

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAE-C-D

PRODUCT NAME: TA11 DATA RELIABILITY

DATE CREATED: 16 MARCH 77

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: JIM LACEY

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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1973,1977 BY DIGITAL EQUIPMEN CORPORATION

CONTENTS

- 1, ABSTRACT
- 2, REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3, LOADING PROCEDURE
- 4, STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM & OPERATOR ACTION
- 5, OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUBROUTINE ABSTRACTS
- 6, ERRORS
 - 6.1 ERROR TYPES
 - 6.2 DATA ERRORS
 - 6.3 ERROR RECOVERY
- 7, RESTRICTIONS
- 8, MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 END OF TEST
 - 8.4 DRIVE COMPATIBILITY
 - 8.5 DATA FORMAT
 - 8.6 TEST TYPEOUT
- 9, PROGRAM DESCRIPTION
 - 9.1 FORMAT PASS
 - 9.2 READ ONLY PASS
 - 9.3 WRITE ONLY PASS

1. ABSTRACT

THIS PROGRAM COLLECTS STATISTICAL INFORMATION PERTAINING TO THE DATA RELIABILITY OF THE TA11/TU60 WHEN RUN FOR EXTENDED PERIODS OF TIME. IT USES A NUMBER OF DIFFERENT PARAMETERS CONTROLLING THE DATA PATTERNS, THE NUMBER OF BYTES PER BLOCK (RECORD) AND THE NUMBER OF BLOCKS PER FILE.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TA11 CASSETTE

2.2 STORAGE

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZTAA
MAINDEC-11-DZTAB
MAINDEC-11-DZTAC
MAINDEC-11-DZTAD

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES OR A CASSETTE TAPE.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

4.2 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS
204 SELECT DRIVE(S) BEFORE STARTING TEST
210 SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST

4.3

PROGRAM & OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A WRITE ENABLED CASSETTE IN BOTH DRIVES
3. LOAD ADDRESS 200.
4. SET SWITCHES (SEE SECTION 4.1)
5. PRESS START.

*** NOTE: IF USING THE SOFTWARE SWITCH REGISTER THE PROGRAM WILL TYPE "SWR=XXXXXX NEW=" AFTER TYPING THE NAME OF THE PROGRAM.

4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF DRIVES "A" AND "B" TO BE TESTED.

STARTING THE PROGRAM AT 204 OR 210 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED.

THE PROGRAM WILL TYPE "DRIVE(S)?".

EITHER OR BOTH DRIVES CAN BE SELECTED BY TYPING "A" AND/OR "B" FOLLOWED BY A CARRIAGE RETURN.

4.3.1.1 DRIVE SELECTION EXAMPLES

DRIVE(S)? A,B	;DRIVES A AND B SELECTED
DRIVE(S)? AB	;DRIVES A AND B SELECTED
DRIVE(S)? B,A	;DRIVES B AND A SELECTED
DRIVE(S)? B	;DRIVE B SELECTED
DRIVE(S)? SB?	;DRIVE S IS IMPOSSIBLE
DRIVE(S)?	;ASK FUR DRIVE

INFORMATION
OF THE
PERIODS
CURRENT
TERMS,
AND

AM AT 210 ALLOWS THE OPERATOR
CONTROL AND STATUS" AND "DATA BUFFER"
THE VECTOR ADDRESS AND THE

ASK FOR THE DRIVES TO BE TESTED AS PER
DRIVES HAVE BEEN SELECTED IT WILL ASK

OF THE CONTROL AND STATUS REGISTER (TACS)
S
L

OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER
: RETURN (WHICH IMPLIES LEAVE AS IS).
PARAMETERS HAVE BEEN DEFINED THE PROGRAM
: GO BACK OUT AND ASK IF THEY ARE OK AT
THE OPERATOR RESPONDES WITH A "Y" OR A
"N" FOR "YES" ANYTHING ELSE IS A "NO".

EXAMPLES

A=177502 VECTOR=000260 PRIORITY=000300

B=177472 VECTOR=000260 PRIORITY=000300

WARE SWITCH REGISTER
ASSETTE

PAGE.

IS TAPES

TEST
BEFORE STARTING TEST

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL:

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

- 1) TYPE CONTROL G <"G"> THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (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 TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)
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 2.

WITH SW<15:10>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST.
THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR
SW<14>=1...LOOP ON TEST
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<10>=1...RING BELL ON ERROR
SW<09>=1...HALT AFTER NEXT "END-OF-TEST" TYPEOUT
SW<08>=1...AT NEXT "END-OF-TAPE" (EOT) GO TO "END-OF-TEST"
SW<07>=1...PERFORM PASS AS PER SWR<1:0>
 SWR<1:0>=00=FORMAT
 SWR<1:0>=01=READ ONLY
 SWR<1:0>=10=WRITE ONLY
 SWR<1:0>=11=READ ONLY

5,2 SUBROUTINE ABSTRACTS

5,2,1 SCOPE

THIS SUBROUTINE CALL (VIA AN IUT INSTRUCTION) IS PLACED AT AN OPTIMUM POSITION IN THE INSTRUCTION SECTION OF THE "FORMAT", "READONLY" AND "WRITEONLY" CODE.

IF SWR<14>*1 THE PROGRAM WILL LOOP THROUGH A SPECIFIC SEQUENCE DEPENDING ON THE TYPE OF PASS BEING PERFORMED.
*** THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

5,2,1.1 FORMAT PASS SCOPE LOOP

1. SETUP FOR A WRITE
2. WRITE
3. BACKSPACE BLOCK GAP
4. SETUP FOR A READ
5. READ
6. REPEAT STEPS 1-5 UNTIL "EOT"

5,2,1.2 READONLY PASS SCOPE LOOP

1. SETUP FOR READ
2. READ
3. CHECK FOR SYNC & DATA ERROR
4. BACK SPACE BLOCK GAP
5. REPEAT STEPS 2-4 INDEFINITELY

5,2,1.3 WRITEONLY PASS SCOPE LOOP

1. SETUP FOR WRITE
2. WRITE
3. REPEAT STEP 2 UNTIL "EUT"

5,2,2 TRAPCATCHER

A ".+2" = "HALT" SEQUENCE IS REPEATED FROM LOC. 0 TO LOC. 776 TO CATCH ANY UNEXPECTED TRAPS. THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2.

5,2,3 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6.)

5,2,4 TRAP

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION, FOLLOWING IS THE CALLS USED AND THE STARTING ADDRESS OF THE ROUTINE.

5,2,4,1 TYPE (\$TYPE)

TYPE AN ASCIZ STRING ON THE TTY

5,2,4,2 RDCHR(\$RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

5,2,4,3 RDLIN(\$RDLIN)

READ AN ASCII STRING FROM THE TTY

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A "JSR".

5.2.6.1 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.2 ASKDRV

THIS ROUTINE IS USED TO ASK THE OPERATOR WHICH DRIVE(S)
ARE TO BE TESTED

5.2.6.3 ASKADR

THIS ROUTINE IS USED TO INPUT THE ADDRESSES OF THE "TACS"
AND THE VECTOR TO USE.

5.2.6.4 TYPERR

THIS ROUTINE IS USED TO TYPE OUT THE "ERROR" DATA

5.2.6.5 CSRERR

THIS ROUTINE IS USED WHEN AN ERROR
IS DETECTED. IT WILL EXAMINE THE
"CONTROL AND STATUS" REGISTER TO
DETERMINE THE TYPE OF ERROR AND
TAKE THE APPROPRIATE ACTION.

5.2.6.6 SETUPW

THIS ROUTINE IS USED TO SETUP THE
PARAMETER BLOCK AND THE WRITE
BUFFER BEFORE STARTING A "WRITE" FUNCTION

5.2.6.7 FILL

USE TO FILL THE WRITE BUFFER
WITH A DATA PATTERN.

5.2.6.8 SETUPR

THIS ROUTINE IS USED TO SETUP THE
PARAMETER BLOCK BEFORE DOING A
"READ" FUNCTION.

5.2.6.9 SYNCK

THIS ROUTINE IS CALLED AFTER
PERFORMING A "READ".
IT CHECKS THE FIRST FOUR BYTES
OF THE DATA TO INSURE THAT THEY
CONTAIN THE PROPER FILE AND
BLOCK NUMBERS.

5.2.6.10 DATCMP

THIS ROUTINE IS USED TO CHECK THE
DATA IN THE READ BUFFER TO INSURE
IT IS CORRECT.

5.2.6.11 CNTSFT

THIS ROUTINE IS USED TO
COUNT SOFT DATA ERRORS

5.2.6.12 CNTHRD

THIS ROUTINE IS USED TO
COUNT HARD DATA ERRORS

5.2.6.13 SAVRGI OR SAVREG

ROUTINE TO SAVE ALL THE REGISTERS

5.2.6.14 CASSETTE PRIMITIVE

THIS IS THE CASSETTE DRIVER

5.2.6.15 CASINT

CASSETTE INTERRUPT HANDLER

5.2.6.16 DBCD

CHANGES A DOUBLE LENGTH BINARY
NUMBER TO A DECIMAL ASCIZ STRING

5.2.6.17 SBCD

CHANGES A SINGLE LENGTH BINARY
NUMBER TO A DECIMAL ASCIZ STRING.

5.2.6.18 SUPRES

TYPES A DECIMAL ASCII STRING
SUPPRESSING LEADING ZEROS.

5.2.6.19 TYPF

USED TO TYPE THE ASCIZ STRING
IMMEDIATELY FOLLOWING THE CALL.

5.2.6.20 TPDRV

TYPES THE DRIVE TO BE TESTED

5.2.6.21 EXAM

THIS ROUTINE EXAMINES THE SELECTED DRIVES
TO INSURE THEY ARE AVAILABLE FOR TESTING.

5.2.6.22 \$B2OCT

TYPES AN OCTAL NUMBER

6. ERRORS

THERE ARE A NUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR(ERROR) ROUTINE IS MADE AND IF SW<13> IS NOT SET AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

REFER TO THE LISTING UNDER \$ERKTB FOR THE DIFFERENT ERRORS THAT CAN OCCUR.

6.1 ERROR TYPES

THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO THREE (3) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

6.1.1 PRETEST ERROR

THESE ERRORS WILL BE DETECTED BEFORE TRYING TO TEST THE DATA RELIABILITY OF THE TA11/TU60.

6.1.2 NON-FATAL ERROR

THESE ERRORS WILL BE DUE TO "CRC" OR "DATA" FAILURES WHICH WILL BE REPORTED AS THEY OCCUR. AFTER REPORTING THE ERROR THE PROGRAM WILL CONTINUE TESTING.

6.1.3 FATAL ERROR

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND OF ERROR THAT CAUSES THE PROGRAM TO LOSE TRACK OF THE TAPE POSITION, OR THE MAXIMUM NUMBER OF DATA ERRORS HAVE OCCURRED.

THIS ERROR WILL BE REPORTED WHEN IT OCCURS, THEN THE PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

6.2 DATA ERRORS

THERE ARE TWO TYPES OF DATA ERRORS THAT CAN OCCUR WHICH ARE DEFINED AND EXPLAINED AS FOLLOWS:

6.2.1 SOFT ERROR

A SOFT ERROR IS BY DEFINITION ANY "CRC" OR "READ DATA" ERROR THAT OCCURS WHILE READING A BLOCK OF DATA. A SOFT ERROR WILL INVOKE A REREAD OF THE BLOCK.

6.2.2 HARD ERROR

A HARD ERROR IS DEFINED AS ANY
"CRC" OR "READ DATA" ERROR THAT OCCURS
ON THE INITIAL READ OF A BLOCK
OF DATA AND CAN NOT BE READ
CORRECTLY WITHIN THREE (3) RETRYS.

6.3 ERROR RECOVERY

6.3.1 PRETEST ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THEN DEPENDING
ON HOW THE PROGRAM WAS STARTED IT WILL ASK FOR THE DRIVES AND
ADDRESSES FOR TESTING OR RETURN TO MONITOR.

6.3.2 NON-FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED AND
THE PROGRAM WILL CONTINUE IN TEST.

6.3.3 FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THE
PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

7. RESTRICTIONS

BEFORE STARTING THE PROGRAM THE OPERATOR MUST INSURE THAT A
CASSETTE IS LOADED AND WRITE ENABLED IN THE DRIVE(S)
TO BE TESTED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

TESTING THE TAI1/TU60 TO SPECIFICATION TAKES APPROXIMATELY
2 HOURS 30 MINUTES WITH EACH DRIVE TAKING 75 MINUTES.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 END OF TEST

WITH ALL SWITCHES ON A "0" THE END OF TEST TYPEOUT WILL OCCUR WHEN THE PROGRAM COMPLETES 18 TAPE PASSES FROM "BOT" TO "EOT" ON THE DRIVE UNDER TEST OR A FATAL ERROR OCCURS.

8.3.1 EXAMPLE OF AN END-OF TEST TYPEOUT

```
*** END-OF-TEST ***
SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1471488
BYTES WRITTEN=369000
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16
```

8.4 DRIVE COMPATIBILITY

THE COMPATIBILITY BETWEEN DRIVES CAN BE TESTED BY DOING A "FORMAT" PASS ON ONE DRIVE AND THEN READING IT ON ANOTHER DRIVE.

8.4.1 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #1

THIS EXAMPLE FORMATS ON DRIVE A AND READS FROM DRIVE B

8.4.1.1 "FORMAT" DRIVE "A"

1. PLACE A WRITE ENABLED TAPE IN DRIVE "A"
2. INSURE DRIVE "B" IS EMPTY
3. LOAD ADDRESS 200
4. SET SW09 AND SW08 TO "1" ALL OTHERS TO "0"
5. PRESS START
6. PROGRAM WILL PERFORM A "FORMAT" PASS ON DRIVE "A", TYPE "END-OF-TEST" STATISTICS AND HALT

8.4.1.2 "READ" DRIVE "B"

1. REMOVE THE TAPE FROM DRIVE "A" AND PLACE IT IN DRIVE "B"
2. LOAD ADDRESS 200
3. SET SW09, SW08, SW07 AND SW00 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS ON DRIVE "B", TYPE "END-OF-TEST" STATISTICS AND HALT.

8,4,2 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #2

THIS EXAMPLE READS KNOWN GOOD TAPE(S)

8,4,2,1 "READ"

1. PLACE THE KNOWN GOOD TAPE(S) IN THE DRIVE(S) TO BE TESTED. (NOTE: IT MIGHT BE WISE TO HAVE THE TAPE(S) WRITE LOCKED.)
2. LOAD ADDRESS 200
3. SET SW08, SW07 AND SW00 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS AND TYPE "END-OF-TEST" STATISTICS ON DRIVE(S) TO BE TESTED.

8,5 DATA FORMAT

THE DATA FORMAT USED IN THIS PROGRAM WILL RESULT IN APPROXIMATELY ELEVEN (11) FILES OF 8192 BYTES TO BE WRITTEN ON TAPE.

8,5,1 FILE STRUCTURE

EACH FILE WILL CONSIST OF SIXTEEN (16) BLOCKS OF DATA, WITH EACH BLOCK CONTAINING AN UNIQUE DATA PATTERN.

8,5,2 BLOCK STRUCTURE

EACH BLOCK WILL HAVE AN "ID" CODE AS THE FIRST FOUR (4) BYTES OF DATA, THIS "ID" WILL BE THE FILE NUMBER AND BLOCK NUMBER AND THEIR COMPLEMENTS. THE DATA FOLLOWING THE "ID" IS A PATTERN THAT REPEATS ITSELF EVERY EIGHT (8) BYTES FOR THE LENGTH OF THE BLOCK.

8,5,3 BLOCK SIZE AND PATTERN

BLOCK NUMBER -----	BLOCK SIZE -----	BLOCK PATTERN -----
0	1024	514 063 514 063 146 231 146 231
1	512	514 231 063 146 514 231 063 146

8.5.3 (CONT.)

BLOCK NUMBER -----	BLOCK SIZE -----	BLOCK PATTERN -----
2	1024	001 002 004 010 020 040 100 200
3	256	177 277 337 357 367 373 375 376
4	1024	252 125 252 125 252 125 252 125
5	128	000 111 222 333 044 155 266 377
6	1024	000 044 111 155 222 266 333 377

8,5,3 (CONT.)

BLOCK NUMBER -----	BLOCK SIZE -----	BLOCK PATTERN -----
7	64	000 222 044 266 111 533 155 377
8	1024	001 003 007 017 037 077 177 377
9	52	376 374 370 360 340 300 200 000
10	128	001 376 002 375 004 373 010 367
11	256	020 357 040 337 100 277 200 177

8.5.3 (CONT.)

BLOCK NUMBER -----	BLOCK SIZE -----	BLOCK PATTERN -----
12	512	000 000 000 000 000 000 000
13	1024	377 377 377 377 377 377 377
14	32	000 377 000 377 000 377
15	128	017 360 207 170 503 074 541 036

8.6 TEST TYPEOUT

THE FOLLOWING EXAMPLE SHOWS A TYPICAL TYPEOUT WHERE BOTH
DRIVES WERE TESTED AND NO ERRORS OCCURRED.

DRIVE A AND DRIVE B WILL BE TESTED

*** FORMAT *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** WRITE *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** HEAD *** DRIVE A
*** HEAD *** DRIVE A
*** FORMAT *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** WRITE *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** HEAD *** DRIVE A

*** END-OF-TEST ***
SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1476608
BYTES WRITTEN=370209
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16

*** FORMAT *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** WRITE *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** FORMAT *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** WRITE *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** HEAD *** DRIVE B

*** END-OF-TEST ***
SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1504096
BYTES WRITTEN=376868
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16

9. PROGRAM DESCRIPTION

THIS PROGRAM IS DESIGNED AROUND THREE PRIMARY ROUTINES THAT WILL TRANSFER DATA TO AND/OR FROM THE TAII/TU60 GOING FROM "BOT" TO "EOT" OF THE TAPE. EACH OF THESE ROUTINES MAKE USE OF COMMON SUBROUTINES TO MANIPULATE TAPE MOTION, KEEP TRACK OF TAPE POSITION, SETUP DATA BUFFERS AND CHECK, COUNT AND REPORT ERRORS. THESE ROUTINES ARE DEFINED AND EXPLAINED BELOW.

9.1 FORMAT PASS

THIS IS A WRITE, BACKSPACE AND READ ROUTINE STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. WRITE A BLOCK OF DATA
2. BACKSPACE A BLOCK GAP
3. READ THE BLOCK
4. CHECK FOR SYNC ERROR
5. CHECK FOR DATA ERRUR
6. REPEAT STEPS 1-5 SIXTEEN TIMES
7. WRITE A FILE GAP
8. REPEAT STEPS 1-7 UNTIL "EOT"

9.2 WRITEONLY PASS

THIS IS A WRITE ONLY ROUTINE. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. WRITE SIXTEEN BLOCKS OF DATA
2. WRITE A FILE GAP
3. REPEAT STEPS 1 & 2 TO "EOT"

9.3 READONLY PASS

THIS IS A READ ONLY ROUTINE AND REQUIRES THAT A "FORMAT" OR "WRITEONLY" PASS HAS ALREADY BEEN PERFORMED. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. READ A BLOCK OF DATA
2. CHECK FOR SYNC ERROR
3. CHECK FOR DATA ERRUR
4. REPEAT STEPS 1-3 SIXTEEN (16) TIMES
5. SPACE FORWARD FILE GAP
6. REPEAT STEPS 1-5 UNTIL THE LAST BLOCK OF THE LAST FILE HAS BEEN READ.

13	GENERAL INFORMATION
59	OPERATIONAL SWITCH SETTINGS
77	BASIC DEFINITIONS
218	STARTING ADDRESSES
219	TRAP CATCHER
228	STARTING ADDRESS(ES)
234	COMMON TAGS
323	BLOCK SIZE TABLE
344	TABLE OF POINTERS TO THE DIFFERENT PATTERNS
366	DATA PATTERNS
420	PARAMETER BLOCK USED WITH ALL FUNCTIONS
428	ERROR POINTER TABLE
530	START UP TEST
552	INITIALIZE THE COMMON TAGS
586	TYPE PROGRAM NAME
593	GET VALUE FOR SOFTWARE SWITCH REGISTER
812	"FORMAT" ROUTINE
964	"READ ONLY" ROUTINE
1101	"WRITE ONLY" ROUTINE
1187	CHECK "EOTS"
1211	END OF PASS ROUTINE
1307	SCOPE HANDLER ROUTINE
1340	ERROR HANDLER ROUTINE
1382	ERROR TIMEOUT ROUTINE
1487	DETERMINE CONTROL AND STATUS ERROR
1575	ROUTINE TO SETUP FOR A WRITE OPERATION
1597	ROUTINE TO FILL THE BUFFER BEFORE A WRITE
1642	ROUTINE TO SETUP FOR A READ
1660	ROUTINE TO CHECK FOR SYNC PROBLEMS
1699	ROUTINE TO CHECK THE READ DATA
1749	COUNT SOFT DATA ERROR
1766	COUNT HARD ERROR
1780	SAVE AND RESTORE R0-R5 ROUTINES
1829	CASSETTE PRIMITIVES ROUTINE
1879	CASSETTE INTERRUPT HANDLER
1924	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
1986	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
2004	TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZERUS
2027	READ AN OCTAL NUMBER FROM THE TTY
2066	ROUTINE TO TYPE DRIVE
2084	ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2119	ROUTINE TO INPUT CSR, DBR, AND VECTOR ADDRESS AND PRIORITY
2180	ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2210	TYPE ROUTINE
2280	TTY INPUT ROUTINE
2419	BINARY TO OCTAL (ASCII) AND TYPE
2496	TRAP DECODER
2519	TRAP TABLE
2540	POWER DOWN AND UP ROUTINES
2585	DATA TABLE POINTERS AND DATA FORMATS FOR ERRORS
2623	ASCII MESSAGES
2785	READ AND WRITE BUFFER

```

1
2
3 .TITLE TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C
4 ;*COPYRIGHT (C) 1973,1977
5 ;*DIGITAL EQUIPMENT COMP,
6 ;*MAYNARD, MASS, 01754
7 ;*
8 ;*PROGRAM BY JIM LACEY
9 ;*
10 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
11 ;*PACKAGE (MAINDEC-11-UZUAC-CS), JAN 19, 1977.
12 ;*
13 ;*****
14 ;*****
15 ;*****
16 .REM!
    
```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

```

777500 TACS CONTROL AND STATUS REGISTER
777502 TADB DATA BUFFER REGISTER
260 TAVEC INTERRUPT VECTOR
    
```

TACS REGISTER DESCRIPTION

BIT	NAME	INIT STATE	READ AND/OR WRITE?
15	ERROR	?	READ ONLY
14	BLOCK CHECK ERROR	0	READ ONLY
13	CLEAR LEADER	?	READ ONLY
12	WRITE LOCK	?	READ ONLY
11	FILE GAP	0	READ ONLY
10	TIMING ERROR	0	READ ONLY
09	OFF LINE	?	READ ONLY
08	UNIT SELECT	0	READ/WRITE
07	TRANSFER REQUEST	0	READ ONLY
06	INTERRUPT ENABLE	0	READ/WRITE
05	READY	1	READ ONLY
04	ILBS	0	READ/WRITE
03	FUNCTION BIT 03	0	READ/WRITE
02	FUNCTION BIT 02	0	READ/WRITE
01	FUNCTION BIT 01	0	READ/WRITE
00	WRITE-FILE-GAP		
	1=WRITE		
	2=READ		
	3=BACK SPACE FILE GAP		
	4=BACK SPACE BLOCK GAP		
	5=SPACE FORWARD FILE GAP		
	6=SPACE FORWARD BLOCK GAP		
	7=REWIND		
00	GO BIT	0	WRITE ONLY!

```

57 .SHTTL OPERATIONAL SWITCH SETTINGS
58 ;*
59 ;* SWITCH USE
60 ;* -----
61 ;* 15 HALT ON ERROR
62 ;* 14 LOOP ON TEST
63 ;* 13 INHIBIT ERROR TIMEOUTS
64 ;* 10 DING BELL ON ERROR
65 ;* 9 HALT AFTER NEXT "END-OF-TEST" TYPE OUT
66 ;* 8 AT NEXT "EOT" GOTO "END-OF-TEST"
67 ;* 7 PERFORM AS PER SWR<1:0>
68 ; 00=FORMAT
69 ; 01=READONLY
70 ; 10=WRITEONLY
71 ; 11=READONLY
72
73
74
75
76
    
```

