TA11

DATA RELIABILITY

MD-11-DZTAE-C

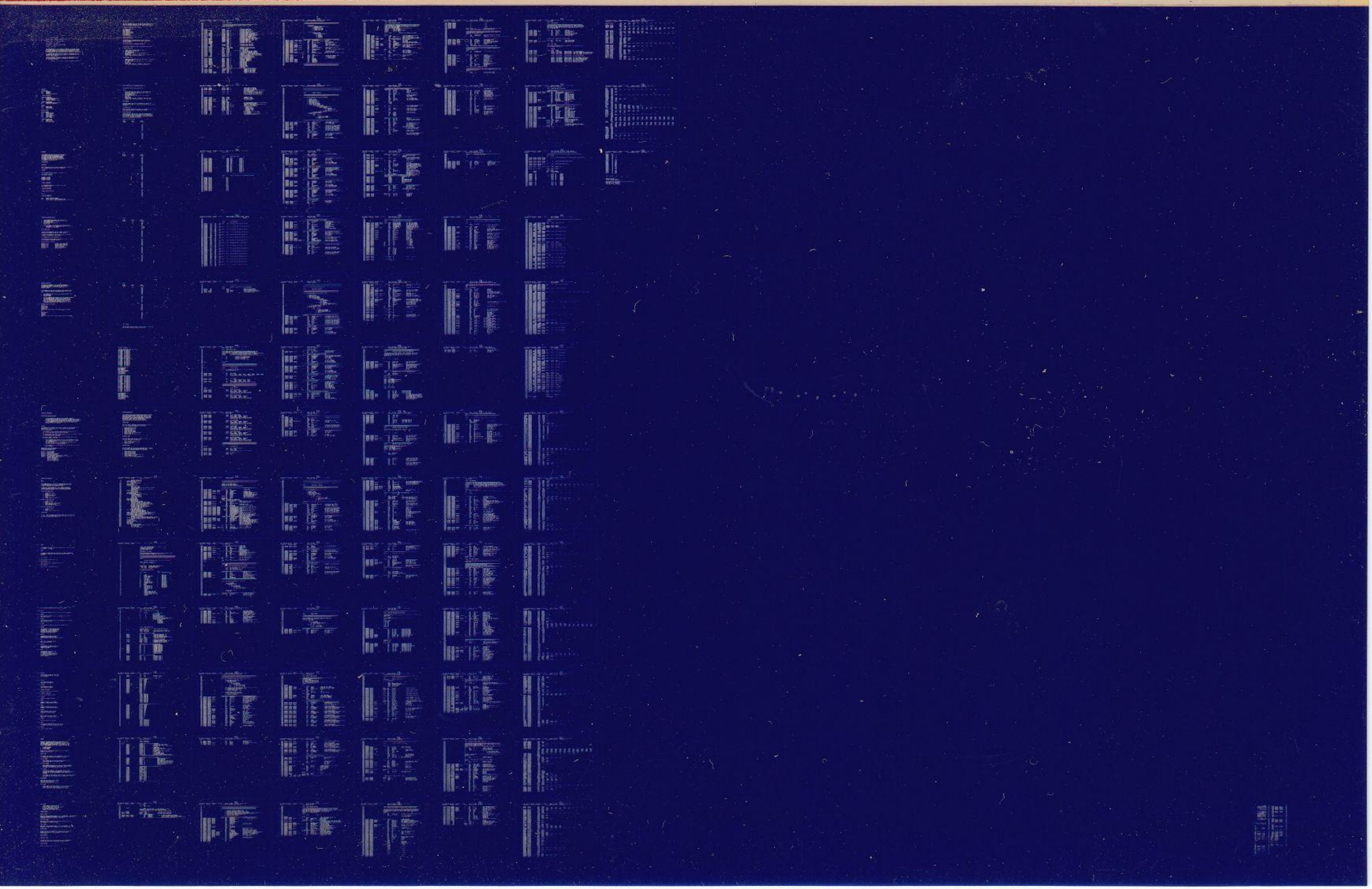
EP-DZTAE-C-DL-B

JUN 1977

digital

FICHE 1 OF 1

MADE IN USA



IDENTIFICATION

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

PRODUCT NAME: TA11 DATA RELIABILITY

DATE CREATED: 16 MARCH 77

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: JIM LACEY

CONTENTS

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 FOR DRIVE
```

4.3.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 210 ALLOWS THE OPERATOR TO CHANGE THE "CONTROL AND STATUS" AND "DATA BUFFER" REGISTER ADDRESSES, THE VECTOR ADDRESS AND THE PRIORITY LEVEL.

THE PROGRAM WILL ASK FOR THE DRIVES TO BE TESTED AS PER 4.3.1. AFTER THE DRIVES HAVE BEEN SELECTED IT WILL ASK FOR:

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
2. VECTOR ADDRESS
3. PRIORITY LEVEL

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONSES WITH A "Y" OR A "CARRIAGE RETURN" FOR "YES" ANYTHING ELSE IS A "NO".

4.3.2.1 ADDRESS SELECTION EXAMPLES

```
DRIVES(S) A
TACS? 177500
VECTOR? 260
PRIORITY? 6
TACS=177500 TADB=177502 VECTOR=000260 PRIORITY=000300
OK?

DRIVES(S) A,B
TACS? 470
VECTOR?
PRIORITY?
TACS=177470 TADB=177472 VECTOR=000260 PRIORITY=000300
OK?
```

G01

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 IOT 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 "EOT"

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
PREFORMING 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 $ERRTB 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 TA11/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	314 062 143 063 146 031 146 031
1	512	063 063 146 146 031 031 146 146

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
5	128	000
		111
		222
		333
		444
		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	000 222 044 266 111 333 155 377
8	1024	001 003 007 017 037 077 177 377
9	32	376 374 370 360 [illegible]
10	128	[illegible]
11	256	[illegible]

8.5.3 (CONT.)

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
12	512	000 000 000 000 000 000 000 000
13	1024	377 377 377 377 377 377 377 377
14	32	000 377 000 377
15	128	017 360 207 170 303 074 341 036

8.6 TEST TYPEOUT

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

```
MAINDEC-11-DZTAE-A

 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
*** READ   *** 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
*** READ   *** 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
*** 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

*** 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 TA11/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 PREFORMED:

1. WRITE A BLOCK OF DATA
2. BACKSPACE A BLOCK GAP
3. READ THE BLOCK
4. CHECK FOR SYNC ERROR
5. CHECK FOR DATA ERROR
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 PREFORMED:

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 PREFORMED. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PREFORMED:

