19$: CLR -(SP) ;;CLEAR COUNTER
4484 025510 005046 CLR -(SP) ;;THE NEW SWR
4485 025512 105777 153426 7$: TSTB @STKS ;;CHAR THERE?
4486 025516 100375 BPL 7$ ;;IF NOT TRY AGAIN
4487
4488 025520 117746 153422 MOVB @STKB,-(SP) ;;PICK UP CHAR
4489 025524 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
4490
4491 025530 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL-C?
4492 025534 001015 BNE 9$ ;;BRANCH IF NOT
4493 025536 104401 026220 TYPE , $CNTLC ;;YES, ECHO CONTROL-C (^C)
4494 025542 062706 000006 ADD #6,SP ;;CLEAN UP STACK
4495 025546 123727 001135 000001 CMPB $INTAG,#1 ;;REENABLE TTY KEYBOARD INTERRUPTS?
4496 025554 001003 BNE 8$ ;;BRANCH IF NO
4497 025556 012777 000100 153360 MOV #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
4498 025564 000137 012152 8$: JMP GT$DEV ;;CONTROL-C RESTART
4499
4500
4501 025570 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
4502 025574 001005 BNE 10$ ;;BRANCH IF NOT
4503 025576 104401 026225 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (^U)
4504 025602 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
4505 025606 000737 BR 19$ ;;LET'S TRY IT AGAIN
4506
4507
4508 025610 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
4509 025614 001022 BNE 16$ ;;BRANCH IF NO
4510 025616 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
4511 025622 001403 BEQ 11$ ;;BRANCH IF YES
4512 025624 016677 000002 153306 MOV 2(SP),@SWR ;;SAVE NEW SWR

```



4513	025632	062706	000006	
4514	025636	104401	001165	
4515	025642	123727	001135	000001
4516	025650	001003		
4517	025652	012777	000100	153264
4518	025660	000002		
4519	025662	004737	025350	
4520	025666	021627	000060	
4521	025672	002420		
4522	025674	021627	000067	
4523	025700	003015		
4524	025702	042726	000060	
4525	025706	005766	000002	
4526	025712	001403		
4527	025714	006316		
4528	025716	006316		
4529	025720	006316		
4530	025722	005266	000002	
4531	025726	056616	177776	
4532	025732	000667		
4533	025734	104401	001164	
4534	025740	000720		

```

11$: ADD #6,SP          ;;CLEAR UP STACK
14$: TYPE ,%CRLF        ;;ECHO <CR> AND <LF>
      CMPB $INTAG,#1    ;;RE-ENABLE TTY KBD INTERRUPTS?
      BNE 15$           ;;BRANCH IF NOT
      MOV #100,@%TKS    ;;RE-ENABLE TTY KBD INTERRUPTS
15$: RTI                ;;RETURN
16$: JSR PC,%TYPEPC     ;;ECHO CHAR
      CMP (SP),#60      ;;CHAR < 0?
      BLT 18$           ;;BRANCH IF YES
      CMP (SP),#67      ;;CHAR > 7?
      BGT 18$           ;;BRANCH IF YES
      BIC #60,(SP)+     ;;STRIP-OFF ASCII
      TST 2(SP)         ;;IS THIS THE FIRST CHAR
      BEQ 17$           ;;BRANCH IF YES
      ASL (SP)          ;;NO, SHIFT PRESENT
      ASL (SP)          ;; CHAR OVER TO MAKE
      ASL (SP)          ;; ROOM FOR NEW ONE.
17$: INC 2(SP)          ;;KEEP COUNT OF CHAR
      BIS -2(SP),%SP    ;;SET IN NEW CHAR
      BR 7$             ;;GET THE NEXT ONE
18$: TYPE ,%QUES        ;;TYPE ?<CR><LF>
      BR 20$           ;;SIMULATE CONTROL-U
.DSABL LSB

```

4535				
4536				
4537				
4538				
4539				
4540				
4541				
4542				
4543				
4544				
4545				
4546	025742	011646		
4547	025744	016666	000004	000002
4548	025752	105777	153166	
4549	025756	100375		
4550	025760	117766	153162	000004
4551	025766	042766	177600	000004
4552	025774	026627	000004	000023
4553	026002	001013		
4554	026004	105777	153134	
4555	026010	100375		
4556	026012	117746	153130	
4557	026016	042716	177600	
4558	026022	022627	000021	
4559	026026	001366		
4560	026030	000750		
4561	026032	026627	000004	000140
4562	026040	002407		
4563	026042	026627	000004	000175
4564	026050	003003		
4565	026052	042766	000040	000004
4566	026060	000002		
4567				
4568				