.SHTTL BASIC DEFINITIONS

```

77 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
78 STACK= 1100
79
80 .EQUIV EMT,ERROR ;*BASIC DEFINITION OF ERROR CALL
81 .EQUIV IOT,SCOPE ;*BASIC DEFINITION OF SCOPE CALL
82
83 ;*MISCELLANEOUS DEFINITIONS
84 MT= 11 ;*CODE FOR HORIZONTAL TAB
85 LF= 12 ;*CODE FOR LINE FEED
86 CR= 15 ;*CODE FOR CARRIAGE RETURN
87 CRLF= 200 ;*CODE FOR CARRIAGE RETURN-LINE FEED
88 PS= 177776 ;*PROCESSOR STATUS WORD
89 .EQUIV PS,PSW
90 STKLM= 177774 ;*STACK LIMIT REGISTER
91 PIRQ= 177772 ;*PROGRAM INTERRUPT REQUEST REGISTER
92 DSWR= 177570 ;*HARDWARE SWITCH REGISTER
93 DDISP= 177570 ;*HARDWARE DISPLAY REGISTER
94
95 ;*GENERAL PURPOSE REGISTER DEFINITIONS
96 R0= X0 ;*GENERAL REGISTER
97 R1= X1 ;*GENERAL REGISTER
98 R2= X2 ;*GENERAL REGISTER
99 R3= X3 ;*GENERAL REGISTER
100 R4= X4 ;*GENERAL REGISTER
101 R5= X5 ;*GENERAL REGISTER
102 R6= X6 ;*GENERAL REGISTER
103 R7= X7 ;*GENERAL REGISTER
104 SP= X6 ;*STACK POINTER
105 PC= X7 ;*PROGRAM COUNTER
106
107 ;*PRIORITY LEVEL DEFINITIONS
108 PR0= 0 ;*PRIORITY LEVEL 0
109 PR1= 40 ;*PRIORITY LEVEL 1
110 PR2= 100 ;*PRIORITY LEVEL 2
111 PR3= 140 ;*PRIORITY LEVEL 3
112 PR4= 200 ;*PRIORITY LEVEL 4
113 PR5= 240 ;*PRIORITY LEVEL 5
    
```

```

113      000300      PR6=      300      //PRIORITY LEVEL 6
114      000340      PR7=      340      //PRIORITY LEVEL 7
115
116      //SWITCH REGISTER SWITCH DEFINITIONS
117      100000      SW15=     100000
118      040000      SW14=     400000
119      020000      SW13=     200000
120      010000      SW12=     100000
121      004000      SW11=     400000
122      002000      SW10=     200000
123      001000      SW09=     100000
124      000400      SW08=     400000
125      000200      SW07=     200000
126      000100      SW06=     100000
127      000040      SW05=     400000
128      000020      SW04=     200000
129      000010      SW03=     100000
130      000004      SW02=     400000
131      000002      SW01=     200000
132      000001      SW00=     100000
133      .EQUIV SW09,SW9
134      .EQUIV SW08,SW8
135      .EQUIV SW07,SW7
136      .EQUIV SW06,SW6
137      .EQUIV SW05,SW5
138      .EQUIV SW04,SW4
139      .EQUIV SW03,SW3
140      .EQUIV SW02,SW2
141      .EQUIV SW01,SW1
142      .EQUIV SW00,SW0
143
144      //DATA BIT DEFINITIONS (BIT00 TO BIT15)
145      100000      BIT15=    100000
146      040000      BIT14=    400000
147      020000      BIT13=    200000
148      010000      BIT12=    100000
149      004000      BIT11=    400000
150      002000      BIT10=    200000
151      001000      BIT09=    100000
152      000400      BIT08=    400000
153      000200      BIT07=    200000
154      000100      BIT06=    100000
155      000040      BIT05=    400000
156      000020      BIT04=    200000
157      000010      BIT03=    100000
158      000004      BIT02=    400000
159      000002      BIT01=    200000
160      000001      BIT00=    100000
161      .EQUIV BIT09,BIT9
162      .EQUIV BIT08,BIT8
163      .EQUIV BIT07,BIT7
164      .EQUIV BIT06,BIT6
165      .EQUIV BIT05,BIT5
166      .EQUIV BIT04,BIT4
167      .EQUIV BIT03,BIT3
168      .EQUIV BIT02,BIT2
    
```

```

169      .EQUIV BIT01,BIT1
170      .EQUIV BIT00,BIT0
171
172      //BASIC "CPU" TRAP VECTOR ADDRESSES
173      000004      ERKVEC= 4      //TIME OUT AND OTHER ERRORS
174      000010      RESVEC= 10     //RESERVED AND ILLEGAL INSTRUCTIONS
175      000014      TBITVEC=14    //"T" BIT
176      000014      TRTVEC= 14     //TRACE TRAP
177      000014      BPTVEC= 14    //BREAKPOINT TRAP (BPT)
178      000020      IOTVEC= 20    //INPUT/OUTPUT TRAP (IOT) **SCOPE**
179      000024      PFWVEC= 24    //POWER FAIL
180      000030      EMTVEC= 30    //EMULATOR TRAP (EMT) **ERROR**
181      000034      TRAPVEC=34    //"TRAP" TRAP
182      000060      TRKVEC= 60    //TTY KEYBOARD VECTOR
183      000064      TPVEC= 64     //TTY PRINTER VECTOR
184      000240      PIRQVEC=240   //PROGRAM INTERRUPT REQUEST VECTOR
185      //*****
186
187      //*****TA11 FUNCTIONS*****
188      XWFG= 0      //WRITE FILE GAP FUNCTION
189      XWRITE= 2    //WRITE FUNCTION
190      XREAD= 4     //READ FUNCTION
191      XBSFG= 6    //BACK SPACE FILE GAP FUNCTION
192      XBSBG= 10   //BACK SPACE BLOCK GAP FUNCTION
193      XSPFG= 12   //SPACE FWD FILE GAP FUNCTION
194      XSPBG= 14   //SPACE FWD BLOCK GAP FUNCTION
195      XRWD= 16    //REWIND FUNCTION
196
197      //*****TA11 BIT ASSIGNMENT*****
198      ERROR= BIT15
199      CRCERR= BIT14
200      LEADER= BIT13
201      WRTLOCK=BIT12
202      FGAP= BIT11
203      TIMERR= BIT10
204      OFFLINE=BIT09
205      UNIT= BIT08
206      TR,REQ= BIT07
207      INT,EN= BIT06
208      READY= BIT05
209      ILBS= BIT04
210      FUNC2= BIT03
211      FUNC1= BIT02
212      FUNC0= BIT01
213      GO= BIT00
214      FUNCTION=   FUNC2+FUNC1+FUNC0
215
    
```

```

216 .SBTTL TRAP CATCHER
217
218 000000 .=0
219 /*ALL UNUSED LOCATIONS FROM 4 = 776 CONTAIN A "+2,HALT"
220 /*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
221 /*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
222 .*=174
223 000174 000000 DISPREG: ,WORD 0 /*SOFTWARE DISPLAY REGISTER
224 000176 000000 SWREG: ,WORD 0 /*SOFTWARE SWITCH REGISTER
225 .SBTTL STARTING ADDRESS(ES)
226 000200 000137 001752 JMP @#BEGIN1 /*JUMP TO STARTING ADDRESS OF PROGRAM
227 000204 000137 002004 JMP @#BEGIN2 /*SELECT DRIVE(S) BEFORE STARTING TEST
228 000210 000137 002012 JMP @#BEGIN3 /*SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
229 ;*****
230
    
```

```

231 .SBTTL COMMON TAGS
232
233 ;*****
234 /*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
235 /*USED IN THE PROGRAM.
236
237 001100 .*=1100
238 001100 SCMTAG: /*START OF COMMON TAGS
239 001100 000000 SPASS: ,WORD 0 /*CONTAINS PASS COUNT
240 001102 000 STSNM: ,BYTE 0 /*CONTAINS THE TEST NUMBER
241 001103 000 SERFLG: ,BYTE 0 /*CONTAINS ERROR FLAG
242 001104 000000 SICNT: ,WORD 0 /*CONTAINS SUBTEST ITERATION COUNT
243 001106 000000 SLPADR: ,WORD 0 /*CONTAINS SCOPE LOOP ADDRESS
244 001110 000000 SLPERR: ,WORD 0 /*CONTAINS SCOPE RETURN FOR ERRORS
245 001112 000000 SERTTL: ,WORD 0 /*CONTAINS TOTAL ERRORS DETECTED
246 001114 000 SITEMB: ,BYTE 0 /*CONTAINS ITEM CONTROL BYTE
247 001115 001 SERRMAX: ,BYTE 1 /*CONTAINS MAX. ERRORS PER TEST
248 001116 000000 SERRPC: ,WORD 0 /*CONTAINS PC OF LAST ERROR INSTRUCTION
249 001120 000000 SGDADR: ,WORD 0 /*CONTAINS ADDRESS OF "GOOD" DATA
250 001122 000000 SBDADR: ,WORD 0 /*CONTAINS ADDRESS OF "BAD" DATA
251 001124 000000 SGDDAT: ,WORD 0 /*CONTAINS "GOOD" DATA
252 001126 000000 SBDAT: ,WORD 0 /*CONTAINS "BAD" DATA
253 001130 000000 /*RESERVED--NOT TO BE USED
254 001132 000000 /*WORD 0
255 001134 000 SAUTOB: ,BYTE 0 /*AUTOMATIC MODE INDICATOR
256 001135 000 SINTAG: ,BYTE 0 /*INTERRUPT MODE INDICATOR
257 001136 000000 /*WORD 0
258 001140 177570 SWR: ,WORD DSWR /*ADDRESS OF SWITCH REGISTER
259 001142 177570 DISPLAY: ,WORD DDISP /*ADDRESS OF DISPLAY REGISTER
260 001144 177560 STKS: 177560 /*TTY KBD STATUS
261 001146 177562 STKB: 177562 /*TTY KBD BUFFER
262 001150 177564 STPS: 177564 /*TTY PRINTER STATUS REG. ADDRESS
263 001152 177566 STPB: 177566 /*TTY PRINTER BUFFER REG. ADDRESS
264 001154 000 SNULL: ,BYTE 0 /*CONTAINS NULL CHARACTER FOR FILLS
265 001155 002 SFILLS: ,BYTE 2 /*CONTAINS # OF FILLER CHARACTERS REQUIRED
266 001156 012 SFILLC: ,BYTE 12 /*INSERT FILL CHARS. AFTER A "LINE FEED"
267 001157 000 STPFLG: ,BYTE 0 /*"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
268 001160 000000 SREGAD: ,WORD 0 /*CONTAINS THE ADDRESS FROM
269 /*WHICH ((SREGAD) WAS OBTAINED
270 001162 000000 SREG0: ,WORD 0 /*CONTAINS ((SREGAD)+0)
271 001164 000000 SREG1: ,WORD 0 /*CONTAINS ((SREGAD)+1)
272 001166 000000 SREG2: ,WORD 0 /*CONTAINS ((SREGAD)+2)
273 001170 000000 SREG3: ,WORD 0 /*CONTAINS ((SREGAD)+3)
274 001172 000000 STMP0: ,WORD 0 /*USER DEFINED
275 001174 000000 STMP1: ,WORD 0 /*USER DEFINED
276 001176 000000 STMP2: ,WORD 0 /*USER DEFINED
277 001200 000000 STMP3: ,WORD 0 /*USER DEFINED
278 001202 077 SQUES: ,ASCII /?/ /*QUESTION MARK
279 001203 015 SCRLF: ,ASCII <15> /*CARRIAGE RETURN
280 001204 00012 SLF: ,ASCII <12> /*LINE FEED
281 ;*****
282 001206 000000 SOFTNM: ,WORD 0 /*NUMBER OF SOFT ERRORS
283 001210 000000 HARDNM: ,WORD 0 /*NUMBER OF HARD ERRORS
284 001212 000000 EOTS: ,WORD 0 /*NUMBER OF TAPE PASSES
285 001214 000000 000000 RBYTTL: ,WORD 0,0 /*NUMBER OF BYTES READ
286 001220 000000 000000 WBYTTL: ,WORD 0,0 /*NUMBER OF BYTES WRITTEN
    
```


287	001224	000000		RDRY6:	WORD	0		JKEEPS COUNT OF REREADS
288	001226	000000		WRTRY6:	WORD	0		JKEEPS COUNT OF REWRITES
289	001230	000000	000000	BYNUM:	WORD	0,0		JTHE NUMBER OF BYTES "READ" OR "WRITTEN
290								J DURING AN OPERATION
291	001234	000017		LASTBK:	WORD	15.		JWILL CONTAIN THE # OF THE LAST BLOCK
292								J AFTER A "FORMAT" OR "WRITE" PASS
293	001236	000011		LASTFL:	WORD	9.		JWILL CONTAIN THE # OF THE LAST FILE
294								J AFTER A "FORMAT" OR "WRITE" PASS
295								
296								
297	001240	177500		TACSL:	WORD	177500		J ADDRESS OF TACS
298	001242	177502		TADBL:	WORD	177502		J ADDRESS OF TADB
299	001244	000260	000262	TAVEC:	WORD	260,262		J TAIL VECTOR ADDRESS
300	001250	000300		TAPRIO:	WORD	300		J TAIL BR LEVEL 6
301	001252	000000	000000	DRVKEY:	WORD	0,0		
302	001256	001252		DRVPT:	WORD	001252		
303	001260	000003		MAXRDS:	WORD	3		JMAX REREADS BEFORE CALLING IT A HARD ERROR
304	001262	000003		MAXERR:	WORD	3		JMAX HARD ERRORS ALLOWED
305	001264	000022		MAXEOT:	WORD	18.		JNUMBER OF TAPE PASSES BEFORE END-OF-TEST
306	001266	000000		PASCNT:	WORD	0		JCOUNT # OF TAPE PASSES
307	001270	001274	000000	PSCNTL:	WORD	FORPAS,0		JCONTROLS THE TYPE OF PASS
308	001274	000001		FDKPAS:	WORD	1		J 1 FORMAT PASS
309	001276	000003		RDIPAS:	WORD	3		J 3 READONLY PASSES
310	001300	000001		WRTPAS:	WORD	1		J 1 WRITEDONLY PASS
311	001302	000004		RDPAS:	WORD	0		J 4 READONLY PASSES
312	001304	000		FILE:	WORD	0		JFILE NUMBER
313	001305	377		BLOCK:	WORD	377		J1'S COMPLEMENT ON FILE NUMBER
314	001306	000		BLOCK:	WORD	0		JBLOCK NUMBER
315	001307	377		BLOCK:	WORD	377		J1'S COMPLEMENT OF BLOCK NUMBER
316	001310	000020		FILESZ:	WORD	16.		JNUMBER OF BLOCKS PER FILE

317								
318								
319								
320								
321								
322	001312	002000		BLKSZ:	WORD	1024.		JBLOCK 0
323	001314	001000			WORD	512.		JBLOCK 1
324	001316	002000			WORD	1024.		JBLOCK 2
325	001320	000400			WORD	256.		JBLOCK 3
326	001322	002000			WORD	1024.		JBLOCK 4
327	001324	002000			WORD	128.		JBLOCK 5
328	001326	002000			WORD	1024.		JBLOCK 6
329	001330	000100			WORD	64.		JBLOCK 7
330	001332	002000			WORD	1024.		JBLOCK 8
331	001334	000040			WORD	32.		JBLOCK 9
332	001336	000200			WORD	128.		JBLOCK 10
333	001340	000400			WORD	256.		JBLOCK 11
334	001342	001000			WORD	512.		JBLOCK 12
335	001344	002000			WORD	1024.		JBLOCK 13
336	001346	000040			WORD	32.		JBLOCK 14
337	001350	000200			WORD	128.		JBLOCK 15
338								
339								
340								
341								
342								
343	001352	001412		PATS:	PAT0			
344	001354	001422			PAT1			
345	001356	001432			PAT2			
346	001360	001442			PAT3			
347	001362	001452			PAT4			
348	001364	001462			PAT5			
349	001366	001472			PAT6			
350	001370	001502			PAT7			
351	001372	001512			PAT8			
352	001374	001522			PAT9			
353	001376	001532			PAT10			
354	001400	001542			PAT11			
355	001402	001552			PAT12			
356	001404	001562			PAT13			
357	001406	001572			PAT14			
358	001410	001602			PAT15			
359								

```

360
361
362
363
364
365 001412 314 063 314 PAT0: .BYTE 314,063,314,063,146,231,146,231
366 001415 063 146 231
367 001420 146 231
368 001422 314 231 063 PAT1: .BYTE 314,231,063,146,314,231,063,146
369 001425 146 314 231
370 001430 063 146
371 001432 001 002 004 PAT2: .BYTE 001,002,004,010,020,040,100,200
372 001435 010 020 040
373 001440 100 200
374 001442 177 277 337 PAT3: .BYTE 177,277,337,367,373,375,376
375 001445 357 367 373
376 001450 375 376
377 001452 252 125 252 PAT4: .BYTE 252,125,252,125,252,125,252,125
378 001455 125 252 125
379 001460 252 125
380 001462 000 111 222 PAT5: .BYTE 000,111,222,333,044,155,266,377
381 001465 333 044 155
382 001470 266 377
383 001472 000 044 111 PAT6: .BYTE 000,044,111,155,222,266,333,377
384 001475 155 222 266
385 001500 333 377
386 001502 000 222 044 PAT7: .BYTE 000,222,044,266,111,333,155,377
387 001505 266 111 333
388 001510 155 377
389 001512 001 003 007 PAT8: .BYTE 001,003,007,017,037,077,177,377
390 001515 017 037 077
391 001520 177 377
392 001522 376 374 370 PAT9: .BYTE 376,374,370,360,340,300,200,000
393 001525 360 340 300
394 001530 200 000
395 001532 001 376 002 PAT10: .BYTE 001,376,002,375,004,373,010,367
396 001535 375 004 373
397 001540 010 367
398 001542 020 357 040 PAT11: .BYTE 020,357,040,337,100,277,200,177
399 001545 337 100 277
400 001550 200 177
401 001552 000 000 000 PAT12: .BYTE 000,000,000,000,000,000,000,000
402 001555 000 000 000
403 001560 000 000
404 001562 377 377 377 PAT13: .BYTE 377,377,377,377,377,377,377,377
405 001565 377 377 377
406 001570 377 377
407 001572 000 377 000 PAT14: .BYTE 000,377,000,377,000,000,377,377
408 001575 377 000 000
409 001600 377 377
410 001602 017 360 207 PAT15: .BYTE 017,360,207,170,303,074,341,036
411 001605 170 303 074
412 001610 341 036
413
    
```

```

414
415
416
417
418
419 001612 000 PAMMBK: .BYTE 0 IUSED FOR STATUS/ERROR
420 001613 000 .BYTE 0 IDRIVE # (DRIVE A=0, B=1)
421 001614 001612 .WORD PAMMBK IPOINTS TO STATUS/ERROR BYTE
422 001616 013464 .WORD BUFFER IFIRST ADDRESS OF DATA BUFFER
423 001620 000000 .WORD 0 IUSED FOR BYTE COUNT.
424
    
```

```

425 .SRTTL ERROR POINTER TABLE
426
427 /*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR,
428 /*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
429 /*LOCATION SITMB, THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
430 /*NOTE1: IF SITMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC),
431 /*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
432
433 /* EM //POINTS TO THE ERROR MESSAGE
434 /* DM //POINTS TO THE DATA HEADER
435 /* DT //POINTS TO THE DATA
436 /* DF //POINTS TO THE DATA FORMAT
437
438
439 001622 SENRTB1
440
441 /******
442 /******
443 /******
444
445 001622 ITEMS0: ITEMS 001-002
446
447 /*NOTE1 ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS
448 /* UNLESS OTHERWISE NOTED
449
450
451 001622 012244 /*ITEM 1
452 001624 012535 EM1 /*DATA ERROR
453 001626 011770 DM1 /*PC FILE BLOCK BYTE GDDAT BDDAT GDADR BODR
454 001630 012052 DT1 /*SERRPC SREG0 SREG2 BYTNUM SGDDAT SBDDAT SGADR SBADR
455 DF1 /*FILE,BLOCK AND BYTE ARE TYPED IN DECIMAL
456
457 001632 012257 /*ITEM 2
458 001634 012633 EM2 /*SYNC ERROR
459 DM2 /* EXPT'D EXPT'D RCV'D RCV'D
460 001636 012012 DT2 /*SERRPC SREG0 SREG2 STMP0 STMP2
461 001640 012062 DF2 /*ALL NUMBERS EXCEPT SERRPC ARE TYPED IN DECIMAL
462
463 /******
464 /******
465 /******
466
467 001642 ITEMS1: ITEMS 101-107
468
469 001642 012272 EM101 /*DRIVE IS OFF-LINE
470 001644 012750 DM101 /*PC FILE BLOCK FUNCTION
471 001646 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
472 001650 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
473 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
474
475 001652 012314 EM102 /*DRIVE IS WRITE-LOCK
476 001654 012750 DM101 /*PC FILE BLOCK FUNCTION
477 001656 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
478 001660 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
479 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
480
    
```

```

481 001662 012340 EM103 /*CLEAR LEADER ERROR
482 001664 012750 DM101 /*PC FILE BLOCK FUNCTION
483 001666 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
484 001670 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
485 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
486
487 001672 012363 EM104 /*FILE GAP ERROR
488 001674 012750 DM101 /*PC FILE BLOCK FUNCTION
489 001676 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
490 001700 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
491 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
492
493 001702 012402 EM105 /*TIMING ERROR
494 001704 012750 DM101 /*PC FILE BLOCK FUNCTION
495 001706 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
496 001710 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
497 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
498
499 001712 012417 EM106 /*BLOCK CHECK ERROR
500 001714 012750 DM101 /*PC FILE BLOCK FUNCTION
501 001716 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
502 001720 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
503 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
504
505 001722 012441 EM107 /*UNKNOWN INTERRUPT
506 001724 012750 DM101 /*PC FILE BLOCK FUNCTION
507 001726 012026 DT101 /*SERRPC SREG0 SREG2 STMP0
508 001730 000000 0 /*FILE AND BLOCK WILL BE TYPED IN DECIMAL
509 /*STMP0 WILL BE TYPED AS A FUNCTION NAME
510
511 /******
512 /******
513 /******
514
515 001732 ITEMS2: ITEMS 201-202
516
517 001732 012463 EM201 /*TA11 FAILED TO RESPOND
518 001734 013011 DM201 /*PC TACS
519 001736 012040 DT201 /*SERRPC TACS
520 001740 000000 0 /*BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
521
522 001742 012512 EM202 /*NO DRIVES AVAILABLE
523 001744 013026 DM202 /*PC
524 001746 012046 DT202 /*SERRPC
525 001750 000000 0
526
    
```

```

527 ////////////////////////////////////////////////////////////////////
528 ////////////////////////////////////////////////////////////////////
529 //*****
530
531 //BEGIN1 IS FOR NORMAL START
532 //BEGIN2 IS FOR DRIVE SELECTION
533 //BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
534
535 //*****
536
537 001752 005005          BEGIN1: CLR    R5          ;NORMAL START
538 001754 012737 041101 001252  MOV    #*AB,#DRVKEY
539 001762 122737 000005 000041  CMPB   #5,#41          ;CASSETTE DDP?
540 001770 001012          BGNCMN          ;GO BEGIN COMMON CODE IF NO
541 001772 022737 000260 001244  CMP    #260,#TAVEC    ;STANDARD VECTOR?
542 002000 001006          BGNCMN          ;GO BEGIN COMMON CODE IF NO
543 002002 000403          BR     BEGIN3        ;GET DRIVES AND ADDRESSES
544 002004 012705 000001          BEGIN2: MOV   R5          ;ASK FOR DRIVES FLAG
545 002010 000402          BR     BGNCMN        ;BEGIN COMMON CODE
546 002012 012705 000002          BEGIN3: MOV  R5          ;ASK FOR DRIVES AND ADDRESSES
547 002016 005067 175754          BGNCMN: CLR    PS
548
549 .SBTTL INITIALIZE THE COMMON TAGS
550 //CLEAR THE COMMON TAGS (SCMTAG) AREA
551 002022 012706 001100          MOV    #SCMTAG,R6    ;FIRST LOCATION TO BE CLEARED
552 002026 005026          CLR    (R6)+        ;CLEAR MEMORY LOCATION
553 002030 022706 001140          CMP    #SWR,R6 ;DONE?
554 002036 012706 001100          BNE   #-6           ;LOOP BACK IF NO
555
556 //INITIALIZE A FEW VECTORS
557 002042 012737 004760 000020  MOV    #SSCOPE,#IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
558 002050 012737 000340 000022  MOV    #340,#IOTVEC+2 ;LEVEL 7
559 002056 012737 005060 000030  MOV    #SEMH0K,#EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
560 002064 012737 000340 000032  MOV    #340,#EMTVEC+2 ;LEVEL 7
561 002072 012737 011522 000034  MOV    #STHAP,#THAPVEC ;TRAP VECTOR FOR TRAP CALLS
562 002100 012737 000340 000036  MOV    #340,#THAPVEC+2;LEVEL 7
563 002106 012737 011606 000024  MOV    #SPK0DN,#PWRVEC ;POWER FAILURE VECTOR
564 002114 012737 000340 000026  MOV    #340,#PWRVEC+2 ;LEVEL 7
565 002122 016767 002314 002304  MOV    $ENDUCT,$EOPCT ;SETUP END-OF-PROGRAM COUNTER
566 002130 012767 002130 176750  MOV    #*,$LPADR     ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
567
568 //SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
569 //EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
570 002136 013746 000004          MOV    #ERRVEC,-(SP) ;SAVE ERROR VECTOR
571 002142 012737 002176 000004  MOV    #64$,#ERRVEC  ;SET UP ERROR VECTOR
572 002150 012767 177570 176762  MOV    #DSWR,SWR     ;SETUP FOR A HARDWARE SWICH REGISTER
573 002156 012767 177570 176756  MOV    #DISP,DISPLAY ;HARD A HARDWARE DISPLAY REGISTER
574 002164 022777 177777 176746  CMP    #*1,#SWR     ;TRY TO REFERENCE HARDWARE SWR
575 002172 001012          BNE   66$           ;BRANCH IF NO TIMEOUT TRAP OCCURRED
576
577 002174 000403          BR     65$         ;HARD THE HARDWARE SWR IS NOT = -1
578 002176 012716 002204          64$: MOV   #65$, (SP) ;BRANCH IF NO TIMEOUT
579 002202 000002          RTI
580 002204 012767 000176 176726  65$: MOV   #SWREG,SWR ;POINT TO SOFTWARE SWR
581 002212 012767 000174 176722  66$: MOV   #DISPREG,DISPLAY ;BRANCH IF NO TIMEOUT
582 002220 012637 000004          67$: MOV   (SP)+,#ERRVEC ;RESTORE ERROR VECTOR
583
584 .SBTTL TYPE PROGRAM NAME