1. READ A BLOCK OF DATA
2. CHECK FOR SYNC ERROR
3. CHECK FOR DATA ERROR
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.

```
.TITLE  TA11 DATA RELIABILITY  MAINDEC-11-DZTAE-C
;*COPYRIGHT (C) 1973,1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY JIM LACEY
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*
;;*****************************************************************
;;*****************************************************************
;;*****************************************************************
.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
-------------------------

                                        INIT
BIT     NAME                            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 02                 0       READ/WRITE
02      FUNCTION BIT 01                 0       READ/WRITE
01      FUNCTION BIT 00                 0       READ/WRITE
        0=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!
```

```
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        10            DING BELL ON ERROR
                        ;*         9            HALT AFTER NEXT "END-OF-TEST" TYPE OUT
                        ;*         8            AT NEXT "EOT" GOTO "END-OF-TEST"
                        ;*         7            PERFORM AS PER SWR<1:0>
                        ;                               00=FORMAT
                        ;                               01=READONLY
                        ;                               10=WRITEONLY
                        ;                               11=READONLY



                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
000006                  R6=     %6              ;;GENERAL REGISTER
000007                  R7=     %7              ;;GENERAL REGISTER
000006                  SP=     %6              ;;STACK POINTER
000007                  PC=     %7              ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
```

```
                        PR6=    300             ;;PRIORITY LEVEL 6
                        PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
```

```
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                        ;;*************************************************************

                        ;**********TA11 FUNCTIONS**********
000000                  XWFG=   0                       ;WRITE FILE GAP FUNCTION
000002                  XWRITE= 2                       ;WRITE FUNCTION
000004                  XREAD=  4                       ;READ FUNCTION
000006                  XBSFG=  6                       ;BACK SPACE FILE GAP FUNCTION
000010                  XBSBG=  10                      ;BACK SPACE BLOCK GAP FUNCTION
000012                  XSFFG=  12                      ;SPACE FWD FILE GAP FUNCTION
000014                  XSFBG=  14                      ;SPACE FWD BLOCK GAP FUNCTION
000016                  XRWND=  16                      ;REWIND FUNCTION

                        ;**********TA11 BIT ASSIGNMENT**********
100000                  ERROR=  BIT15
040000                  CRCERR= BIT14
020000                  LEADER= BIT13
010000                  WRTLOCK=BIT12
004000                  FGAP=   BIT11
002000                  TIMERR= BIT10
001000                  OFFLINE=BIT09
000400                  UNIT=   BIT08
000200                  TR.REQ= BIT07
000100                  INT.EN= BIT06
000040                  READY=  BIT05
000020                  ILBS=   BIT04
000010                  FUNC2=  BIT03
000004                  FUNC1=  BIT02
000002                  FUNC0=  BIT01
000001                  GO=     BIT00
000016                  FUNCTION=       FUNC2+FUNC1+FUNC0
```

```
                                .SBTTL  TRAP CATCHER

                        000000          .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174          .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001752                  JMP     @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  002004                  JMP     @#BEGIN2                ;SELECT DRIVE(S) BEFORE STARTING TEST
000210  000137  002012                  JMP     @#BEGIN3                ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
                                ;;**************************************************************
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001174  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001176  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001200  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001202  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001203  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001204  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
001206  000000          $SOFTNM: .WORD  0                       ;NUMBER OF SOFT ERRORS
001210  000000          HARDNM: .WORD   0                       ;NUMBER OF HARD ERRORS
001212  000000          EOTS:   .WORD   0                       ;NUMBER OF TAPE PASSES
001214  000000  000000  RBYTTL: .WORD   0,0                     ;NUMBER OF BYTES READ
001220  000000  000000  WBYTTL: .WORD   0,0                     ;NUMBER OF BYTES WRITTEN
```

```
                                ;KEEPS COUNT OF REREADS
                                ;KEEPS COUNT OF REWRITES
                                ;THE NUMBER OF BYTES "READ" OR "WRITTEN
                                ;DURING AN OPERATION
                                ;WILL CONTAIN THE # OF THE LAST BLOCK
                                ;AFTER A "FORMAT" OR "WRITE" PASS
                                ;WILL CONTAIN THE # OF THE LAST FILE
                                ;AFTER A "FORMAT" OR "WRITE" PASS


                                ;ADDRESS OF TACS
                                ;ADDRESS OF TADB
                                ;TA11 VECTOR ADDRESS
                                ;TA11 BR LEVEL 6

                                ;MAX REREADS BEFORE CALLING IT A HARD ERROR
                                ;MAX HARD ERRORS ALLOWED
                                ;NUMBER OF TAPE PASSES BEFORE END-OF-TEST
                                ;COUNT # OF TAPE PASSES
                                ;CONTROLS THE TYPE OF PASS
                                ; 1 FORMAT PASS
                                ; 3 READONLY PASSES
                                ; 1 WRITEONLY PASS
                                ; 4 READONLY PASSES
                                ;FILE NUMBER
                                ;1'S COMPLEMENT ON FILE NUMBER
                                ;BLOCK NUMBER
                                ;1'S COMPLEMENT OF BLOCK NUMBER
                                ;NUMBER OF BLOCKS PER FILE
```

```
000000          RDTRYS: .WORD   0
000000          WRTRYS: .WORD   0
000000  000000  BYTNUM: .WORD   0,0

000017          LASTBK: .WORD   15.

000011          LASTFL: .WORD   9.


177500          TACSL:  177500
177502          TADBL:  177502
000260  000262  TAVEC:  260,262
000300          TAPRIO: 300
000000  000000  DRVKEY: 0,0
001252          DRVPNT: DRVKEY
000003          MAXRDS: .WORD   3
000003          MAXERR: .WORD   3
000022          MAXEOT: .WORD   18.
000000          PASCNT: .WORD   0
001274  000000  PSCNTL: .WORD   FORPAS,0
000001          FORPAS: .WORD   1
000003          RD1PAS: .WORD   3
000001          WRTPAS: .WORD   1
000004          RD2PAS: .WORD   4
000             FILE:   .BYTE   0
377             .BYTE   377
000             BLOCK:  .BYTE   0
377             .BYTE   377
000020          FILESZ: .WORD   16.
```

```
                                ;////////////////////////////////////////////////////////////////////////////////

                                .SBTTL          BLOCK SIZE TABLE

001312  002000                  BLKSZ:  .WORD   1024.            ;BLOCK 0
001314  001000                          .WORD   512.             ;BLOCK 1
001316  002000                          .WORD   1024.            ;BLOCK 2
001320  000400                          .WORD   256.             ;BLOCK 3
001322  002000                          .WORD   1024.            ;BLOCK 4
001324  000200                          .WORD   128.             ;BLOCK 5
001326  002000                          .WORD   1024.            ;BLOCK 6
001330  000100                          .WORD   64.              ;BLOCK 7
001332  002000                          .WORD   1024.            ;BLOCK 8
001334  000040                          .WORD   32.              ;BLOCK 9
001336  000200                          .WORD   128.             ;BLOCK 10
001340  000400                          .WORD   256.             ;BLOCK 11
001342  001000                          .WORD   512.             ;BLOCK 12
001344  002000                          .WORD   1024.            ;BLOCK 13
001346  000040                          .WORD   32.              ;BLOCK 14
001350  000200                          .WORD   128.             ;BLOCK 15

                                ;////////////////////////////////////////////////////////////////////////////////

                                .SBTTL          TABLE OF POINTERS TO THE DIFFERENT PATTERNS

001352  001412                  PATS:   PAT0
001354  001422                          PAT1
001356  001432                          PAT2
001360  001442                          PAT3
001362  001452                          PAT4
001364  001462                          PAT5
001366  001472                          PAT6
001370  001502                          PAT7
001372  001512                          PAT8
001374  001522                          PAT9
001376  001532                          PAT10
001400  001542                          PAT11
001402  001552                          PAT12
001404  001562                          PAT13
001406  001572                          PAT14
001410  001602                          PAT15
```

```
                        ;//////////////////////////////////////////////////////////////////////////////
                        .SBTTL          DATA PATTERNS
001412  314  063  314  PAT0:   .BYTE   314,063,314,063,146,231,146,231
001415  063  146  231
001420  146  231
001422  314  231  063  PAT1:   .BYTE   314,231,063,146,314,231,063,146
001425  146  314  231
001430  063  146
001432  001  002  004  PAT2:   .BYTE   001,002,004,010,020,040,100,200
001435  010  020  040
001440  100  200
001442  177  277  337  PAT3:   .BYTE   177,277,337,357,367,373,375,376
001445  357  367  373
001450  375  376
001452  252  125  252  PAT4:   .BYTE   252,125,252,125,252,125,252,125
001455  125  252  125
001460  252  125
001462  000  111  222  PAT5:   .BYTE   000,111,222,333,044,155,266,377
001465  333  044  155
001470  266  377
001472  000  044  111  PAT6:   .BYTE   000,044,111,155,222,266,333,377
001475  155  222  266
001500  333  377
001502  000  222  044  PAT7:   .BYTE   000,222,044,266,111,333,155,377
001505  266  111  333
001510  155  377
001512  001  003  007  PAT8:   .BYTE   001,003,007,017,037,077,177,377
001515  017  037  077
001520  177  377
001522  376  374  370  PAT9:   .BYTE   376,374,370,360,340,300,200,000
001525  360  340  300
001530  200  000
001532  001  376  002  PAT10:  .BYTE   001,376,002,375,004,373,010,367
001535  375  004  373
001540  010  367
001542  020  357  040  PAT11:  .BYTE   020,357,040,337,100,277,200,177
001545  337  100  277
001550  200  177
001552  000  000  000  PAT12:  .BYTE   000,000,000,000,000,000,000,000
001555  000  000  000
001560  000  000
001562  377  377  377  PAT13:  .BYTE   377,377,377,377,377,377,377,377
001565  377  377  377
001570  377  377
001572  000  377  000  PAT14:  .BYTE   000,377,000,377,000,000,377,377
001575  377  000  000
001600  377  377
001602  017  360  207  PAT15:  .BYTE   017,360,207,170,303,074,341,036
001605  170  303  074
001610  341  036
```

```
                                ;////////////////////////////////////////////////////////////////////////////////
                                .SBTTL          PARAMETER BLOCK USED WITH ALL FUNCTIONS

001612  000           PARMBK: .BYTE   0               ;USED FOR STATUS/ERROR
001613  000                   .BYTE   0               ;DRIVE # (DRIVE A=0, B=1)
001614  001612                .WORD   PARMBK          ;POINTS TO STATUS/ERROR BYTE
001616  013464                .WORD   BUFFER          ;FIRST ADDRESS OF DATA BUFFER
001620  000000                .WORD   0               ;USED FOR BYTE COUNT.
```

```
                        .SBTTL  ERROR POINTER TABLE

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

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

001622                  $ERRTB:

                        ;;*********************************************************************
                        ;;*********************************************************************
                        ;;*********************************************************************

001622                  ITEMS0:         ;ITEMS 001-002

                        ;NOTE:  ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS
                        ;       UNLESS OTHERWISE NOTED

                        ;ITEM   1
001622  012244                  EM1     ;DATA ERROR
001624  012535                  DH1     ;PC     FILE    BLOCK   BYTE    GDDAT   BDDAT   GDADR   BDADR
001626  011770                  DT1     ;$ERRPC $REG0   $REG2   BYTNUM  $GDDAT  $BDDAT  $GDADR  $BDADR
001630  012052                  DF1     ;FILE,BLOCK AND BYTE ARE TYPED IN DECIMAL

                        ;ITEM   2
001632  012257                  EM2     ;SYNC ERROR
001634  012633                  DH2     ;       EXPT'D  EXPT'D  RCV'D   RCV'D
                                        ;PC     FILE    BLOCK   FILE    BLOCK
001636  012012                  DT2     ;$ERRPC $REG0   $REG2   $TMP0   $TMP2
001640  012062                  DF2     ;ALL NUMBERS EXCEPT $ERRPC ARE TYPED IN DECIMAL

                        ;;*********************************************************************
                        ;;*********************************************************************
                        ;;*********************************************************************

001642                  ITEMS1:         ;ITEMS 101-107

001642  012272                  EM101   ;DRIVE IS OFF-LINE
001644  012750                  DH101   ;PC     FILE    BLOCK   FUNCTION
001646  012026                  DT101   ;$ERRPC $REG0   $REG2   $TMP0
001650  000000                  0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME

001652  012314                  EM102   ;DRIVE IS WRITE-LOCK
001654  012750                  DH101   ;PC     FILE    BLOCK   FUNCTION
001656  012026                  DT101   ;$ERRPC $REG0   $REG2   $TMP0
001660  000000                  0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
```

```
  481  001662  012340                    EM103   ;CLEAR LEADER ERROR
  482  001664  012750                    DH101   ;PC      FILE    BLOCK   FUNCTION
  483  001666  012026                    DT101   ;$ERRPC $REG0   $REG2   $TMP0
  484  001670  000000                    0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
  485                                             ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
  486
  487  001672  012363                    EM104   ;FILE GAP ERROR
  488  001674  012750                    DH101   ;PC      FILE    BLOCK   FUNCTION
  489  001676  012026                    DT101   ;$ERRPC $REG0   $REG2   $TMP0
  490  001700  000000                    0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
  491                                             ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
  492
  493  001702  012402                    EM105   ;TIMING ERROR
  494  001704  012750                    DH101   ;PC      FILE    BLOCK   FUNCTION
  495  001706  012026                    DT101   ;$ERRPC $REG0   $REG2   $TMP0
  496  001710  000000                    0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
  497                                             ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
  498
  499  001712  012417                    EM106   ;BLOCK CHECK ERROR
  500  001714  012750                    DH101   ;PC      FILE    BLOCK   FUNCTION
  501  001716  012026                    DT101   ;$ERRPC $REG0   $REG2   $TMP0
  502  001720  000000                    0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
  503                                             ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
  504
  505  001722  012441                    EM107   ;UNKNOWN INTERRUPT
  506  001724  012750                    DH101   ;PC      FILE    BLOCK   FUNCTION
  507  001726  012026                    DT101   ;$ERRPC $REG0   $REG2   $TMP0
  508  001730  000000                    0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
  509                                             ;$TMP0 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                    DH201   ;PC      TACS
  519  001736  012040                    DT201   ;$ERRPC TACS
  520  001740  000000                    0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
  521
  522  001742  012512                    EM202   ;NO DRIVES AVAILABLE
  523  001744  013026                    DH202   ;PC
  524  001746  012046                    DT202   ;$ERRPC
  525  001750  000000                    0       ;
  526
```

```
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;;**************************************************************
                        
                        ;BEGIN1 IS FOR NORMAL START
                        ;BEGIN2 IS FOR DRIVE SELECTION
                        ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
                        
                        ;;**************************************************************
                        
001752  005005          BEGIN1: CLR     R5                      ;NORMAL START
001754  012737  041101  001252          MOV     #"AB,@#DRVKEY
001762  122737  000005  000041          CMPB    #5,@#41                 ;CASSETTE DDP?
001770  001012                          BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001772  022737  000260  001244          CMP     #260,@#TAVEC            ;STANDARD VECTOR?
002000  001006                          BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
002002  000403                          BR      BEGIN3                  ;GET DRIVES AND ADDRESSES
002004  012705  000001          BEGIN2: MOV     #1,R5                   ;ASK FOR DRIVES FLAG
002010  000402                          BR      BGNCMN                  ;BEGIN COMMON CODE
002012  012705  000002          BEGIN3: MOV     #2,R5                   ;ASK FOR DRIVES AND ADDRESSES
002016  005067  175754          BGNCMN: CLR     PS
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002022  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
002026  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
002030  022706  001140                  CMP     #SWR,R6 ;;DONE?
002034  001374                          BNE     .-6             ;;LOOP BACK IF NO
002036  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
002042  012737  004760  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002050  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002056  012737  005060  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002064  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002072  012737  011522  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002100  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
002106  012737  011606  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
002114  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
002122  016767  002314  002304          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
002130  012767  002130  176750          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002136  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002142  012737  002176  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
002150  012767  177570  176762          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002156  012767  177570  176756          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002164  022777  177777  176746          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
002172  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
002174  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
002176  012716  002204          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
002202  000002                          RTI
002204  012767  000176  176726  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
002212  012767  000174  176722          MOV     #DISPREG,DISPLAY
002220  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                .SBTTL  TYPE PROGRAM NAME
```

```
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002224  005227  177777                  INC     #-1                     ;;FIRST TIME?
002230  001036                          BNE     67$                     ;;BRANCH IF NO
002232  022737  004726  000042          CMP     #$ENDAD,@#42            ;;ACT-11?
002240  001432                          BEQ     67$                     ;;BRANCH IF YES
002242  104401  002300                  TYPE    ,68$                    ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002246  005737  000042                  TST     @#42                    ;;ARE WE RUNNING UNDER XXDP/ACT?
002252  001006                          BNE     69$                     ;;BRANCH IF YES
002254  026727  176660  000176          CMP     SWR,#SWREG              ;;SOFTWARE SWITCH REG SELECTED?
002262  001005                          BNE     70$                     ;;BRANCH IF NO
002264  104405                          GTSWR                           ;;GET SOFT-SWR SETTINGS
002266  000403                          BR      70$
002270  112767  000001  176636  69$:    MOVB    #1,$AUTOB               ;;SET AUTO-MODE INDICATOR
002276                          70$:
002276  000413                          BR      67$                     ;;GET OVER THE ASCIZ
                                ;;68$:  .ASCIZ  <CRLF>/MAINDEC-11-DZTAE-C/<CRLF>
002326                          67$:

                                ;;******************************************************************
                                ;;******************************************************************

                                ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
                                ;
                                ;       R5=2    ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
                                ;       R5=1    ASK FOR DRIVE(S)
                                ;       R5=0    DON'T ASK FOR ANYTHING
                                ;
                                ;;******************************************************************
002326  010504                  BEGINX: MOV     R5,R4                   ;COPY R5
002330  005305                          DEC     R5                      ;ASK FOR DRIVES?
002332  002406                          BLT     CHKADR                  ;BR IF NO
002334  004737  007620                  JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
002340  005305                          DEC     R5                      ;ASK FOR ADDRESSES?
002342  002402                          BLT     CHKADR                  ;BR IF NO
002344  004737  007730                  JSR     PC,@#ASKADR             ;GO GET TA11 ADDRESSES
                                ;;******************************************************************
                                ;;******************************************************************

                                ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
                                ;
                                ;I.     TIMEOUT OCCURRED
                                ;       A. TYPE ERROR MESSAGE
                                ;       B. EXAMINE R4
                                ;               1. R4>0 GOTO BEGINX
                                ;               2. R4=0 EXAMINE (42)
                                ;                       A. (42)=0 GOTO BEGINX
                                ;                       B. (42)>0 GOTO $ENDAD
                                ;
                                ;II.    TIMEOUT DIDN'T OCCUR
                                ;       A. CONTINUE
                                ;
                                ;;******************************************************************
002350  012737  002366  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE OF TIMEOUTS
002356  005000                          CLR     R0                      ;USE AS A SWITCH
002360  005777  176654                  TST     @TACSL                  ;SEE IF TA11 RESPONDS
```

```
  639  002364  000402                        BR      2$                      ;BR IF NO TIMEOUT
  640  002366  005200                  1$:   INC     R0                      ;COME HERE ON TIMEOUT
  641  002370  022626                        CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
  642  002372  012737  000006  000004  2$:   MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
  643  002400  005700                        TST     R0                      ;DID A TIMEOUT OCCUR?
  644  002402  001412                        BEQ     3$                      ;BR IF NO
  645  002404  104201                        ERROR   201                     ;TA11 FAILED TO RESPOND
  646  002406  012705  000002                MOV     #2,R5                   ;DRIVES & ADDRESSES
  647  002412  005704                        TST     R4                      ;OPERATOR INPUTS?
  648  002414  001344                        BNE     BEGINX                  ;BR IF YES
  649  002416  013700  000042                MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
  650  002422  001741                        BEQ     BEGINX                  ;BR IF NO MONITOR
  651  002424  000137  004726                JMP     @#$ENDAD                ;GO TO END
  652  002430  012777  007056  176606  3$:   MOV     #CASINT,@TAVEC          ;SETUP CASSETTE INTERRUPT VECTOR
  653  002436  005077  176604                CLR     @TAVEC+2
```

```
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;
                        ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
                        ;
                        ;I.     DESIRED DRIVES CAN NOT BE TESTED
                        ;       A. TYPE ERROR MESSAGE
                        ;       B. EXAMINE R4
                        ;               1. R4>0 GOTO BEGINX
                        ;               2. R4=0 EXAMINE (42)
                        ;                       A. (42)=0 GOTO BEGINX
                        ;                       B. (42)>0 GOTO $ENDAD
                        ;
                        ;II.    BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
                        ;       A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
                        ;       B. TYPE THE DRIVE TO BE TESTED
                        ;       C. CONTINUE IN PROGRAM
                        ;
                        ;III.   DESIRED DRIVE(S) CAN BE TESTED
                        ;       A. TYPE THE DRIVE(S) TO BE TESTED
                        ;       B. CONTINUE IN PROGRAM
                        ;
                        ;;*****************************************************************
002442  012700  001252  CHKDRV: MOV     #DRVKEY,R0              ;PICKUP ADDRESS OF ASCII DRIVE KEY
002446  004737  010174          JSR     PC,@#EXAM               ;GO EXAMINE FIRST DRIVE
002452  000410                  BR      1$                      ;OK TO TEST---GO CHECK NEXT
002454  116010  000001          MOVB    1(R0),(R0)              ;REPLACE 1ST WITH 2ND
002460  001412                  BEQ     2$                      ;BR IF NO 2ND DRIVE SELECTED
002462  004737  010174          JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002466  000407                  BR      2$                      ;OK TO TEST
002470  005010                  CLR     (R0)                    ;CLEAR DRIVE CODES
002472  000405                  BR      2$
002474  005200          1$:     INC     R0                      ;POINT TO 2ND
002476  004737  010174          JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002502  000401                  BR      2$                      ;OK TO TEST
002504  105010                  CLRB    (R0)                    ;CLEAR 2ND
002506  012700  001252  2$:     MOV     #DRVKEY,R0              ;RESET ADDRESS POINTERS
002512  010037  001256          MOV     R0,@#DRVPNT
002516  121060  000001          CMPB    (R0),1(R0)              ;1ST = 2ND?
002522  001002                  BNE     3$                      ;BR IF NO
002524  105060  000001          CLRB    1(R0)                   ;YES---CLEAR 2ND
002530  005710          3$:     TST     (R0)                    ;ANY DRIVES?
002532  001424                  BEQ     5$                      ;BR IF NO
002534  112037  001613          MOVB    (R0)+,@#PARMBK+1        ;SETUP TO TYPE THIS DRIVE
002540  104401  001203          TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
002544  106237  001613          ASRB    @#PARMBK+1              ;ADJUST FOR TYPING
002550  004737  007564          JSR     PC,@#TPDRV              ;GO TYPE DRIVE
002554  111037  001613          MOVB    (R0),@#PARMBK+1         ;GET 2ND
002560  001406                  BEQ     4$                      ;BR IF NONE
002562  104401  013220          TYPE    ,MSG9                   ;"AND"
002566  106237  001613          ASRB    @#PARMBK+1              ;ADJUST FOR TYPING
002572  004737  007564          JSR     PC,@#TPDRV              ;GO TYPE THE 2ND DRIVE
002576  104401  013225  4$:     TYPE    ,MSG10                  ;"WILL BE TESTED"<CRLF>
002602  000412                  BR      $START
002604  104202          5$:     ERROR   202                     ;NO DRIVES AVAILABLE
002606  012705  000002          MOV     #2,R5                   ;DRIVES & ADDRESS
```