```

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:
*   RDCHR                ;;INPUT A SINGLE CHARACTER FROM THE TTY
*   RETURN HERE         ;;CHARACTER IS ON THE STACK
*                       ;;WITH PARITY BIT STRIPPED OFF
*
$RDCHR: MOV (SP),-(SP)   ;;PUSH DOWN THE PC
        MOV 4(SP),2(SP)  ;;SAVE THE PS
1$:    TSTB @%TKS        ;;WAIT FOR
        BPL 1$           ;;A CHARACTER
        MOVB @%TKB,4(SP) ;;READ THE TTY
        BIC #^C<177>,4(SP) ;;GET RID OF JUNK IF ANY
        CMP 4(SP),#23    ;;IS IT A CONTROL-S?
        BNE 3$           ;;BRANCH IF NO
2$:    TSTB @%TKS        ;;WAIT FOR A CHARACTER
        BPL 2$           ;;LOOP UNTIL ITS THERE
        MOVB @%TKB,-(SP) ;;GET CHARACTER
        BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
        CMP (SP)+,#21    ;;IS IT A CONTROL-Q?
        BNE 2$           ;;IF NOT DISCARD IT
        BR 1$           ;;YES, RESUME
3$:    CMP 4(SP),#140    ;;IS IT UPPER CASE?
        BLT 4$           ;;BRANCH IF YES
        CMP 4(SP),#175  ;;IS IT A SPECIAL CHAR?
        BGT 4$           ;;BRANCH IF YES
        BIC #40,4(SP)   ;;MAKE IT UPPER CASE
4$:    RTI              ;;GO BACK TO USER
*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

```

```

4569
4570
4571
4572
4573
4574 026062 010346
4575 026064 012703 026210
4576 026070 022703 026220
4577 026074 101415
4578 026076 104410
4579 026100 112613
4580 026102 122713 000003
4581 026106 001005
4582 026110 104401 026220
4583 026114 012603
4584 026116 000137 012152
4585 026122 122713 000177
4586 026126 001003
4587 026130 104401 001164
4588 026134 000753
4589 026136 111337 026206
4590 026142 104401 026206
4591 026146 122723 000015
4592 026152 001346
4593 026154 105063 177777
4594 026160 104401 001166
4595 026164 012603
4596 026166 011646
4597 026170 016666 000004 000002
4598 026176 012766 026210 000004
4599 026204 000002
4600 026206 000
4601 026207 000
4602 026210 000010
4603 026220 041536 005015 000
4604 026225 136 006525 000012
4605 026232 043536 005015 000
4606 026237 015 051412 051127
4607 026244 036440 000040
4608 026250 020040 042516 020127
4609 026256 020075 000
4610 026262

```

```

;*CALL:
;* RDLIN
;* RETURN HERE
;*
$RDLIN: MOV R3,-(SP) ;;SAVE R3
1$: MOV #$TTYIN,R3 ;;GET ADDRESS
2$: CMP #$TTYIN+8.,R3 ;;BUFFER FULL?
BLOS 4$ ;;BR IF YES
RDCHR ;;GO READ ONE CHARACTER FROM THE TTY
MOVB (SP)+,(R3) ;;GET CHARACTER
CMPB #3,(R3) ;;IS IT A CONTROL-C?
BNE 10$ ;;BRANCH IF NO
TYPE ,%CNTLC ;;TYPE A CONTROL-C (^C)
MOV (SP)+,R3 ;;RESTORE R3
JMP GT$DEV ;;GOTO CONTROL-C RESTART
10$: CMPB #177,(R3) ;;IS IT A RUBOUT
BNE 3$ ;;SKIP IF NOT
4$: TYPE ,%QUES ;;TYPE A '?'
BR 1$ ;;CLEAR THE BUFFER AND LOOP
3$: MCVB (R3),9$ ;;ECHO THE CHARACTER
TYPE ,9$
CMPB #15,(R3)+ ;;CHECK FOR RETURN
BNE 2$ ;;LOOP IF NOT RETURN
CLRB -1(R3) ;;CLEAR RETURN (THE 15)
TYPE ,%LF ;;TYPE A LINE FEED
MOV (SP)+,R3 ;;RESTORE R3
MOV (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
MOV #$TTYIN,4(SP)
RTI ;;RETURN
9$: .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
.BYTE 0 ;;TERMINATOR
$TTYIN: .BLKB 8. ;;RESERVE 8 BYTES FOR TTY INPUT
%CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
%CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
%CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
$MSWR: .ASCIZ <15><12>/SWR = /
$MNEW: .ASCIZ / NEW = /
.EVEN

```