```

```

583 //TYPE THE NAME OF THE PROGRAM IF FIRST PASS
584 002224 005227 177777          INC    #-1           ;FIRST TIME?
585 002230 001036          BNE   67$           ;BRANCH IF NO
586 002232 022737 004726 000042  CMP    #SENDAD,#42   ;ACT=11?
587 002240 001432          BEQ   67$           ;BRANCH IF YES
588 002242 104401 002300          TYPE ,68$         ;TYPE ASCII STRING
589
590 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
591 002246 005737 000042          TST   #42          ;ARE WE RUNNING UNDER XXDP/ACT?
592 002252 001006          BNE   69$           ;BRANCH IF YES
593 002254 026727 176660 000176  CMP    SWR,#SWREG    ;SOFTWARE SWITCH REG SELECTED?
594 002262 001005          BNE   70$           ;BRANCH IF NO
595 002264 104405          GTSWR          ;GET SOFT=SWR SETTINGS
596 002266 000403          BR     70$         ;
597 002270 112767 000001 176636  69$: MOVB  #1,$AUTOB   ;SET AUTO=MODE INDICATOR
598 002276 000413          70$: BR     67$         ;GET OVER THE ASCII
599 //68$: .ASCII <CRLF> /MAINDEC=11-DZTAE=C /<CRLF>
600 002326          67$:
601
602 //*****
603 //*****
604
605 //THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
606 /
607 /      R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
608 /      R5=1 ASK FOR DRIVE(S)
609 /      R5=0 DON'T ASK FOR ANYTHING
610 /
611 //*****
612 002326 010504          BEGINX: MOV  R5,R4    ;COPY R5
613 002330 005305          DEC    R5           ;ASK FOR DRIVES?
614 002332 002406          BLT   CHKADR        ;BR IF NO
615 002334 004737 007620          JSR   PC,#ASKDRV    ;GO GET DRIVES TO BE TESTED
616 002340 005305          DEC    R5           ;ASK FOR ADDRESSES?
617 002342 002402          BLT   CHKADR        ;BR IF NO
618 002344 004737 007730          JSR   PC,#ASKADR    ;GO GET TA11 ADDRESSES
619
620 //*****
621 //*****
622 //CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
623 /
624 /      I. TIMEOUT OCCURRED
625 /      A. TYPE ERROR MESSAGE
626 /      B. EXAMINE R4
627 /
628 /      1. R4=0 GOTO BEGINX
629 /      2. R4=0 EXAMINE (42)
630 /      A. (42)=0 GOTO BEGINX
631 /      B. (42)=0 GOTO SENDAD
632 /
633 /      II. TIMEOUT DIDN'T OCCUR
634 /      A. CONTINUE
635 /
636 //*****
637 002350 012737 002366 000004  CHKADR: MOV  #1$,#ERRVEC ;IN CASE OF TIMEOUTS
638 002356 005000          CLR    R0           ;USE AS A SWITCH
639 002360 005777 176654          TST   @TACSL       ;SEE IF TA11 RESPONDS

```

```

639 002364 000402          BR      28          JBR IF NO TIMEOUT
640 002366 005200          1S:   INC      R0          JCOME HERE ON TIMEOUT
641 002370 022626          CMP      (SP)+,(SP)+      JCLEANUP THE STACK
642 002372 012737 000006 000004 2S:   MOV      #ERRVEL+2,#ERRVEC JRESTORE TIMEOUT VECTOR
643 002400 005700          TST      R0          JDID A TIMEOUT OCCUR?
644 002402 001412          BEQ      3S          JBR IF NO
645 002404 104201          ERROR  201         JTA11 FAILED TO RESPOND
646 002406 012705 000002          MOV      #2,R5         JDRIVES & ADDRESSES
647 002412 005704          TST      R4          JOPERATOR INPUTS?
648 002414 001344          BNE     BEGINX        JBR IF YES
649 002416 013700 000042          MOV      #42,R0        JGET MONITOR RETURN ADDRESS
650 002422 001741          BEQ     BEGINX        JBR IF NO MONITOR
651 002424 000137 004726          JMP      #SENDAD       JGO TO END
652 002430 012777 007056 176606 3S:   MOV      #CASINT,#TAVEC JSETUP CASSETTE INTERRUPT VECTOR
653 002436 005077 176604          CLR      @TAVEC+2
    
```

```

654                                     J*****
655                                     J*****
656                                     J*****
657                                     J*****
658                                     J*****
659                                     J*****
660                                     J*****
661                                     J*****
662                                     J*****
663                                     J*****
664                                     J*****
665                                     J*****
666                                     J*****
667                                     J*****
668                                     J*****
669                                     J*****
670                                     J*****
671                                     J*****
672                                     J*****
673                                     J*****
674                                     J*****
675                                     J*****
676                                     J*****
677 002442 012700 001252          CHKDRV: MOV     #DRVKEY,R0      JPICKUP ADDRESS OF ASCII DRIVE KEY
678 002446 004737 010174          JSR      PC,#EXAM         JGO EXAMINE FIRST DRIVE
679 002452 000410          BR      1S              JOK TO TEST--GO CHECK NEXT
680 002454 116010 000001          MOV     1(R0),(R0)       JREPLACE 1ST WITH 2ND
681 002460 001412          BEQ     2S              JBR IF NO 2ND DRIVE SELECTED
682 002462 004737 010174          JSR      PC,#EXAM         JGO EXAMINE DRIVE
683 002466 000407          BR      2S              JOK TO TEST
684 002470 005010          CLR     (R0)           JCLEAR DRIVE CODES
685 002472 000405          BR      2S
686 002474 005200          1S:   INC      R0          JPOINT TO 2ND
687 002476 004737 010174          JSR      PC,#EXAM         JGO EXAMINE DRIVE
688 002502 000401          BR      2S              JOK TO TEST
689 002504 105010          CLR     (R0)           JCLEAR 2ND
690 002506 012700 001252          2S:   MOV     #DRVKEY,R0      JRESET ADDRESS POINTERS
691 002512 010037 001256          MOV     R0,#DRVVPNT
692 002516 121060 000001          CMP     (R0),1(R0)      J1ST = 2ND?
693 002522 001002          BNE     3S              JBR IF NO
694 002524 105060 000001          CLR     1(R0)           JYES--CLEAR 2ND
695 002530 005710          3S:   TST     (R0)           JANY DRIVES?
696 002532 001424          BEQ     5S              JBR IF NO
697 002534 112037 001613          MOV     (R0)+,#PAMMBK+1 JSETUP TO TYPE THIS DRIVE
698 002540 104401 001203          TYPE   ,SCLRF          JTYPE "CR" & "LF"
699 002544 106237 001613          ASRB   #PAMMBK+1       JADJUST FOR TYPING
700 002550 004737 007564          JSR     PC,@TPDKV       JGO TYPE DRIVE
701 002554 111037 001613          MOV     (R0),#PAMMBK+1 JGET 2ND
702 002560 001406          BEQ     4S              JBR IF NONE
703 002562 104401 013220          TYPE   ,MSG0           J"AND"
704 002566 106237 001613          ASRB   #PAMMBK+1       JADJUST FOR TYPING
705 002572 004737 007564          JSR     PC,@TPDKV       JGO TYPE THE 2ND DRIVE
706 002576 104401 013225          4S:   TYPE   ,MSG10        J"WILL BE TESTED"<CRLF>
707 002602 000412          BR      START
708 002604 104202          5S:   ERROR  202         JNO DRIVES AVAILABLE
709 002606 012705 000002          MOV     #2,R5          JDRIVES & ADDRESS
    
```

710	002612	005704		TST	R4	OPERATOR INPUTS?
711	002614	001244		BNE	BEGINX	IF YES
712	002616	013700	000042	MOV	#42,R0	GET MONITOR RETURN ADDRESS
713	002622	001641		BEG	BEGINX	NO MONITOR
714	002624	000137	004726	JMP	##SENDAD	GO TO END

```

715 ////////////////////////////////////////////////////////////////////
716 ////////////////////////////////////////////////////////////////////
717 //*****
718 //*****
719 //
720 //      1. CLEAR THE VARIABLE STORAGE AREA
721 //      2. SETUP FOR THE DRIVE THAT IS TO BE TESTED
722 //      3. SETUP "LASTFL" AND "LASTBK" INCASE OPERATOR SELECTS "READONLY" PASS
723 //      4. SETUP THE MAX. NUMBER OF RETRYS FOR READS AND WRITES
724 //      5. SETUP TO START WITH A "FORMAT" PASS
725 //      6. CLEAR THE PASS COUNTER
726 //
727 //*****
728 START: MOV #SITEMB,R0          ;CLEAR VARIABLE STORAGE AREA
729         MOV #SBODAT-SITEMB/2,R1
730         JSR PC,##CLEAR          ;CLEAR AREA
731         MOV #SOFTNM,R0
732         MOV #LASTFL-SOFTNM/2,R1
733         JSR PC,##CLEAR
734         MOV #DRVPNT,R1          ;GET DRIVE POINTER
735         MOV# (R1)+,R0          ;SETUP THE DRIVE
736         ASR R0
737         BIC #CBIT00,R0
738         MOV# R0,##PARMBK+1
739         TST# (R1)
740         BNE ZS
741         MOV #DRVKEY,R1          ;RESET DRIVE TABLE POINTER
742         MOV R1,##DKVPNT        ;RESTORE THE DRIVE POINTER
743         MOV #9,##LASTFL        ;SET "LAST FILE" & "LAST BLOCK" INCASE
744         MOV #15,##LASTBK       ;OPERATOR SELECTS "READONLY" PASS
745         MOV ##MAXRUS,##RDTRYS  ;SETUP MAX. # OF REREAD TRYS
746         MOV ##MAXWR,##WRTRYS  ;SETUP MAX. # OF REWRITE TRYS
747         MOV #FORPAS,##PSCNTL   ;DO A FORMAT PASS
748         CLR ##PSCNTL+2
749         CLR ##PASCNT
750         BR LOOPER              ;CLEAR THE PASS COUNT
751         CLR (R0)+              ;GET AROUND CLEAR
752         DEC R1
753         BGT CLEAR
754         RTS PC
    
```

```

755 //*****
756 //*****
757 //
758 //I. CHECK SWR<7>
759 // A. SWR<7>=1 PERFORM AS PER SWR<110>
760 // 1. 00=FORMAT PASS
761 // 2. 01=READ ONLY PASS
762 // 3. 10=WRITE ONLY PASS
763 // 4. 11=READ ONLY PASS
764 //
765 // B. SWR<7>=0
766 // 1. UPDATE PASCNT
767 // 2. PERFORM AS PER (PSCNTL+2)
768 // A. FURMAT
769 // B. READONLY
770 // C. WRITEOONLY
771 // D. READONLY
772 //
773 //*****
774 //LOOPER: MOVB #SWR,#STMP0 //IF SWR<7>=1 ; DO PER SWR<110>
775 // BICB #C<BIT07;BIT01;BIT00>,#STMP0
776 // BMI J5 //BR IF SWR<7>=1
777 // CMP #PASCNT,#PSCNTL //IS THE COUNTER AT MAX,?
778 // BLT Z5 //BR IF NO
779 // CLR #PASCNT //RESET THE COUNTER
780 // ADD #2,#PSCNTL //MOVE TO THE NEXT PASS TYPE
781 // INC #PSCNTL+2
782 // CMP #PSCNTL,#NDQPAS+2 //TIME TO RESET PASS TYPE?
783 // BNE 15 //BR IF NO
784 // MOV #FORPAS,#PSCNTL //YES---RESET THE PASS CONTROL WORDS
785 // CLR #PSCNTL+2
786 // BR 15 //GO CHECK THE COUNT
787 // INC #PASCNT //COUNT THIS PASS
788 // MOV #PSCNTL+2,#STMP0 //PICKUP THE TYPE OF PASS TO DO NEXT
789 //
790 // CLR ~(SP)
791 // MOVB #STMP0,(SP) //PICKUP THE DISPATCH INDEX
792 // ASLB (SP) //POSITION THE INDEX
793 // ASLB (SP) // BEFORE USING IT
794 // ADD (SP)+,PC //GO TO THE ROUTINE
795 // JMP #FURMAT
796 // JMP #READONLY
797 // JMP #WRITEONLY
798 // JMP #READONLY
799 //
800 ////////////////////////////////////////////////////
801 //*****
802 //*****
    
```

```

803 ////////////////////////////////////////////////////
804 //*****
805 //*****
806 //*****
807 //*****
808 //*****
809 //*****
810 //*****
811 //*****
812 //*****
813 //*****
814 //*****
815 //*****
816 //*****
817 //*****
818 //*****
819 //*****
820 //*****
821 //*****
822 //*****
823 //*****
824 //*****
825 //*****
826 //*****
827 //*****
828 //*****
829 //*****
830 //*****
831 //*****
832 //*****
833 //*****
834 //*****
835 //*****
836 //*****
837 //*****
838 //*****
839 //*****
840 //*****
841 //*****
842 //*****
843 //*****
844 //*****
845 //*****
846 //*****
847 //*****
848 //*****
849 //*****
850 //*****
851 //*****
852 //*****
853 //*****
854 //*****
855 //*****
856 //*****
857 //*****
858 //*****
    
```