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

```
                                ;////////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////////
                                ;;******************************************************************
                                ;;******************************************************************
                                ;
                                ;       1. CLEAR THE VARIABLE STORAGE AREA
                                ;       2. SETUP FOR THE DRIVE THAT IS TO BE TESTED
                                ;       3. SETUP "LASTFL" AND "LASTBK" INCASE OPERATOR SELECTS "READONLY" PASS
                                ;       4. SETUP THE MAX. NUMBER OF RETRYS FOR READS AND WRITES
                                ;       5. SETUP TO START WITH A "FORMAT" PASS
                                ;       6. CLEAR THE PASS COUNTER
                                ;
                                ;;******************************************************************
002630  012700  001114          START:  MOV     #$ITEMB,R0              ;CLEAR VARIABLE STORAGE AREA
002634  012701  000005                  MOV     #$BDDAT-$ITEMB/2,R1
002640  004737  002764.                JSR     PC,@#CLEAR                      ;CLEAR AREA
002644  012700  001206                  MOV     #SOFTNM,R0
002650  012701  000014                  MOV     #LASTFL-SOFTNM/2,R1
002654  004737  002764                  JSR     PC,@#CLEAR
002660  013701  001256                  MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
002664  112100                          MOVB    (R1)+,R0                ;SETUP THE DRIVE
002666  006200                          ASR     R0
002670  042700  177776                  BIC     #^CBIT00,R0
002674  110037  001613                  MOVB    R0,@#PARMBK+1
002700  105711                          TSTB    (R1)                    ;END OF DRIVE TABLE?
002702  001002                          BNE     2$                      ;BR IF NO
002704  012701  001252                  MOV     #DRVKEY,R1              ;RESET DRIVE TABLE POINTER
002710  010137  001256          2$:     MOV     R1,@#DRVPNT             ;RESTORE THE DRIVE POINTER
002714  012737  000011  001236          MOV     #9.,@#LASTFL            ;SET "LAST FILE" & "LAST BLOCK" INCASE
002722  012737  000017  001234          MOV     #15.,@#LASTBK           ;OPERATOR SELECTS "READONLY" PASS
002730  013737  001260  001224          MOV     @#MAXRDS,@#RDTRYS       ;SETUP MAX. # OF REREAD TRYS
002736  013737  001262  001226          MOV     @#MAXERR,@#WRTRYS       ;SETUP MAX. # OF REWRITE TRYS
002744  012737  001274  001270          MOV     #FORPAS,@#PSCNTL        ;DO A FORMAT PASS
002752  005037  001272                  CLR     @#PSCNTL+2
002756  005037  001266                  CLR     @#PASCNT                ;CLEAR THE PASS COUNT
002762  000404                          BR      LOOPER                  ;GET ARCUND CLEAR
002764  005020          CLEAR:  CLR     (R0)+
002766  005301                  DEC     R1
002770  003375                  BGT     CLEAR
002772  000207                  RTS     PC
```

```
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;
                                ;I.     CHECK SWR<7>
                                ;       A. SWR<7>=1 PERFORM AS PER SWR<1:0>
                                ;               1. 00=FORMAT PASS
                                ;               2. 01=READ ONLY PASS
                                ;               3. 10=WRITE ONLY PASS
                                ;               4. 11=READ ONLY PASS
                                ;
                                ;       B. SWR<7>=0
                                ;               1. UPDATE PASCNT
                                ;               2. PERFORM AS PER (PSCNTL+2)
                                ;                       A. FORMAT
                                ;                       B. READONLY
                                ;                       C. WRITEONLY
                                ;                       D. READONLY
                                ;
                                ;;*****************************************************************
002774  117737  176140  001172  LOOPER: MOVB    @SWR,@#$TMP0            ;IF SWR<7>=1 ; DO PER SWR<1:0>
003002  142737  177574  001172          BICB    #^C<BIT07!BIT01!BIT00>,@#$TMP0
003010  100432                          BMI     3$                      ;BR IF SWR<7>=1
003012  023777  001266  176250  1$:     CMP     @#PASCNT,@PSCNTL        ;IS THE COUNTER AT MAX.?
003020  002421                          BLT     2$                      ;BR IF NO
003022  005037  001266                  CLR     @#PASCNT                ;RESET THE COUNTER
003026  062737  000002  001270          ADD     #2,@#PSCNTL             ;MOVE TO THE NEXT PASS TYPE
003034  005237  001272                  INC     @#PSCNTL+2
003040  023727  001270  001304          CMP     @#PSCNTL,#RD2PAS+2      ;TIME TO RESET PASS TYPE?
003046  001361                          BNE     1$                      ;BR IF NO
003050  012737  001274  001270          MOV     #FORPAS,@#PSCNTL        ;YES---RESET THE PASS CONTROL WORDS
003056  005037  001272                  CLR     @#PSCNTL+2
003062  000753                          BR      1$                      ;GO CHECK THE COUNT
003064  005237  001266          2$:     INC     @#PASCNT                ;COUNT THIS PASS
003070  013737  001272  001172          MOV     @#PSCNTL+2,@#$TMP0      ;PICKUP THE TYPE OF PASS TO DO NEXT
003076                          3$:
003076  005046                          CLR     -(SP)
003100  113716  001172                  MOVB    @#$TMP0,(SP)            ;PICKUP THE DISPATCH INDEX
003104  106316                          ASLB    (SP)                    ;POSITION THE INDEX
003106  106316                          ASLB    (SP)                    ;    BEFORE USING IT
003110  062607                          ADD     (SP)+,PC                ;GO TO THE ROUTINE
003112  000137  003132                  JMP     @#FORMAT
003116  000137  003562                  JMP     @#READONLY
003122  000137  004136                  JMP     @#WRITEONLY
003126  000137  003562                  JMP     @#READONLY
                                ;//////////////////////////////////////////////////////////////////
                                ;//////////////////////////////////////////////////////////////////
                                ;;*****************************************************************
                                ;;*****************************************************************
```

```
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////

                        .SBTTL          "FORMAT" ROUTINE

                        ;;**************************************************************************
                        ;;**************************************************************************

                        ;THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE
                        ;
                        ;       1. REWIND TO BOT
                        ;       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.) (RDTRYS)<(MAXRDS) GOTO 4.
                        ;                       2.) (RDTRYS)=(MAXRDS)
                        ;                               A.) (WRTRYS)<(MAXERR)
                        ;                                       1.)  BACK BLOCK GAP
                        ;                                       2.)  GOTO 3.
                        ;                               B.) (WRTRYS)=(MAXERR) GOTO $EOP
                        ;               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.
                        ;
                        ;;**************************************************************************

 003132  012737  003132  001106 FORMAT: MOV     #FORMAT,@#$LPADR        ;SETUP SCOPE LOOP ADDRESS
 003140  104401  013246                 TYPE    ,MSG11                  ;"*** FORMAT ***"
 003144  004737  007564                 JSR     PC,@#TPDRV              ;GO TYPE THE DRIVE
 003150  104401  001203                 TYPE    ,$CRLF
 003154  012737                         MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
 003156      000     377                .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
 003160  001306                         .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
 003162  012737                         MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
 003164      000     377                .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
 003166  001304                         .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////////////
 003170  012700  001612                 MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
 003174  004737  006756                 JSR     PC,@#REWIND             ;GO TO REWIND
 003200  105737  001612         1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
 003204  001775                         BEQ     1$
 003206  100022                         BPL     4$                      ;BR IF NO ERROR
 003210  004037  005536                 JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
```

```
                                        JMP     @#$EOP                  ;GO TO END OF PROGRAM
                                ;///////////////////////////////////////////////////////////////
                                ;       "WRITE-FILL-GAP"
                                ;///////////////////////////////////////////////////////////////
                                2$:
                                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
                                        JSR     PC,@#WFG                ;GO TO WFG
                                3$:     TSTB    @#PARMBK                ;WAIT ON FLAG
                                        BEQ     3$
                                        BPL     4$                      ;BR IF NO ERROR
                                        JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
                                        JMP     @#$EOP                  ;GO TO END OF PROGRAM
                                        JMP     @#CKEOTS
                                ;///////////////////////////////////////////////////////////////
                                ;       "WRITE" A BLOCK OF DATA
                                ;///////////////////////////////////////////////////////////////
                                4$:     MOV     #5$,@#$LPADR            ;SETUP SCOPE LOOP ADDRESS
                                5$:     JSR     PC,@#SETUPW             ;GO SETUP FOR A WRITE
                                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
                                        JSR     PC,@#WRITE              ;GO TO WRITE
                                6$:     TSTB    @#PARMBK                ;WAIT ON FLAG
                                        BEQ     6$
                                        BPL     7$                      ;BR IF NO ERROR
                                        JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
                                        JMP     @#$EOP                  ;GO TO END OF PROGRAM
                                        JMP     @#CKEOTS
                                ;///////////////////////////////////////////////////////////////
                                ;       "BACK-SPACE-BLOCK-GAP"
                                ;///////////////////////////////////////////////////////////////
                                7$:
                                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
                                        JSR     PC,@#BSBG               ;GO TO BSBG
                                8$:     TSTB    @#PARMBK                ;WAIT ON FLAG
                                        BEQ     8$
                                        BPL     9$                      ;BR IF NO ERROR
                                        JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
                                        JMP     @#$EOP                  ;GO TO END OF PROGRAM
                                ;///////////////////////////////////////////////////////////////
                                ;       "READ" A BLOCK OF DATA
                                ;///////////////////////////////////////////////////////////////
                                9$:     JSR     PC,@#SETUPR             ;GO SETUP FOR A READ
                                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
                                        JSR     PC,@#READ               ;GO TO READ
                                10$:    TSTB    @#PARMBK                ;WAIT ON FLAG
                                        BEQ     10$
                                        BPL     11$                     ;BR IF NO ERROR
                                        JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
                                        JMP     @#$EOP
                                11$:    SCOPE
                                ;///////////////////////////////////////////////////////////////
                                ;       CHECK FOR SYNC ERROR
                                ;///////////////////////////////////////////////////////////////
                                        JSR     PC,@#SYNCK              ;SYNC ERROR?
                                        BR      12$                     ;RETURN HERE IF NO
                                        ERROR   2                       ;SYNC ERROR
                                        JMP     @#$EOP
```

```
                                ;//////////////////////////////////////////////////////////////////////////////
                                ;       CHECK FOR DATA ERROR IF SOFT REREAD
                                ;//////////////////////////////////////////////////////////////////////////////
003424  004737  006356      12$:  JSR     PC,@#DATCMP             ;CHECK THE DATA
003430  000430                    BR      15$                     ;RETURN HERE IF DATA IS GOOD
003432  104001                    ERROR   1                       ;DATA ERROR
003434  004037  006550            JSR     R0,@#CNTSFT             ;COUNT SOFT ERROR
003440  000730                    BR      7$                      ;GO REREAD
003442  004037  006606            JSR     R0,@#CNTHRD             ;COUNT HARD ERROR
003446  000402                    BR      13$                     ;REWRITE
003450  000137  004412            JMP     @#$EOP                  ;TO MANY HARD ERRORS
                                ;//////////////////////////////////////////////////////////////////////////////
                                ;       IF HARD ERROR REWRITE
                                ;//////////////////////////////////////////////////////////////////////////////
003454  005337  001226      13$:  DEC     @#WRTRYS                ;TRY TO REWRITE THIS BLOCK?
003460  002414                    BLT     15$                     ;BR IF NO
003462  012700  001612            MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003466  004737  006742            JSR     PC,@#BSBG               ;GO TO BSBG
003472  105737  001612      14$:  TSTB    @#PARMBK                ;WAIT ON FLAG
003476  001775                    BEQ     14$
003500  100265                    BPL     4$                      ;GO START A WRITE IF NO ERROR
003502  004037  005536            JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003506  000137  004412            JMP     @#$EOP
003512  013737  001260  001224  15$:  MOV     @#MAXRDS,@#RDTRYS       ;RESET THE REREAD COUNT
003520  013737  001262  001226        MOV     @#MAXERR,@#WRTRYS       ;RESET THE REWRITE COUNT
                                ;//////////////////////////////////////////////////////////////////////////////
                                ;       UPDATE BLOCK # & FILE #
                                ;//////////////////////////////////////////////////////////////////////////////
003526  062737                    ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
003530     001     377            .BYTE   1,-1
003532  001306                    .WORD   BLOCK
003534  123737  001310  001306    CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
003542  003244                    BGT     4$                      ;BR IF NO--GO START A WRITE
003544  062737                    ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
003546     001     377            .BYTE   1,-1
003550  001304                    .WORD   FILE
003552  012737                    MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
003554     000     377            .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003556  001306                    .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
003560  000617                    BR      2$                      ;GO WRITE FILE GAP
```

```
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////

                        .SBTTL          "READ ONLY" ROUTINE

                        ;;*****************************************************************
                        ;;*****************************************************************

                        ;THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE
                        ;
                        ;       1. REWIND TO BOT
                        ;       2. READ BLOCK OF DATA
                        ;       3. CHECK FOR SYNC ERROR
                        ;       4. CHECK FOR DATA ERROR
                        ;               A. DATA ERROR OCCURRED
                        ;                       1.) (RDTRYS)<(MAXRDS)
                        ;                               A.) BACK BLOCK GAP
                        ;                               B.) GOTO 2.
                        ;                       2.) (RDTRYS)=(MAXRDS)
                        ;                               A.) (WRTRYS)<(MAXERR) GOTO 5.
                        ;                               B.) (WRTRYS)=(MAXERR) GOTO $EOP
                        ;               B. NO DATA ERROR --- GOTO 5.
                        ;       5. UPDATE THE BLOCK NUMBER
                        ;       6. END OF FILE?
                        ;               A. NO --- GOTO 2.
                        ;               B. YES
                        ;                       1.) UPDATE FILE NUMBER
                        ;                       2.) RESET BLOCK NUMBER TO 0
                        ;                       3.) SPACE FORWARD FILE GAP
                        ;                       4.) GOTO 2.
                        ;
                        ;*****************************************************************
003562                  READONLY:
003562  012737  003562  001106          MOV     #READONLY,@#$LPADR      ;SETUP SCOPE LOOP ADDRESS
003570  104401  013265                  TYPE    ,MSG12                  ;"*** READ    ***"
003574  004737  007564                  JSR     PC,@#TPDRV              ;GO TYPE DRIVE TO BE TESTED
003600  104401  001203                  TYPE    ,$CRLF
003604  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
003606     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003610  001306                          .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
003612  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
003614     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003616  001304                          .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////////////
003620  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003624  004737  006756                  JSR     PC,@#REWIND             ;GO TO REWIND
003630  105737  001612          1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003634  001775                          BEQ     1$
003636  100004                          BPL     2$                      ;BR IF NO ERROR
003640  004037  005536                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003644  000137  004412                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                        ;////////////////////////////////////////////////////////////////////////////////
```

```
                                        ;       SETUP FOR "READ"
                                        ;////////////////////////////////////////////////////////////////////////////
003650  004737  006220          2$:     JSR     PC,@#SETUPR             ;SETUP FOR "READ"
003654  000417                          BR      5$                      ;GO START A READ
                                        ;////////////////////////////////////////////////////////////////////////////
                                        ;       "BSBG"--IF SCOPE OR SOFT DATA ERROR
                                        ;////////////////////////////////////////////////////////////////////////////
003656  012737  003714  001106  3$:     MOV     #5$,@#$LPADR            ;SETUP THE SCOPE LOOP ADDRESS
003664  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003670  004737  006742                  JSR     PC,@#BSBG               ;GO TO BSBG
003674  105737  001612          4$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003700  001775                          BEQ     4$
003702  100004                          BPL     5$                      ;GO TRY TO REREAD
003704  004037  005536                  JSR     R0,@#CSRERR             ;GO CHECK CSR ERROR
003710  000137  004412                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////////
                                        ;       "READ" BLOCK OF DATA
                                        ;////////////////////////////////////////////////////////////////////////////
003714  012737  003656  001106  5$:     MOV     #3$,@#$LPADR            ;SET THE LOOP ADDRESS
003722  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003726  004737  006732                  JSR     PC,@#READ               ;GO TO READ
003732  105737  001612          6$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003736  001775                          BEQ     6$
003740  100004                          BPL     7$                      ;BR IF NO ERROR
003742  004037  005536                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003746  000137  004412                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////////
                                        ;       CHECK FOR SYNC ERROR
                                        ;////////////////////////////////////////////////////////////////////////////
003752  004737  006254          7$:     JSR     PC,@#SYNCK              ;GO CHECK FOR SYNC ERROR
003756  000403                          BR      8$                      ;RETURN HERE IF NO ERROR
003760  104002                          ERROR   2                       ;SYNC ERROR
003762  000137  004412                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////////
                                        ;       CHECK FOR DATA ERROR
                                        ;////////////////////////////////////////////////////////////////////////////
003766  004737  006356          8$:     JSR     PC,@#DATCMP             ;GO CHECK THE DATA
003772  000411                          BR      9$                      ;RETURN HERE IF NO ERROR
003774  104001                          ERROR   1                       ;DATA ERROR
003776  004037  006550                  JSR     R0,@#CNTSFT             ;COUNT SOFT ERROR
004002  000725                          BR      3$                      ;TRY TO REREAD
004004  004037  006606                  JSR     R0,@#CNTHRD             ;COUNT HARD ERROR
004010  000402                          BR      9$                      ;MOVE TO NEXT BLOCK
004012  000137  004412                  JMP     @#$EOP                  ;TO MANY HARD ERRORS
004016  000004                  9$:     SCOPE
                                        ;////////////////////////////////////////////////////////////////////////////
                                        ;       IF LAST FILE OF LAST BLOCK GOTO CKEOTS
                                        ;////////////////////////////////////////////////////////////////////////////
004020  013737  001260  001224          MOV     @#MAXRDS,@#RDTRYS       ;RESET THE REREAD TRYS
004026  123737  001236  001304          CMPB    @#LASTFL,@#FILE         ;LAST FILE
004034  003007                          BGT     10$                     ;BR IF NO
004036  123737  001234  001306          CMPB    @#LASTBK,@#BLOCK        ;LAST BLOCK
004044  003003                          BGT     10$                     ;BR IF NO
004046  005237  001212                  INC     @#EOTS                  ;COUNT END-OF-TAPE
004052  000544                          BR      CKEOTS                  ;CHECK FOR END-OF-PASS
                                        ;////////////////////////////////////////////////////////////////////////////
```

```
1067                                    ;       UPDATE BLOCK # & FILE #
1068                                    ;///////////////////////////////////////////////////////////////////////////////
1069  004054                            10$:
1070  004054  062737                           ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