```

4611      .SBTTL  APT COMMUNICATIONS ROUTINE
4612
4613      ;:*****
4614 026262 112737 000001 026526 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
4615 026270 112737 000001 026524 $ATY3:  MOVB  #1,$MFLG      ;;TO TYPE A MESSAGE
4616 026276 000403          BR      $ATYC
4617 026300 112737 000001 026526 $ATY4:  MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
4618 026306          $ATYC:
4619 026306 010046          MOV    R0,-(SP)      ;;PUSH R0 ON STACK
4620 026310 010146          MOV    R1,-(SP)      ;;PUSH R1 ON STACK
4621 026312 105737 026524          TSTB  $MFLG      ;;SHOULD TYPE A MESSAGE?
4622 026316 001450          BEQ    5$          ;;IF NOT: BR
4623 026320 122737 000001 001210  CMPB  #APTENV,$ENV      ;;OPERATING UNDER APT?
4624 026326 001031          BNE   3$          ;;IF NOT: BR
4625 026330 132737 000100 001211  BITB  #APTPOOL,$ENVM    ;;SHOULD SPOOL MESSAGES?
4626 026336 001425          BEQ    3$          ;;IF NOT: BR
4627 026340 017600 000004          MOV    @4(SP),R0      ;;GET MESSAGE ADDR.
4628 026344 062766 000002 000004  ADD   #2,4(SP)        ;;BUMP RETURN ADDR.
4629 026352 005737 001170          1$:  TST   $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
4630 026356 001375          BNE   1$          ;;IF NOT: WAIT
4631 026360 010037 001204          MOV    R0,$MSGAD      ;;PUT ADDR IN MAILBOX
4632 026364 105720          2$:  TSTB  (R0)+          ;;FIND END OF MESSAGE
4633 026366 001376          BNE   2$
4634 026370 163700 001204          SUB   $MSGAD,R0      ;;SUB START OF MESSAGE
4635 026374 006200          ASR   R0          ;;GET MESSAGE LGTH IN WORDS
4636 026376 010037 001206          MOV    R0,$MSGGLT      ;;PUT LENGTH IN MAILBOX
4637 026402 012737 000004 001170  MOV   #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
4638 026410 000413          BR    5$
4639 026412 017637 000004 026436  3$:  MOV   @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
4640 026420 062766 000002 000004  ADD   #2,4(SP)        ;;BUMP RETURN ADDRESS
4641 026426 013746 177776          MOV   177776,-(SP)    ;;PUSH 177776 ON STACK
4642 026432 004737 025136          JSR   PC,$TYPE      ;;CALL TYPE MACRO
4643 026436 000000          4$:  .WORD  0
4644 026440          5$:
4645 026440 105737 026526          10$: TSTB  $FFLG      ;;SHOULD REPORT FATAL ERROR?
4646 026444 001416          BEQ   12$          ;;IF NOT: BR
4647 026446 005737 001210          TST   $ENV          ;;RUNNING UNDER APT?
4648 026452 001413          BEQ   12$          ;;IF NOT: BR
4649 026454 005737 001170          11$: TST   $MSGTYPE      ;;FINISHED LAST MESSAGE?
4650 026460 001375          BNE   11$          ;;IF NOT: WAIT
4651 026462 017637 000004 001172  MOV   @4(SP),$FATAL    ;;GET ERROR #
4652 026470 062766 000002 000004  ADD   #2,4(SP)        ;;BUMP RETURN ADDR.
4653 026476 005237 001170          INC   $MSGTYPE      ;;TELL APT TO TAKE ERROR
4654 026502 105037 026526          12$: CLRB  $FFLG      ;;CLEAR FATAL FLAG
4655 026506 105037 026525          CLRB  $LFLG      ;;CLEAR LOG FLAG
4656 026512 105037 026524          CLRB  $MFLG      ;;CLEAR MESSAGE FLAG
4657 026516 012601          MOV   (SP)+,R1      ;;POP STACK INTO R1
4658 026520 012600          MOV   (SP)+,R0      ;;POP STACK INTO R0
4659 026522 000207          RTS   PC          ;;RETURN
4660 026524 000          $MFLG: .BYTE 0      ;;MESSG. FLAG
4661 026525 000          $LFLG: .BYTE 0      ;;LOG FLAG
4662 026526 000          $FFLG: .BYTE 0      ;;FATAL FLAG
4663          .EVEN
4664          APTSIZE=200
4665          APTENV=001
4666          APTPOOL=100

```

4667 000040  
4668  
4669  
4670  
4671  
4672  
4673  
4674  
4675  
4676  
4677  
4678  
4679  
4680  
4681 026530  
4682 026530 104407  
4683 026532 105237 001103  
4684 026536 001775  
4685 026540 013777 001102 152374  
4686 026546 032777 002000 152364  
4687 026554 001402  
4688 026556 104401 001160  
4689 026562 005237 001112  
4690 026566 011637 001116  
4691 026572 162737 000002 001116  
4692 026600 117737 152312 001114  
4693 026606 032777 020000 152324  
4694 026614 001004  
4695 026616 004737 027044  
4696 026622 104401 001165  
4697 026626  
4698 026626 122737 000001 001210  
4699 026634 001007  
4700 026636 113737 001114 026650  
4701 026644 004737 026300  
4702 026650 000  
4703 026651 000  
4704 026652 000777  
4705 026654 005777 152260  
4706 026660 100002  
4707 026662 000000  
4708 026664 104407  
4709 026666  
4710 026666 000002

APTCSUP=040  
.SBITL 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     ,SCRLF
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!
3$:  CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
     RTI      ;;RETURN