.SHTTL "FORMAT" ROUTINE

```

//*****
//THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE
//
// 1. REWIND TO BUT
// 2. WRITE FILE GAP
// 3. WRITE BLOCK OF DATA
// 4. BACK BLOCK GAP
// 5. READ BLOCK OF DATA
// 6. CHECK FOR SYNC ERROR
// 7. CHECK FOR DATA ERROR
//
// A. DATA ERROR OCCURRED
// 1.) (RDRTRY)<(MAXRDS) GOTO 4.
// 2.) (RDRTRY)=(MAXRDS)
// A.) (WRTRY)<(MAXERR)
// 1.) BACK BLOCK GAP
// 2.) GOTO 3.
// B.) (WRTRY)=(MAXERR) GOTO SEOP
//
// B. NO DATA ERROR --- GOTO 8.
//
// 8. UPDATE THE BLOCK NUMBER
// 9. END OF FILE?
// A. NO --- GOTO 3.
// B. YES
// 1.) UPDATE FILE NUMBER
// 2.) RESET BLOCK NUMBER TO 0
// 3.) GOTO 2.
//*****
//*****
840 //*****
841 //*****
842 //*****
843 //*****
844 //*****
845 //*****
846 //*****
847 //*****
848 //*****
849 //*****
850 //*****
851 //*****
852 //*****
853 //*****
854 //*****
855 //*****
856 //*****
857 //*****
858 //*****
    
```

```

859 003214 000137 004412          JMP     ##SEUP          IGO TO END OF PROGRAM
860                                     I//////////
861                                     I "WRITE=FILE-GAP"
862                                     I//////////
863 003220          2S:
864 003220 012700 001612          MOV     #PARMBK,R0          IRO=1ST ADDRESS OF PARAMETER BLOCK
865 003224 004737 006722          JSR     PC,#WFG          IGO TO WFG
866 003230 105737 001612          3S:  TSTB  #PAMMBK          IWAIT ON FLAG
867 003234 001775          BEQ     3S
868 003236 100006          BPL     4S          IBR IF NO ERROR
869 003240 004037 005536          JSR     R0,#CSRERR          IGO TO CSR ERROR CHECK
870 003244 000137 004412          JMP     ##SEUP          IGO TO END OF PROGRAM
871 003250 000137 004364          JMP     #CKLEOTS
872                                     I//////////
873                                     I "WRITE" A BLOCK OF DATA
874                                     I//////////
875 003254 012737 003262 001106 4S:  MOV     #S,#SLPADR          ISETUP SCOPE LOOP ADDRESS
876 003262 004737 006072          5S:  JSR     PC,#SETUPH          IGO SETUP FOR A WRITE
877 003266 012700 001612          MOV     #PARMBK,R0          IRO=1ST ADDRESS OF PARAMETER BLOCK
878 003272 004737 006726          JSR     PC,#WRITE          IGO TO WRITE
879 003276 105737 001612          6S:  TSTB  #PAMMBK          IWAIT ON FLAG
880 003302 001775          BEQ     6S
881 003304 100006          BPL     7S          IBR IF NO ERROR
882 003306 004037 005536          JSR     R0,#CSRERR          IGO TO CSR ERROR CHECK
883 003312 000137 004412          JMP     ##SEUP          IGO TO END OF PROGRAM
884 003316 000137 004364          JMP     #CKLEOTS
885                                     I//////////
886                                     I "BACK-SPACE=BLUCK-GAP"
887                                     I//////////
888 003322          7S:
889 003322 012700 001612          MOV     #PARMBK,R0          IRO=1ST ADDRESS OF PARAMETER BLOCK
890 003326 004737 006742          JSR     PC,#BSBG          IGO TO BSBG
891 003332 105737 001612          8S:  TSTB  #PAMMBK          IWAIT ON FLAG
892 003336 001775          BEQ     8S
893 003340 100004          BPL     9S          IBR IF NO ERROR
894 003342 004037 005536          JSR     R0,#CSRERR          IGO TO CSR ERROR CHECK
895 003346 000137 004412          JMP     ##SEUP          IGO TO END OF PROGRAM
896                                     I//////////
897                                     I "READ" A BLOCK OF DATA
898                                     I//////////
899 003352 004737 006220          9S:  JSR     PC,#SETUPR          IGO SETUP FOR A READ
900 003356 012700 001612          MOV     #PARMBK,R0          IRO=1ST ADDRESS OF PARAMETER BLOCK
901 003362 004737 006732          JSR     PC,#RLAD          IGO TO READ
902 003366 105737 001612          10S: TSTB  #PAMMBK          IWAIT ON FLAG
903 003372 001775          BEQ     10S
904 003374 100004          BPL     11S          IBR IF NO ERROR
905 003376 004037 005536          JSR     R0,#CSRERR          IGO TO CSR ERROR CHECK
906 003402 000137 004412          JMP     ##SEUP
907 003406 000004          11S: SCOPE
908                                     I//////////
909                                     I CHECK FOR SYNC ERROR
910                                     I//////////
911 003410 004737 006254          JSR     PC,#SYNCK          ISYNC ERROR?
912 003414 000403          BR      12S          IRETURN HERE IF NO
913 003416 104002          ERROR  2          ISYNC ERROR
914 003420 000137 004412          JMP     ##SEUP
    
```

```

915                                     I//////////
916                                     I CHECK FOR DATA ERROR IF SOFT REREAD
917                                     I//////////
918 003424 004737 006356          12S: JSR     PC,#DATCMP          ICHECK THE DATA
919 003430 000430          BR      15S          IRETURN HERE IF DATA IS GOOD
920 003432 104001          ERROR  1          IDATA ERROR
921 003434 004037 006550          JSR     R0,#CNTSFT          ICOUNT SOFT ERROR
922 003440 000730          BR      7S          IGO REREAD
923 003442 004037 006606          JSR     R0,#CNTHRD          ICOUNT HARD ERROR
924 003446 000402          BR      13S          IREWRITE
925 003450 000137 004412          JMP     ##SEUP          ITO MANY HARD ERRORS
926                                     I//////////
927                                     I IF HARD ERROR REWRITE
928                                     I//////////
929 003454 005337 001226          13S: DEC     #WRTRYS          ITRY TO REWRITE THIS BLOCK?
930 003460 002414          BLT     15S          IBR IF NO
931 003462 012700 001612          MOV     #PARMBK,R0          IRO=1ST ADDRESS OF PARAMETER BLOCK
932 003466 004737 006742          JSR     PC,#BSBG          IGO TO BSBG
933 003472 105737 001612          14S: TSTB  #PAMMBK          IWAIT ON FLAG
934 003476 001775          BEQ     14S
935 003500 100265          BPL     4S          IGO START A WRITE IF NO ERROR
936 003502 004037 005536          JSR     R0,#CSRERR          IGO TO CSR ERROR CHECK
937 003506 000137 004412          JMP     ##SEUP
938 003512 013737 001260 001224 15S: MOV     #MAXRUS,#WRTRYS          IRESET THE REREAD COUNT
939 003520 013737 001260 001226 15S: MOV     #MAXENR,#WRTRYS          IRESET THE REWRITE COUNT
940                                     I//////////
941                                     I UPDATE BLOCK # & FILE #
942                                     I//////////
943 003526 062737          ADD     (PC)+,(PC)+          IINCREMENT THE BLOCK NUMBER
944 003530 001 377          .BYTE  1,-1
945 003532 001306          .WORD  BLOCK
946 003534 123737 001310 001306  CMPB   #FILESZ,#BLOCK          ITIME FOR A FILE GAP?
947 003542 003244          BGT     4S          IBR IF NO=GO START A WRITE
948 003544 062737          ADD     (PC)+,(PC)+          IINCREMENT THE FILE NUMBER
949 003546 001 377          .BYTE  1,-1
950 003550 001304          .WORD  FILE
951 003552 012737          MOV     (PC)+,(PC)+          IINITIALIZE THE BLOCK NUMBER
952 003554 000 377          .BYTE  0,377          ISET BYTE 0 TO 0 AND BYTE 1
953 003556 001306          .WORD  BLOCK          ITO THE 1'S COMP. OF BYTE 0
954 003560 000617          BR      2S          IGO WRITE FILE GAP
    
```



```
955 ///////////////////////////////////////////////////  
956 ///////////////////////////////////////////////////  
957 ///////////////////////////////////////////////////  
958 ///////////////////////////////////////////////////  
959 ///////////////////////////////////////////////////  
960 .SBTTL "READ ONLY" ROUTINE  
961 //*****  
962 //*****  
963 //*****  
964 //*****  
965 JTHIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE  
966 /  
967 /  
968 / 1. REWIND TO BUT  
969 / 2. READ BLOCK OF DATA  
970 / 3. CHECK FOR SYNC ERROR  
971 / 4. CHECK FOR DATA ERROR  
972 / A. DATA ERROR OCCURRED  
973 / 1.) (RDTRYS)<(MAXRDS)  
974 / A.) BACK BLOCK GAP  
975 / B.) GOTO 2.  
976 / 2.) (RDTRYS)=(MAXRDS)  
977 / A.) (WRTRYS)<(MAXERR) GOTO 5.  
978 / B.) (WRTRYS)=(MAXERR) GOTO 3EOP  
979 / B. NO DATA ERROR --- GOTO 5;  
980 / 5. UPDATE THE BLOCK NUMBER  
981 / 6. END OF FILE?  
982 / A. NO --- GOTO 2,  
983 / B. YES  
984 / 1.) UPDATE FILE NUMBER  
985 / 2.) RESET BLOCK NUMBER TO 0  
986 / 3.) SPACE FORWARD FILE GAP  
987 / 4.) GOTO 2.  
988 //*****  
989 READONLY!  
990 003562 012737 003562 001106 MOV #REAUONLY,##SLPADR ISETUP SCOPE LOOP ADDRESS  
991 003570 104401 013265 TYPE #MSG12 I"*** HEAD ***"  
992 003574 004737 007564 JSR PC,##TPDRV IGO TYPE DRIVE TO BE TESTED  
993 003600 104401 001203 TYPE ,SCLRF  
994 003604 012737 MOV (PC)+,(PC)+ IINITIALIZE THE BLOCK NUMBER  
995 003606 000 377 .BYTE 0,377 ISET BYTE 0 TO 0 AND BYTE 1  
996 003610 001306 .WORD BLOCK ITO THE 1'S COMP. OF BYTE 0  
997 003612 012737 MOV (PC)+,(PC)+ IINITIALIZE THE FILE NUMBER  
998 003614 000 377 .BYTE 0,377 ISET BYTE 0 TO 0 AND BYTE 1  
999 003616 001304 .WORD FILE ITO THE 1'S COMP. OF BYTE 0  
1000 ///////////////////////////////////////////////////  
1001 / "REWIND" TO "BUT"  
1002 ///////////////////////////////////////////////////  
1003 003620 012700 001612 MOV #PARMBK,R0 IR0=1ST ADDRESS OF PARAMETER BLOCK  
1004 003624 004737 006756 JSR PC,##REWIND IGO TO REWIND  
1005 003630 105737 001612 1S: TSTB ##PARMBK IWAIT ON FLAG  
1006 003634 001775 BEQ 1S  
1007 003636 100004 BPL 2S IBR IF NO ERROR  
1008 003640 004037 005536 JSR R0,##CSRERR IGO TO CSR ERROR CHECK  
1009 003644 000137 004412 JMP ##SEUP IGO TO END-OF-PROGRAM  
1010 ///////////////////////////////////////////////////
```

```
1011 / SETUP FOR "REAU"  
1012 ///////////////////////////////////////////////////  
1013 003650 004737 006220 2S: JSR PC,##SETUPR ISETUP FOR "READ"  
1014 003654 000417 BR 5S IGO START A READ  
1015 ///////////////////////////////////////////////////  
1016 / "BSBG"---IF SCOPE OR SOFT DATA ERROR  
1017 ///////////////////////////////////////////////////  
1018 003656 012737 003714 3S: MOV #5S,##SLPADR ISETUP THE SCOPE LOOP ADDRESS  
1019 003664 012700 001612 MOV #PARMBK,R0 IR0=1ST ADDRESS OF PARAMETER BLOCK  
1020 003670 004737 006742 JSR PC,##BSBG IGO TO BSBG  
1021 003674 105737 001612 4S: TSTB ##PARMBK IWAIT ON FLAG  
1022 003700 001775 BEQ 4S  
1023 003702 100004 BPL 5S IGO TRY TO REREAD  
1024 003704 004037 005536 JSR R0,##CSRERR IGO CHECK CSR ERROR  
1025 003710 000137 004412 JMP ##SEUP IGO TO END-OF-PROGRAM  
1026 ///////////////////////////////////////////////////  
1027 / "READ" BLOCK OF DATA  
1028 ///////////////////////////////////////////////////  
1029 003714 012737 003656 5S: MOV #3S,##SLPADR ISET THE LOOP ADDRESS  
1030 003722 012700 001612 MOV #PARMBK,R0 IR0=1ST ADDRESS OF PARAMETER BLOCK  
1031 003726 004737 006732 JSR PC,##READ IGO TO READ  
1032 003732 105737 001612 6S: TSTB ##PARMBK IWAIT ON FLAG  
1033 003736 001775 BEQ 6S  
1034 003740 100004 BPL 7S IBR IF NO ERROR  
1035 003742 004037 005536 JSR R0,##CSRERR IGO TO CSR ERROR CHECK  
1036 003746 000137 004412 JMP ##SEUP IGO TO END-OF-PROGRAM  
1037 ///////////////////////////////////////////////////  
1038 / CHECK FOR SYNC ERROR  
1039 ///////////////////////////////////////////////////  
1040 003752 004737 006254 7S: JSR PC,##SYNCK IGO CHECK FOR SYNC ERROR  
1041 003756 000403 BR 8S IRETURN HERE IF NO ERROR  
1042 003760 104002 ERROR 2 ISYNC ERROR  
1043 003762 000137 004412 JMP ##SEUP IGO TO END-OF-PROGRAM  
1044 ///////////////////////////////////////////////////  
1045 / CHECK FOR DATA ERROR  
1046 ///////////////////////////////////////////////////  
1047 003766 004737 006356 8S: JSR PC,##DATCMP IGO CHECK THE DATA  
1048 003772 000411 BR 9S IRETURN HERE IF NO ERROR  
1049 003774 104001 ERROR 1 IDATA ERROR  
1050 003776 004037 006550 JSR R0,##CNTSFT ICOUNT SOFT ERROR  
1051 004002 000725 BR 3S ITRY TO REREAD  
1052 004004 004037 006606 JSR R0,##CNTHDD ICOUNT HARD ERROR  
1053 004010 000402 BR 9S IMOVE TO NEXT BLOCK  
1054 004012 000137 004412 JMP ##SEUP ITO MANY HARD ERRORS  
1055 004016 000004 9S: SCOPE  
1056 ///////////////////////////////////////////////////  
1057 / IF LAST FILE OF LAST BLOCK GOTO CKEOTS  
1058 ///////////////////////////////////////////////////  
1059 004020 013737 001260 001224 MOV #MAXRUS,##RDTRYS IRESET THE REREAD TRYS  
1060 004026 123737 001236 001304 CMPB ##LASTPL,##FILE ILAST FILE  
1061 004034 003007 BGT 10S IBR IF NO  
1062 004036 123737 001234 001306 CMPB ##LASTBK,##BLOCK ILAST BLOCK  
1063 004044 003003 BGT 10S IBR IF NO  
1064 004046 005237 001212 INC ##EOTS ICOUNT END-OF-TAPE  
1065 004052 000544 BR CKEOTS ICHECK FOR END-OF-PASS  
1066 ///////////////////////////////////////////////////
```

```

1067 / UPDATE BLOCK # & FILE #
1068 /
1069 ///////////////////////////////////////////////////
1070 105: ADD (PC)+,@(PC)+ ;INCREMENT THE BLOCK NUMBER
      ,BYTE 1,-1
1071 004054 062737 001 377 ,WORD BLOCK
1072 004060 001306 CMPB @#FILESZ,@#BLOCK ;TIME FOR A FILE GAP?
1073 004062 125737 001310 001306 BGT 25 ;BR IF NO
1074 004070 062737 ADD (PC)+,@(PC)+ ;INCREMENT THE FILE NUMBER
1075 004072 062737 ,BYTE 1,-1
1076 004074 001 377 ,WORD FILE
1077 004076 001304 MOV (PC)+,@(PC)+ ;INITIALIZE THE BLOCK NUMBER
1078 004100 012737 ,BYTE 0,377 ;SET BYTE 0 TO 0 AND BYTE 1
1079 004102 000 377 ,WORD BLOCK ;TO THE 1'S COMP, OF BYTE 0
1080 004104 001306
1081 /
1082 / "SPACE=FORWARD=FILE=GAP"
1083 /
1084 115: MOV #PARMBK,R0 ;R0=1ST ADDRESS OF PARAMETER BLOCK
1085 004106 012700 001612 JSR PC,@#SFFG ;GO TO SFFG
1086 004112 004737 006746 TSTB @#PARMBK ;WAIT ON FLAG
1087 004116 105737 001612 BEQ 15
1088 004122 001775 BPL 25 ;GO READ
1089 004124 100251 JSR R0,@#CSRERR ;GO CHECK CSR ERROR
1090 004126 004037 005536 JMP @#SEUP ;GO TO END-OF-PROGRAM
1091 004132 000137 004412
1091

```

```

1092 /
1093 /
1094 /
1095 /
1096 /
1097 ///////////////////////////////////////////////////
1098 .SSTTL "WRITE ONLY" ROUTINE
1099 /
1100 /
1101 /
1102 /
1103 /
1104 /
1105 /
1106 /
1107 /
1108 /
1109 /
1110 /
1111 /
1112 /
1113 /
1114 /
1115 /
1116 /
1117 WRITENO:
1118 004136 MOV #WRITEONLY,@#SLPADR ;SETUP SCOPE LOOP ADDRESS
1119 004144 104401 013304 TYPE ,MSG15 ;"*** WRITE ***"
1120 004150 004737 007564 JSR PC,@#TPDRV ;TYPE DRIVE TO BE TESTED
1121 004154 104401 001203 TYPE ,SCRLF
1122 004160 012737 MOV (PC)+,@(PC)+ ;INITIALIZE THE BLOCK NUMBER
1123 004162 000 377 ,BYTE 0,377 ;SET BYTE 0 TO 0 AND BYTE 1
1124 004164 001306 ,WORD BLOCK ;TO THE 1'S COMP, OF BYTE 0
1125 004166 012737 MOV (PC)+,@(PC)+ ;INITIALIZE THE FILE NUMBER
1126 004170 000 377 ,BYTE 0,377 ;SET BYTE 0 TO 0 AND BYTE 1
1127 004172 001304 ,WORD FILE ;TO THE 1'S COMP, OF BYTE 0
1128 /
1129 /
1130 /
1131 /
1132 /
1133 /
1134 /
1135 /
1136 /
1137 /
1138 /
1139 /
1140 /
1141 /
1142 /
1143 /
1144 /
1145 /
1146 /
1147 /
1147