1071  004056     001     377                   .BYTE   1,-1
1072  004060  001306                           .WORD   BLOCK
1073  004062  123737  001310  001306           CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
1074  004070  003267                           BGT     2$                      ;BR IF NO
1075  004072  062737                           ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
1076  004074     001     377                   .BYTE   1,-1
1077  004076  001304                           .WORD   FILE
1078  004100  012737                           MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
1079  004102     000     377                   .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
1080  004104  001306                           .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
1081                                    ;///////////////////////////////////////////////////////////////////////////////
1082                                    ;       "SPACE-FORWARD-FILE-GAP"
1083                                    ;///////////////////////////////////////////////////////////////////////////////
1084  004106  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1085  004112  004737  006746                   JSR     PC,@#SFFG               ;GO TO SFFG
1086  004116  105737  001612           11$:    TSTB    @#PARMBK                ;WAIT ON FLAG
1087  004122  001775                           BEQ     11$
1088  004124  100251                           BPL     2$                      ;GO READ
1089  004126  004037  005536                   JSR     R0,@#CSRERR             ;GO CHECK CSR ERROR
1090  004132  000137  004412                   JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
1091
```

```
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////////////////////

                        .SBTTL          "WRITE ONLY" ROUTINE

                        ;;**************************************************************
                        ;;**************************************************************

                        ;THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE
                        ;
                        ;       1. REWIND TO BOT
                        ;       2. WRITE FILE GAP
                        ;       3. WRITE BLOCK OF DATA
                        ;       4. UPDATE THE BLOCK NUMBER
                        ;       5. END OF FILE?
                        ;               A. NO --- GOTO 3.
                        ;               B. YES
                        ;                       1.) UPDATE FILE NUMBER
                        ;                       2.) RESET BLOCK NUMBER TO 0
                        ;                       3.) GOTO 2.
                        ;
                        ;;**************************************************************

004136                  WRITEONLY:
004136  012737  004136  001106          MOV     #WRITEONLY,@#$LPADR     ;SETUP SCOPE LOOP ADDRESS
004144  104401  013304                  TYPE    ,MSG13                  ;"*** WRITE  ***"
004150  004737  007564                  JSR     PC,@#TPDRV              ;TYPE DRIVE TO BE TESTED
004154  104401  001203                  TYPE    ,$CRLF
004160  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
004162     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
004164  001306                          .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
004166  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
004170     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
004172  001304                          .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////////////
004174  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
004200  004737  006756                  JSR     PC,@#REWIND             ;GO TO REWIND
004204  105737  001612          1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
004210  001775                          BEQ     1$
004212  100022                          BPL     4$                      ;BR IF NO ERROR
004214  004037  005536                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
004220  000137  004412                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;       "WRITE-FILE-GAP"
                        ;////////////////////////////////////////////////////////////////////////////////
004224                  2$:
004224  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
004230  004737  006722                  JSR     PC,@#WFG                ;GO TO WFG
004234  105737  001612          3$:     TSTB    @#PARMBK                ;WAIT ON FLAG
004240  001775                          BEQ     3$
004242  100006                          BPL     4$                      ;BR IF NO ERROR
004244  004037  005536                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
```

```
1148  004250  000137  004412                JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
1149  004254  000137  004364                JMP     @#CKEOTS
1150                                  ;////////////////////////////////////////////////////////////////
1151                                  ;       "WRITE" A BLOCK OF DATA
1152                                  ;////////////////////////////////////////////////////////////////
1153  004260  004737  006072          4$:   JSR     PC,@#SETUPW             ;GO SETUP FOR "WRITE"
1154  004264  012737  004272  001106        MOV     #5$,@#$LPADR            ;SETUP SCOPE LOOP ADDRESS
1155  004272                          5$:
1156  004272  012700  001612                MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1157  004276  004737  006726                JSR     PC,@#WRITE              ;GO TO WRITE
1158  004302  105737  001612          6$:   TSTB    @#PARMBK                ;WAIT ON FLAG
1159  004306  001775                        BEQ     6$
1160  004310  100006                        BPL     7$                      ;BR IF NO ERROR
1161  004312  004037  005536                JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
1162  004316  000137  004412                JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
1163  004322  000137  004364                JMP     @#CKEOTS
1164  004326  000004                   7$:  SCOPE
1165                                  ;////////////////////////////////////////////////////////////////
1166                                  ;       UPDATE BLOCK # & FILE #
1167                                  ;////////////////////////////////////////////////////////////////
1168  004330  062737                        ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
1169  004332     001     377                .BYTE   1,-1
1170  004334  001306                        .WORD   BLOCK
1171  004336  123737  001310  001306        CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
1172  004344  003345                        BGT     4$                      ;BR IF NO
1173  004346  062737                        ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
1174  004350     001     377                .BYTE   1,-1
1175  004352  001304                        .WORD   FILE
1176  004354  012737                        MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
1177  004356     000     377                .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
1178  004360  001306                        .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
1179  004362  000720                        BR      2$                      ;START A FILE GAP
```

```
1180                                    ;////////////////////////////////////////////////////////////////////////////////
1181                                    ;////////////////////////////////////////////////////////////////////////////////
1182
1183                                    .SBTTL          CHECK "EOTS"
1184
1185                                    ;;**************************************************************
1186                                    ;;**************************************************************
1187
1188                                    ;WHEN A "FORMAT" OR  "WRITEONLY" PASS HITS "EOT" OR A
1189                                    ;"READONLY" PASS READS THE LAST BLOCK OF THE LAST FILE
1190                                    ;THEY WILL COME HERE
1191                                    ;
1192                                    ;       1. (EOTS) IS CHECKED
1193                                    ;               A. (EOTS)=(MAXEOT) GOTO $EOP
1194                                    ;               B. (EOTS)<(MAXEOT) GOTO 2
1195                                    ;       2. SWR<8> = 1?
1196                                    ;               A. YES -- GOTO $EOP
1197                                    ;               B. NO  -- GOTO LOOPER
1198                                    ;
1199                                    ;**************************************************************
1200  004364  000004                    CKEOTS: SCOPE
1201  004366  023737  001212  001264            CMP     @#EOTS,@#MAXEOT         ;MAX. EOTS OCCURRED?
1202  004374  002006                            BGE     $EOP                    ;BR IF YES
1203  004376  032777  000400  174534            BIT     #SWO8,@SWR              ;IF SWR<08> = 1
1204  004404  001002                            BNE     $EOP                    ;GOTO END-OF-TEST
1205  004406  000137  002774                    JMP     @#LOOPER                ;NO--LOOP
1206
```

```
1207                                    .SBTTL  END OF PASS ROUTINE
1208
1209                                    ;;**********************************************************************
1210                                    ;*INCREMENT THE PASS NUMBER ($PASS)
1211                                    ;*TYPE "END PASS"
1212                                    ;*IF THERES A MONITOR GO TO IT
1213                                    ;*IF THERE ISN'T JUMP TO START
1214                                    ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
1215                                    ;*$ENDMG CAN BE CHANGED TO 7.
1216
1217  004412                            $EOP:
1218  004412  000004                           SCOPE
1219  004414  005067  174462                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
1220  004420  005267  174454                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
1221  004424  042767  100000  174446          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
1222  004432  005327                           DEC     (PC)+           ;;LOOP?
1223  004434  000001                    $EOPCT: .WORD   1
1224  004436  003137                           BGT     $DOAGN          ;;YES
1225  004440  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
1226  004442  000001                    $ENDCT: .WORD   1
1227  004444  004434                           $EOPCT
1228  004446  104401  004745                   TYPE    ,$ENDMG         ;;TYPE "END PASS"
1229  004452  104401  004742                   TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
1230  004456  013700  000042            $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
1231  004462  001525                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
1232  004464  000004                           SCOPE
1233                                  ;////////////////////////////////////////////////////////////////////////
1234                                  ;       TYPE END-OF-TEST STATISTICS
1235                                  ;////////////////////////////////////////////////////////////////////////
1236  004466  012700  001612                   MOV     #PARMBK,R0      ;R0=PARAMETER BLOCK ADDRESS
1237  004472  004737  006756                   JSR     PC,@#REWIND     ;START A REWIND
1238  004476  104401  013031                   TYPE    ,MSG1           ;<CRLF>"*** END-OF-TEST ***"
1239
1240  004502  104401  013056                   TYPE    ,MSG2           ;<CRLF>"SOFT ERRORS="
1241  004506  013746  001206                   MOV     @#SOFTNM,-(SP)  ;PICKUP SINGLE PRECISION BINARY NUMBER
1242  004512  004737  007366                   JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
1243  004516  004737  007422                   JSR     PC,@#$SUPRS     ;TYPE  WITHOUT LEADING ZEROS
1244
1245  004522  104401  013074                   TYPE    ,MSG3           ;<CRLF>"HARD ERRORS="
1246  004526  013746  001210                   MOV     @#HARDNM,-(SP)  ;PICKUP SINGLE PRECISION BINARY NUMBER
1247  004532  004737  007366                   JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
1248  004536  004737  007422                   JSR     PC,@#$SUPRS     ;TYPE  WITHOUT LEADING ZEROS
1249
1250  004542  104401  013112                   TYPE    ,MSG4           ;<CRLF>"BYTES READ="
1251  004546  012746  001214                   MOV     #RBYTTL,-(SP)   ;GET ADDRESS OF DOUBLE PRECISION BINARY #
1252  004552  004737  007172                   JSR     PC,@#$DB2D      ;CHANGE IT TO DECIMAL ASCIZ
1253  004556  004737  007422                   JSR     PC,@#$SUPRS     ;TYPE IT WITHOUT LEADING ZERO
1254
1255  004562  104401  013127                   TYPE    ,MSG5           ;<CRLF>"BYTES WRITTEN="
1256  004566  012746  001220                   MOV     #WBYTTL,-(SP)   ;GET ADDRESS OF DOUBLE PRECISION BINARY #
1257  004572  004737  007172                   JSR     PC,@#$DB2D      ;CHANGE IT TO DECIMAL ASCIZ
1258  004576  004737  007422                   JSR     PC,@#$SUPRS     ;TYPE IT WITHOUT LEADING ZERO
1259
1260  004602  104401  013147                   TYPE    ,MSG6           ;<CRLF>"TAPE PASSES="
1261  004606  013746  001212                   MOV     @#EOTS,-(SP)    ;PICKUP SINGLE PRECISION BINARY NUMBER
1262  004612  004737  007366                   JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
```

```
1263  004616  004737  007422                  JSR     PC,@#$SUPRS             ;TYPE  WITHOUT LEADING ZEROS
1264
1265  004622  104401  013165                  TYPE    ,MSG7                   ;<CRLF>"FILES/PASS="
1266  004626  013700  001236                  MOV     @#LASTFL,R0             ;PICKUP THE LAST FILE NUMBER
1267  004632  105200                          INCB    R0                      ;ADD 1 TO MAKE IT # OF FILES
1268  004634  010046                          MOV     R0,-(SP)                ;PUT IT ON THE STACK
1269  004636  004737  007366                  JSR     PC,@#$B2D               ;CHANGE IT TO DECIMAL ASCIZ
1270  004642  004737  007422                  JSR     PC,@#$SUPRS             ;TYPE IT WITHOUT LEADING ZEROS
1271
1272  004646  104401  013202                  TYPE    ,MSG8                   ;<CRLF>"BLOCKS/FILE="
1273  004652  013746  001310                  MOV     @#FILESZ,-(SP)          ;PICKUP SINGLE PRECISION BINARY NUMBER
1274  004656  004737  007366                  JSR     PC,@#$B2D               ;CHANGE IT TO DECIMAL ASCIZ
1275  004662  004737  007422                  JSR     PC,@#$SUPRS             ;TYPE  WITHOUT LEADING ZEROS
1276
1277  004666  104401  001204                  TYPE    ,$LF
1278                                  ;////////////////////////////////////////////////////////////////
1279                                  ;       CHECK HALT AT END-OF-TEST SWITCH
1280                                  ;////////////////////////////////////////////////////////////////
1281  004672  032777  001000  174240          BIT     #SW09,@SWR              ;HALT AT END-OF-TEST?
1282  004700  001406                          BEQ     100$                    ;BR IF NO
1283  004702  000000                          HALT                            ;YES
1284
1285  004704  022767  000176  174226          CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
1286  004712  001001                          BNE     20$                     ;NO- GET OUT
1287  004714  104405                          GTSWR                           ;GET VALUE
1288  004716                          20$:                                    ;CONTINUE
1289  004716                          100$:
1290  004716  013700  000042                  MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
1291  004722  001405                          BEQ     $DOAGN          ;;RETURN ADDRESS
1292  004724  000005                          RESET                   ;;CLEAR THE WORLD
1293  004726  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
1294  004730  000240                          NOP                     ;;SAVE ROOM
1295  004732  000240                          NOP                     ;;FOR
1296  004734  000240                          NOP                     ;;ACT11
1297  004736                          $DOAGN:
1298  004736  000137                          JMP     @(PC)+          ;;RETURN
1299  004740  002630                  $RTNAD: .WORD   START
1300  004742     377        377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
1301  004745     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS/
1302  004752  050040  051501  000123
```

```
                                .SBTTL  SCOPE HANDLER ROUTINE

                                ;;**********************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW14=1         LOOP ON TEST
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT

004760                          $SCOPE:
004760  104406                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
004762  032777  040000  174150  1$:    BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
004770  001025                         BNE     $OVER           ;;YES IF SW14=1
                                ;*****START OF CODE FOR THE XOR TESTER*****
004772  000416                $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
004774  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
005000  012737  005020  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
005006  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
005012  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
005016  000404                         BR      $SVLAD          ;;GO TO THE NEXT TEST
005020  022626                 5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
005022  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
005026  000406                         BR      $OVER           ;;LOOP ON THE PRESENT TEST
005030                          6$:;*****END OF CODE FOR THE XOR TESTER*****
005030  105267  174046         $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
005034  011667  174046                 MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
005040  105067  174037                 CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
005044  016777  174032  174070  $OVER: MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
005052  016716  174030                 MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
005056  000002                         RTI                     ;;FIXES PS
```

```
                                .SBTTL  ERROR HANDLER ROUTINE

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

005060                          $ERROR:
005060  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
005062  105267  174015          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
005066  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
005070  016777  174006  174044          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
005076  032777  002000  174034          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
005104  001402                          BEQ     1$              ;;NO - SKIP
005106  104401  005204                  TYPE    ,$BELL          ;;RING BELL
005112  005267  173774          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
005116  011667  173774                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
005122  162767  000002  173766          SUB     #2,$ERRPC
005130  117767  173762  173756          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
005136  032777  020000  173774          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
005144  001004                          BNE     20$             ;;SKIP TYPEOUTS
005146  004767  000036                  JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
005152  104401  001203                  TYPE    ,$CRLF
005156                          20$:
005156  005777  173756          2$:     TST     @SWR            ;;HALT ON ERROR
005162  100002                          BPL     3$              ;;SKIP IF CONTINUE
005164  000000                          HALT                    ;;HALT ON ERROR!
005166  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
005170                          3$:
005170  022737  004726  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
005176  001001                          BNE     6$              ;;BRANCH IF NO
005200  000000                          HALT                    ;;YES
005202                          6$:
005202  000002                          RTI                     ;;RETURN
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    TYPERR: TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
1380  005214  010046                    MOV     R0,-(SP)                ;SAVE R0
1381  005216  010146                    MOV     R1,-(SP)                ;SAVE R1
1382  005220  005046                    CLR     -(SP)                   ;PICKUP THE ITEM BYTE
1383  005222  113716  001114            MOVB    @#$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  000001            MOVB    1(SP),R1                ;GET THE ERROR TYPE
1389  005242  112600                    MOVB    (SP)+,R0                ;GET THE ERROR NUMBER
1390  005244  066100  005252            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)+,3$                ;PICKUP "ERROR MESSAGE" POINTER
1397  005264  104401                    TYPE                            ;TYPE "ERROR MESSAGE"
1398  005266  000000             3$:    0
1399  005270  104401  001203            TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
1400  005274  012067  000002            MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
1401  005300  104401                    TYPE                            ;TYPE "DATA HEADER"
1402  005302  000000             4$:    0
1403  005304  104401  001203            TYPE    ,$CRLF                  ;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)+,R1                ;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                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1421  005350  000767                    BR      1$                      ;GO GET NEXT DATA WORD
1422  005352  010046             3$:    MOV R0,-(SP)
1423  005354  004737  007172            JSR     PC,@#$DB2D              ;CHANGE NUMBER TO ASCIZ
1424  005360  062716  000004            ADD     #4,(SP)                 ;TYPE 6 DIGITS
1425  005364  012667  000002            MOV     (SP)+,4$                ;
1426  005370  104401                    TYPE                            ;CALL TYPE ASCIZ MESSAGE ROUTINE
1427  005372  000000             4$:    .WORD   0                       ;ADDRESS OF ASCIZ STRING
1428  005374  000755                    BR      1$                      ;GO GET NEXT DATA WORD
```

```
1429
1430  005376  011001                ;////////////////////////////////////////////////////////////////////
1431  005400  013146          ERROR1: MOV     (R0),R1                 ;PICKUP THE "DATA POINTER"
1432                                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1433  005402  104402                  TYPOC                   ;;$ERRPC
1434  005404  012767  000003  000006        MOV     #3,2$           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1435                                                                  ;PRINT 2 SPACES 3 TIMES
1436  005412  104401  013323          1$:     TYPE    ,MSG14          ;AND 2 DECIMAL #'S 2 TIMES
1437  005416  005327                  DEC     (PC)+                   ;"  "
1438  005420  000000          2$:     .WORD   0                       ;TYPE ANOTHER # ?
1439  005422  001412                  BEQ     4$                      ;BR IF NO
1440  005424  012146                  MOV     (R1)+,-(SP)             ;YES---PICKUP ADDRESS OF #
1441  005426  004737  007172          JSR     PC,@#$DB2D              ;GO CHANGE TO DECIMAL ASCIZ
1442  005432  062716  000004          ADD     #4,(SP)                 ;ONLY TYPE THE LAST 6 DIGITS
1443  005436  012667  000002          MOV     (SP)+,3$                ;SAVE ADDRESS OF ASCIZ STRING
1444  005442  104401                  TYPE                            ;CALL THE TYPE ROUTINE
1445  005444  000000          3$:     .WORD   0                       ;POINTER GOES HERE
1446  005446  000761                  BR      1$                      ;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$:     .WORD   0                       ;MESSAGE POINTER GOES HERE
1451  005464  000421                  BR      EREXT2                  ;GO TO EXIT
1452                          ;THIS TABLE CONTAINS THE POINTERS TO THE DIFFERENT ASCIZ MESSAGES
1453                          ;FOR THE FUNCTION BEING PERFORMED WHEN THE ERROR OCCURRED.
1454  005466  012067          BADFUN: MXWFG           ;       WRITE-FILE-GAP
1455  005470  012106                  MXWRIT          ;       WRITE
1456  005472  012114                  MXREAD          ;       READ
1457  005474  012121                  MXBSFG          ;       BACK-SPACE-FILE-GAP