```

```

4711
4712
4713
4714
4715
4716
4717
4718 026670
4719 026670 104401 001165
4720 026674 010046
4721 026676 005000
4722 026700 153700 001114
4723 026704 001004
4724
4725 026706 013746 001116
4726
4727 026712 104402
4728 026714 000445
4729 026716 005300
4730 026720 006300
4731 026722 006300
4732 026724 006300
4733 026726 062700 001250
4734 026732 012037 026742
4735 026736 001404
4736 026740 104401
4737 026742 000000
4738 026744 104401 001165
4739 026750 012037 026760
4740 026754 001404
4741 026756 104401
4742 026760 000000
4743 026762 104401 001165
4744 026766 010146
4745 026770 012001
4746 026772 001415
4747 026774 012000
4748 026776 105720
4749 027000 001003
4750 027002 013146
4751 027004 104402
4752 027006 000402
4753 027010
4754 027010 013146
4755 027012 104405
4756 027014 005711
4757 027016 001403
4758 027020 104401 027040
4759 027024 000764
4760
4761 027026 012601
4762 027030 012600
4763 027032 104401 001165
4764 027036 000207
4765 027040 020040 000
4766 027044

```

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

```

```
4767
4768
4769
4770 027044 113737 001114 027076 MYTYPE: MOVB $ITEMB,ERRNUM ;FOR ERROR TYPEOUT
4771 027052 013737 014244 001174 MOV SRXSA,$TESTN ;FOR APT
4772 ;APT PRINTS $TESTN & $FATAL IN THAT ORDER.
4773 ;$TESTN WILL BE FLOPPY RXSA
4774 ;$FATAL IS ERR #
4775 027060 004737 026670 JSR PC,$ERRTYP ;TYPE ERR MSG
4776 027064 005237 027100 INC TERR
4777 027070 005237 027104 INC PERR
4778 027074 000207 RTS PC
4779
4780 027076 000000 ERRNUM: 0 ;FOR ERR TYPEOUT
4781
4782 027100 000000 TERR: 0 ;TOTAL ERROR COUNT
4783 027102 000000 TSERR: 0 ;TOTAL SOFT ERR COUNT
4784 027104 000000 PERR: 0 ;PASS ERR COUNT
4785 027106 000000 PSERR: 0 ;PASS SOFT ERR COUNT... ALL FOR EOP TYPEOUT
```

4786  
4787  
4788  
4789  
4790  
4791  
4792  
4793  
4794  
4795  
4796  
4797  
4798 027110  
4799 027110 010046  
4800 027112 010146  
4801 027114 010246  
4802 027116 010346  
4803 027120 010546  
4804 027122 012746 020200  
4805 027126 016605 000020  
4806 027132 100004  
4807 027134 005405  
4808 027136 112766 000055 000001  
4809 027144 005000  
4810 027146 012703 027324  
4811 027152 112723 000040  
4812 027156 005002  
4813 027160 016001 027314  
4814 027164 160105  
4815 027166 002402  
4816 027170 005202  
4817 027172 000774  
4818 027174 060105  
4819 027176 005702  
4820 027200 001002  
4821 027202 105716  
4822 027204 100407  
4823 027206 106316  
4824 027210 103003  
4825 027212 116663 000001 177777  
4826 027220 052702 000060  
4827 027224 052702 000040  
4828 027230 110223  
4829 027232 005720  
4830 027234 020027 000010  
4831 027240 002746  
4832 027242 003002  
4833 027244 010502  
4834 027246 000764  
4835 027250 105726  
4836 027252 100003  
4837 027254 116663 177777 177776  
4838 027262 105013  
4839 027264 012605  
4840 027266 012603  
4841 027270 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.