```

```
1148 004250 000137 004412 JMP #SEUP IGO TO END-OF-PROGRAM
1149 004254 000137 004364 JMP #CKEOTS
1150
1151 ////////////////////////////////////////////////////
1152 // "WRITE" A BLOCK OF DATA
1153 ////////////////////////////////////////////////////
1154 004260 004737 006072 48: JSR PC,##SETUPH IGO SETUP FOR "WRITE"
1155 004272 004272 004272 001106 MOV #55,##SLPADR ISETUP SCOPE LOOP ADDRESS
1156 004272 012700 001612 55: MOV #PARAMK,R0 IRO=1ST ADDRESS OF PARAMETER BLOCK
1157 004276 004737 006726 JSR PC,##WRITE IGO TO WRITE
1158 004302 105737 001612 65: TSTB ##PARAMBK IWAIT ON FLAG
1159 004306 001775 BEQ 65
1160 004310 100006 BPL 75 IBR IF NO ERROR
1161 004312 004037 005536 JSR R0,##CSREMK IGO TO CSK ERROR CHECK
1162 004316 000137 004412 JMP #SEUP IGO TO END-OF-PROGRAM
1163 004322 000137 004364 JMP #CKEOTS
1164 004326 000004 75: SCOPE
1165 ////////////////////////////////////////////////////
1166 // UPDATE BLOCK # & FILE #
1167 ////////////////////////////////////////////////////
1168 004330 062737 ADD (PC)+,(PC)+ IINCREMENT THE BLOCK NUMBER
1169 004332 001 377 .BYTE 1,-1
1170 004334 001306 .WORD BLOCK
1171 004336 123737 001310 001306 CMPB #FILESZ,##BLOCK ITIME FOR A FILE GAP?
1172 004344 003345 BGT 45 IBR IF NO
1173 004346 062737 ADD (PC)+,(PC)+ IINCREMENT THE FILE NUMBER
1174 004350 001 377 .BYTE 1,-1
1175 004352 001304 .WORD FILE
1176 004354 012737 MOV (PC)+,(PC)+ IINITIALIZE THE BLOCK NUMBER
1177 004356 000 377 .BYTE 0,377 ISET BYTE 0 TO 0 AND BYTE 1
1178 004360 001306 .WORD BLOCK ITO THE 1'S COMP. OF BYTE 0
1179 004362 000720 BR 25 ISTART A FILE GAP
```

```
1180 ////////////////////////////////////////////////////
1181 ////////////////////////////////////////////////////
1182 ////////////////////////////////////////////////////
1183 .SBTTL CHECK "EOTS"
1184
1185 //*****
1186 //*****
1187 //*****
1188 IWHEN A "FORMAT" OR "WRITEONLY" PASS HITS "EOT" OR A
1189 I"READONLY" PASS REAUS THE LAST BLOCK OF THE LAST FILE
1190 ITHEY WILL COME HERE
1191 I
1192 I 1. (EOTS) IS CHECKED
1193 I A. (EOTS)<(MAXEOT) GOTO SEOP
1194 I B. (EOTS)<(MAXEOT) GOTO 2
1195 I 2. SWR<0> = 1?
1196 I A. YES -- GOTO SEOP
1197 I B. NU -- GOTO LOOPER
1198 I
1199 //*****
1200 004364 000004 CKEOTS: SCOPE
1201 004366 023737 001212 001264 CMP #EOTS,##MAXEOT IMAX, EOTS OCCURRED?
1202 004374 002006 BGE SEOP IBR IF YES
1203 004376 032777 000400 174534 BIT #SWR0,##SWR IIF SWR<00> = 1
1204 004404 001002 BNE SEOP IGO TO END-OF-TEST
1205 004406 000137 002774 JMP #LOOPER IEND-LOOP
1206
```

```

1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217 004412
1218 004412 000004
1219 004414 005067 174462
1220 004420 005267 174454
1221 004424 042767 100000 174446
1222 004432 005327
1223 004434 000001
1224 004436 003137
1225 004440 012737
1226 004442 000001
1227 004444 004434
1228 004446 104401 004745
1229 004452 104401 004742
1230 004456 013700 000042
1231 004462 001525
1232 004464 000004
1233
1234
1235
1236 004466 012700 001612
1237 004472 004737 006756
1238 004476 104401 013031
1239
1240 004502 104401 013056
1241 004506 013746 001206
1242 004512 004737 007366
1243 004516 004737 007422
1244
1245 004522 104401 013074
1246 004526 013746 001210
1247 004532 004737 007366
1248 004536 004737 007422
1249
1250 004542 104401 013112
1251 004546 012746 001214
1252 004552 004737 007172
1253 004556 004737 007422
1254
1255 004562 104401 013127
1256 004566 012746 001220
1257 004572 004737 007172
1258 004576 004737 007422
1259
1260 004602 104401 013147
1261 004606 013746 001212
1262 004612 004737 007366

      .SBTTL END OF PASS ROUTINE
      *****
      *INCREMENT THE PASS NUMBER (SPASS)
      *TYPE "END PASS"
      *IF THERES A MONITUM GO TO IT
      *IF THERE ISN'T JUMP TO START
      *IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
      *SENDMG CAN BE CHANGED TO 7.

      SEOP1
      SCOPE
      CLR STSTNM //ZERO THE TEST NUMBER
      INC SPASS //INCREMENT THE PASS NUMBER
      BIC #100000,SPASS //DON'T ALLOW A NEG. NUMBER
      DEC (PC)+ //LOOP?
      SEOPCT: ,WORD 1
      BGT SDOAGN //YES
      MOV (PC)+,0(PC)+ //RESTORE COUNTER
      SENDCT: ,WORD 1
      SEOPCT
      TYPE ,SENDMG //TYPE "END PASS"
      TYPE ,SENULL //TYPE A NULL CHARACTER
      SGET42: MOV #42,R0 //GET MONITOR ADDRESS
      BEQ SDOAGN //BRANCH IF NO MONITOR
      SCOPE
      ///////////////////////////////////////////////////////////////////
      // TYPE END-OF-TEST STATISTICS
      ///////////////////////////////////////////////////////////////////
      MOV #PARMBK,R0 //R0=PARAMETER BLOCK ADDRESS
      JSR PC,#REWIND //START A REWIND
      TYPE ,MSG1 //<CRLF>"*** END-OF-TEST ***"
      TYPE ,MSG2 //<CRLF>"SOFT ERRORS="
      MOV #SDPTNM,=(SP) //PICKUP SINGLE PRECISION BINARY NUMBER
      JSR PC,#SSB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE WITHOUT LEADING ZEROS
      TYPE ,MSG3 //<CRLF>"HARD ERRORS="
      MOV #HAKDNM,=(SP) //PICKUP SINGLE PRECISION BINARY NUMBER
      JSR PC,#SSB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE WITHOUT LEADING ZEROS
      TYPE ,MSG4 //<CRLF>"BYTES READ="
      MOV #RBYTTL,=(SP) //GET ADDRESS OF DOUBLE PRECISION BINARY #
      JSR PC,#SUB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE IT WITHOUT LEADING ZERO
      TYPE ,MSG5 //<CRLF>"BYTES WRITTEN="
      MOV #WBYTTL,=(SP) //GET ADDRESS OF DOUBLE PRECISION BINARY #
      JSR PC,#SUB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE IT WITHOUT LEADING ZERO
      TYPE ,MSG6 //<CRLF>"TAPE PASSES="
      MOV #EDOTS,=(SP) //PICKUP SINGLE PRECISION BINARY NUMBER
      JSR PC,#SSB2D //CHANGE IT TO DECIMAL ASCIZ
    
```

```

1263 004616 004737 007422
1264
1265 004622 104401 015165
1266 004626 013700 001236
1267 004632 105200
1268 004634 010046
1269 004636 004737 007366
1270 004642 004737 007422
1271
1272 004646 104401 015202
1273 004652 013746 001310
1274 004656 004737 007366
1275 004662 004737 007422
1276
1277 004666 104401 001204
1278
1279
1280
1281 004672 032777 001000 174240
1282 004700 001406
1283 004702 000000
1284
1285 004704 022767 000176 174226
1286 004712 001001
1287 004714 104405
1288 004716
1289 004716
1290 004716 013700 000042
1291 004722 001405
1292 004724 000005
1293 004726 004710
1294 004730 000240
1295 004732 000240
1296 004734 000240
1297 004736
1298 004736 000137
1299 004740 002630
1300 004742 377 000
1301 004745 015 042412 042116
1302 004752 050040 051501 000123

      JSR PC,#SSUPRS //TYPE WITHOUT LEADING ZEROS
      TYPE ,MSG7 //<CRLF>"FILES/PASS="
      MOV #LASTFL,R0 //PICKUP THE LAST FILE NUMBER
      INCB R0 //ADD 1 TO MAKE IT # OF FILES
      MOV R0,=(SP) //PUT IT ON THE STACK
      JSR PC,#SSB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE IT WITHOUT LEADING ZEROS
      TYPE ,MSG8 //<CRLF>"BLOCKS/FILE="
      MOV #FILESZ,=(SP) //PICKUP SINGLE PRECISION BINARY NUMBER
      JSR PC,#SSB2D //CHANGE IT TO DECIMAL ASCIZ
      JSR PC,#SSUPRS //TYPE WITHOUT LEADING ZEROS
      TYPE ,SLF
      ///////////////////////////////////////////////////////////////////
      // CHECK MALT AT END-OF-TEST SWITCH
      ///////////////////////////////////////////////////////////////////
      BIT #SW04,SWR //MALT AT END-OF-TEST?
      BEQ 1005 //BR IF NO
      HALT //YES
      CMP #SWREG,SWR //USING S/W SWITCH REG?
      BNE 203 //NO- GET OUT
      GTSWR //GET VALUE
      //CONTINUE
      203:
      1005:
      MOV #42,R0 //INSURE R0 CONTAINS THE MONITORS
      BEQ SDOAGN //RETURN ADDRESS
      RESET //CLEAR THE WORLD
      SENDAD: JSR PC,(R0) //GO TO MONITOR
      NOP //SAVE ROOM
      NOP //FOR
      NOP //ACT11
      SDOAGN: JMP 0(PC)+ //RETURN
      SRNAD: ,WORD START
      SENULL: ,BYTE =1,-1,0 //NULL CHARACTER STRING
      SENDMG: ,ASCIZ <15><12>/END PASS/
    
```

```

1303 .SBTTL SCOPE HANDLER ROUTINE
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314 004760          .SSCOPE:
1315 004760 104406          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
1316 004762 032777 040000 174150 1S: BIT          #0IT14,#SWR          ;;LOOP ON PRESENT TEST?
1317 004770 001025          BNE          #0VER          ;;YES IF SW14=1
1318
1319 004772 000416          *****START OF CODE FOR THE XOR TESTER*****
1320
1321 004774 013746 000004          MOV          #ERRVEC,(SP)          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1322 005000 012737 005020 000004          MOV          #SS,#ERRVEC          ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1323 005006 005737 177060          TST          #177060          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1324 005012 012637 000004          MOV          (SP),#ERRVEC          ;;SET FOR TIMEOUT
1325 005016 000404          BR          $SVLAD          ;;TIME OUT ON XOR?
1326 005020 022626          SS: CMP          (SP),#ERRVEC          ;;RESTORE THE ERROR VECTOR
1327 005022 012637 000004          MOV          (SP),#ERRVEC          ;;GO TO THE NEXT TEST
1328 005026 000406          BR          #0VER          ;;CLEAR THE STACK AFTER A TIME OUT
1329 005030          BR          $SVLAD          ;;RESTORE THE ERROR VECTOR
1330 005030 105267 174046          BSI;*****END OF CODE FOR THE XOR TESTER*****          ;;LOOP ON THE PRESENT TEST
1331 005034 011667 174046          $SVLAD: INCB          $STSTM          ;;COUNT TEST NUMBERS
1332 005040 105067 174037          MOV          (SP),$LPADR          ;;SAVE SCOPE LOOP ADDRESS
1333 005044 016777 174032 174070          CLR          CLRBB          ;;ZERO THE ERROR FLAG
1334 005052 016716 174030          $OVER: MOV          $STSTM,#DISPLAY          ;;DISPLAY TEST NUMBER
1335 005056 000002          MOV          $LPADR,(SP)          ;;FUDGE RETURN ADDRESS
          RTI          ;;FIXES PS
    
```

```

1336 .SBTTL ERROR HANDLER ROUTINE
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349 005060          .SENROR:
1350 005060 104406          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
1351 005062 105267 174015          7S: INCB          $ERFLG          ;;SET THE ERROR FLAG
1352 005066 001775          BEQ          7S          ;;DON'T LET THE FLAG GO TO ZERO
1353 005070 016777 174006 174044          MOV          $STSTM,#DISPLAY          ;;DISPLAY TEST NUMBER AND ERROR FLAG
1354 005076 032777 002000 174034          BIT          #0IT10,#SWR          ;;BELL ON ERROR?
1355 005104 001402          BEQ          1S          ;;NO - SKIP
1356 005106 104401 005204          TYPE          $BELL          ;;RING BELL
1357 005112 005267 173774          1S: INC          $ERTTL          ;;COUNT THE NUMBER OF ERRORS
1358 005116 011667 173774          MOV          (SP),$ERRPC          ;;GET ADDRESS OF ERROR INSTRUCTION
1359 005122 162767 000002 173766          SUB          #2,$ERRPC          ;;STRIP AND SAVE THE ERROR ITEM CODE
1360 005130 117767 173762 173756          MOV          #ERRHPC,$ITEMB          ;;SKIP TYPEOUT IF SET
1361 005136 032777 020000 173774          BIT          #0IT13,#SWR          ;;SKIP TYPEOUTS
1362 005144 001004          BNE          20S          ;;SKIP TYPEOUTS
1363 005146 004767 000036          JSR          PC,TYPEERR          ;;GO TO USER ERROR ROUTINE
1364 005152 104401 001203          TYPE          $CRFL          ;;
1365 005156          20S:
1366 005156 005777 173756          2S: TST          #SWR          ;;HALT ON ERROR
1367 005162 100002          BPL          3S          ;;SKIP IF CONTINUE
1368 005164 000000          HALT          ;;HALT ON ERROR!
1369 005166 104406          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
1370 005170          3S:
1371 005170 022737 004726 000042          CMP          #SENDAD,#42          ;;ACT=11 AUTO-ACCEP?
1372 005176 001001          BNE          6S          ;;BRANCH IF NO
1373 005200 000000          HALT          ;;YES
1374 005202          6S:
1375 005202 000002          RTI          ;;RETURN
1376 005204 177607 000377          $BELL: .ASCIZ <207><377><377>          ;;ASCII CODE FOR BELL
    
```

```
1377 ;*****  
1378 ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES  
1379 005210 104401 001203 TYPERR1 TYPE ,SCLRF ;TYPE "CR" & "LF"  
1380 005214 010046 MOV R0,=(SP) ;SAVE R0  
1381 005216 010146 MOV R1,=(SP) ;SAVE R1  
1382 005220 005046 CLR =(SP) ;PICKUP THE ITEM BYTE  
1383 005222 113716 MOV8 #$(ITEMB,(SP) ;  
1384 005226 005316 DEC (SP) ;ADJUST THE INDEX SO IT  
1385 005230 006316 ASL (SP) ;WILL WORK FOR THE  
1386 005232 006316 ASL (SP) ;ERROR TABLE  
1387 005234 006316 ASL (SP) ;  
1388 005236 116601 MOV8 1(SP),M1 ;GET THE ERROR TYPE  
1389 005242 112600 MOV8 (SP)+,R0 ;GET THE ERROR NUMBER  
1390 005244 066100 ADD 1$(R1),R0 ;FORM THE TABLE POINTER  
1391 005250 000403 BR 2$  
1392 005252 001622 1$: ITEMS0  
1393 005254 001642 ITEMS1  
1394 005256 001732 ITEMS2  
1395  
1396 005260 012067 000002 2$: MOV (R0)+,$$ ;PICKUP "ERROR MESSAGE" POINTER  
1397 005264 104401 TYPE ;TYPE "ERROR MESSAGE"  
1398 005266 000000 3$: 0  
1399 005270 104401 TYPE ,SCLRF ;TYPE "CR" & "LF"  
1400 005274 012067 000002 MOV (R0)+,$$ ;PICKUP "DATA HEADER" POINTER  
1401 005300 104401 TYPE ;TYPE "DATA HEADER"  
1402 005302 000000 4$: 0  
1403 005304 104401 TYPE ,SCLRF ;TYPE "CR" & "LF"  
1404 005310 060107 ADD R1,PC ;GO TYPE THE DATA  
1405 005312 000402 BR ERROR0  
1406 005314 000430 BR ERROR1  
1407 005316 000473 BR ERROR2  
1408  
1409  
1410 005320 010246 ERROR0: MOV R2,=(SP) ;SAVE R2  
1411 005322 012001 MOV (R0)+,M1 ;PICKUP THE "DATA POINTER"  
1412 005324 012002 MOV (R0)+,R2 ;PICKUP "FORMAT" POINTER  
1413 005326 000402 BR 2$  
1414 005330 104401 013323 1$: TYPE ,MSG14 ;" "  
1415 005334 012100 2$: MOV (R1)+,R0 ;GET ADDRESS OF DATA WORD  
1416 005336 001473 BEQ EREXT1 ;GO TO EXIT IF 0  
1417 005340 105722 TSTB (R2)+ ;TYPE DECIMAL OR OCTAL?  
1418 005342 001003 BNE 3$ ;BR IF DECIMAL  
1419 005344 011046 MOV (R0)+,=(SP) ;SAVE (R0) FOR TYPEOUT  
1420 005346 104402 TYPCC ;GO TYPE--OCTAL ASCII(ALL DIGITS)  
1421 005350 000767 BR 1$ ;GO GET NEXT DATA WORD  
1422 005352 010046 5$: MOV R0,=(SP)  
1423 005354 004737 JSR PC,#SUB20 ;CHANGE NUMBER TO ASCII  
1424 005360 062716 ADD #4,(SP) ;TYPE 6 DIGITS  
1425 005364 012667 000002 MOV (SP)+,$$ ;  
1426 005370 104401 TYPE ;CALL TYPE ASCII MESSAGE ROUTINE  
1427 005372 000000 4$: ,WORD 0 ;ADDRESS OF ASCII STRING  
1428 005374 000755 BR 1$ ;GO GET NEXT DATA WORD
```

```
1429 ;*****  
1430 005376 011001 ERROR1: MOV (R0),R1 ;PICKUP THE "DATA POINTER"  
1431 005400 013146 MOV @(R1)+,=(SP) ;SAVE @(R1)+ FOR TYPEOUT  
1432 ;SERRPC  
1433 005402 104402 TYPCC ;GO TYPE--OCTAL ASCII(ALL DIGITS)  
1434 005404 012767 000003 000006 MOV #3,2$ ;PRINT 2 SPACES 3 TIMES  
1435 ;AND 2 DECIMAL #'S 2 TIMES  
1436 005412 104401 015323 1$: TYPE ,MSG14 ;" "  
1437 005416 005327 DEC (PC)+ ;TYPE ANOTHER # ?  
1438 005420 000000 2$: ,WORD 0  
1439 005422 001412 BEQ 4$ ;BR IF NO  
1440 005424 012146 MOV (R1)+,=(SP) ;YES==PICKUP ADDRESS OF #  
1441 005426 004737 JSR PC,#SUB20 ;GO CHANGE TO DECIMAL ASCII  
1442 005432 062716 ADD #4,(SP) ;ONLY TYPE THE LAST 6 DIGITS  
1443 005436 012667 000002 MOV (SP)+,$$ ;SAVE ADDRESS OF ASCII STRING  
1444 005442 104401 TYPE ;CALL THE TYPE ROUTINE  
1445 005444 000000 ,WORD 0 ;POINTER GOES HERE  
1446 005446 000761 BR 1$ ;LOOP  
1447 005450 013101 4$: MOV @(R1)+,R1 ;PICKUP FUNCTION INDEX  
1448 005452 016167 005466 000002 MOV BADFUN(R1),5$ ;GET THE FUNCTION MESSAGE  
1449 005460 104401 TYPE ;AND TYPE IT  
1450 005462 000000 5$: ,WORD 0 ;MESSAGE POINTER GOES HERE  
1451 005464 000421 BR EREXT2 ;GO TO EXIT  
1452 ;THIS TABLE CONTAINS THE POINTERS TO THE DIFFERENT ASCII MESSAGES  
1453 ;FOR THE FUNCTION BEING PERFORMED WHEN THE ERROR OCCURRED.  
1454 005466 012067 BADFUN: MXWFG ; WRITE-FILE-GAP  
1455 005470 012106 MXWRIT ; WRITE  
1456 005472 012114 MXREAD ; READ  
1457 005474 012121 MXRFBG ; BACK-SPACE=FILE-GAP  
1458 005476 012145 MXBSBG ; BACK-SPACE=BLK-GAP  
1459 005500 012170 MXSFBG ; SPACE-FWD=FILE-GAP  
1460 005502 012213 MXSFBD ; SPACE-FWD=BLK-GAP  
1461 005504 012235 MXRWND ; REWIND  
1462  
1463  
1464 005506 011000 ERROR2: MOV (R0),R0 ;PICKUP THE DATA POINTER  
1465 005510 005710 1$: TST (R0) ;ALL DATA TYPED?  
1466 005512 001406 BEQ EREXT2 ;BR IF YES  
1467 005514 013046 MOV @(R0)+,=(SP) ;SAVE @(R0)+ FOR TYPEOUT  
1468 005516 104402 TYPCC ;GO TYPE--OCTAL ASCII(ALL DIGITS)  
1469 005520 104401 013323 TYPE ,MSG14 ;" "  
1470 005524 000771 BR 1$ ;LOOP  
1471  
1472  
1473 005526 012602 EREXT1: MOV (SP)+,R2 ;RESTORE R2  
1474 005530 012601 EREXT2: MOV (SP)+,M1 ;RESTORE R1  
1475 005532 012600 MOV (SP)+,R0 ;RESTORE R0  
1476 005534 000207 RTS PC ;RETURN  
1477  
1478
```

```

1479
1480
1481
1482
1483 005536 113737 001304 001162 CSRERR: MOV# #FILE,##$REG0 ;GET THE FILE NUMBER
1484 005544 113737 001306 001166 MOV# #BLOCK,##$REG2 ;GET THE BLOCK NUMBER
1485 005552 013746 007170 MOV# #FUNLOC,-(SP) ;GET THE LAST FUNCTION
1486 005556 042716 177761 BIC #FUNCTION,(SP) ;CLEAR AWAY THE JUNK
1487 005562 011637 001172 MOV# (SP),##$TMP0 ;SAVE THE FUNCTION CODE
1488 005566 113766 001612 000001 MOV# #PAMMK,1(SP) ;COMBINE THE ST/ER WITH
1489 005574 012601 MOV# (SP)+,M1 ;THE FUNCTION
1490 005576 005046 CLR -(SP) ;FIND OUT WHAT CAUSED THE ERROR
1491 005600 032701 001000 BIT #OFFLINE,M1 ;OFF LINE?
1492 005604 001025 BNE B5 ;BR IF YES
1493 005606 032701 010000 BIT #WRTLOCK,M1 ;WRITE LOCK?
1494 005612 001021 BNE S5 ;BR IF YES
1495 005614 032701 020000 BIT #LEADEN,M1 ;BOT/EOT?
1496 005620 001015 BNE 45 ;BR IF YES
1497 005622 032701 004000 BIT #FGAP,M1 ;FILE GAP?
1498 005626 001011 BNE S5 ;BR IF YES
1499 005630 032701 002000 BIT #TIMERN,M1 ;TIMING?
1500 005634 001005 BNE Z5 ;BR IF YES
1501 005636 032701 040000 BIT #CRCLRN,M1 ;BLOCK CHECK?
1502 005642 001001 BNE I5 ;BR IF YES
1503 005644 005216 INC (SP) ; 6 = UNKNOWN
1504 005646 005216 15: INC (SP) ; 5 = CRCLRN
1505 005650 005216 25: INC (SP) ; 4 = TIMERN
1506 005652 005216 35: INC (SP) ; 3 = FGAP
1507 005654 005216 45: INC (SP) ; 2 = LEADER
1508 005656 005216 55: INC (SP) ; 1 = WRTLOCK
1509 005660 65:
1510 005660 106316 ASLB (SP) ;POSITION INDEX
1511 005662 062607 ADD (SP)+,PC ;BR TO THE ROUTINE
1512 005664 000406 BR UL,ERR
1513 005666 000407 BR WL,ERR
1514 005670 000410 BR CL,ERR
1515 005672 000462 BR FG,ERR
1516 005674 000463 BR TM,ERR
1517 005676 000464 BR BC,ERR
1518 005700 000472 BR X,ERR
1519
1520 005702 104101 OL,ERR: ERROR 101 ;DRIVE IS OFFLINE
1521 005704 000200 RTS R0
1522
1523 005706 104102 WL,ERR: ERROR 102 ;DRIVE IS WRITE LOCK
1524 005710 000200 RTS R0
1525
    
```

```

1526 005712 022737 000000 001172 CL,ERR: CMP #XFWG,##$TMP0 ;FUNCTION "WRITE FILE GAP"?
1527 005720 001014 BNE Z5 ;BR IF NO
1528 005722 15:
1529 005722 162737 SUB (PC)+,(PC)+ ;DECREMENT THE FILE NUMBER
1530 005724 001 377 ,BYTE 1,-1
1531 005726 001304 ,WORD FILE
1532 005730 113737 001304 001236 MOV# #FILE,##LASTFL ;SAVE LAST FILE NUMBER
1533 005736 113737 001310 001234 MOV# #FILESZ,##LASTBK ;AND LAST BLOCK NUMBER
1534 005744 005337 001234 DEC #LASTBK
1535 005750 000425 BR S5 ;BR IF NO
1536 005752 022737 000002 001172 25: CMP #XWRITE,##$TMP0 ;FUNCTION "WRITE"?
1537 005760 001025 BNE 45 ;BR IF NO
1538 005762 113737 001304 001236 MOV# #FILE,##LASTFL ;SAVE LAST FILE NUMBER
1539 005770 162737 SUB (PC)+,(PC)+ ;DECREMENT THE BLOCK NUMBER
1540 005772 001 377 ,BYTE 1,-1
1541 005774 001306 ,WORD BLOCK
1542 005776 113737 001306 001234 MOV# #BLOCK,##LASTBK ;SAVE LAST BLOCK NUMBER
1543 006004 163737 007164 001220 SUB #CBCNT,##BYTTL ;SUBTRACT RESIDUE COUNT FROM
1544 006012 005637 001222 SBC #BYTTL+2 ; THE TOTAL WRITE COUNT
1545 006016 005137 001306 COM #BLOCK ;IF NO BLOCKS IN THIS FILE
1546 006022 001737 BEQ I5 ;BACK TO LAST FILE
1547 006024 022020 35: CMP (R0)+,(R0)+ ;ADJUST FOR RETURN
1548 006026 005237 001212 INC #EOTS ;COUNT END-OF-TAPE
1549 006032 000401 BR S5
1550 006034 104103 45: ERROR 103 ;"CLEAR LEADER" ERROR
1551 006036 000200 55: RTS R0 ;RETURN
1552
1553 006040 104104 FG,ERR: ERROR 104 ;"FILE GAP ERROR"
1554 006042 000200 RTS R0
1555
1556 006044 104105 TM,ERR: ERROR 105 ;TIMING ERROR
1557 006046 000200 RTS R0
1558
1559 006050 104106 BC,ERR: ERROR 106 ;BLOCK CHECK ERROR
1560 006052 022737 000004 001172 CMP #XREAD,##$TMP0 ;IF FUNCTION ISN'T A READ
1561 006060 001001 BNE I5 ;TAKE THE "EOP" EXIT
1562 006062 022020 15: CMP (R0)+,(R0)+ ;ADJUST FOR RETURN
1563 006064 000200 RTS R0
1564
1565 006066 104107 X,ERR: ERROR 107 ;UNKNOWN INTERRUPT
1566 006070 000200 RTS R0
    
```

```

1567 ;*****
1568 ;*****
1569 ;*****
1570 ;*****
1571 ;*****
1572 ;*****
1573 ;*****
1574 ;*****
1575 ;*****
1576 ;*****
1577 ;*****
1578 ;*****
1579 ;*****
1580 006072 013700 001306          SETUP# MOV    ##BLUCK,R0          /GET THE BLOCK NUMBER
1581 006076 042700 177760          BIC    #'C15,,R0          /KEEP MAX PATTERN
1582 006102 006300                   ASL    R0                    /X2
1583 006104 006037 001312 001220  ADD    BLKSZ(M0),##WBYTTL    /ADD THE BLOCK SIZE TO THE TOTAL
1584 006112 005537 001222                   ADC    ##WBYTTL+2          /NUMBER OF BYTES WRITTEN
1585 006116 016037 001312 001620  MOV    BLKSZ(M0),##PARMBK+6 /SET THE BYTE COUNT
1586 006124 016000 001352                   MOV    PATS(M0),R0         /GET PATTERN POINTER
1587 006130 004737 006130                   JSR    PC,##FILL           /GO FILL WRITE BUFFER
1588 006134 000207                   RTS    PC                    /RETURN
1589
1590 ;*****
1591 ;*****
1592 ;*****
1593 ;*****
1594 ;*****
1595 ;*****
1596 ;*****
1597 ;*****
1598 ;*****
1599 ;*****
1600 ;*****
1601 ;*****
1602 ;*****
1603 ;*****
1604 ;*****
1605 ;*****
1606 ;*****
1607 ;*****
1608 ;*****
1609 ;*****
1610 ;*****
1611 006136
1612 006136 010046          FILL#  MOV    R0,=(SP)          ;;PUSH R0 ON STACK
1613 006140 010146          MOV    R1,=(SP)          ;;PUSH R1 ON STACK
1614 006142 010246          MOV    R2,=(SP)          ;;PUSH R2 ON STACK
1615 006144 010346          MOV    R3,=(SP)          ;;PUSH R3 ON STACK
1616 006146 012701 000200  MOV    #1024,/8,,R1       /FILL 1024. BYTES WITH PATTERN
1617 006152 012702 013464  MOV    #BUFFER,R2        /1ST ADDRESS OF DATA BUFFER
1618 006156 013722 001304  MOV    ##FILE,(R2)+      /FILE NUMBER AND ITS COMPLEMENT
1619 006162 013722 001306  MOV    ##BLUCK,(R2)+    /BLOCK NUMBER AND ITS COMPLEMENT
1620 006166 010003          MOV    R0,R3            /SAVE FIRST ADDRESS OF PATTERN
1621 006170 010300 151  MOV    R3,R0            /RESET PATTERN POINTER
1622 006172 012022          MOV    (R0)+,(R2)+      /MOVE THE PATTERN
    
```

```

1623 006174 012022          MOV    (R0)+,(R2)+      /INTO THE BUFFER AREA.
1624 006176 012022          MOV    (R0)+,(R2)+
1625 006200 012022          MOV    (R0)+,(R2)+
1626 006202 005301          DEC    R1
1627 006204 001371          BNE    158
1628 006206 012603          MOV    (SP)+,R3         ;;POP STACK INTO R3
1629 006210 012602          MOV    (SP)+,R2         ;;POP STACK INTO R2
1630 006212 012601          MOV    (SP)+,R1         ;;POP STACK INTO R1
1631 006214 012600          MOV    (SP)+,R0         ;;POP STACK INTO R0
1632 006216 000207          RTS    PC
1633
1634 ;*****
1635 ;*****
1636 ;*****
1637 ;*****
1638 ;*****
1639 ;*****
1640 ;*****
1641 ;*****
1642 ;*****
1643 ;*****
1644 ;*****
1645 006220 013700 001306          SETUP# MOV    ##BLUCK,R0          /GET THE BLOCK NUMBER
1646 006224 042700 177760          BIC    #'C15,,R0          /KEEP MAX PATTERN
1647 006230 006300                   ASL    R0                    /X2
1648 006232 006037 001312 001214  ADD    BLKSZ(M0),##WBYTTL    /ADD THE BLOCK SIZE TO THE TOTAL
1649 006240 005537 001216                   ADC    ##WBYTTL+2          /NUMBER OF BYTES READ
1650 006244 016037 001312 001620  MOV    BLKSZ(M0),##PARMBK+6 /SET THE BYTE COUNT
1651 006252 000207                   RTS    PC                    /RETURN
1652
1653 ;*****
1654 ;*****
1655 ;*****
1656 ;*****
1657 ;*****
1658 ;*****
1659 ;*****
1660 ;*****
1661 ;*****
1662 ;*****
1663 ;*****
1664 ;*****
1665 ;*****
1666 ;*****
1667 ;*****
1668 ;*****
1669 ;*****
1670 006254 012700 013464          SYNCK# MOV    #BUFFER,R0        /ADDRESS OF FIRST WORD
1671 006260 011001          MOV    (R0),R1          /CHECK FILE # WAS READ OK
1672 006262 000301          SWAB  R1                /BYTE 1 SHOULD BE THE
1673 006264 062001          ADD    (R0)+,R1         /COMPLEMENT OF BYTE 0
1674 006266 005101          COM   R1                /
1675 006270 001031          BNE   Z8                /BR IF FILE NUMBER READ WRONG
1676 006272 011001          MOV    (R0),R1          /CHECK BLOCK # WAS READ OK
1677 006274 000301          SWAB  R1                /BYTE 3 SHOULD BE THE
1678 006276 062001          ADD    (R0)+,R1         /COMPLEMENT OF BYTE 2
    
```



```

1679 006300 005101 COM R1
1680 006302 001024 BNE 25
1681 006304 023740 001306 CMP #BLUCK,=(R0) ;BFR IF BLOCK NUMBER READ WRONG
1682 006310 001003 BNE 15 ;MAKE SURE PROPER BLOCK WAS READ
1683 006312 023740 001304 CMP #FILE,=(R0) ;BFR IF AT THE WRONG BLOCK
1684 006316 001416 BEQ 25 ;MAKE SURE PROPER FILE
1685 006320 113737 001304 001162 15: MOVB #FILE,##SREG0 ;BFR IF IN THE PROPER FILE
1686 006326 113737 001306 001166 MOVB #BLUCK,##SREG2
1687 006334 113737 013464 001172 MOVB #BUFFER,##STMP0
1688 006342 113737 013466 001176 MOVB #BUFFER+2,##STMP2
1689 006350 062716 000002 ADD #2,(SP) ;TAKE ERROR RETURN
1690 006354 000207 25: RTS ;RETURN
1691
1692 ;*****
1693 .SBTTL ROUTINE TO CHECK THE READ DATA
1694
1695 ;CALL
1696 ; JSR PC,DATCMP
1697 ; NORMAL RETURN ;ERROR FLAG=0 AND DATA IS GOOD
1698 ; ERROR1 RETURN ;DATA IS BAD (ERROR FLAG=?)
1699 ; ERROR2 RETURN ;DATA IS GOOD BUT ERROR FLAG=1
1700
1701 006356 013700 001306 DATCMP: MOV #BLUCK,R0 ;GET THE BLOCK NUMBER
1702 006362 042700 177760 BIC #C15,R0 ;KEEP MAX PATTERN
1703 006366 006300 ASL R0 ;X2
1704 006370 016001 001312 MOV BLKSZ(R0),R1 ;PICKUP THE BLOCK SIZE
1705 006374 012702 013464 MOV #BUFFER,R2 ;FIRST ADDRESS OF DATA
1706 006400 012703 000004 MOV #4,R3 ;SETUP TO CHECK FILE AND BLOCK
1707 006404 012704 001304 MOV #FILE,R4
1708 006410 005301 15: DEC R1 ;COUNT THIS BYTE
1709 006412 122224 CMPB (R2)+(R4)+ ;CHECK IT
1710 006414 001015 BNE 45 ;BFR IF BAD
1711 006416 005303 DEC R3 ;AGAIN?
1712 006420 001373 BNE 15 ;BFR IF YES
1713 006422 012703 000010 25: MOV #0,R3 ;GET NUMBER OF BYTES/PATTERN
1714 006426 016004 001352 MOV PATS(R0),R4 ;GET FIRST ADDRESS OF PATTERN
1715 006432 005301 35: DEC R1 ;LAST BYTE BEEN CHECKED?
1716 006434 002437 BLT 55 ;BFR IF YES
1717 006436 122224 CMPB (R2)+(R4)+ ;NO-CHECK IT
1718 006440 001003 BNE 45 ;BFR IF BAD
1719 006442 005303 DEC R3 ;END OF PATTERN?
1720 006444 003372 BGT 35 ;BFR IF NO
1721 006446 000765 BR 25 ;YES
1722 006450 005037 001124 45: CLR #SGUDAT
1723 006454 005037 001126 CLR #SBUDAT
1724 006460 114437 001124 MOVB #SGUDAT ;SAVE GOOD DATA
1725 006464 010437 001120 MOV R4,#SGOADR ;SAVE GOOD ADDRESS
1726 006470 114237 001126 MOVB #SBUDAT ;SAVE BAD DATA
1727 006474 010237 001122 MOV R2,#SBOADR ;SAVE BAD ADDRESS
1728 006500 005401 NEG R1 ;SAVE THE BYTE NUMBER
1729 006502 066001 001312 ADD BLKSZ(R0),R1
1730 006506 010137 001230 MOV R1,#BYTNUM
1731 006512 113737 001304 001162 MOVB #FILE,##SREG0
1732 006520 113737 001306 001166 MOVB #BLUCK,##SREG2
1733 006526 062716 000002 ADD #2,(SP) ;SETUP TO TAKE ERROR1 EXIT
1734 006532 000405 BR 65 ;TAKE ERROR1 EXIT