1458  005476  012145                  MXBSBG          ;       BACK-SPACE-BLK-GAP
1459  005500  012170                  MXSFFG          ;       SPACE-FWD-FILE-GAP
1460  005502  012213                  MXSFBG          ;       SPACE-FWD-BLK-GAP
1461  005504  012235                  MXRWND          ;       REWIND
1462
1463                          ;////////////////////////////////////////////////////////////////////
1464  005506  011000          ERROR2: MOV     (R0),R0                 ;PICKUP THE DATA POINTER
1465  005510  005710          1$:     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                  TYPOC                   ;;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)+,R1                ;RESTORE R1
1475  005532  012600                  MOV     (SP)+,R0                ;RESTORE R0
1476  005534  000207                  RTS     PC                      ;RETURN
1477
1478
```

```
1479                                    ;////////////////////////////////////////////////////////////////////////////
1480                                    ;////////////////////////////////////////////////////////////////////////////
1481                                    .SBTTL          DETERMINE CONTROL AND STATUS ERROR
1482
1483  005536  113737  001304  001162  CSRERR: MOVB    @#FILE,@#$REG0          ;GET THE FILE NUMBER
1484  005544  113737  001306  001166          MOVB    @#BLOCK,@#$REG2         ;GET THE BLOCK NUMBER
1485  005552  013746  007170                  MOV     @#FUNLOC,-(SP)          ;GET THE LAST FUNCTION
1486  005556  042716  177761                  BIC     #^CFUNCTION,(SP)        ;CLEAR AWAY THE JUNK
1487  005562  011637  001172                  MOV     (SP),@#$TMP0            ;SAVE THE FUNCTION CODE
1488  005566  113766  001612  000001          MOVB    @#PARMBK,1(SP)          ;COMBINE THE ST/ER WITH
1489  005574  012601                          MOV     (SP)+,R1                ;THE FUNCTION
1490  005576  005046                          CLR     -(SP)                   ;FIND OUT WHAT CAUSED THE ERROR
1491  005600  032701  001000                  BIT     #OFFLINE,R1             ;OFF LINE?
1492  005604  001025                          BNE     6$                      ;BR IF YES
1493  005606  032701  010000                  BIT     #WRTLOCK,R1             ;WRITE LOCK?
1494  005612  001021                          BNE     5$                      ;BR IF YES
1495  005614  032701  020000                  BIT     #LEADER,R1              ;BOT/EOT?
1496  005620  001015                          BNE     4$                      ;BR IF YES
1497  005622  032701  004000                  BIT     #FGAP,R1                ;FILE GAP?
1498  005626  001011                          BNE     3$                      ;BR IF YES
1499  005630  032701  002000                  BIT     #TIMERR,R1              ;TIMING?
1500  005634  001005                          BNE     2$                      ;BR IF YES
1501  005636  032701  040000                  BIT     #CRCERR,R1              ;BLOCK CHECK?
1502  005642  001001                          BNE     1$                      ;BR IF YES
1503  005644  005216                          INC     (SP)                    ; 6 = UNKNOWN
1504  005646  005216                  1$:     INC     (SP)                    ; 5 = CRCERR
1505  005650  005216                  2$:     INC     (SP)                    ; 4 = TIMERR
1506  005652  005216                  3$:     INC     (SP)                    ; 3 = FGAP
1507  005654  005216                  4$:     INC     (SP)                    ; 2 = LEADER
1508  005656  005216                  5$:     INC     (SP)                    ; 1 = WRTLOCK
1509  005660                          6$:
1510  005660  106316                          ASLB    (SP)                    ;POSITION INDEX
1511  005662  062607                          ADD     (SP)+,PC                ;BR TO THE ROUTINE
1512  005664  000406                          BR      OL.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     #XWFG,@#$TMPO           ;FUNCTION "WRITE FILE GAP"?
1527  005720  001014                          BNE     2$                      ;BR IF NO
1528  005722                          1$:
1529  005722  162737                          SUB     (PC)+,@(PC)+            ;DECREMENT THE FILE NUMBER
1530  005724     001     377                  .BYTE   1,-1
1531  005726  001304                          .WORD   FILE
1532  005730  113737  001304  001236          MOVB    @#FILE,@#LASTFL         ;SAVE LAST FILE NUMBER
1533  005736  113737  001310  001234          MOVB    @#FILESZ,@#LASTBK       ;AND LAST BLOCK NUMBER
1534  005744  005337  001234                  DEC     @#LASTBK
1535  005750  000425                          BR      3$                      ;BR IF NO
1536  005752  022737  000002  001172  2$:     CMP     #XWRITE,@#$TMPO         ;FUNCTION "WRITE"?
1537  005760  001025                          BNE     4$                      ;BR IF NO
1538  005762  113737  001304  001236          MOVB    @#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          MOVB    @#BLOCK,@#LASTBK        ;SAVE LAST BLOCK NUMBER
1543  006004  163737  007164  001220          SUB     @#CBCNT,@#WBYTTL        ;SUBTRACT RESIDUE COUNT FROM
1544  006012  005637  001222                  SBC     @#WBYTTL+2              ;       THE TOTAL WRITE COUNT
1545  006016  005137  001306                  COM     @#BLOCK                 ;IF NO BLOCKS IN THIS FILE
1546  006022  001737                          BEQ     1$                      ;BACK TO LAST FILE
1547  006024  022020                  3$:     CMP     (R0)+,(R0)+             ;ADJUST FOR RETURN
1548  006026  005237  001212                  INC     @#EOTS                  ;COUNT END-OF-TAPE
1549  006032  000401                          BR      5$
1550  006034  104103                  4$:     ERROR   103                     ;"CLEAR LEADER" ERROR
1551  006036  000200                  5$:     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,@#$TMPO          ;IF FUNCTION ISN'T A READ
1561  006060  001001                          BNE     1$                      ;TAKE THE "EOP" EXIT
1562  006062  022020                          CMP     (R0)+,(R0)+             ;ADJUST FOR RETURN
1563  006064  000200                  1$:     RTS     R0
1564
1565  006066  104107                  X.ERR:  ERROR   107                     ;UNKNOWN INTERRUPT
1566  006070  000200                          RTS     R0
```

```
1567                                    ;;*****************************************************************
1568                                    ;;*****************************************************************
1569                                    .SBTTL          ROUTINE TO SETUP FOR A WRITE OPERATION
1570
1571                                    ;THIS ROUTINE WILL SETUP THE BYTE COUNT, DETERMINE THE PATTERN TO
1572                                    ;BE WRITTEN ON TAPE, ADD THE SIZE OF THIS BLOCK TO THE TOTAL
1573                                    ;NUMBER OF BYTES TRANSFERRED AND CALL THE ROUTINE TO FILL THE
1574                                    ;WRITE BUFFER.
1575
1576                                    ;CALL
1577                                    ;       JSR     PC,@#SETUPW
1578
1579
1580  006072  013700  001306           SETUPW: MOV     @#BLOCK,R0              ;GET THE BLOCK NUMBER
1581  006076  042700  177760                   BIC     #^C15.,R0               ;KEEP MAX PATTERN
1582  006102  006300                           ASL     R0                      ;*2
1583  006104  066037  001312  001220           ADD     BLKSZ(R0),@#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(R0),@#PARMBK+6    ;SET THE BYTE COUNT
1586  006124  016000  001352                   MOV     PATS(R0),R0             ;GET PATTERN POINTER
1587  006130  004737  006136                   JSR     PC,@#FILL               ;GO FILL WRITE BUFFER
1588  006134  000207                           RTS     PC                      ;RETURN
1589
1590                                    ;;*****************************************************************
1591                                    .SBTTL          ROUTINE TO FILL THE BUFFER BEFORE A WRITE
1592                                    ;CALL
1593                                    ;       MOV     #PATADR,R0              ;1ST ADDRESS OF 8 BYTE PATADR
1594                                    ;       JSR     PC,@#FILL
1595                                    ;THE BUFFER IS FILLED AS FOLLOWS:
1596                                    ;
1597                                    ;BYTE#  DATA
1598                                    ;
1599                                    ;0001   FILE NUMBER
1600                                    ;0002   FILE NUMBER COMPLEMENTED
1601                                    ;0003   BLOCK NUMBER
1602                                    ;0004   BLOCK NUMBER COMPLEMENTED
1603                                    ;0005   DATA PATTERN
1604                                    ;0006   DATA PATTERN
1605                                    ; *      *       *
1606                                    ; *      *       *
1607                                    ; *      *       *
1608                                    ;1027.  DATA PATTERN
1609                                    ;1028.  DATA PATTERN
1610
1611  006136                            FILL:
1612  006136  010046                           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     @#BLOCK,(R2)+           ;BLOCK NUMBER AND ITS COMPLEMENT
1620  006166  010003                           MOV     R0,R3                   ;SAVE FIRST ADDRESS OF PATTERN
1621  006170  010300                   1$:     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     1$
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                                   .SBTTL          ROUTINE TO SETUP FOR A READ
1637
1638                                   ;THIS ROUTINE WILL SETUP THE BYTE COUNT AND ADD THE SIZE OF THIS
1639                                   ;BLOCK TO THE TOTAL NUMBER OF BYTES READ.
1640
1641                                   ;CALL
1642                                   ;       JSR     PC,SETUPR
1643
1644
1645   006220  013700  001306          SETUPR: MOV     @#BLOCK,R0              ;GET THE BLOCK NUMBER
1646   006224  042700  177760                  BIC     #^C15.,R0               ;KEEP MAX PATTERN
1647   006230  006300                          ASL     R0                      ;X2
1648   006232  066037  001312  001214          ADD     BLKSZ(R0),@#RBYTTL      ;ADD THE BLOCK SIZE TO THE TOTAL
1649   006240  005537  001216                  ADC     @#RBYTTL+2              ;NUMBER OF BYTES READ
1650   006244  016037  001312  001620          MOV     BLKSZ(R0),@#PARMBK+6    ;SET THE BYTE COUNT
1651   006252  000207                          RTS     PC                      ;RETURN
1652
1653                                   ;;*****************************************************************
1654                                   .SBTTL          ROUTINE TO CHECK FOR SYNC PROBLEMS
1655
1656                                   ;THIS ROUTINE CHECKS THE FIRST FOUR BYTES OF THE DATA
1657                                   ;BUFFER TO INSURE THE PROPER BLOCK WAS READ
1658                                   ;BYTE 1 = FILE NUMBER
1659                                   ;BYTE 2 = FILE NUMBER COMPLEMENT
1660                                   ;BYTE 3 = BLOCK NUMBER
1661                                   ;BYTE 4 = BLOCK NUMBER COMPLEMENT
1662
1663                                   ;CALL
1664                                   ;       JSR     PC,@#SYNCK
1665                                   ;       NORMAL RETURN
1666                                   ;       ERROR RETURN
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     2$                      ;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     2$                      ;BR IF BLOCK NUMBER READ WRONG
1681   006304  023740  001306                  CMP     @#BLOCK,-(R0)           ;MAKE SURE PROPER BLOCK WAS READ
1682   006310  001003                          BNE     1$                      ;BR IF AT THE WRONG BLOCK
1683   006312  023740  001304                  CMP     @#FILE,-(R0)            ;MAKE SURE PROPER FILE
1684   006316  001416                          BEQ     2$                      ;BR IF IN THE PROPER FILE
1685   006320  113737  001304  001162  1$:     MOVB    @#FILE,@#$REG0
1686   006326  113737  001306  001166          MOVB    @#BLOCK,@#$REG2
1687   006334  113737  013464  001172          MOVB    @#BUFFER,@#$TMP0
1688   006342  113737  013466  001176          MOVB    @#BUFFER+2,@#$TMP2
1689   006350  062716  000002                  ADD     #2,(SP)                 ;TAKE ERROR RETURN
1690   006354  000207                  2$:     RTS     PC                      ;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     @#BLOCK,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                  1$:     DEC     R1                      ;COUNT THIS BYTE
1709   006412  122224                          CMPB    (R2)+,(R4)+             ;CHECK IT
1710   006414  001015                          BNE     4$                      ;BR IF BAD
1711   006416  005303                          DEC     R3                      ;AGAIN?
1712   006420  001373                          BNE     1$                      ;BR IF YES
1713   006422  012703  000010          2$:     MOV     #8.,R3                  ;GET NUMBER OF BYTES/PATTERN
1714   006426  016004  001352                  MOV     PAT$(R0),R4             ;GET FIRST ADDRESS OF PATTERN
1715   006432  005301                  3$:     DEC     R1                      ;LAST BYTE BEEN CHECKED?
1716   006434  002437                          BLT     5$                      ;BR IF YES
1717   006436  122224                          CMPB    (R2)+,(R4)+             ;NO--CHECK IT
1718   006440  001003                          BNE     4$                      ;BR IF BAD
1719   006442  005303                          DEC     R3                      ;END OF PATTERN?
1720   006444  003372                          BGT     3$                      ;BR IF NO
1721   006446  000765                          BR      2$                      ;YES
1722   006450  005037  001124          4$:     CLR     @#$GDDAT
1723   006454  005037  001126                  CLR     @#$BDDAT
1724   006460  114437  001124                  MOVB    -(R4),@#$GDDAT          ;SAVE GOOD DATA
1725   006464  010437  001120                  MOV     R4,@#$GDADR             ;SAVE GOOD ADDRESS
1726   006470  114237  001126                  MOVB    -(R2),@#$BDDAT          ;SAVE BAD DATA
1727   006474  010237  001122                  MOV     R2,@#$BDADR             ;SAVE BAD ADDRESS
1728   006500  005401                          NEG     R1                      ;SAVE THE BYTE NUMBER
1729   006502  066001  001312                  ADD     BLKSZ(R0),R1
1730   006506  010137  001230                  MOV     R1,@#BYTNUM
1731   006512  113737  001304  001162          MOVB    @#FILE,@#$REG0
1732   006520  113737  001306  001166          MOVB    @#BLOCK,@#$REG2
1733   006526  062716  000002                  ADD     #2,(SP)                 ;SETUP TO TAKE ERROR1 EXIT
1734   006532  000405                          BR      6$                      ;TAKE ERROR1 EXIT
```

```
1735  006534  105737  001612          5$:     TSTB    @#PARMBK                ;DATA IS GOOD BUT WHAT ABOUT
1736                                                                            ;THE ERROR FLAG
1737  006540  100002                          BPL     6$                      ;BR IF ERROR FLAG=0
1738  006542  062716  000004                  ADD     #4,(SP)                 ;SETUP TO TAKE ERROR2 EXIT
1739  006546  000207                  6$:     RTS     PC                      ;RETURN
1740
1741                                  ;;**********************************************************************
1742
1743                                  .SBTTL          COUNT SOFT DATA ERROR
1744
1745                                  ;CALL
1746                                  ;       JSR     R0,@#CNTSFT
1747                                  ;       RETURN  FOR SOFT RETRY
1748                                  ;       RETURN  FOR HARD
1749                                  ;
1750  006550  005237  001206          CNTSFT: INC     @#SOFTNM                ;COUNT THIS SOFT ERROR
1751  006554  005337  001224                  DEC     @#RDTRYS                ;REREAD THE RECORD?
1752  006560  002011                          BGE     1$                      ;BR IF YES
1753  006562  013737  001260  001224          MOV     @#MAXRDS,@#RDTRYS       ;RESET REREAD COUNT
1754  006570  163737  001260  001206          SUB     @#MAXRDS,@#SOFTNM       ;DON'T COUNT AS SOFT
1755  006576  005337  001206                  DEC     @#SOFTNM
1756  006602  005720                          TST     (R0)+                   ;STEP OVER THE REREAD RETURN
1757  006604  000200                  1$:     RTS     R0                      ;RETURN
1758                                  ;;**********************************************************************
1759
1760                                  .SBTTL          COUNT HARD ERROR
1761
1762                                  ;CAL
1763                                  ;       JSR     R0,@#CNTHRD
1764                                  ;       RETURN  MORE ERRORS ALLOWED
1765                                  ;       RETURN  MAX. ERRORS HAVE OCCURRED
1766                                  ;
1767  006606  005237  001210          CNTHRD: INC     @#HARDNM                ;COUNT THIS HARD ERROR
1768  006612  023737  001210  001262          CMP     @#HARDNM,@#MAXERR       ;MAX. HARD ERRORS OCCURRED
1769  006620  002401                          BLT     1$                      ;BR IF NO
1770  006622  005720                          TST     (R0)+                   ;TAKE FATAL EXIT
1771  006624  000200                  1$:     RTS     R0                      ;RETURN
```

```
                        ;;*****************************************************************

                        .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES

                        ;;*****************************************************************
                        ;*SAVE R0-R5
                        ;*CALL:
                        ;*      SAVREG
                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                        ;*
                        ;*TOP---(+16)
                        ;* +2---(+18)
                        ;* +4---R5
                        ;* +6---R4
                        ;* +8---R3
                        ;*+10---R2
                        ;*+12---R1
                        ;*+14---R0

006626                  $SAVREG:
006626  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006630  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006632  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006634  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006636  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
006640  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006642  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
006646  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
006652  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
006656  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
006662  000002                  RTI

                        ;*RESTORE R0-R5
                        ;*CALL:
                        ;*      RESREG

006664                  $RESREG:
006664  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
006670  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
006674  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
006700  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
006704  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
006706  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
006710  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
006712  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
006714  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
006716  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
006720  000002                  RTI
```

```
                                ;;*****************************************************************
                                .SBTTL          CASSETTE PRIMITIVES ROUTINE
                                ;CALLED VIA     JSR  PC,(FUNCTION)
                                ;WITH R0 POINTING TO:
                                ;
                                ;       .BYTE   0
                                ;       .BYTE   DRIVE # (0=A,1=B)
                                ;       .WORD   POINTER TO STATUS/ERROR BYTE
                                ;       .WORD   BUFFER ADDRESS
                                ;       .WORD   BYTE COUNT
                                ;
006722  010746                  WFG:    MOV     PC,-(SP)        ; WRITE A FILE GAP
006724  000415                  FUNTAB: BR      IOCOM
006726  010746                  WRITE:  MOV     PC,-(SP)        ; WRITE A BLOCK OF DATA
006730  000413                          BR      IOCOM
006732  010746                  READ:   MOV     PC,-(SP)        ; READ A BLOCK OF DATA
006734  000411                          BR      IOCOM
006736  010746                  BSFG:   MOV     PC,-(SP)        ; BACK SPACE A FILE GAP
006740  000407                          BR      IOCOM
006742  010746                  BSBG:   MOV     PC,-(SP)        ; BACK SPACE A BLOCK GAP
006744  000405                          BR      IOCOM
006746  010746                  SFFG:   MOV     PC,-(SP)        ; SPACE FORWARD A FILE GAP
006750  000403                          BR      IOCOM
006752  010746                  SFBG:   MOV     PC,-(SP)        ; SPACE FORWARD A BLOCK GAP
006754  000401                          BR      IOCOM
006756  010746                  REWIND: MOV     PC,-(SP)        ; REWIND TO BEGINNING OF TAPE
006760  162716  006724          IOCOM:  SUB     #FUNTAB,(SP)    ;PC ON STACK IS USED TO DETERMINE
006764  006216                          ASR     (SP)            ;  FUNCTION
006766  032777  000040  172244  CHKRDY: BIT     #40,@TACSL      ;READY BIT UP?
006774  001774                          BEQ     CHKRDY          ;WAIT ON IT
006776  012667  000166                  MOV     (SP)+,FUNLOC    ;PUT FUNC. IN BITS 1-3 OF FUNLOC
007002  010146                          MOV     R1,-(SP)
007004  010046                          MOV     R0,-(SP)
007006  012701  007162                  MOV     #CASPAR,R1      ;PT. TO INT. HANDLER PARAM. BLK.
007012  016011  000002                  MOV     2(R0),(R1)      ;PTR. TO ST/ERR WD. INTO P.BLK.
007016  105031                          CLRB    @(R1)+          ;ZERO ST/ERR BYTE
007020  005200                          INC     R0
007022  152067  000143          CHANOK: BISB    (R0)+,FUNLOC+1  ;COPY CHAN. NUM. TO FUNC. WORD
007026  005720                          TST     (R0)+
007030  012061  000002                  MOV     (R0)+,2(R1)     ;BUFF. ADR.
007034  011021                          MOV     (R0),(R1)+      ;BYTE COUNT
007036  005721                          TST     (R1)+
007040  152711  000101                  BISB    #101,(R1)       ;ADD INT. ENABLE AND GO BITS
                                                                ;TO THE FUNCTION
007044  011177  172170          DOFUN:  MOV     (R1),@TACSL     ;START THE FUNCTION
007050  012600                          MOV     (SP)+,R0
007052  012601                          MOV     (SP)+,R1
007054  000207                          RTS     PC
```

```
                        ;;*****************************************************************
                        ;
                        .SBTTL          CASSETTE INTERRUPT HANDLER
                        ;
                        ;
007056                  CASINT:
007056  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
007060  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007062  016702  172152          MOV     TACSL,R2
007066  012701  007164          MOV     #CBCNT,R1               ;PARAM. BLOCK
007072  105712                  TSTB    (R2)                    ;TRANSFER REQUEST?
007074  100411                  BMI     TREQ                    ;YES
007076  011204                  MOV     (R2),R4
007100  000304                  SWAB    R4
007102  052704  000001          BIS     #1,R4                   ;SET THE DONE BIT
007106  110451                  MOVB    R4,@-(R1)               ;LOAD ST/ERR BYTE
007110  105012                  CLRB    (R2)                    ;TURN OFF INTERS.
007112  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007114  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007116  000002                  RTI
                        ;
007120  005321          TREQ:   DEC     (R1)+
007122  100412                  BMI     RWILBS                  ;WHEN COUNT NEG., DONE
007124  032712  000004          BIT     #4,(R2)                 ;CHECK READ OR WRITE FUNC.
007130  001003                  BNE     READR
007132  113177  172104          MOVB    @(R1)+,@TADBL           ;WRITE BYTE
007136  000402                  BR      RWDONE
007140  117731  172076  READR:  MOVB    @TADBL,@(R1)+           ;READ BYTE
007144  005241          RWDONE: INC     -(R1)                   ;BUMP   ADDR.
007146  000402                  BR BACK                         ;GET BACK
                        ;
007150  052712  000020  RWILBS: BIS     #20,(R2)                ;INIT. LAST BYTE SEQ.