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

```

4842	027272	012601		
4843	027274	012600		
4844	027276	104401	027324	
4845	027302	016666	000002	000004
4846	027310	012616		
4847	027312	000002		
4848	027314	023420		
4849	027316	001750		
4850	027320	000144		
4851	027322	000012		
4852	027324	000004		
4853				
4854				
4855				
4856				
4857				
4858				
4859				
4860				
4861				
4862				
4863				
4864				
4865				
4866				
4867				
4868				
4869				
4870				
4871				
4872				
4873				
4874				
4875				
4876				
4877				
4878	027334	017646	000000	
4879	027340	116637	000001	027557
4880	027346	112637	027561	
4881	027352	062716	000002	
4882	027356	000406		
4883	027360	112737	000001	027557
4884	027366	112737	000006	027561
4885	027374	112737	000005	027556
4886	027402	010346		
4887	027404	010446		
4888	027406	010546		
4889	027410	113704	027561	
4890	027414	005404		
4891	027416	062704	000006	
4892	027422	110437	027560	
4893	027426	113704	027557	
4894	027432	016605	000012	
4895	027436	005003		
4896	027440	006105		
4897	027442	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

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

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

```



4898	027444	006105		2\$:	ROL	R5	::FORM THIS DIGIT
4899	027446	006105			ROL	R5	
4900	027450	006105			ROL	R5	
4901	027452	010503			MOV	R5,R3	
4902	027454	006103		3\$:	ROL	R3	::GET LSB OF THIS DIGIT
4903	027456	105337	027560		DECB	\$OMODE	::TYPE THIS DIGIT?
4904	027462	1000i6			BPL	7\$	::BR IF NO
4905	027464	042703	177770		BIC	#177770,R3	::GET RID OF JUNK
4906	027470	001002			BNE	4\$	::TEST FOR 0
4907	027472	005704			TST	R4	::SUPPRESS THIS 0?
4908	027474	001403			BEQ	5\$	::BR IF YES
4909	027476	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
4910	027500	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
4911	027504	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
4912	027510	110337	027554		MOVB	R3,8\$	::SAVE FOR TYPING
4913	027514	104401	027554		TYPE	,8\$	::GO TYPE THIS DIGIT
4914	027520	105337	027556	7\$:	DECB	\$OCNT	::COUNT BY i
4915	027524	003347			BGT	2\$	::BR IF MORE TO DO
4916	027526	002402			BLT	6\$	::BR IF DONE
4917	027530	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
4918	027532	000744			BR	2\$	::GO DO THE LAST DIGIT
4919	027534	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
4920	027536	012604			MOV	(SP)+,R4	::RESTORE R4
4921	027540	012603			MOV	(SP)+,R3	::RESTORE R3
4922	027542	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
4923	027550	012616			MOV	(SP)+,(SP)	
4924	027552	000002			RTI		::RETURN
4925	027554	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
4926	027555	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
4927	027556	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
4928	027557	000		\$UFILL:	.BYTE	0	::ZERO FILL SWITCH
4929	027560	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