```

```

1735 006534 105737 001612 55: TSTB #PAKMBK ;DATA IS GOOD BUT WHAT ABOUT
1736 ;THE ERROR FLAG
1737 006540 100002 BPL 65 ;BFR IF ERROR FLAG=0
1738 006542 062716 000004 ADD #4,(SP) ;SETUP TO TAKE ERROR2 EXIT
1739 006546 000207 65: RTS ;RETURN
1740
1741 ;*****
1742 .SBTTL COUNT SOFT DATA ERROR
1743
1744 ;CALL
1745 ; JSR R0,#CNTSFT
1746 ; RETURN FOR SOFT RETRY
1747 ; RETURN FOR HARD
1748 ;
1749
1750 006550 005237 001206 CNTSFT: INC #SOFTNM ;COUNT THIS SOFT ERROR
1751 006554 005337 001224 DEC #RDTRYS ;REREAD THE RECORD?
1752 006560 002011 BGE 15 ;BFR IF YES
1753 006562 013737 001260 001224 MOV #MAXRUS,#RDTRYS ;RESET REREAD COUNT
1754 006570 163737 001260 001206 SUB #MAXRUS,##SUFTNM ;DON'T COUNT AS SOFT
1755 006576 005337 DEC #SOFTNM
1756 006602 005720 TST (R0)+ ;STEP OVER THE REREAD RETURN
1757 006604 000200 15: RTS ;RETURN
1758
1759 ;*****
1760 .SBTTL COUNT HARD ERROR
1761
1762 ;CAL
1763 ; JSR R0,#CNTMRD
1764 ; RETURN MORE ERRORS ALLOWED
1765 ; RETURN MAX. ERRORS HAVE OCCURRED
1766 ;
1767 006606 005237 001210 CNTMRD: INC #HARDNM ;COUNT THIS HARD ERROR
1768 006612 023737 001210 001262 CMP #HARDNM,##MAXERR ;MAX. HARD ERRORS OCCURRED
1769 006620 002401 BLT 15 ;BFR IF NO
1770 006622 005720 TST (R0)+ ;TAKE FATAL EXIT
1771 006624 000200 15: RTS ;RETURN

```

```

1772                                     ;*****
1773                                     ;
1774                                     .SBTTL   SAVE AND RESTORE R0-R5 ROUTINES
1775                                     ;
1776                                     ;*****
1777                                     ;SAVE R0-R5
1778                                     ;*CALL:
1779                                     ;*   SAVREG
1780                                     ;*UPON RETURN FROM SSAVREG THE STACK WILL LOOK LIKE:
1781                                     ;*
1782                                     ;*TOP---(+16)
1783                                     ;* +2---(+18)
1784                                     ;* +4---R5
1785                                     ;* +6---R4
1786                                     ;* +8---R3
1787                                     ;**10---R2
1788                                     ;**12---R1
1789                                     ;**14---R0
1790
1791 006626                                     SSAVREG:
1792 006626 010046                             MOV    R0,=(SP)           ;;PUSH R0 ON STACK
1793 006630 010146                             MOV    R1,=(SP)           ;;PUSH R1 ON STACK
1794 006632 010246                             MOV    R2,=(SP)           ;;PUSH R2 ON STACK
1795 006634 010346                             MOV    R3,=(SP)           ;;PUSH R3 ON STACK
1796 006636 010446                             MOV    R4,=(SP)           ;;PUSH R4 ON STACK
1797 006640 010546                             MOV    R5,=(SP)           ;;PUSH R5 ON STACK
1798 006642 016646 000022                     MOV    22(SP),=(SP)      ;;SAVE PS OF MAIN FLOW
1799 006646 016646 000022                     MOV    22(SP),=(SP)      ;;SAVE PC OF MAIN FLOW
1800 006652 016646 000022                     MOV    22(SP),=(SP)      ;;SAVE PS OF CALL
1801 006656 016646 000022                     MOV    22(SP),=(SP)      ;;SAVE PC OF CALL
1802 006662 000002                             RTI
1803
1804                                     ;*RESTORE R0-R5
1805                                     ;*CALL:
1806                                     ;*   RESREG
1807                                     ;*RESREG:
1808 006664 012666 000022                     MOV    (SP)+,22(SP)      ;;RESTORE PC OF CALL
1809 006670 012666 000022                     MOV    (SP)+,22(SP)      ;;RESTORE PS OF CALL
1810 006674 012666 000022                     MOV    (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
1811 006700 012666 000022                     MOV    (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
1812 006704 012605                             MOV    (SP)+,R5          ;;POP STACK INTO R5
1813 006706 012604                             MOV    (SP)+,R4          ;;POP STACK INTO R4
1814 006710 012603                             MOV    (SP)+,R3          ;;POP STACK INTO R3
1815 006712 012602                             MOV    (SP)+,R2          ;;POP STACK INTO R2
1816 006714 012601                             MOV    (SP)+,R1          ;;POP STACK INTO R1
1817 006716 012600                             MOV    (SP)+,R0          ;;POP STACK INTO R0
1818 006720 000002                             RTI
1819
1820

```

```

1821                                     ;*****
1822                                     ;
1823                                     .SBTTL   CASSETTE PRIMITIVES ROUTINE
1824                                     ;
1825                                     ;CALLED VIA JSR PC,(FUNCTION)
1826                                     ;WITH R0 POINTING TO:
1827                                     ;
1828                                     ;   .BYTE 0
1829                                     ;   .BYTE DRIVE # (0=A,1=B)
1830                                     ;   .WORD POINTER TO STATUS/ERROR BYTE
1831                                     ;   .WORD BUFFER ADDRESS
1832                                     ;   .WORD BYTE COUNT
1833                                     ;
1834 006722 010746                             WFG:   MOV    PC,=(SP)           ; WRITE A FILE GAP
1835 006724 000415                             FUNTAB: BR    IOCOM           ;
1836 006726 010746                             WRITE: MOV   PC,=(SP)           ; WRITE A BLOCK OF DATA
1837 006730 000413                             BR     IOCOM           ;
1838 006732 010746                             READ:  MOV   PC,=(SP)           ; READ A BLOCK OF DATA
1839 006734 000411                             BR     IOCOM           ;
1840 006736 010746                             BSFG:  MOV   PC,=(SP)           ; BACK SPACE A FILE GAP
1841 006740 000407                             BR     IOCOM           ;
1842 006742 010746                             BSHG:  MOV   PC,=(SP)           ; BACK SPACE A BLOCK GAP
1843 006744 000405                             BR     IOCOM           ;
1844 006746 010746                             SFFG:  MOV   PC,=(SP)           ; SPACE FORWARD A FILE GAP
1845 006750 000403                             BR     IOCOM           ;
1846 006752 010746                             SFBG:  MOV   PC,=(SP)           ; SPACE FORWARD A BLOCK GAP
1847 006754 000401                             BR     IOCOM           ;
1848 006756 010746                             REWIND: MOV  PC,=(SP)           ; REWIND TO BEGINNING OF TAPE
1849 006760 162716 006724                     IOCOM: SUB   #FUNTAB,(SP)      ;PC ON STACK IS USED TO DETERMINE
1850 006764 006216                             ASR    (SP)                ; FUNCTION
1851 006766 032777 000040 172244                CHKRDY: BIT   #40,#TACSL      ;READY BIT UP?
1852 006774 001774                             BEQ    CHKRDY              ;WAIT ON IT
1853 006776 012667 000166                     MOV    (SP)+,FUNLUC        ;PUT FUNC. IN BITS 1-3 OF FUNLOC
1854 007002 010146                             MOV    R1,=(SP)
1855 007004 010046                             MOV    R0,=(SP)
1856 007006 012701 007162                     MOV    #CASPAR,R1          ;PT. TO INT. HANDLER PARAM, BLK,
1857 007012 010011 000002                     MOV    2(R0),(R1)          ;PTM. TO ST/ERR WD. INTO P,BLK.
1858 007016 105031                             CLR    B(R1)+             ;ZERO ST/ERR BYTE
1859 007020 005200                             INC    R0
1860 007022 152067 000143                     CHANOK: BISH (R0)+,FUNLUC+1   ;COPY CHAN. NUM. TO FUNC. WORD
1861 007026 005720                             TST    (R0)
1862 007030 012061 000002                     MOV    (R0)+,2(R1)        ;BUFF. ADR.
1863 007034 011021                             MOV    (R0),(R1)+         ;BYTE COUNT
1864 007036 005721                             TST    (R1)+
1865 007040 152711 000101                     BISH   #01,(R1)          ;ADD INT. ENABLE AND GO BITS
1866                                     ;TO THE FUNCTION
1867 007044 011177 172170                     DOPUN: MOV   (R1),#TACSL     ;START THE FUNCTION
1868 007050 012600                             MOV    (SP)+,R0
1869 007052 012601                             MOV    (SP)+,R1
1870 007054 000207                             RTS    PC

```

```

1871                                     ;*****
1872                                     ;
1873                                     ;SBTTL      CASSETTE INTERRUPT HANDLER
1874                                     ;
1875                                     ;
1876 007056                               CASINT:
1877 007056 010146                         MOV    R1,-(SP)      ;PUSH R1 ON STACK
1878 007060 010246                         MOV    R2,-(SP)      ;PUSH R2 ON STACK
1879 007062 016702 172152                   MOV    TACSL,R2
1880 007066 012701 007164                   MOV    #CBCNT,R1    ;PARAM. BLOCK
1881 007072 105712                           TSTB   (R2)         ;TRANSFER REQUEST?
1882 007074 100411                           BMI    TREQ         ;YES
1883 007076 011204                           MOV    (R2),R4
1884 007100 000304                           SWAB   R4
1885 007102 052704 000001                   BIS    #1,R4        ;SET THE DONE BIT
1886 007106 110451                           MOVB   R4,0+(R1)    ;LOAD ST/ERR BYTE
1887 007110 105012                           CLRB   (R2)         ;TURN OFF INTERS.
1888 007112 012602                           MOV    (SP)+,R2     ;POP STACK INTO R2
1889 007114 012601                           MOV    (SP)+,R1     ;POP STACK INTO R1
1890 007116 000002                           RTI
1891
1892 007120 005321                               ;
1893 007122 100412                           TREQ:  DEC    (R1)+
1894 007124 032712 000004                   BMI    RWILBS       ;WHEN COUNT NEG., DONE
1895 007130 001003                           BIT    #4,(R2)     ;CHECK READ OR WRITE FUNC.
1896 007132 113177 172104                   BNE    READM
1897 007136 000402                           MOVB   0(R1)+,0TAUBL ;WRITE BYTE
1898 007140 117731 172076                   BR     RWDONE
1899 007144 005241                           HEADR: MOVB   0TADBL,0(R1)+ ;READ BYTE
1900 007146 000402                           RWDONE: INC    -(R1) ;BUMP ADDR.
1901                                     ;
1902 007150 052712 000020                   BR     BACK
1903 007154                                     ;
1904 007154 012602                           RWILBS: BIS    #20,(R2) ;INIT. LAST BYTE SEQ.
1905 007156 012601                           BACK:
1906 007160 000002                           MOV    (SP)+,R2     ;POP STACK INTO R2
1907                                     ;
1908                                     ;
1909                                     ;
1910                                     ;
1911 007162 000000                           CASPAR: ,WORD 0     ;STATUS/ERROR POINTER
1912 007164 000000                           CBCNT:  ,WORD 0     ;WORKING BYTE COUNT
1913 007166 000000                           ,WORD 0             ;WORKING BUFFER ADDRESS
1914 007170 000000                           FUNLOC: ,WORD 0     ;FUNC. WORD BUILT HERE
1915
1916

```

```

1917                                     ;*****
1918                                     ;SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
1919                                     ;
1920                                     ;*****
1921                                     ;THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
1922                                     ;DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
1923                                     ;POSITIVE.
1924                                     ;CALL
1925                                     ;*   MOV    #PNTR,-(SP)      ;POINTER TO LOW WORD OF BINARY NUMBER
1926                                     ;*   JSR   PC,#SUB2D      ;THE FIRST ADDRESS OF ASCII
1927                                     ;*   RETURN                      ;IS ON THE STACK
1928
1929
1930
1931 007172 104412                               SDB2D:  SAVREG      ;SAVE REGISTERS
1932 007174 016602 000002                       MOV    2(SP),R2    ;PICKUP THE DATA POINTER
1933 007200 012700 007352                       MOV    #SDECVL,R0 ;GET ADDRESS OF "SDECVL" STRING
1934 007204 010066 000002                       MOV    R0,2(SP)    ;PUT ADDRESS OF ASCII STRING ON STACK
1935 007210 012201                               MOV    (R2)+,R1    ;PICKUP THE BINARY NUMBER
1936 007212 012202                               MOV    (R2)+,R2
1937 007214 012767 000012 000046               MOV    #10,40     ;SET UP TO DO 10 CONVERSIONS
1938 007222 012704 007302                       MOV    #STNPMH,R4 ;ADDRESS OF TEN POWER
1939 007226 012705 007304                       MOV    #STNPMH+2,R5
1940 007232 005003                               1S:   CLR    R3     ;CLEAR PARTIAL
1941 007234 161401                               2S:   SUB    (R4),R1 ;SUBTRACT TEN POWER
1942 007236 005602                               SBC   R2
1943 007240 161502                               SUB   (R5),R2
1944 007242 002402                               BLT   3S          ;BR IF TEN POWER TO LARGE
1945 007244 005203                               INC   R3          ;ADD 1 TO PARTIAL
1946 007246 000772                               BR    2S         ;LOOP
1947 007250 062401                               3S:   ADD    (R4)+,R1 ;RESTORE SUBTRACTED VALUE
1948 007252 005502                               ADC   R2
1949 007254 062402                               ADD   (R4)+,R2
1950 007256 022525                               CMP   (R5)+,(R5)+ ;MOVE TO NEXT TEN POWER
1951 007260 052703 000060                       BIS   #0,R3      ;CHANGE PARTIAL TO ASCII
1952 007264 110320                               MOVB  R3,(R0)+   ;SAVE IT
1953 007266 005327                               DEC   (PC)+      ;JUNE?
1954 007270 000000                               4S:   ,WORD 0
1955 007272 001357                               BNE   1S        ;BR IF NO
1956 007274 105020                               CLRB  (R0)+     ;TERMINATOR
1957 007276 104413                               RESREG ;RESTORE REGISTERS
1958 007300 000207                               RTS   PC        ;RETURN
1959 007302 145000                               STNPMH: 145000   ;1,0E09
1960 007304 035632                               145000         ;1,0E08
1961 007306 160000                               145000         ;1,0E07
1962 007310 002765                               2765          ;1,0E06
1963 007312 113200                               113200        ;1,0E05
1964 007314 000230                               230           ;1,0E04
1965 007316 041100                               041100        ;1,0E03
1966 007320 000017                               17            ;1,0E02
1967 007322 103240                               103240        ;1,0E01
1968 007324 000001                               1             ;1,0E00
1969 007326 023420                               23420        ;1,0E-01
1970 007330 000000                               0             ;1,0E-02
1971 007332 001750                               1750         ;1,0E-03
1972 007334 000000                               0

```

```

1973 007336 000144          144          111,0E02
1974 007340 000000          0
1975 007342 000012          12          111,0E01
1976 007344 000000          0
1977 007346 000001          1          111,0E00
1978 007350 000000          0
1979 007352 000014          0
SDCVLI  ,BLKB 12,          //RESERVE STORAGE FOR ASCII STRING
,SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
1981
1982
1983 //*****
1984 //THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
1985 //UNSIGNED DECIMAL ASCII NUMBER.
1986 //CALL
1987 //*   MOV   NUMBER,=(SP)   //PUT BINARY NUMBER ON THE STACK
1988 //*   JSR   PC,#$SB20     //CALL
1989 //*   RETURN              //ADDRESS OF THE 1ST ASCII CHAR,IS ON THE STACK
1990
1991 007366 016667 000002 000022  SS02D: MOV   2(SP),1$          //SAVE BINARY NUMBER
1992 007374 012746 007416          MOV   #1$,-(SP)         //SET POINTER
1993 007400 004737 007172          JSR   PC,#$SB20         //CALL DOUBLE LENGTH CONVERT
1994 007404 062716 000005          ADD   #5,(SP)          //ONLY ALLOW FIVE CHARACTERS
1995 007410 012666 000002          MOV   (SP)+,2(SP)      //PICKUP POINTER
1996 007414 000207          RTS   PC               //RETURN
1997 007416 000000 000000          1$:  .WORD 0,0
1998          ,SBTTL  TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
1999
2000 //*****
2001 //THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
2002 //LEADING NUMBERS.
2003 //CALL
2004 //*   MOV   #NUMADH,=(SP) //FIRST ADDRESS OF ASCII STRING
2005 //*   JSR   PC,#$SUPMS
2006
2007
2008 007422 010046          SSUPRS: MOV   R0,=(SP)          //SAVE R0
2009 007424 016600 000004          MOV   4(SP),R0        //PICKUP THE POINTER
2010 007430 105710          1$:  TSTB  (R0)          //TERMINATED?
2011 007432 001403          BEQ   2$              //BR IF YES
2012 007434 122720 000050          CMPB  #'0',(R0)+      //IS THIS AN ASCII "0" ?
2013 007440 001773          BEQ   1$              //BR IF YES
2014 007442 005300          2$:  DEC   R0           //BACKUP BY "1"
2015 007444 010067 000002          MOV   R0,3$          //SAVE FOR TYPING
2016 007450 104401          TYPE  TYPE           //GO TYPE
2017 007452 000000          3$:  .WORD 0           //ASCII POINTER GOES HERE
2018 007454 012600          MOV   (SP)+,R0        //RESTORE R0
2019 007456 012616          MOV   (SP)+,(SP)      //RESTORE THE STACK
2020 007460 000207          RTS   PC               //RETURN
2021          ,SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2022
2023 //*****
2024 //THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2025 //CHANGE IT TO BINARY.
2026 //CALL:
2027 //*   RDOCT              //READ AN OCTAL NUMBER
2028 //*   RETURN HERE       //LOW ORDER BITS ARE ON TOP OF THE STACK

```

```

2029 //*
2030 //HIGH ORDER BITS ARE IN SHIOCT
2031 007462 011646          SHDOCT: MOV   (SP),-(SP)          //PROVIDE SPACE FOR THE
2032 007464 016666 000004 000002          MOV   4(SP),2(SP)      //INPUT NUMBER
2033 007472 010046          MOV   R0,=(SP)         //PUSH R0 ON STACK
2034 007474 010146          MOV   R1,=(SP)         //PUSH R1 ON STACK
2035 007476 010246          MOV   R2,=(SP)         //PUSH R2 ON STACK
2036 007500 104410          1$:  ROLIN          //READ AN ASCII LINE
2037 007502 012600          MOV   (SP)+,R0        //GET ADDRESS OF 1ST CHARACTER
2038 007504 005001          CLR   R1               //CLEAR DATA WORD
2039 007506 005002          CLR   R2
2040 007510 112046          2$:  MOVB  (R0)+,(SP)      //PICKUP THIS CHARACTER
2041 007512 001412          BEQ   3$               //IF ZERO GET OUT
2042 007514 006301          ASL   R1               //*2
2043 007516 006102          ROL   R2               //
2044 007520 006301          ASL   R1               //*4
2045 007522 006102          ROL   R2               //
2046 007524 006301          ASL   R1               //*8
2047 007526 006102          ROL   R2               //
2048 007530 042716 177770          BIC   #'C7,(SP)       //STRIP THE ASCII JUNK
2049 007534 062601          ADD   (SP)+,R1        //ADD IN THIS DIGIT
2050 007536 000764          BR    2$              //LOOP
2051 007540 005726          3$:  TST  (SP)+          //CLEAN TERMINATOR FROM STACK
2052 007542 010166 000012          MOV   R1,12(SP)       //SAVE THE RESULT
2053 007546 010267 000010          MOV   R2,SHIOCT
2054 007552 012602          MOV   (SP)+,R2        //POP STACK INTO R2
2055 007554 012601          MOV   (SP)+,R1        //POP STACK INTO R1
2056 007556 012600          MOV   (SP)+,R0        //POP STACK INTO R0
2057 007560 000002          RTI                    //RETURN
2058 007562 000000          SHIOCT: .WORD 0       //HIGH ORDER BITS GO HERE