007154                  BACK:
007154  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007156  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007160  000002                  RTI
                        ;
                        ; CASSETTE INTER. HANDLER PARAM. BLOCK
                        ;
007162  000000          CASPAR: .WORD   0                       ;STATUS/ERROR POINTER
007164  000000          CBCNT:  .WORD   0                       ;WORKING BYTE COUNT
007166  000000                  .WORD   0                       ;WORKING BUFFER ADDRESS
007170  000000          FUNLOC: .WORD   0                       ;FUNC. WORD BUILT HERE
```

```
                        ;;**********************************************************************
                        .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                        ;;**********************************************************************
                        ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
                        ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                        ;*POSITIVE.
                        ;*CALL
                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                        ;*      JSR     PC,@#$DB2D
                        ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                                        ;;IS ON THE STACK

007172  104412          $DB2D:  SAVREG                  ;;SAVE REGISTERS
007174  016602  000002          MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
007200  012700  007352          MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
007204  010066  000002          MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
007210  012201                  MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
007212  012202                  MOV     (R2)+,R2
007214  012767  000012  000046  MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
007222  012704  007302          MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
007226  012705  007304          MOV     #$TNPWR+2,R5
007232  005003                  1$:     CLR     R3      ;;CLEAR PARTIAL
007234  161401                  2$:     SUB     (R4),R1 ;;SUBTRACT TEN POWER
007236  005602                  SBC     R2
007240  161502                  SUB     (R5),R2
007242  002402                  BLT     3$              ;;BR IF TEN POWER TO LARGE
007244  005203                  INC     R3              ;;ADD 1 TO PARTIAL
007246  000772                  BR      2$              ;;LOOP
007250  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
007252  005502                  ADC     R2
007254  062402                  ADD     (R4)+,R2
007256  022525                  CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
007260  052703  000060          BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
007264  110320                  MOVB    R3,(R0)+        ;;SAVE IT
007266  005327                  DEC     (PC)+           ;;DONE?
007270  000000                  4$:     .WORD   0
007272  001357                  BNE     1$              ;;BR IF NO
007274  105020                  CLRB    (R0)+           ;;TERMINATOR
007276  104413                  RESREG                  ;;RESTORE REGISTERS
007300  000207                  RTS     PC              ;;RETURN
007302  145000                  $TNPWR: 145000          ;;1.0E09
007304  035632                  35632
007306  160400                  160400                  ;;1.0E08
007310  002765                  2765
007312  113200                  113200                  ;;1.0E07
007314  000230                  230
007316  041100                  041100                  ;;1.0E06
007320  000017                  17
007322  103240                  103240                  ;;1.0E05
007324  000001                  1
007326  023420                  23420                   ;;1.0E04
007330  000000                  0
007332  001750                  1750                    ;;1.0E03
007334  000000                  0
```

```
1973   007336  000144                          144                     ;;1.0E02
1974   007340  000000                          0
1975   007342  000012                          12                      ;;1.0E01
1976   007344  000000                          0
1977   007346  000001                          1                       ;;1.0E00
1978   007350  000000                          0
1979   007352  000014                $DECVL: .BLKB   12.               ;;RESERVE STORAGE FOR ASCIZ STRING
1980                                  .SBTTL  SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
1981
1982                                  ;;**********************************************************************
1983                                  ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
1984                                  ;*UNSIGNED DECIMAL ASCIZ NUMBER.
1985                                  ;*CALL
1986                                  ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
1987                                  ;*      JSR     PC,@#$SB2D      ;;CALL
1988                                  ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
1989
1990
1991   007366  016667  000002  000022  $SB2D:  MOV     2(SP),1$        ;;SAVE BINARY NUMBER
1992   007374  012746  007416                  MOV     #1$,-(SP)       ;;SET POINTER
1993   007400  004737  007172                  JSR     PC,@#$DB2D      ;;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$:     .WORD   0,0
1998                                  .SBTTL  TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
1999
2000                                  ;;**********************************************************************
2001                                  ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
2002                                  ;*LEADING NUMBERS.
2003                                  ;*CALL
2004                                  ;*      MOV     #NUMADR,-(SP)   ;;FIRST ADDRESS OF ASCIZ STRING
2005                                  ;*      JSR     PC,@#$SUPRS
2006
2007
2008   007422  010046                  $SUPRS: MOV     R0,-(SP)        ;;SAVE R0
2009   007424  016600  000004                  MOV     4(SP),R0        ;;PICKUP THE POINTER
2010   007430  105710          1$:     TSTB    (R0)                    ;;TERMINATEOR?
2011   007432  001403                          BEQ     2$              ;;BR IF YES
2012   007434  122720  000060                  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                    ;;GO TYPE
2017   007452  000000          3$:     .WORD   0                       ;;ASCIZ 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                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
2030
2031   007462  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2032   007464  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
2033   007472  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2034   007474  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2035   007476  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2036   007500  104410                  1$:     RDLIN                   ;;READ AN ASCIZ 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$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2052   007542  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2053   007546  010267  000010                  MOV     R2,$HIOCT
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                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
2059                                    ;;*****************************************************************
2060                                    .SBTTL          ROUTINE TO TYPE DRIVE
2061
2062                                    ;CALL
2063                                    ;       JSR     PC,@#TPDRV
2064                                    ;
2065
2066
2067  007564  104401  013326            TPDRV:  TYPE    ,MSG15                  ;" DRIVE "
2068  007570  113746  001613                    MOVB    @#PARMBK+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)+,1$                ;SAVE IT FOR TYPEOUT
2072  007610  104401                            TYPE                            ;TYPE IT
2073  007612  007615                            1$
2074  007614  000207                            RTS     PC
2075  007616  000000                    1$:     .WORD   0
```

```
2076                                ;;*****************************************************************
2077
2078                                .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2079
2080                                ;CALL
2081                                ;       JSR     PC,@#ASKDRV
2082                                ;       RETURN                  ;NOTE: R0 AND R1 ARE DESTROYED
2083
2084  007620  104401  013336       ASKDRV: TYPE    ,MSGDRV                 ;<CRLF>"DRIVE(S)? "
2085  007624  005067  171422               CLR     DRVKEY
2086  007630  104410                       RDLIN                           ;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       LOOP:   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    ,$QUES                  ;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
```

```
                                ;;*****************************************************************
                                .SBTTL  ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
                                ;CALL
                                ;JSR     PC,@#ASKADR

007730  010046                  ASKADR: MOV     R0,-(SP)                ;SAVE R0
007732  104401  013352          1$:     TYPE    ,MSGASK                 ;"TACS?"
007736  104411                          RDOCT                           ;GET VALUE
007740  012600                          MOV     (SP)+,R0                ;PICK UP THE OCTAL NUMBER
007742  001411                          BEQ     3$                      ;IF "0" USE OLD VALUES
007744  020027  160000                  CMP     R0,#160000              ;MAKE SURE IT IS A BUS ADDRESS
007750  103770                          BLO     1$
007752  010037  001240                  MOV     R0,@#TACSL              ;SAVE THE TACS
007756  062700  000002                  ADD     #2,R0                   ;STEP TO TADB ADDRESS
007762  010037  001242                  MOV     R0,@#TADBL              ;AND SAVE IT
007766  104401  013361          3$:     TYPE    ,MSGVEC                 ;"VECTOR?"
007772  104411                          RDOCT
007774  012600                          MOV     (SP)+,R0
007776  001411                          BEQ     5$
010000  020027  001000                  CMP     R0,#1000                ;MAKE SURE ADDRESS IS IN VECTOR AREA
010004  103370                          BHIS    3$
010006  010037  001244                  MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
010012  062700  000002                  ADD     #2,R0
010016  010037  001246                  MOV     R0,@#TAVEC+2
010022  104401  013372          5$:     TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
010026  104411                          RDOCT
010030  012600                          MOV     (SP)+,R0
010032  001413                          BEQ     6$                      ;IF "0" USE OLD VALUE
010034  020027  000007                  CMP     R0,#7                   ;MAKE SURE ITS VALID
010040  101370                          BHI     5$
010042  000300                          SWAB    R0                      ;PUT INTO HIGH BYTE
010044  006200                          ASR     R0                      ;AND SHIFT
010046  006200                          ASR     R0                      ;INTO PROPER
010050  006200                          ASR     R0                      ;POSITION
010052  042700  177437                  BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
010056  010037  001250                  MOV     R0,@#TAPRIO             ;STORE IT AWAY
010062  104401  013405          6$:     TYPE    ,MTACS                  ;"TACS="
010066  016746  171146                  MOV     TACSL,-(SP)     ;;SAVE TACSL FOR TYPEOUT
010072  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010074  104401  013414                  TYPE    ,MTADB                  ;"TADB="
010100  016746  171136                  MOV     TADBL,-(SP)     ;;SAVE TADBL FOR TYPEOUT
010104  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010106  104401  013423                  TYPE    ,MTAVEC                 ;"VECTOR="
010112  016746  171126                  MOV     TAVEC,-(SP)     ;;SAVE TAVEC FOR TYPEOUT
010116  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010120  104401  013435                  TYPE    ,MTAPRI                 ;"PRIORITY="
010124  016746  171120                  MOV     TAPRIO,-(SP)    ;;SAVE TAPRIO FOR TYPEOUT
010130  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010132  104401  013451                  TYPE    ,MSGOK                  ;"OK?"
010136  104407                          RDCHR                   ;GO READ ONE CHARACTER
010140  012600                          MOV     (SP)+,R0                ;GET IT
010142  022700  000015                  CMP     #15,R0                  ;IS IT "CR"?
010146  001406                          BEQ     7$                      ;BRANCH IF YES
010150  022700  000131                  CMP     #'Y,R0                  ;IS IT "Y"?
010154  001403                          BEQ     7$                      ;IT WAS
010156  104401  001202                  TYPE    .$QUES                  ;TYPE "?"
```

```
2168  010162  000663                    BR      1$                      ;AND LET HIM CORRECT THEM
2169  010164  104401  013456    7$:     TYPE    ,MYES                   ;TYPE OUT "YES"
2170  010170  012600                    MOV     (SP)+,R0                ;RESTORE R0
2171  010172  000207                    RTS     PC                      ;AND RETURN
```

```
                                        ;;*****************************************************************
                                        ;
                                        .SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY

                                        ;CALL:
                                        ;       MOV     #DRVKEY,R0
                                        ;       JSR     PC,@#EXAM               ;R1 IS DESTROYED
                                        ;       NORMAL RETURN
                                        ;       ERROR RETURN

010174  013701  001240          EXAM:   MOV     @#TACSL,R1              ;PICKUP THE "CONTROL & STATUS" REG. ADR.
010200  005011                          CLR     (R1)                    ;DRIVE="A", FUNCTION="WFG"
010202  122710  000101                  CMPB    #'A,(R0)                ;EXAMINE DRIVE "A"?
010206  001402                          BEQ     1$                      ;BR IF YES
010210  052711  000400                  BIS     #UNIT,(R1)              ;SELECT DRIVE "B"
010214  032711  000040          1$:     BIT     #READY,(R1)             ;WAIT ON READY
010220  001775                          BEQ     1$
010222  005711                          TST     (R1)                    ;ANY ERROR?
010224  100024                          BPL     4$                      ;BR IF NO
010226  032711  001000                  BIT     #OFFLINE,(R1)           ;ERROR DUE TO "OFF LINE"?
010232  001017                          BNE     3$                      ;BR IF YES
010234  032711  010000                  BIT     #WRTLOCK,(R1)           ;ERROR DUE TO "WRITE LOCK"?
010240  001411                          BEQ     2$                      ;BR IF NO
010242  122777  000201  170670          CMPB    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
010250  001412                          BEQ     4$                      ;BR IF YES
010252  122777  000203  170660          CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
010260  001406                          BEQ     4$                      ;BR IF YES
010262  000403                          BR      3$                      ;TAKE THE ERROR EXIT
010264  032711  020000          2$:     BIT     #LEADER,(R1)            ;ERROR DUE TO "CLEAR LEADER"?
010270  001002                          BNE     4$                      ;BR IF YES
010272  062716  000002          3$:     ADD     #2,(SP)                 ;TAKE ERROR RETURN
010276  000207                  4$:     RTS     PC                      ;RETURN
```

```
                                .SBTTL  TYPE ROUTINE

                                ;;*****************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*

010300  105767  170653          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010304  100002                          BPL     1$              ;;BR IF YES
010306  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
010310  000407                          BR      3$              ;;LEAVE
010312  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
010314  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010320  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010322  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010324  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010326  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
010330  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
010334  000002                          RTI                     ;;RETURN
010336  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
010342  001430                          BEQ     8$
010344  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
010350  001006                          BNE     5$
010352  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
010354  104401                          TYPE                    ;;TYPE A CR AND LF
010356  001203                          $CRLF
010360  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
010364  000755                          BR      2$              ;;GET NEXT CHARACTER
010366  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
010372  126726  170560          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
010376  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
010400  016746  170550                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
010404  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
010410  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
010412  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
010416  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
010422  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

010424  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
010430  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
010434  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
010442  001372                          BNE     9$              ;;TAB STOP
010444  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
```

```
2260  010446  000724                          BR      2$              ;;GET NEXT CHARACTER
2261  010450  105777  170474          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
2262  010454  100375                          BPL     $TYPEC
2263  010456  116677  000002  170466        MOVB    2(SP),@$TPB     ;;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     1$              ;;BRANCH IF NO
2266  010474  105067  000014                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
2267  010500  000406                          BR      $TYPEX          ;;EXIT
2268  010502  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
2269  010510  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
2270  010512  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
2271  010514  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
2272  010516  000207                $TYPEX: 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  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2285  010526  001074                          BNE     15$             ;;BRANCH IF NO
2286  010530  105777  170410                TSTB    @$TKS           ;;CHAR THERE?
2287  010534  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
2288  010536  117746  170404                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
2289  010542  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
2290  010546  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
2291  010552  001062                          BNE     15$             ;;NO, RETURN TO USER
2292  010554  126727  170354  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
2293  010562  001456                          BEQ     15$             ;;BRANCH IF YES
2294
2295  010564  104401  011245                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
2296  010570  104401  011252        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
2297  010574  016746  167376                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
2298  010600  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2299  010602  104401  011263                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
2300  010606  005046                19$:     CLR     -(SP)           ;;CLEAR COUNTER
2301  010610  005046                        CLR     -(SP)           ;;THE NEW SWR
2302  010612  105777  170326        7$:     TSTB    @$TKS           ;;CHAR THERE?
2303  010616  100375                          BPL     7$              ;;IF NOT TRY AGAIN
2304
2305  010620  117746  170322                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
2306  010624  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2307
2308
2309
2310  010630  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
2311  010634  001005                          BNE     10$             ;;BRANCH IF NOT
2312  010636  104401  011240                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
2313  010642  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
2314  010646  000757                          BR      19$             ;;LET'S TRY IT AGAIN
2315
```

```
2316
2317  010650  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
2318  010654  001022                          BNE     16$             ;;BRANCH IF NO
2319  010656  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
2320  010662  001403                          BEQ     11$             ;;BRANCH IF YES
2321  010664  016677  000002  170246          MOV     2(SP),@SWR      ;;SAVE NEW SWR
2322  010672  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
2323  010676  104401  001203          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
2324  010702  126727  170227  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2325  010710  001003                          BNE     15$             ;;BRANCH IF NOT
2326  010712  012777  000100  170224          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2327  010720  000002          15$:    RTI                             ;;RETURN
2328  010722  004767  177522          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
2329  010726  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
2330  010732  002420                          BLT     18$             ;;BRANCH IF YES
2331  010734  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
2332  010740  003015                          BGT     18$             ;;BRANCH IF YES
2333  010742  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2334  010746  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
2335  010752  001403                          BEQ     17$             ;;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          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
2340  010766  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2341  010772  000707                          BR      7$              ;;GET THE NEXT ONE
2342  010774  104401  001202          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
2343  011000  000720                          BR      20$             ;;SIMULATE CONTROL-U
2344                                  .DSABL  LSB
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          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
2356  011004  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
2357  011012  105777  170126          1$:     TSTB    @$TKS           ;;WAIT FOR
2358  011016  100375                          BPL     1$              ;;A CHARACTER
2359  011020  117766  170122  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
2360  011026  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
2361  011034  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2362  011042  001013                          BNE     3$              ;;BRANCH IF NO
2363  011044  105777  170074          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
2364  011050  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
2365  011052  117746  170070                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
2366  011056  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2367  011062  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
2368  011066  001366                          BNE     2$              ;;IF NOT DISCARD IT
2369  011070  000750                          BR      1$              ;;YES, RESUME
2370  011072  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
2371  011100  002407                          BLT     4$              ;;BRANCH IF YES
```