```

4930
4931
4932
4933
4934
4935
4936
4937
4938 027562 010046
4939 027564 016600 000002
4940 027570 005740
4941 027572 111000
4942 027574 006300
4943 027576 016000 027616
4944 027602 000200
4945
4946
4947
4948
4949 027604 011646
4950 027606 016666 000004 000002
4951 027614 000002
4952
4953
4954
4955
4956
4957
4958
4959
4960 027616 027604
4961 027620 025136
4962 027622 027360
4963 027624 027334
4964 027626 027374
4965 027630 027110
4966
4967 027632 025470
4968
4969 027634 025420
4970 027636 025742
4971 027640 026062
4972 027642 012152
4973
4974 027644 000040
4975
4976 027744
4977
4978
4979 003334

.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 ;;RESTORE 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
$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
GT$DEV ;;CALL=GTDEV 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

```







DD	= 000040	1067#	3681												
DDISP	= 177570	732#	1154	1610											
DEVN	021500	2995	4377#												
DF	025110	1233	1236	1239	1242	1245	1248	1251	1254	1257	1260	1263	1266	1269	
		1308	1311	1314	1317	1320	1323	1326	1329	1332	1335	1338	1341	1344	
		1347	1350	1353	1356	1362	1365	1368	1371	1374	1377	1380	1383	1410	
		1413	1416	1419	1425	1428	1431	1434	1467	1470	1473	1476	1479	1482	
		1485	1488	1491	1494	1502	1505	4377#							
DF1	025116	1272	1275	1278	1281	1290	1293	1296	1299	1386	1389	1392	1395	1398	
		1401	1404	1407	1449	1458	4377#								
DF2	025126	1284	1287	1302	1305	1359	1452	1461	4377#						
DH1	024170	1248	1308	1314	1317	1320	1323	1326	1335	1338	1341	1502	1505	4377#	
DH2	024207	1233	1236	1242	4377#										
DH3	024233	1239	1245	4377#											
DH4	024275	1251	1311	4377#											
DH5	024326	1254	1257	1260	1263	1266	1416	1419	4377#						
DH6	024363	1269	1329	1332	1344	1347	1350	1353	1356	1362	1365	1368	1371	1374	
		1377	1380	1383	1410	1413	1425	1428	1431	1434	1467	1470	1473	1476	
		1479	1482	1485	1488	1491	1494	4377#							
DH7	024430	1272	1275	1278	1281	1290	1293	1296	1299	1386	1389	1392	1395	1398	
		1401	1404	1407	1449	1458	4377#								
DH8	024517	1284	1287	1302	1305	1359	1452	1461	4377#						
DISPLA	001142	1154#	1610*	1618*	4685*										
DISPRE	000174	1081#	1618												
DIVO	005172	1967	1974#												
DNA	= 100000	1028#													
DNAIE	= 000040	1034#													
DONE	= 000200	958#	993#	1048#	2131	2326	2334	2353	2366	4328	4347				
DROP	020663	2906	4377#												
DRVO	020266	2911	3793	3807	3826	4377#									
DRV1	020273	2912	4154	4168	4188	4377#									
DSC	= 100000	985#													
DSI	= 100000	952#													
DSIE	= 000040	960#	995#												
DSRDY	= 001000	957#	991#												
DSWR	= 177570	731#	1153	1609											
DTR	= 000002	963#	999#	1659	1660	1690	2124	2174	2207	2208	2936	2937			
DT1	024606	1248	1308	1314	1317	1320	1323	1326	1335	1338	1341	1502	1505	4377#	
DT10	024736	1272	1275	1278	1281	1386	1389	1392	1395	1449	4377#				