```

```

2059 ;*****
2060 ,SBTTL ROUTINE TO TYPE DRIVE
2061
2062 ;CALL
2063 ; JSR PC,#TMDRV
2064 ;
2065
2066
2067 007564 104401 013326 TPURV: TYPE ,MSG15 ;" DRIVE "
2068 007570 113746 001613 MOVB #PAMMBK+1,-(SP) ;PICKUP THE DRIVE NUMBER
2069 007574 042716 177776 BIC #'C1,(SP) ;STRIP ANY JUNK
2070 007600 062716 000101 ADD #'A,(SP) ;MAKE IT ASCIZ
2071 007604 012667 000006 MOV (SP)+,IS ;SAVE IT FOR TYPEOUT
2072 007610 104401 TYPE ;TYPE IT
2073 007612 007616 IS
2074 007614 000207 RTS PC
2075 007616 000000 151 .WORD 0
    
```

```

2076 ;*****
2077 ,SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2078
2079 ;CALL
2080 ; JSR PC,#ASKDRV
2081 ; RETURN ;NOTE: R0 AND R1 ARE DESTROYED
2082 ;
2083
2084 007620 104401 013336 ASKDRV: TYPE ,MSGDRV ;<CRLF>"DRIVE(S)? "
2085 007624 005067 171422 CLR DRVKEY
2086 007630 104410 ROLIN ;GO GET A DRIVE
2087 007632 012600 MOV (SP)+,R0 ;SETUP TO CHECK FOR VALID DRIVE(S)
2088 007634 105710 TSTB #R0 ;WAS A DRIVE SELECTED?
2089 007636 001425 BEQ NOTLGL ;BR IF NO
2090 007640 012701 001252 MOV #DRVKEY,R1
2091 007644 122710 000101 LOUP: CMPB #'A,#R0 ;WAS DRIVE "A" SELECTED?
2092 007650 001002 BNE NOTA ;BR IF NO
2093 007652 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "A"
2094 007654 000411 BR NEXT
2095 007656 122710 000102 NOTA: CMPB #'B,#R0 ;WAS DRIVE "B" SELECTED?
2096 007662 001002 BNE NOTB ;BR IF NO
2097 007664 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "B"
2098 007666 000404 BR NEXT
2099 007670 122710 000054 NOTB: CMPB #54,#R0 ;WAS A COMMA TYPED?
2100 007674 001006 BNE NOTLGL ;BR IF NO
2101 007676 105720 TSTB (R0)+ ;DUMP THE COMMA
2102 007700 105710 NEXT: TSTB #R0 ;TERMINATOR?
2103 007702 001406 BEQ EXIT ;BR IF YES
2104 007704 022701 001254 CMP #DRVKEY+2,R1 ;TWO DRIVES SELECTED?
2105 007710 101355 BHI LOOP ;BR IF NO
2106 007712 104401 001202 NOTLGL: TYPE ,SQULS ;ILLEGAL INPUT DETECTED
2107 007716 000740 BR ASKDRV ;GO TRY AGAIN
2108 007720 005767 171326 EXIT: TST DRVKEY ;ANY DRIVE SELECTED?
2109 007724 001772 BEQ NOTLGL ;BR IF NO
2110 007726 000207 RTS PC
2111
    
```

```

2112                                     ;*****
2113 ,SBTTL ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2114 ;CALL
2115 ;JSR PC,0#ASKADR
2116
2117 007730 010046 ASKADR: MOV R0,=(SP) ;SAVE R0
2118 007732 104401 015352 1S: TYPE ,MSGASK ;"TACS?"
2119 007736 104411 RDOCT ;GET VALUE
2120 007740 012600 MOV (SP)+,R0 ;PICK UP THE OCTAL NUMBER
2121 007742 001411 BEQ 55 ;IF "0" USE OLD VALUES
2122 007744 020027 160000 CMP R0,#160000 ;MAKE SURE IT IS A BUS ADDRESS
2123 007750 103770 BLD 15
2124 007752 010037 MOV R0,##TACSL ;SAVE THE TACS
2125 007756 062700 000002 ADD #2,R0 ;STEP TO TADB ADDRESS
2126 007762 010037 001242 MOV R0,##TADBL ;AND SAVE IT
2127 007766 104401 015361 5S: TYPE ,MSGVEC ;"VECTOR?"
2128 007772 104411 RDOCT
2129 007774 012600 MOV (SP)+,R0
2130 007776 001411 5S
2131 010000 020027 001000 CMP R0,#1000 ;MAKE SURE ADDRESS IS IN VECTOR AREA
2132 010004 103370 BMS 35
2133 010006 010037 001244 MOV R0,##TAVEC ;SAVE AS VECTOR ADDRESS
2134 010012 062700 000002 ADD #2,R0
2135 010016 010037 001246 MOV R0,##TAVEC+2
2136 010022 104401 015372 5S: TYPE ,MSGPRI ;ASK FOR PRIORITY
2137 010026 104411 RDOCT
2138 010030 012600 MOV (SP)+,R0
2139 010032 001413 BEQ 65 ;IF "0" USE OLD VALUE
2140 010034 020027 000007 CMP R0,#/ ;MAKE SURE ITS VALID
2141 010040 101370 BMI 55
2142 010042 000300 SWAB R0 ;PUT INTO HIGH BYTE
2143 010044 006200 ASR R0 ;AND SHIFT
2144 010046 006200 ASR R0 ;INTO PROPER
2145 010050 006200 ASR R0 ;POSITION
2146 010052 042700 177437 BIC #<340>,R0 ;SAVE ONLY PRIORITY BITS
2147 010056 010037 001250 MOV R0,##TAPRI ;STORE IT AWAY
2148 010062 104401 015405 6S: TYPE ,MTALS ;TACS="
2149 010066 016746 171146 MOV TACSL,=(SP) ;SAVE TACSL FOR TYPEOUT
2150 010072 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2151 010074 104401 015414 TYPE ,MTAUB ;"TADB="
2152 010100 016746 171136 MOV TADBL,=(SP) ;SAVE TADBL FOR TYPEOUT
2153 010104 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2154 010106 104401 015423 TYPE ,MTAVEC ;"VECTOR="
2155 010112 016746 171126 MOV TAVEC,=(SP) ;SAVE TAVEC FOR TYPEOUT
2156 010116 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2157 010120 104401 015435 TYPE ,MTAPRI ;"PRIORITY="
2158 010124 016746 171120 MOV TAPRI,=(SP) ;SAVE TAPRI FOR TYPEOUT
2159 010130 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2160 010132 104401 015451 TYPE ,MSGUK ;"OK?"
2161 010136 104407 RDCHR ;GO READ ONE CHARACTER
2162 010140 012600 MOV (SP)+,R0 ;GET IT
2163 010142 022700 000015 CMP #15,R0 ;IS IT "CR"?
2164 010146 001406 BEQ 75 ;BRANCH IF YES
2165 010150 022700 000131 CMP #'Y',R0 ;IS IT "Y"?
2166 010154 001403 BEQ 75 ;IT WAS
2167 010156 104401 001202 TYPE ,SQUES ;TYPE "?"

```

```

2168 010162 000663 BR 15 ;AND LET HIM CORRECT THEM
2169 010164 104401 015456 7S: TYPE ,MYES ;TYPE OUT "YES"
2170 010170 012600 MOV (SP)+,R0 ;RESTORE R0
2171 010172 000207 RTS PC ;AND RETURN

```

```

2172 ;/*****
2173 ;
2174 ;SMTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2175 ;
2176 ;CALL:
2177 ; MOV #DRVKEY,R0
2178 ; JSR PC,#EXAM ;R1 IS DESTROYED
2179 ; NORMAL RETURN
2180 ; ERROR RETURN
2181
2182 010174 013701 001240 EXAM: MOV #TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2183 010200 005011 CLR (R1) ;DRIVE="A", FUNCTION="WFG"
2184 010202 122710 000101 CMPB #A,(R0) ;EXAMINE DRIVE "A"?
2185 010206 001402 BEQ 15 ;BR IF YES
2186 010210 052711 000400 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2187 010214 032711 000040 15: BIT #READY,(R1) ;WAIT ON READY
2188 010220 001775 BEQ 15
2189 010222 005711 TST (R1) ;ANY ERROR?
2190 010224 100024 BPL 45 ;BR IF NO
2191 010226 032711 001000 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2192 010232 001017 BNE 35 ;BR IF YES
2193 010234 032711 010000 BIT #WRITELOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2194 010240 001411 BEQ 25 ;BR IF NO
2195 010242 122777 000201 170670 CMPB #BIT07;BIT00,#SWR ;"READONLY" SELECTED? (NDIPAS)
2196 010250 001412 BEQ 45 ;BR IF YES
2197 010252 122777 000203 170660 CMPB #BIT07;BIT01;BIT00,#SWR ;(RDZPAS)?
2198 010260 001406 BEQ 45 ;BR IF YES
2199 010262 000403 BR 35 ;TAKE THE ERROR EXIT
2200 010264 032711 020000 25: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2201 010270 001002 BNE 45 ;BR IF YES
2202 010272 062716 000002 35: ADD #2,(SP) ;TAKE ERROR RETURN
2203 010276 000207 45: RTS PC ;RETURN
    
```

```

2204 ;SMTTL TYPE ROUTINE
2205 ;/*****
2206 ;
2207 ;ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE,
2208 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2209 ;*NOTE1: #NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2210 ;*NOTE2: #FILLC CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2211 ;*NOTE3: #FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2212 ;
2213 ;CALL:
2214 ;#1) USING A TRAP INSTRUCTION
2215 ;* TYPE #MESADR ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2216 ;#DR TYPE
2217 ;* TYPE
2218 ;* MESADR
2219 ;*
2220
2221 010300 105767 170653 STYPE: TSTB #STPLG ;IS THERE A TERMINAL?
2222 010304 100002 BPL 15 ;BR IF YES
2223 010306 000000 HALT ;HALT HERE IF NO TERMINAL
2224 010310 000407 BR 35 ;LEAVE
2225 010312 010046 15: MOV R0,#(SM) ;SAVE R0
2226 010314 017600 000002 MOV #2(SM),R0 ;GET ADDRESS OF ASCIZ STRING
2227 010320 112046 25: MOVB (R0)*,#(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
2228 010322 001005 BNE 45 ;BR IF IT ISN'T THE TERMINATOR
2229 010324 005726 TST (SP)* ;IF TERMINATOR POP IT OFF THE STACK
2230 010326 012600 60: MOV (SP)*,R0 ;RESTORE R0
2231 010330 062716 000002 35: ADD #2,(SP) ;ADJUST RETURN PC
2232 010334 000002 RTI ;RETURN
2233 010336 122716 000011 45: CMPB #HT,(SM) ;BRANCH IF <HT>
2234 010342 001430 BEQ 65
2235 010344 122716 000200 CMPB #CRLF,(SP) ;BRANCH IF NOT <CRLF>
2236 010350 001006 BNE 55
2237 010352 005726 TST (SP)* ;POP <CR><LF> EQUIV
2238 010354 104401 TYPE ;TYPE A CR AND LF
2239 010356 001203 SCRLF
2240 010360 105067 000130 CLRB #CHARCNT ;CLEAR CHARACTER COUNT
2241 010364 000755 BR 25 ;GET NEXT CHARACTER
2242 010366 004767 000056 55: JSR PC,STYPEC ;GO TYPE THIS CHARACTER
2243 010372 126726 170560 65: CMPB #FILLC,(SP)* ;IS IT TIME FOR FILLER CHARS.?
2244 010376 001350 BNE 25 ;IF NO GO GET NEXT CHAR.
2245 010400 016746 170550 MOV #NULL,#(SP) ;GET # OF FILLER CHARS. NEEDED
2246 ;AND THE NULL CHAR.
2247 010404 105366 000001 75: DECB 1(SM) ;DOES A NULL NEED TO BE TYPED?
2248 010410 002770 BLT 65 ;BR IF NO--GO POP THE NULL OFF OF STACK
2249 010412 004767 000032 JSR PC,STYPEC ;GO TYPE A NULL
2250 010416 105367 000072 DECB #CHARCNT ;DO NOT COUNT AS A COUNT
2251 010422 000770 BR 75 ;LOOP
2252
2253 ;HORIZONTAL TAB PROCEDURE
2254
2255 010424 112716 000040 85: MOVB # ,(SM) ;REPLACE TAB WITH SPACE
2256 010430 004767 000014 95: JSR PC,STYPEC ;TYPE A SPACE
2257 010434 132767 000007 000052 BITB #7,#CHARCNT ;BRANCH IF NOT AT
2258 010442 001372 BNE 95 ;TAB STOP
2259 010444 005726 TST (SP)* ;POP SPACE OFF STACK
    
```

```

2260 010446 000720
2261 010450 105777 170474 STYPEC: TSTB #STPS //GET NEXT CHARACTER
2262 010454 100375 BPL STYPEC //WAIT UNTIL PRINTER IS READY
2263 010456 116677 000002 170466 MOVB 2(SP),#STPB //LOAD CHAR TO BE TYPED INTO DATA REG.
2264 010464 122766 000015 000002 CMPB #CR,#2(SP) //IS CHARACTER A CARRIAGE RETURN?
2265 010472 001003 BNE 15 //BRANCH IF NO
2266 010474 105067 000014 CLRB SCHARCNT //YES--CLEAR CHARACTER COUNT
2267 010500 000406 BR STYPEX //EXIT
2268 010502 122766 000012 000002 1S: CMPB #LF,#2(SP) //IS CHARACTER A LINE FEED?
2269 010510 001402 BEQ STYPEX //BRANCH IF YES
2270 010512 105227 (PC)* INCB //COUNT THE CHARACTER
2271 010514 000000 SCHARCNT:WORD 0 //CHARACTER COUNT STORAGE
2272 010516 000207 STYPEX: RTS PC
2273
2274 .SBTTL TTY INPUT ROUTINE
2275
2276 //*****
2277 .ENABL LSB
2278
2279 //*****
2280 //SOFTWARE SWITCH REGISTER CHANGE ROUTINE,
2281 //ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2282 //SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2283 //WHEN OPERATING IN TTY FLAG MODE.
2284 010520 022767 000176 170412 SCKSWR: CMP #SWREG,SWR //IS THE SOFT-SWR SELECTED?
2285 010526 001074 BNE 15S //BRANCH IF NO
2286 010530 105777 170410 TSTB #STKS //CHAR THERE?
2287 010534 100071 BPL 15S //IF NO, DON'T WAIT AROUND
2288 010536 117746 170404 MOVB #STKB,#(SP) //SAVE THE CHAR
2289 010542 042716 177600 BIC #'C177,(SP) //STRIP-OFF THE ASCII
2290 010546 022726 000007 CMP #7,(SP)+ //IS IT A CONTROL G?
2291 010552 001062 BNE 15S //NO, RETURN TO USER
2292 010554 126727 170354 000001 CMPB SAUTOB,#1 //ARE WE RUNNING IN AUTO-MODE?
2293 010562 001456 BEQ 15S //BRANCH IF YES
2294
2295 010564 104401 011245 TYPE ,SCNTLU //ECHO THE CONTROL-G ('G)
2296 010570 104401 011252 SGTSWR: TYPE ,MSWR //TYPE CURRENT CONTENTS
2297 010574 016746 167376 MOV SWREG,#(SP) //SAVE SWREG FOR TYPEDOUT
2298 010600 104402 TYPDC //GO TYPE--OCTAL ASCII(ALL DIGITS)
2299 010602 104401 011263 TYPE ,SMNEW //PROMPT FOR NEW SWR
2300 010606 005046 CLR -(SP) //CLEAR COUNTER
2301 010610 005046 CLR -(SP) //THE NEW SWR
2302 010612 105777 170326 7S: TSTB #STKS //CHAR THERE?
2303 010616 100375 BPL 7S //IF NOT TRY AGAIN
2304
2305 010620 117746 170322 MOVB #STKB,#(SP) //PICK UP CHAR
2306 010624 042716 177600 BIC #'C177,(SP) //MAKE IT 7-BIT ASCII
2307
2308
2309
2310 010630 021627 000025 9S: CMP (SP),#25 //IS IT A CONTROL-U?
2311 010634 001005 BNE 10S //BRANCH IF NOT
2312 010636 104401 011240 TYPE ,SCNTLU //YES, ECHO CONTROL-U ('U)
2313 010642 062706 000006 20S: ADD #6,SP //IGNORE PREVIOUS INPUT
2314 010646 000757 BR 19S //LET'S TRY IT AGAIN
2315
    
```

```

2316
2317 010650 021627 000015 10S: CMP (SP),#15 //IS IT A <CR>?
2318 010654 001022 BNE 16S //BRANCH IF NO
2319 010656 005766 000004 TST 4(SP) //YES, IS IT THE FIRST CHAR?
2320 010662 001403 BEQ 11S //BRANCH IF YES
2321 010664 016677 000002 170246 MOV 2(SP),#SWR //SAVE NEW SWR
2322 010672 062706 000006 11S: ADD #6,SP //CLEAR UP STACK
2323 010676 104401 001203 14S: TYPE ,SCLLF //ECHO <CR> AND <LF>
2324 010702 126727 170227 000001 CMPB SINTAG,#1 //RE-ENABLE TTY KBD INTERRUPTS?
2325 010710 001003 BNE 15S //BRANCH IF NOT
2326 010712 012777 000100 170224 MOV #100,#STKS //RE-ENABLE TTY KBD INTERRUPTS
2327 010720 000002 15S: RTI //RETURN
2328 010722 004767 177522 16S: JSR PC,STYPEC //ECHO CHAR
2329 010726 021627 000000 CMP (SP),#00 //CHAR < 0?
2330 010732 002420 BLT 18S //BRANCH IF YES
2331 010734 021627 000007 CMP (SP),#07 //CHAR > 7?
2332 010740 003015 BGT 18S //BRANCH IF YES
2333 010742 042726 000006 BIC #07,(SP)+ //STRIP-OFF ASCII
2334 010746 005766 000002 TST 2(SP) //IS THIS THE FIRST CHAR
2335 010752 001403 BEQ 17S //BRANCH IF YES
2336 010754 006316 ASL (SP) //NO, SHIFT PRESENT
2337 010756 006316 ASL (SP) // CHAR OVER TO MAKE
2338 010760 006316 ASL (SP) // ROOM FOR NEW ONE.
2339 010762 005266 000002 17S: INC 2(SP) //KEEP COUNT OF CHAR
2340 010766 056616 177776 BIS #2,(SP), (SP) //SET IN NEW CHAR
2341 010772 000707 BR 7S //GET THE NEXT ONE
2342 010774 104401 001202 18S: TYPE ,SQUELS //TYPE ?<CR><LF>
2343 011000 000720 BR 20S //SIMULATE CONTROL-U
2344
2345
2346
2347 //*****
2348 //THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2349 //CALL:
2350 // * RDCHR //INPUT A SINGLE CHARACTER FROM THE TTY
2351 // * RETURN HERE //CHARACTER IS ON THE STACK
2352 // * //WITH PARITY BIT STRIPPED OFF
2353 //
2354
2355 011002 011646 SRDCHR: MOV (SP),-(SP) //PUSH DOWN THE PC
2356 011004 016666 000004 000002 MOV 4(SP),2(SP) //SAVE THE PS
2357 011012 105777 170126 1S: TSTB #STKS //WAIT FOR
2358 011016 100375 BPL 1S //A CHARACTER
2359 011020 117766 170122 000004 MOVB #STKB,4(SP) //READ THE TTY
2360 011026 042766 177600 000004 BIC #'C177,4(SP) //GET RID OF JUNK IF ANY
2361 011034 026627 000004 000023 CMP 4(SP),#23 //IS IT A CONTROL-S?
2362 011042 001013 BNE 1S //BRANCH IF NO
2363 011044 105777 170074 2S: TSTB #STKS //WAIT FOR A CHARACTER
2364 011050 100375 BPL 2S //LOOP UNTIL ITS THERE
2365 011052 117746 170070 MOVB #STKB,#(SP) //GET CHARACTER
2366 011056 042716 177600 BIC #'C177,(SP) //MAKE IT 7-BIT ASCII
2367 011062 022627 000021 CMP (SP),#21 //IS IT A CONTROL-O?
2368 011066 001366 BNE 2S //IF NOT DISCARD IT
2369 011070 000750 BR 1S //YES, RESUME
2370 011072 026627 000004 000140 3S: CMP 4(SP),#140 //IS IT UPPER CASE?
2371 011100 002407 BLT 4S //BRANCH IF YES
    
```



```

2372 011102 026627 000004 000175 CMP 4(SP),#175 ;IS IT A SPECIAL CHAR?
2373 011110 003003 BGT 4S ;BRANCH IF YES
2374 011112 042766 000040 000004 BIC #40,4(SP) ;MAKE IT UPPER CASE
2375 011120 000002 4S: RTI ;GO BACK TO USER
2376 ;*****
2377 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2378 ;*CALL:
2379 ;* ROLIN ;INPUT A STRING FROM THE TTY
2380 ;* RETURN HERE ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2381 ;* ;TERMINATOR WILL BE A BYTE OF ALL 0'S
2382
2383 011122 010346 SRULIN: MOV R3,-(SP) ;SAVE R3
2384 011124 012703 011230 1S: MOV #STTYIN,R3 ;GET ADDRESS
2385 011130 022703 011240 2S: CMP #STTYIN+8,R3 ;BUFFER FULL?
2386 011134 101405 BLOS 4S ;BR IF YES
2387 011136 104407 ROCHR ;GO READ ONE CHARACTER FROM THE TTY
2388 011140 112613 MOVB (SP)+,(R3) ;GET CHARACTER
2389 011142 122713 000177 10S: CMPB #177,(R3) ;IS IT A RUBOUT
2390 011146 001003 BNE 3S ;SKIP IF NOT
2391 011150 104401 001202 4S: TYPE ,8QUE5 ;TYPE A "?"
2392 011154 000763 BR 1S ;CLEAR THE BUFFER AND LOUP
2393 011156 111367 000044 3S: MOVB (R3),9S ;ECHO THE CHARACTER
2394 011162 104401 011226 TYPE ,9S
2395 011166 122723 000015 CMPB #15,(R3)+ ;CHECK FOR RETURN
2396 011172 001356 BNE 2S ;LOOP IF NOT RETURN
2397 011174 105063 177777 CLRB -1(R3) ;CLEAR RETURN (THE 15)
2398 011200 104401 001204 TYPE ,5LF ;TYPE A LINE FEED
2399 011204 012603 MOV (SP)+,R3 ;RESTORE R3
2400 011206 011646 MOV (SP)-,(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
2401 011210 016666 000004 000002 MOV 4(SP),2(SP) ; FIRST ASCII CHARACTER ON IT
2402 011216 012766 011230 000004 MOV #STTYIN,4(SP)
2403 011224 000002 RTI ;RETURN
2404 011226 000 ;BYTE 0 ;STORAGE FOR ASCII CHAR, TO TYPE
2405 011227 000 ;BYTE 0 ;TERMINATOR
2406 011230 000010 ;BLKB 0 ;RESERVE 8 BYTES FOR TTY INPUT
2407 011240 052536 005015 000 SCNTLUI ,ASCIZ /"U/<15><12> ;CONTROL "U"
2408 011245 136 006507 000012 SCNTLGI ,ASCIZ /"G/<15><12> ;CONTROL "G"
2409 011252 005015 053523 020122 SM$WR: ,ASCIZ <15><12>/SMR = /
2410 011260 020075 000
2411 011263 040 047040 053505 SM$NEW: ,ASCIZ / NEW = /
2412 011270 036440 000040
    
```

```

2413 ;SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2414
2415 ;*****
2416 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2417 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2418 ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2419 ;*CALL:
2420 ;* MOV NUM,-(SP) ;NUMBER TO BE TYPED
2421 ;* TYPOS ;CALL FOR TYPEOUT
2422 ;* ,BYTE N ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2423 ;* ,BYTE M ;M=1 OR 0
2424 ;* ;I=TYPE LEADING ZEROS
2425 ;* ;O=SUPPRESS LEADING ZEROS
2426 ;*
2427 ;*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2428 ;*STYPOS OR STYPOC
2429 ;*CALL:
2430 ;* MOV NUM,-(SP) ;NUMBER TO BE TYPED
2431 ;* TYPON ;CALL FOR TYPEOUT
2432 ;*
2433 ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2434 ;*CALL:
2435 ;* MOV NUM,-(SP) ;NUMBER TO BE TYPED
2436 ;* TYPOC ;CALL FOR TYPEOUT
2437 ;*
2438 011274 017646 000000 STYPOS: MOV 0(SP),-(SP) ;PICKUP THE MODE
2439 011300 116667 000001 000211 MOVB 1(SP),80FILL ;LOAD ZERO FILL SWITCH
2440 011306 112667 000207 MOVB (SP)+,80MODE+1 ;NUMBER OF DIGITS TO TYPE
2441 011312 062716 000002 ADD #2,(SP) ;ADJUST RETURN ADDRESS
2442 011316 000406 BR STYPON
2443 011320 112767 000001 000171 STYPOC: MOVB #1,80FILL ;SET THE ZERO FILL SWITCH
2444 011326 112767 000006 000165 MOVB #6,80MODE+1 ;SET FOR SIX(6) DIGITS
2445 011334 112767 000005 000154 STYPON: MOVB #5,80CNT ;SET THE ITERATION COUNT
2446 011342 010346 MOV R3,-(SP) ;SAVE R3
2447 011344 010446 MOV R4,-(SP) ;SAVE R4
2448 011346 010546 MOV R5,-(SP) ;SAVE R5
2449 011350 116704 000145 MOVB 80MODE+1,R4 ;GET THE NUMBER OF DIGITS TO TYPE
2450 011354 005404 NEG R4
2451 011356 062704 000006 ADD #6,R4 ;SUBTRACT IT FOR MAX, ALLOWED
2452 011362 110467 000132 MOVB R4,80MODE ;SAVE IT FOR USE
2453 011366 116704 000125 MOVB 80FILL,R4 ;GET THE ZERO FILL SWITCH
2454 011372 016605 000012 MOV 12(SP),R5 ;PICKUP THE INPUT NUMBER
2455 011376 005003 CLR R3 ;CLEAR THE OUTPUT WORD
2456 011400 006105 1S: ROL R5 ;ROTATE MSB INTO "C"
2457 011402 000404 BR 3S ;GO DO MSB
2458 011404 006105 2S: ROL R5 ;FORM THIS DIGIT
2459 011406 006105 ROL R5
2460 011410 006105 ROL R5
2461 011412 010503 MOV R5,R3
2462 011414 006103 3S: ROL R3 ;GET LSB OF THIS DIGIT
2463 011416 105367 000076 DECB 80MODE ;TYPE THIS DIGIT?
2464 011422 100016 BPL 7S ;BR IF NO
2465 011424 042703 177770 BIC #177770,R3 ;GET RID OF JUNK
2466 011430 001002 BNE 4S ;TEST FOR 0
2467 011432 005704 TST R4 ;SUPPRESS THIS 0?
2468 011434 001403 BEQ 5S ;BR IF YES
    
```

```

2469 011436 005204          4S: INC R0          ;IDON'T SUPPRESS ANYMORE 0'S
2470 011440 052703 000060          BIS #0,M0          ;MAKE THIS DIGIT ASCII
2471 011444 052703 000040          5S: BIS #1,M0          ;MAKE ASCII IF NOT ALREADY
2472 011450 110367 000040          MOVB R3,R0          ;SAVE FOR TYPING
2473 011454 104401 011514          TYPE ,R0          ;GO TYPE THIS DIGIT
2474 011460 105367 000032          7S: DECB SOCNT          ;COUNT BY 1
2475 011464 003347          BGT 2S          ;BR IF MORE TO DO
2476 011466 002402          BLT 6S          ;BR IF DONE
2477 011470 005204          INC R4          ;INSURE LAST DIGIT ISN'T A BLANK
2478 011472 007744          BR 2S          ;GO DO THE LAST DIGIT
2479 011474 012605          6S: MOV (SP),R5          ;RESTORE R5
2480 011476 012604          MOV (SP),R4          ;RESTORE R4
2481 011500 012603          MOV (SP),R3          ;RESTORE R3
2482 011502 016666 000002 000004          MOV 2(SP),4(SP)      ;SET THE STACK FOR RETURNING
2483 011510 012616          MOV (SP)+,(SP)
2484 011512 000002          RTI          ;RETURN
2485 011514 000          8S: .BYTE 0          ;STORAGE FOR ASCII DIGIT
2486 011515 000          .BYTE 0          ;TERMINATOR FOR TYPE ROUTINE
2487 011516 000          SOCNT: .BYTE 0      ;OCTAL DIGIT COUNTER
2488 011517 000          30FILL: .BYTE 0     ;ZERO FILL SWITCH
2489 011520 000000          30MODE: .WORD 0      ;NUMBER OF DIGITS TO TYPE
    
```

```

2490          .SBTTL TRAP DECODER
2491
2492          ;*****
2493          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2494          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2495          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2496          ;*GO TO THAT ROUTINE.
2497
2498 011522 010046          STRAP1: MOV R0,-(SP)          ;SAVE R0
2499 011524 016600 000002          MOV 2(SP),R0          ;GET TRAP ADDRESS
2500 011530 005740          TST -(R0)          ;BACKUP BY 2
2501 011532 111000          MOVB (R0),R0          ;GET RIGHT BYTE OF TRAP
2502 011534 006300          ASL R0          ;POSITION FOR INDEXING
2503 011536 016000 011556          MOV STRPAD(R0),R0      ;INDEX TO TABLE
2504 011542 000200          RTS R0          ;GO TO ROUTINE
2505
2506
2507          ;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2508
2509 011544 011646          STRAP2: MOV (SP),-(SP)          ;MOVE THE PC DOWN
2510 011546 016666 000004 000002          MOV 4(SP),2(SP)      ;MOVE THE PSW DOWN
2511 011554 000002          RTI          ;RESTORE THE PSW
2512
2513          .SBTTL TRAP TABLE
2514
2515          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2516          ;*BY THE "TRAP" INSTRUCTION.
2517
2518          ;
2519          ; ROUTINE
2520          ;-----
2521          ;
2522          STRPAD: .WORD STRAP2
2523          STYPE ;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2524          STYPOC ;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2525          STYPOS ;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2526          STYPON ;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2527
2528          SGTSWR ;CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
2529
2530          SCKSWR ;CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
2531          SRDCHR ;CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
2532          SRDLIN ;CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE
2533          SRODOCT ;CALL=RODOCT TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
2534          S$AVR6G ;CALL=$AVR6G TRAP+12(104412) SAVE R0-R5 ROUTINE
2535          S$RES6G ;CALL=$RES6G TRAP+13(104413) RESTORE R0-R5 ROUTINE
    
```

```

2534 .SMTTL POWER DOWN AND UP ROUTINES
2535
2536
2537
2538 011606 012737 011752 000024 SPWRDN: MOV #SILLUP,#PWRVEC ;SET FOR FAST UP
2539 011614 012737 000340 000026 MOV #340,#PWRVEC+2 ;PRIO:7
2540 011622 010046 MOV R0,=(SP) ;PUSH R0 ON STACK
2541 011624 010146 MOV R1,=(SP) ;PUSH R1 ON STACK
2542 011626 010246 MOV R2,=(SP) ;PUSH R2 ON STACK
2543 011630 010346 MOV R3,=(SP) ;PUSH R3 ON STACK
2544 011632 010446 MOV R4,=(SP) ;PUSH R4 ON STACK
2545 011634 010546 MOV R5,=(SP) ;PUSH R5 ON STACK
2546 011636 017746 167276 MOV #SWR,=(SP) ;PUSH #SWR ON STACK
2547 011642 010667 000110 MOV SP,$SAVR6 ;SAVE SP
2548 011646 012737 011660 000024 MOV #SP*NUMP,#PWRVEC ;SET UP VECTOR
2549 011654 000000 HALT
2550 011656 000776 BR -2 ;HANG UP
2551
2552
2553
2554 011660 012737 011752 000024 SPWRUP: MOV #SILLUP,#PWRVEC ;SET FOR FAST DOWN
2555 011666 016706 000064 MOV $SAVR6,SP ;GET SP
2556 011672 005067 000060 CLR $SAVR6 ;WAIT LOOP FOR THE TTY
2557 011676 005267 000054 1S: INC $SAVR6 ;WAIT FOR THE INC
2558 011702 001375 BNE 1S ;OF WORD
2559 011704 012677 167230 MOV (SP)+,#SWH ;POP STACK INTO #SWR
2560 011710 012605 MOV (SP)+,R5 ;POP STACK INTO R5
2561 011712 012604 MOV (SP)+,R4 ;POP STACK INTO R4
2562 011714 012603 MOV (SP)+,R3 ;POP STACK INTO R3
2563 011716 012602 MOV (SP)+,R2 ;POP STACK INTO R2
2564 011720 012601 MOV (SP)+,R1 ;POP STACK INTO R1
2565 011722 012600 MOV (SP)+,R0 ;POP STACK INTO R0
2566 011724 012737 011606 000024 MOV #SP*NUMD,#PWRVEC ;SET UP THE POWER DOWN VECTOR
2567 011732 012737 000340 000026 MOV #340,#PWRVEC+2 ;PRIO:7
2568 011740 104401 SPWRMG: TYPE ;REPORT THE POWER FAILURE
2569 011742 011760 SPWRMG: .WORD $POWER ;POWER FAIL MESSAGE POINTER
2570 011744 012716 SPWRMG: MOV (PC)+,(SP) ;RESTART AT SEOP
2571 011746 004412 SPWRAD: .WORD SEOP ;RESTART ADDRESS
2572 011750 000002 RTI
2573 011752 000000 $ILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
2574 011754 000776 BR -2 ;BEFORE THE POWER DOWN WAS COMPLETE
2575 011756 000000 $SAVR6: 0 ;PUT THE SP HERE
2576 011760 005015 $POWER: .ASCIZ <15><12>"POWER"
2577 011766 000122
2578 .EVEN
    
```

```

2579
2580
2581
2582
2583 011770 001116 001162 001166 DT1: .WORD SERRPC,SREG0,SREG2,BYTNUM,SGDDAT,SBDDAT,SGDADR,SBDAADR,0
2584 011776 001230 001124 001126
2585 012004 001120 001122 000000
2586
2587 012012 001116 001162 001166 DT2: .WORD SERRPC,SREG0,SREG2,STMP0,STMP2,0
2588 012020 001172 001176 000000
2589
2590 012026 001116 001162 001166 DT101: .WORD SERRPC,SREG0,SREG2,STMP0,0
2591 012034 001172 000000
2592
2593 012040 001116 001240 000000 DT201: .WORD SERRPC,TACSL,0
2594
2595 012046 001116 000000 DT202: .WORD SERRPC,0
2596
2597
2598
2599
2600
2601 012052 000 DF1: .BYTE DN ;SERRPC
2602 012053 001 .BYTE DN ;SREG0
2603 012054 001 .BYTE DN ;SREG2
2604 012055 001 .BYTE DN ;BYTNUM
2605 012056 000 .BYTE DN ;SGDDAT
2606 012057 000 .BYTE DN ;SBDDAT
2607 012060 000 .BYTE DN ;SGDADR
2608 012061 000 .BYTE DN ;SBDAADR
2609
2610 012062 000 DF2: .BYTE DN ;SERRPC
2611 012063 001 .BYTE DN ;SREG0
2612 012064 001 .BYTE DN ;SREG2
2613 012065 001 .BYTE DN ;STMP0
2614 012066 001 .BYTE DN ;STMP1
2615
    
```

```

2616 /*****
2617 JMESSAGES
2618
2619 2619 012067 127 044522 042524 046111 026585
2620 2620 012074 043055 046111 026585
2621 2621 012106 040507 000120
2622 2622 012106 051127 052111 000105 MXRIT: *ASCIZ /WRITE/
2623 2623 012114 042922 042101 000 MHEAD: *ASCIZ /READ/
2624 2624 012121 102 041501 026513 MXBSFG: *ASCIZ /BACK-SPACE-FILE-GAP/
2625 2625 012126 050123 041501 026505
2626 2626 012134 044906 042514 043455
2627 2627 012142 050101 000
2628 2628 012145 102 041501 026513 MXBSBG: *ASCIZ /BACK-SPACE-BLK-GAP/
2629 2629 012152 050123 041501 026505
2630 2630 012160 046102 026511 040507
2631 2631 012166 000120
2632 2632 012179 050123 026504 044506
2633 2633 012179 050123 026504 044506
2634 2634 012179 050123 026504 044506
2635 2635 012204 042514 043455 050101
2636 2636 012212 000
2637 2637 012213 123 040520 042503 MXSFBG: *ASCIZ /SPACE-FWD-BLK-GAP/
2638 2638 012220 043055 042127 041055
2639 2639 012226 045514 043455 050101
2640 2640 012234 000
2641 2641 012235 122 053505 047111 MXRIND: *ASCIZ /REWIND/
2642 2642 012242 000104 040524 042440
2643 2643 012244 040504 051122 051117
2644 2644 012252 051122 051117 000
2645 2645 012257 123 047131 020103
2646 2646 012284 051105 047522 000122
2647 2647 012284 051104 053111 020105
2648 2648 012300 051511 047440 043106
2649 2649 012306 040655 047111 000105
2650 2650 012314 051104 053111 020105
2651 2651 012330 051511 053440 044522
2652 2652 012330 042524 046055 041517
2653 2653 012336 000113 040505 020122
2654 2654 012337 046103 042101 051105
2655 2655 012336 042514 051122 051117
2656 2656 012346 042440 051122 051117
2657 2657 012347 000
2658 2658 012352 000
2659 2659 012355 109 046111 020105
2660 2660 012357 042507 020120
2661 2661 012356 042522 000522
2662 2662 012362 042524 043516
2663 2663 012364 042524 051122 051117
2664 2664 012410 000
2665 2665 012417 000
2666 2666 012424 042440 047514 045503
2667 2667 012446 042440 051122 051117
2668 2668 012446 000
2669 2669 012446 125 045516 047516
2670 2670 012454 047127 044440 052116
2671 2671 012462 051105 052522 052120
2672 2672 012462 000
  