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

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

011274  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
011300  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
011306  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
011312  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
011316  000406                          BR      $TYPON
011320  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
011326  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
011334  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
011342  010346                          MOV     R3,-(SP)        ;;SAVE R3
011344  010446                          MOV     R4,-(SP)        ;;SAVE R4
011346  010546                          MOV     R5,-(SP)        ;;SAVE R5
011350  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
011354  005404                          NEG     R4
011356  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
011362  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
011366  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
011372  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
011376  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
011400  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
011402  000404                          BR      3$              ;;GO DO MSB
011404  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
011406  006105                          ROL     R3
011410  006105                          ROL     R5
011412  010503                          MOV     R5,R3
011414  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
011416  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
011422  100016                          BPL     7$              ;;BR IF NO
011424  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
011430  001002                          BNE     4$              ;;TEST FOR 0
011432  005704                          TST     R4              ;;SUPPRESS THIS 0?
011434  001403                          BEQ     5$              ;;BR IF YES
```

```
 2469  011436  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 2470  011440  052703  000060                   BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 2471  011444  052703  000040          5$:      BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 2472  011450  110367  000040                   MOVB    R3,8$           ;;SAVE FOR TYPING
 2473  011454  104401  011514                   TYPE    ,8$             ;;GO TYPE THIS DIGIT
 2474  011460  105367  000032          7$:      DECB    $OCNT           ;;COUNT BY 1
 2475  011464  003347                           BGT     2$              ;;BR IF MORE TO DO
 2476  011466  002402                           BLT     6$              ;;BR IF DONE
 2477  011470  005204                           INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 2478  011472  000744                           BR      2$              ;;GO DO THE LAST DIGIT
 2479  011474  012605                  6$:      MOV     (SP)+,R5        ;;RESTORE R5
 2480  011476  012604                           MOV     (SP)+,R4        ;;RESTORE R4
 2481  011500  012603                           MOV     (SP)+,R3        ;;RESTORE R3
 2482  011502  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 2483  011510  012616                           MOV     (SP)+,(SP)
 2484  011512  000002                           RTI                     ;;RETURN
 2485  011514     000                  8$:      .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 2486  011515     000                           .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 2487  011516     000                  $OCNT:   .BYTE   0               ;;OCTAL DIGIT COUNTER
 2488  011517     000                  $OFILL:  .BYTE   0               ;;ZERO FILL SWITCH
 2489  011520  000000                  $OMODE:  .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .SBTTL  TRAP DECODER

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

011522  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
011524  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
011530  005740                          TST     -(R0)           ;;BACKUP BY 2
011532  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
011534  006300                          ASL     R0              ;;POSITION FOR INDEXING
011536  016000  011556                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
011542  000200                          RTS     R0              ;;GO TO ROUTINE


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

011544  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
011546  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
011554  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
011556  011544                  $TRPAD: .WORD   $TRAP2
011560  010300                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
011562  011320                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
011564  011274                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
011566  011334                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

011570  010570                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

011572  010520                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
011574  011002                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
011576  011122                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
011600  007462                          $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
011602  006626                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
011604  006664                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;**********************************************************************
                                        ;POWER DOWN ROUTINE
011606  012737  011752  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
011614  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
011622  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011624  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011626  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011630  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011632  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
011634  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011636  017746  167276                  MOV     @$WR,-(SP)      ;;PUSH @$WR ON STACK
011642  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
011646  012737  011660  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
011654  000000                          HALT
011656  000776                          BR      .-2             ;;HANG UP

                                        ;;**********************************************************************
                                        ;POWER UP ROUTINE
011660  012737  011752  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
011666  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
011672  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
011676  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
011702  001375                          BNE     1$              ;;OF  WORD
011704  012677  167230                  MOV     (SP)+,@$WR      ;;POP STACK INTO @$WR
011710  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
011712  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
011714  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
011716  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
011720  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
011722  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
011724  012737  011606  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
011732  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
011740  104401                          TYPE                    ;REPORT THE POWER FAILURE
011742  011760          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
011744  012716                  MOV     (PC)+,(SP)              ;;RESTART AT $EOP
011746  004412          $PWRAD: .WORD   $EOP                    ;;RESTART ADDRESS
011750  000002                  RTI
011752  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
011754  000776                  BR      .-2                     ;; BEFORE THE POWER DOWN WAS COMPLETE
011756  000000          $SAVR6: 0                               ;;PUT THE SP HERE
011760  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
011766  000122
                                        .EVEN
```

```
2579                                    ;;*****************************************************************
2580
2581                                    ;DATA POINTERS
2582
2583  011770  001116  001162  001166    DT1:    .WORD   $ERRPC,$REG0,$REG2,BYTNUM,$GDDAT,$BDDAT,$GDADR,$BDADR,0
2584  011776  001230  001124  001126
2585  012004  001120  001122  000000
2586
2587  012012  001116  001162  001166    DT2:    .WORD   $ERRPC,$REG0,$REG2,$TMP0,$TMP2,0
2588  012020  001172  001176  000000
2589
2590  012026  001116  001162  001166    DT101:  .WORD   $ERRPC,$REG0,$REG2,$TMP0,0
2591  012034  001172  000000
2592
2593  012040  001116  001240  000000    DT201:  .WORD   $ERRPC,KACSL,0
2594
2595  012046  001116  000000            DT202:  .WORD   $ERRPC,0
2596
2597                                    ;DATA FORMATS
2598          000000                    ON=0    ;OCTAL NUMBER
2599          000001                    DN=1    ;DECIMAL NUMBER
2600
2601  012052      000                   DF1:    .BYTE   ON      ;$ERRPC
2602  012053      001                           .BYTE   DN      ;$REG0
2603  012054      001                           .BYTE   DN      ;$REG2
2604  012055      001                           .BYTE   DN      ;BYTNUM
2605  012056      000                           .BYTE   ON      ;$GDDAT
2606  012057      000                           .BYTE   ON      ;$BDDAT
2607  012060      000                           .BYTE   ON      ;$GDADR
2608  012061      000                           .BYTE   ON      ;$BDADR
2609
2610  012062      000                   DF2:    .BYTE   ON      ;$ERRPC
2611  012063      001                           .BYTE   DN      ;$REG0
2612  012064      001                           .BYTE   DN      ;$REG2
2613  012065      001                           .BYTE   DN      ;$TMP0
2614  012066      001                           .BYTE   DN      ;$TMP1
2615
```

```
                                ;;*****************************************************************
                                ;MESSAGES
012067   127  044522  042524  MXWFG:  .ASCIZ  /WRITE-FILE-GAP/
012074  043055  046111  026505
012102  040507  000120
012106  051127  052111  000105  MXWRIT: .ASCIZ  /WRITE/
012114  042522  042101     000  MXREAD: .ASCIZ  /READ/
012121     102  041501  026513  MXBSFG: .ASCIZ  /BACK-SPACE-FILE-GAP/
012126  050123  041501  026505
012134  044506  042514  043455
012142  050101     000
012145     102  041501  026513  MXBSBG: .ASCIZ  /BACK-SPACE-BLK-GAP/
012152  050123  041501  026505
012160  046102  026513  040507
012166  000120
012170  050123  041501  026505  MXSFFG: .ASCIZ  /SPACE-FWD-FILE-GAP/
012176  053506  026504  044506
012204  042514  043455  050101
012212     000
012213     123  040520  042503  MXSFBG: .ASCIZ  /SPACE-FWD-BLK-GAP/
012220  043055  042127  041055
012226  045514  043455  050101
012234     000
012235     122  053505  047111  MXRWND: .ASCIZ  /REWIND/
012242  000104
012244  040504  040524  042440  EM1:    .ASCIZ  "DATA ERROR"
012252  051122  051117     000
012257     123  047131  020103  EM2:    .ASCIZ  "SYNC ERROR"
012264  051105  047522  000122
012272  051104  053111  020105  EM101:  .ASCIZ  "DRIVE IS OFF-LINE"
012300  051511  047440  043106
012306  046055  047111  000105
012314  051104  053111  020105  EM102:  .ASCIZ  "DRIVE IS WRITE-LOCK"
012322  051511  053440  044522
012330  042524  046055  041517
012336  000113
012340  046103  040505  020122  EM103:  .ASCIZ  "CLEAR LEADER ERROR"
012346  042514  042101  051105
012354  042440  051122  051117
012362     000
012363     106  046111  020105  EM104:  .ASCIZ  "FILE GAP ERROR"
012370  040507  020120  051105
012376  047522  000122
012402  044524  044515  043516  EM105:  .ASCIZ  "TIMING ERROR"
012410  042440  051122  051117
012416     000
012417     102  047514  045503  EM106:  .ASCIZ  "BLOCK CHECK ERROR"
012424  041440  042510  045503
012432  042440  051122  051117
012440     000
012441     125  045516  047516  EM107:  .ASCIZ  "UNKNOWN INTERRUPT"
012446  047127  044440  052116
012454  051105  052522  052120
012462     000
```

```
2672  012463     124  030501  020061  EM201:  .ASCIZ  "TA11 FAILED TO RESPOND"
2673  012470  040506  046111  042105
2674  012476  052040  020117  042522
2675  012504  050123  047117  000104
2676  012512  047516  042040  044522  EM202:  .ASCIZ  "NO DRIVE AVAILABLE"
2677  012520  042526  040440  040526
2678  012526  046111  041101  042514
2679  012534     000
2680  012535     120  020103  020040  DH1:    .ASCIZ  /PC      FILE    BLOCK   BYTE    GDDAT   BDDAT   GDADR   BDADR/
2681  012542  020040  043040  046111
2682  012550  020105  020040  041040
2683  012556  047514  045503  020040
2684  012564  041040  052131  020105
2685  012572  020040  043440  042104
2686  012600  052101  020040  041040
2687  012606  042104  052101  020040
2688  012614  043440  040504  051104
2689  012622  020040  041040  040504
2690  012630  051104     000
2691  012633     040  020040  020040  DH2:    .ASCII  /       EXPT'D  EXPT'D  RCV'D   RCV'D/<15><12>
2692  012640  020040  042440  050130
2693  012646  023524  020104  042440
2694  012654  050130  023524  020104
2695  012662  051040  053103  042047
2696  012670  020040  051040  053103
2697  012676  042047  005015
2698  012702  041520  020040  020040          .ASCIZ  /PC      FILE    BLOCK   FILE    BLOCK/
2699  012710  020040  044506  042514
2700  012716  020040  020040  046102
2701  012724  041517  020113  020040
2702  012732  044506  042514  020040
2703  012740  020040  046102  041517
2704  012746  000113
2705  012750  041520  020040  020040  DH101:  .ASCIZ  /PC      FILE    BLOCK   FUNCTION/
2706  012756  020040  044506  042514
2707  012764  020040  020040  046102
2708  012772  041517  020113  020040
2709  013000  052506  041516  044524
2710  013006  047117     000
2711  013011     120  020103  020040  DH201:  .ASCIZ  /PC      TACS/
2712  013016  020040  052040  041501
2713  013024  000123
2714  013026  041520     000          DH202:  .ASCIZ  /PC/
2715  013031     200  025052  020052  MSG1:   .ASCIZ  <CRLF>"*** END-OF-TEST ***"
2716  013036  047105  026504  043117
2717  013044  052055  051505  020124
2718  013052  025052  000052
2719  013056  051600  043117  020124  MSG2:   .ASCIZ  <CRLF>"SOFT ERRORS="
2720  013064  051105  047522  051522
2721  013072  000075
2722  013074  044200  051101  020104  MSG3:   .ASCIZ  <CRLF>"HARD ERRORS="
2723  013102  051105  047522  051522
2724  013110  000075
2725  013112  041200  052131  051505  MSG4:   .ASCIZ  <CRLF>"BYTES READ="
2726  013120  051040  040505  036504
2727  013126     000
```

```
2728  013127  000200  054502  042524  MSG5:   .ASCIZ  <CRLF>"BYTES WRITTEN="
2729  013134  020123  051127  052111
2730  013142  042524  036516  000000
2731  013147  000200  040524  042520  MSG6:   .ASCIZ  <CRLF>"TAPE PASSES="
2732  013154  050040  051501  042523
2733  013162  036523  000000
2734  013165  000200  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  000000  MSG9:   .ASCIZ  " AND"
2741  013225  000040  044527  046114  MSG10:  .ASCIZ  " WILL BE TESTED"<CRLF>
2742  013232  041040  020105  042524
2743  013240  052123  042105  000200
2744  013246  025052  020052  047506  MSG11:  .ASCIZ  "*** FORMAT ***"
2745  013254  046522  052101  025040
2746  013262  025052  000000
2747  013265  000052  025052  051040  MSG12:  .ASCIZ  "*** READ   ***"
2748  013272  040505  020104  020040
2749  013300  025052  000052
2750  013304  025052  020052  051127  MSG13:  .ASCIZ  "*** WRITE  ***"
2751  013312  052111  020105  025040
2752  013320  025052  000000
2753  013323  000040  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  000000
2760  013361  000200  042526  052103  MSGVEC: .ASCIZ  <CRLF>"VECTOR?"
2761  013366  051117  000077
2762  013372  050200  044522  051117  MSGPRI: .ASCIZ  <CRLF>"PRIORITY?"
2763  013400  052111  037531  000000
2764  013405  000200  040524  051503  MTACS:  .ASCIZ  <CRLF>"TACS="
2765  013412  000075
2766  013414  052040  042101  036502  MTADB:  .ASCIZ  / TADB=/
2767  013422  000000
2768  013423  000040  053040  041505  MTAVEC: .ASCIZ  "  VECTOR="
2769  013430  047524  036522  000000
2770  013435  000040  050040  044522  MTAPRI: .ASCIZ  "  PRIORITY="
2771  013442  051117  052111  036531
2772  013450  000000
2773  013451  000200  045517  000077  MSGOK:  .ASCIZ  <CRLF>"OK?"
2774  013456  042531  100123  000000  MYES:   .ASCIZ  "YES"<CRLF>
2775          013464                  .EVEN
2776
2777                                  .SBTTL          READ AND WRITE BUFFER
2778
2779  013464  002004                  BUFFER: .BLKB   1028.
2780          015470                  LASTADDRESS=    .
2781
2782          000001                          .END
```

DZTAEC.P11 17-MAR-77 14:52 CROSS REFERENCE TABLE -- USER SYMBOLS

```
ASKADR  007730            618    2117#
ASKDRV  007620            615    2084#   2107
BACK    007154           1900    1903#
BADFUN  005466           1448    1454#
BC.ERR  006050           1517    1559#
BEGINX  002326            612#    648     650     711     713
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#   1583    1585    1648    1650    1704    1729
BLOCK   001306            314#    846     945     946     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#
CKSWR = 104406           1315    1350    1369    2528#
CLEAR   002764            730     733     751#    753
CL.ERR  005712           1514    1526#
```

```
CNTHRD  006606            923    1052    1767#
CNTSFT  006550            921    1050    1750#
CR    = 000015             85#   2264    2274
CRCERR= 040000            199#   1501
CRLF  = 000200             86#    600    2235    2274    2715    2719    2722    2725    2728    2731    2734    2737    2741
                          2756    2758    2760    2762    2764    2773    2774
CSRERR  005536            858     869     882     894     905     936    1008    1024    1035    1089    1136    1147    1161
                          1483#
DATCMP  006356            918    1047    1701#
DDISP = 177570             92#    259     571
DF1     012052            454    2601#
DF2     012062            461    2610#
DH1     012535            452    2680#
DH101   012750            470     476     482     488     494     500     506    2705#
DH2     012633            458    2691#
DH201   013011            518    2711#
DH202   013026            523    2714#
DISPLA  001142            259#    571*    579*   1333*   1353*
DISPRE  000174            223#    579
DN    = 000001           2599#   2602    2603    2604    2611    2612    2613    2614
DOFUN   007044           1867#
DRVKEY  001252            301#    302     538*    677     690     741    2085*   2090    2104    2108
DRVPNT  001256            302#    691*    734     742*
DSWR  = 177570             91#    258     570
DT1     011770            453    2583#
DT101   012026            471     477     483     489     495     501     507    2590#
DT2     012012            460    2587#
DT201   012040            519    2593#
DT202   012046            524    2595#
EMTVEC= 000030            180#    558*    559*
EM1     012244            451    2643#
EM101   012272            469    2647#
EM102   012314            475    2650#
EM103   012340            481    2654#
EM104   012363            487    2658#
EM105   012402            493    2661#
EM106   012417            499    2664#
EM107   012441            505    2668#
EM2     012257            457    2645#
EM201   012463            517    2672#
EM202   012512            522    2676#
EOTS    001212            284#   1064*   1201    1261    1548*
EREXT1  005526           1416    1473#
EREXT2  005530           1451    1466    1474#
ERROR = 100000            198#
ERROR0  005320           1405    1410#
ERROR1  005376           1406    1430#
ERROR2  005506           1407    1464#
ERRVEC= 000004            173#    568     569*    580*    636*    642*   1321    1322*   1324*   1327*
EXAM    010174            678     682     687    2182#
EXIT    007720           2103    2108#
FGAP  = 004000            202#   1497
FG.ERR  006040           1515    1553#
FILE    001304            312#    849     950     999    1060    1077    1127    1175    1483    1531    1532    1538    1618
                          1683    1685    1707    1731
FILESZ  001310            316#    946    1073    1171    1273    1533
```