DT11	024760	1284	1287	1359	1452	4377#									
DT12	025002	1290	1293	1296	1299	1398	1401	1404	1407	1458	4377#				
DT13	025024	1302	1305	1461	4377#										
DT14	025046	1329	1332	1344	1347	1350	1353	1368	1374	1413	1425	1428	1431	1434	
		1467	1470	1473	1476	1479	1482	1485	1488	1491	1494	4377#			
DT15	025064	1416	4377#												
DT16	025076	1419	4377#												
DT2	024614	1233	1236	1242	4377#										
DT3	024624	1239	1245	4377#											
DT4	024640	1251	1311	4377#											
DT5	024650	1254	4377#												
DT6	024662	1257	4377#												
DT7	024674	1260	4377#												
DT8	024706	1263	1266	4377#											
DT9	024720	1269	1356	1362	1365	1371	1377	1380	1383	1410	4377#				
DUN	021011	1807	2040	2160	3797	3809	3828	4158	4170	4190	4377#				
DXTST	003260	1543#	2317*	2404*	2505*	2595*	2640*	2706*	2744*	3553	3679	3741	3765	3836	





















CVKDAC  
CVKDAC.P11

PDT11-150 EXERCISER  
16-FEB-79 11:12

MACY11 30G(1063) 16-FEB-79 11:13 PAGE 128  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0127

STRAP	027562	1603	4938#											
STRAP2	027604	4949#	4960											
STRP =	000013	4953#	4962#	4963#	4964#	4965#	4966#	4967	4968#	4969	4970#	4971#	4972#	4973#
STRPAD	027616	4943	4960#											
STSTM	001004	1122#												
STSTNM	001102	1135#	4685	4711										
STTYIN	026210	4575	4576	4598	4602#									
STYPBN=	***** U	4966												
STYPDS	027110	4798#	4965											
STYPE	025136	4395#	4642	4953	4961									
STYPEC	025350	3017	4425	4432	4439	4444#	4445	4519						
STYPEX	025416	4450	4452	4455#										
STYPOC	027360	4883#	4962											
STYPON	027374	4882	4885#	4964										
STYPOS	027334	4878#	4963											
SUNIT	001202	1178#												
SUNITM	001010	1124#												
SUSWR	001214	1185#												
SVECT1	001240	1210#												
SVECT2	001242	1211#												
\$OFILL	027557	4879*	4883*	4893	4928#									
\$LOCAT=	***** U	4695												
.	= 027744	1080#	1084#	1088#	1091#	1094#	1096#	1098#	1103#	1109	1110#	1112#	1114#	1132#
		1167	1522#	1523#	1540#	1541#	1598	2729	2776	2814#	3077	3103	3131	3144
		3166	4220	4232	4248	4261	4364	4457	4460	4602#	4603	4610#	4663#	4711
		4766#	4852#	4974#										
.\$ASTA=	***** U	4615	4618											
.\$X =	001000	1109#	1114											

Handwritten mark or signature.





CVKDAC  
CVKDAC.P11

PDT11-150 EXERCISER  
16-FEB-79 11:12

MACV11 30G(1063) 16-FEB-79 11:13 PAGE 131  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0129

.EQUAT	691#	715
.HEADE	691#	692
.SETUP	691#	1560
.SWRHI	691#	
.SACT1	691#	
.SAPT8	691#	1168#
.SAPTH	691#	1104
.SAPTY	691#	4611
.SCATC	691#	
.SCMTA	691#	1126
.SEOP	691#	
.SERRO	691#	4668
.SERRT	691#	4711
.SREAD	691#	4457
.SSCOP	691#	
.STRAP	691#	4930
.STYPD	691#	4786
.STYPE	691#	4378
.STYPO	691#	4853

. ABS. 027744 000 OVR RW REL GBL D

ERRORS DETECTED: 0

Z:CVKDAC,Z:CVKDAC.SEQ/CRF/SOL=CVKDAC.P11  
RUN-TIME: 25 15 2 SECONDS  
RUN-TIME RATIO: 138/42=3.2  
CORE USED: 24K (47 PAGES)