```

```

T-11 DATA RELIABILITY MAINDEC-11-DZTAE-C
DZTAC-P11 17-MAR-77 14:52
MACY11 27(1006) 17-MAR-77 14:55 PAGE 62

*****
JMESSAGES
*****

2672 2672 012463 124 030501 020061
2673 2673 012470 040506 046111 042105
2674 2674 012476 052040 020117 042522
2675 2675 012504 050123 047117 000104
2676 2676 012512 047516 042040 044522
2677 2677 012550 042526 040440 040526
2678 2678 012556 046111 041101 042514
2679 2679 012556 000
2680 2680 012556 120 020103 020040
2681 2681 012556 020040 046111
2682 2682 012550 020105 020040 041040
2683 2683 012556 047514 045503 020040
2684 2684 012554 041040 052131 020105
2685 2685 012572 020040 043440 042104
2686 2686 012600 052101 020040 041040
2687 2687 012606 042104 052101 020040
2688 2688 012614 043440 040504 051104
2689 2689 012622 020040 041040 040504
2690 2690 012630 051104 000
2691 2691 012635 040 020040
2692 2692 012640 046440 050130
2693 2693 012646 051520 020100
2694 2694 012652 051105 020100
2695 2695 012652 051105 020107
2696 2696 012652 051105 053103
2697 2697 012670 042847 005015
2698 2698 012670 041520 020040
2699 2699 012676 042506 044506 042514
2700 2700 012676 020040 044506 042514
2701 2701 012672 041517 020113 020040
2702 2702 012672 044506 042514 020040
2703 2703 012674 020040 046102 041517
2704 2704 012676 000113
2705 2705 012670 041520 020040 020040
2706 2706 012676 020040 044506 042514
2707 2707 012674 020040 020040 046102
2708 2708 012672 041517 020113 020040
2709 2709 013000 052506 041516 044584
2710 2710 013006 047117 000
2711 2711 013011 120 020103 020040
2712 2712 013016 020040 041501
2713 2713 013024 000123
2714 2714 013026 041520 000
2715 2715 013031 200 020052
2716 2716 013036 047105 026504 043117
2717 2717 013044 052055 051505 020124
2718 2718 013052 025052 000052
2719 2719 013056 051000 043117 020124
2720 2720 013084 051105 047522 051522
2721 2721 013072 000075
2722 2722 013074 044200 051101 020104
2723 2723 013102 051105 047522 051522
2724 2724 013110 000075
2725 2725 013112 041200 052131 051505
2726 2726 013120 051040 040505 036504
2727 2727 013126 000
  
```

```

2728 013127 200 054502 042524 MSG5: .ASCIZ <CRLF>"BYTES WRITTEN="
2729 013134 020123 051127 052111
2730 013142 042524 036516 000
2731 013147 200 040524 042520 MSG6: .ASCIZ <CRLF>"TAPE PASSES="
2732 013154 050040 051501 042523
2733 013162 036523 000
2734 013165 200 044506 042514 MSG7: .ASCIZ <CRLF>"FILES/PASS="
2735 013172 027523 040520 051523
2736 013200 000075
2737 013202 041200 047514 045503 MSG8: .ASCIZ <CRLF>"BLOCKS/FILE="
2738 013210 027523 044506 042514
2739 013216 000075
2740 013220 040440 042116 000 MSG9: .ASCIZ " AND"
2741 013225 040 044527 046114 MSG10: .ASCIZ " WILL BE TESTED"<CRLF>
2742 013232 041040 020105 042526
2743 013240 052123 042105 000200
2744 013246 025052 020052 047506 MSG11: .ASCIZ "*** FURMAT ***"
2745 013250 046522 052101 025040
2746 013262 025052 000
2747 013265 052 025052 051040 MSG12: .ASCIZ "*** READ ***"
2748 013272 040505 020104 020040
2749 013300 025052 000052
2750 013304 025052 020052 051127 MSG13: .ASCIZ "*** WHITE ***"
2751 013312 052111 020105 025040
2752 013320 025052 000
2753 013323 040 000040 MSG14: .ASCIZ " "
2754 013326 042040 044522 042526 MSG15: .ASCIZ " DRIVE "
2755 013334 000040
2756 013336 042200 044522 042526 MSGDRV: .ASCIZ <CRLF>"DRIVE(S)? "
2757 013344 051450 037451 000040
2758 013352 052200 041501 037523 MSGASK: .ASCIZ <CRLF>"TACS?"
2759 013360 000
2760 013361 200 042526 052103 MSGVECT: .ASCIZ <CRLF>"VECTOR?"
2761 013366 051117 000077
2762 013372 050200 044522 051117 MSGPRI: .ASCIZ <CRLF>"PRIORITY?"
2763 013400 052111 037531 000
2764 013405 200 040524 051503 MTACS: .ASCIZ <CRLF>"TACS="
2765 013412 000075
2766 013414 052040 042101 036502 MTADB: .ASCIZ / TAUB=/
2767 013422 000
2768 013423 040 053040 041505 MTAVEC: .ASCIZ " VECTOR="
2769 013430 047524 036522 000
2770 013435 040 050040 044522 MTAPRI: .ASCIZ " PRIORITY="
2771 013442 051117 052111 036531
2772 013450 000
2773 013451 200 045517 000077 MSGOK: .ASCIZ <CRLF>"OK?"
2774 013456 042531 100123 000 MYES: .ASCIZ "YES"<CRLF>
2775 013464 013464 ,EVEN
2776
2777 .SBTTL READ AND WRITE BUFFER
2778
2779 013464 002004 BUFFER: .BLKB 102d.
2780 015470 LASTADDRESS=
2781
2782 000001 .END
    
```

```

ASKADR 007730 618 2117#
ASKDRV 007620 615 2084# 2107
BACK 007124 1900 1903#
BADFUN 005466 1448 1454#
BC_ERR 006050 1517 1559#
BEGINX 002326 612# 648 650 711 715
BEGIN1 001752 226 537#
BEGIN2 002004 227 544#
BEGIN3 002012 228 543 546#
BGNCMN 002016 540 542 545 547#
BIT0 = 000001 170#
BIT00 = 000001 160# 170 213 737 775 2195 2197
BIT01 = 000002 159# 169 212 775 2197
BIT02 = 000004 158# 168 211
BIT03 = 000010 157# 167 210
BIT04 = 000020 156# 166 209
BIT05 = 000040 155# 165 208
BIT06 = 000100 154# 164 207
BIT07 = 000200 153# 163 206 775 2195 2197
BIT08 = 000400 152# 162 205
BIT09 = 001000 151# 161 204
BIT1 = 000002 169#
BIT10 = 002000 150# 203 1354
BIT11 = 004000 149# 202
BIT12 = 010000 148# 201
BIT13 = 020000 147# 200 1361
BIT14 = 040000 146# 199 1316
BIT15 = 100000 145# 198
BIT2 = 000004 168#
BIT3 = 000010 167#
BIT4 = 000020 166#
BIT5 = 000040 165#
BIT6 = 000100 164#
BIT7 = 000200 163#
BIT8 = 000400 162#
BIT9 = 001000 161#
BLKSZ 001312 322# 1503 1585 1648 1650 1704 1729
BLOCK 001306 314# 846 945 906 953 996 1062 1072 1073 1080 1124 1170 1171
1178 1484 1541 1542 1545* 1580 1619 1645 1681 1686 1701 1732
BPTVEC = 000014 177#
BSBG 006742 890 932 1020 1842#
BSFG 006736 1840#
BUFFER 013464 422 1617 1670 1687 1688 1705 2779#
BYTNUM 001230 289# 1730* 2583
C = 000016 790# 791 1510#
CASINT 007056 652 1876#
CASPAR 007162 1856 1911#
CBCNT 007164 1543 1880 1912#
CHANOK 007022 1860#
CHKADR 002350 614 617 636#
CHKDRV 002442 677#
CHKRDY 006766 1851# 1852
CKEOTS 004364 871 884 1065 1149 1163 1200#
CKWR = 104406 1315 1350 1369 2528#
CLEAR 002764 730 733 751# 753
CL_ERR 005712 1514 1526#
    
```


\$\$SKIP	185#	
.EQUAT	2#	75
.HEADE	2#	
.SETUP	2#	537
.SWRMI	2#	57
.SWRLO	67#	
.SCATC	2#	216
.SCMTA	2#	231
.SDB2D	2#	1918
.SEOP	2#	1207
.SERKO	2#	1336
.SPOWF	2#	2534
.SRDUC	2#	2021
.SREAD	2#	2274
.SSAVE	2#	1774
.SSB2D	2#	1980
.SSCUP	2#	1303
.SSPAC	2#	
.SSUPR	2#	1998
.STRAP	2#	2490
.STYPE	2#	2204
.STYPD	2#	2413

. ABS. 015470 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSKZ:DZTAECP11,DSKZ:DZTAECP11,DSKZ:DZTAECP11,DSKZ:DZTAECP11
RUN-TIME: 15 8 .9 SECONDS
RUN-TIME RATIO: 108/25=4.3
CORE USED: 25K (50 PAGES)