```
FILL    006136          1587    1611#
FORMAT  003132           795     840#
FORPAS  001274           307     308#    747     784
FUNCTI= 000016           214#   1486
FUNCO = 000002           212#    214
FUNC1 = 000004           211#    214
FUNC2 = 000010           210#    214
FUNLOC  007170          1485    1853*   1860*   1914#
FUNTAB  006724          1835#   1849
GNS   = ****** U         222     599    2521    2522    2523    2524    2526    2528    2529    2530    2531    2532    2533
GO    = 000001           213#
GTSWR = 104405           594    1287    2526#
HARDNM  001210           283#   1246    1767*   1768
HT    = 000011            83#   2233    2274
ILBS  = 000020           209#
INT.EN= 000100           207#
IOCOM   006760          1835    1837    1839    1841    1843    1845    1847    1849#
IOTVEC= 000020           178#    556*    557*
ITEMS0  001622           445#   1392
ITEMS1  001642           467#   1393
ITEMS2  001732           515#   1394
LASTAD= 015470          2780#
LASTBK  001234           291#    744*   1062    1533*   1534*   1542*
LASTFL  001236           293#    732     743*   1060    1266    1532*   1538*
LEADER= 020000           200#   1495    2200
LF    = 000012            84#   2268    2274
LOOP    007644          2091#   2105
LOOPER  002774           750     774#   1205
MAXEOT  001264           305#   1201
MAXERR  001262           304#    746     939    1768
MAXRDS  001260           303#    745     938    1059    1753    1754
MSGASK  013352          2118    2758#
MSGDRV  013336          2084    2756#
MSGOK   013451          2160    2773#
MSGPRI  013372          2136    2762#
MSGVEC  013361          2127    2760#
MSG1    013031          1238    2715#
MSG10   013225           706    2741#
MSG11   013246           841    2744#
MSG12   013265           991    2747#
MSG13   013304          1119    2750#
MSG14   013323          1414    1436    1469    2753#
MSG15   013326          2067    2754#
MSG2    013056          1240    2719#
MSG3    013074          1245    2722#
MSG4    013112          1250    2725#
MSG5    013127          1255    2728#
MSG6    013147          1260    2731#
MSG7    013165          1265    2734#
MSG8    013202          1272    2737#
MSG9    013220           703    2740#
MTACS   013405          2148    2764#
MTADB   013414          2151    2766#
MTAPRI  013435          2157    2770#
MTAVEC  013423          2154    2768#
MXBSBG  012145          1458    2629#
```

DZTAEC.P11 17-MAR-77 14:52 CROSS REFERENCE TABLE -- USER SYMBOLS

```
MXBSFG  012121          1457   2625#
MXREAD  012114          1456   2624#
MXRWND  012235          1461   2641#
MXSFBG  012213          1460   2637#
MXSFFG  012170          1459   2633#
MXWFG   012067          1454   2620#
MXWRIT  012106          1455   2623#
MYES    013456          2169   2774#
NEXT    007700          2094   2098   2102#
NOTA    007656          2092   2095#
NOTB    007670          2096   2099#
NOTLGL  007712          2089   2100   2106#  2109
OFFLIN= 001000           204#  1491   2191
OL.ERR  005702          1512   1520#
ON    = 000000          2598#  2601   2605   2606   2607   2608   2610
PARMBK  001612           419#   421    697*   699*   701*   704*   738*   853    855    864    866    877    879
                         889    891    900    902    931    933   1003   1005   1019   1021   1030   1032   1084
                        1086   1131   1133   1142   1144   1156   1158   1236   1488   1585*  1650*  1735   2068
PASCNT  001266           306#   749*   777    779*   787*
PATS    001352           343#  1586   1714
PAT0    001412           343    365#
PAT1    001422           344    368#
PAT10   001532           353    395#
PAT11   001542           354    398#
PAT12   001552           355    401#
PAT13   001562           356    404#
PAT14   001572           357    407#
PAT15   001602           358    410#
PAT2    001432           345    371#
PAT3    001442           346    374#
PAT4    001452           347    377#
PAT5    001462           348    380#
PAT6    001472           349    383#
PAT7    001502           350    386#
PAT8    001512           351    389#
PAT9    001522           352    392#
PIRQ  = 177772            90#
PIRQVE= 000240           184#
PR0   = 000000           107#
PR1   = 000040           108#
PR2   = 000100           109#
PR3   = 000140           110#
PR4   = 000200           111#
PR5   = 000240           112#
PR6   = 000300           113#
PR7   = 000340           114#
PS    = 177776            87#    88    547#
PSCNTL  001270           307#   747*   748*   777    780*   781*   782    784*   785*   788
PSW   = 177776            88#
PWRVEC= 000024           179#   562*   563*  2538*  2539*  2548*  2554*  2566*  2567*
RBYTTL  001214           285#  1251   1648*  1649*
RDCHR = 104407          2161   2387   2529#
RDLIN = 104410          2036   2086   2530#
RDOCT = 104411          2119   2128   2137   2531#
RDTRYS  001224           287#   745*   938*  1059*  1751*  1753*
RD1PAS  001276           309#
```

```
RD2PAS  001302            311#    782
READ    006732            901    1031    1838#
READON  003562            796     798     989#    990
READR   007140           1895    1898#
READY = 000040            208#   2187
RESREG= 104413           1957    2533#
RESVEC= 000010            174#
REWIND  006756            854    1004    1132    1237    1848#
RWDONE  007144           1897    1899#
RWILBS  007150           1893    1902#
SAVREG= 104412           1931    2532#
SETUPR  006220            899    1013    1645#
SETUPW  006072            876    1153    1580#
SFBG    006752           1846#
SFFG    006746           1085    1844#
SOFTNM  001206            282#    731     732    1241    1750*   1754*   1755*
STACK = 001100             78#    554
START   002630            707     728#   1299
STKLMT= 177774             89#
SWR     001140            258#    552     570*    572     578*    592     774    1203    1281    1285    1316    1354    1361
                         1366    2195    2197    2284    2321*   2546    2559*
SWREG   000176            224#    578     592    1285    2284    2297
SW0   = 000001            142#
SW00  = 000001            132#    142
SW01  = 000002            131#    141
SW02  = 000004            130#    140
SW03  = 000010            129#    139
SW04  = 000020            128#    138
SW05  = 000040            127#    137
SW06  = 000100            126#    136
SW07  = 000200            125#    135
SW08  = 000400            124#    134    1203
SW09  = 001000            123#    133    1281
SW1   = 000002            141#
SW10  = 002000            122#   1336
SW11  = 004000            121#
SW12  = 010000            120#
SW13  = 020000            119#
SW14  = 040000            118#
SW15  = 100000            117#
SW2   = 000004            140#
SW3   = 000010            139#
SW4   = 000020            138#
SW5   = 000040            137#
SW6   = 000100            136#
SW7   = 000200            135#
SW8   = 000400            134#
SW9   = 001000            133#
SYNCK   006254            911    1040    1670#
TACSL   001240            297#    638    1851    1867*   1879    2124*   2149    2182    2593
TADBL   001242            298#   1896*   1898    2126*   2152
TAPRIO  001250            300#   2147*   2158
TAVEC   001244            299#    541     652*    653*   2133*   2135*   2155
TBITVE= 000014            175#
TIMERR= 002000            203#   1499
TKVEC = 000060            182#
```

DZTAEC.P11 17-MAR-77 14:52 CROSS REFERENCE TABLE -- USER SYMBOLS

```
TMPSWR= 160000          1336#  1377
TM.ERR  006044          1516   1556#
TPDRV   007564           700    705    842    992   1120   2067#
TPVEC = 000064           183#
TRAPVE= 000034           181#   560*   561*
TREQ    007120          1882   1892#
TRTVEC= 000014           176#
TR.REQ= 000200           206#
TYPE  = 104401           588    698    703    706    841    843    991    993   1119   1121   1228   1229   1238
                        1240   1245   1250   1255   1260   1265   1272   1277   1356   1364   1379   1397   1399
                        1401   1403   1414   1426   1436   1444   1449   1469   2016   2067   2072   2084   2106
                        2118   2127   2136   2148   2151   2154   2157   2160   2167   2169   2238   2295   2296
                        2299   2312   2323   2342   2391   2394   2398   2473   2521#  2568
TYPERR  005210          1363   1379#
TYPOC = 104402          1420   1433   1468   2150   2153   2156   2159   2298   2522#
TYPON = 104404          2524#
TYPOS = 104403          2523#
UNIT  = 000400           205#  2186
WBYTTL  001220           286#  1256   1543*  1544*  1583*  1584*
WFG     006722           865   1143   1834#
WL.ERR  005706          1513   1523#
WRITE   006726           878   1157   1836#
WRITEQ  004136           797   1117#  1118
WRTLOC= 010000           201#  1493   2193
WRTPAS  001300           310#
WRTRYS  001226           288#   746*   929*   939*
XBSBG = 000010           192#
XBSFG = 000006           191#
XREAD = 000004           190#  1560
XRWND = 000016           195#
XSFBG = 000014           194#
XSFFG = 000012           193#
XWFG  = 000000           188#  1526
XWRITE= 000002           189#  1536
X.ERR   006066          1518   1565#
$AUTOB  001134           255#   596*  2292   2413
$BDADR  001122           250#  1727*  2583
$BDDAT  001126           252#   729   1723*  1726*  2583
$BELL   005204          1356   1376#
$CHARC  010514          2240*  2250*  2257   2266*  2271#
$CKSWR  010520          2284#  2528
$CMTAG  001100           238#   549    550    558    564    565
$CM1  = 000004           270#   271#   272#   273#   274#
$CM2  = 000010           270#   271#   272#   273#   274#
$CM3  = 000004           268#   270
$CM4  = 000004           274#   275#   276#   277#   278#
$CNTLG  011245          2295   2408#
$CNTLU  011240          2312   2407#
$CRLF   001203           279#   698    843    993   1121   1364   1377   1379   1399   1403   2239   2274   2323
                        2407
$DB2D   007172          1252   1257   1423   1441   1931#  1993
$DECVL  007352          1933   1979#
$DOAGN  004736          1224   1231   1291   1297#
$ENDAD  004726           586    651    714   1293#  1371
$ENDCT  004442           564   1226#
$ENDMG  004745          1228   1301#
```

DZTAEC.P11 17-MAR-77 14:52 CROSS REFERENCE TABLE -- USER SYMBOLS

```
$ENULL  004742          1229    1300#
$EOP    004412           859     870     883     895     906     914     925     937    1009    1025    1036    1043    1054
                        1090    1137    1148    1162    1202    1204    1217#   2571
$EOPCT  004434           564*   1223#   1227
$ERFLG  001103           241#   1308    1332*   1336    1351*   1376
$ERMAX  001115           247#   1336
$ERROR  005060           558    1349#
$ERRPC  001116           248#   1358*   1359*   1360    1376    2583    2587    2590    2593    2595
$ERRTB  001622           439#
$ERTTL  001112           245#   1357*   1376
$FILLC  001156           266#   2243    2274
$FILLS  001155           265#   2274
$GDADR  001120           249#   1725*   2583
$GDDAT  001124           251#   1722*   1724*   2583
$GET42  004456          1230#
$GTSWR  010570          2296#   2526
$HD   = 000000            12
$HIOCT  007562          2053*   2058#
$ICNT   001104           242#
$ILLUP  011752          2538    2554    2573#
$INTAG  001135           256#   2324    2413
$ITEMB  001114           246#    728     729    1360*   1377    1383
$LF     001204           280#   1277    1377    2274    2398    2407
$LPADR  001106           243#    565*    840*    875*    990*   1018*   1029*   1118*   1154*   1331*   1334    1336
$LPERR  001110           244#
$MAIL = ****** U         582     592    1331    1366    2227
$MNEW   011263          2299    2411#
$MSWR   011252          2296    2409#
$NULL   001154           264#   2245    2274
$OCNT   011516          2445*   2474*   2487#
$OMODE  011520          2440*   2444*   2449    2452*   2463*   2489#
$OVER   005044          1317    1328    1333#
$PASS   001100           239#   1220*   1221*   1300
$POWER  011760          2569    2576#
$PWRAD  011746          2571#
$PWRDN  011606           562    2538#   2566
$PWRMG  011742          2569#
$PWRUP  011660          2548    2554#
$QUES   001202           278#   1377    2106    2167    2274    2342    2391    2407
$RDCHR  011002          2355#   2529
$RDDEC= ****** U        2532
$RDLIN  011122          2383#   2530
$RDOCT  007462          2031#   2531
$RDSZ = 000010          2376#
$REGAD  001160           268#
$REG0   001162           270#   1483*   1685*   1731*   2583    2587    2590
$REG1   001164           271#
$REG2   001166           272#   1484*   1686*   1732*   2583    2587    2590
$REG3   001170           273#
$RESRE  006664          1807#   2533
$RTNAD  004740          1299#
$R2A  = ****** U        2534
$SAVRE  006626          1791#   2532
$SAVR6  011756          2547*   2555    2556*   2557*   2575#
$SB2D   007366          1242    1247    1262    1269    1274    1991#
$SCOPE  004760           556    1314#
```

```
$SETUP= 000137          537*   555    556    558    560    562    564    565    586    589   1219   1315   1350
                       1369   1371   2279   2413
$STUP  = 177777         537*
$SUPRS  007422         1243   1248   1253   1258   1263   1270   1275   2008*
$SVLAD  005030         1325   1330*
$SWR   = 160000           2*     12     61     62     63     64     65     66    278    565    566   1212   1220
                       1232   1298   1300   1309   1310   1311   1316   1328   1330   1332   1336*  1342   1343
                       1344   1345   1346   1354   1361   1366   1370   1376   1377*  2572
$SWRMK= 000000         1311
$TKB    001146          261*   2277   2288   2305   2359   2365
$TKS    001144          260*   2277   2286   2302   2326*  2357   2363
$TMP0   001172          274*    774*   775*   788*   790    791   1487*  1526   1536   1560   1687*  2587   2590
$TMP1   001174          275*
$TMP2   001176          276*   1688*  2587
$TMP3   001200          277*
$TN    = 000001           2*     12
$TNPWR  007302         1938   1939   1959*
$TPB    001152          263*   2263*  2274
$TPFLG  001157          267*   2221   2274
$TPS    001150          262*   2261   2274
$TRAP   011522          560    2498*
$TRAP2  011544         2509*  2520
$TRP   = 000014         2513*  2522*  2523*  2524*  2525*  2526   2527*  2528   2529*  2530*  2531*  2532*  2533*
                       2534*
$TRPAD  011556         2503   2520*
$TSTNM  001102          240*   1219*  1308   1330*  1333   1336   1353   1376
$TTYIN  011230         2384   2385   2402   2406*
$TYPBN= ****** U       2525
$TYPDS= ****** U       2525
$TYPE   010300         2221*  2513   2521
$TYPEC  010450         2242   2249   2256   2261*  2262   2328
$TYPEX  010516         2267   2269   2272*
$TYPOC  011320         2443*  2522
$TYPON  011334         2442   2445*  2524
$TYPOS  011274         2438*  2523
$XTSTR  004772         1319*
$$GET4= 000001         1232*  1290*
$OFILL  011517         2439*  2443*  2453   2488*
$40CAT= ****** U       1316   1363
.      = 015470          218*    222*   237*   281    553    565    600*   1300   1303   1336   1376   1377   1979*
                       2274   2277   2406*  2407   2413   2550   2574   2775*  2779*  2780
```

```
BGOTO     216#  1509
CALL      216#   853   863   877   888   900   931  1003  1019  1030  1084  1131  1141  1155
COMMEN    185#
DECBLK    216#  1539
DECFIL    216#  1528
ENDCOM    185#
EOPCOD    216#  1232
ERROR      79#   645   708   913   920  1042  1049  1520  1523  1550  1553  1556  1559  1565
ESCAPE    185#
GETPRI    185#
GETSWR    185#   589#
INCBLK    216#   943  1069  1168
INCFIL    216#   948  1075  1173
JGOTO     216#   789
MORETA    216#   282
MULT      185#
NEWTST    185#
OUTDBL    216#  1251  1256
OUTSGL    216#  1241  1246  1261  1273
POP       185#  1628  1812  1888  1903  2054  2559  2560
PUSH      185#  1611  1792  1876  2033  2540  2546
REMARK    216#   850   860   872   885   896   908   915   926   940  1000  1010  1015  1026  1037
         1044  1056  1066  1081  1128  1138  1150  1165  1233  1278
REPORT    185#
SCOPE      60#   907  1055  1164  1200  1218  1232
SETBLK    216#   844   951   994  1078  1122  1176
SETFIL    216#   847   997  1125
SETPRI    185#
SETTRA   2513#  2522  2523  2524  2526  2528  2529  2530  2531  2532  2533
SETUP     185#   548
SKIP      185#
SLASH     185#   318   339   361   415   527   715   799   803   850   852   860   862   872   874
           885   887   896   898   908   910   915   917   926   928   940   942   955  1000  1002
          1010  1012  1015  1017  1026  1028  1037  1039  1044  1046  1056  1058  1066  1068  1081
          1083  1092  1128  1130  1138  1140  1150  1152  1165  1167  1180  1233  1235  1278  1280
          1409  1429  1463  1472  1479
SPACE     185#
STARS      12   185#  229   233   281   441   463   511   529   535   602   611   619   635   654
           676   717   727   755   773   801   810   838   962   988  1099  1115  1185  1199  1209
          1305  1338  1377  1567  1590  1634  1653  1692  1741  1758  1772  1776  1821  1871  1917
          1920  1982  2000  2023  2059  2076  2112  2172  2206  2276  2279  2347  2376  2415  2492
          2536  2552  2579  2616
SWRSU     185#   566#
TRMTRP   2513#
TYPBIN    185#
TYPDEC    185#
TYPNAM    185#   582
TYPNUM    185#
TYPOCS    185#
TYPOCT    185#  1419  1431  1467  2149  2152  2155  2158  2297
TYPTXT    185#
$$CMRE    231#   270   271   272   273
$$CMTM    231#   274   275   276   277
$$ESCA    185#
$$NEWT    185#
$$SET    2513#  2522  2523  2524  2526  2528  2529  2530  2531  2532  2533
```

```
$$SKIP    185#
.EQUAT      2#     75
.HEADE      2#
.SETUP      2#    537
.SWRHI      2#     57
.SWRLO     67#
.$CATC      2#    216
.$CMTA      2#    231
.$DB2D      2#   1918
.$EOP       2#   1207
.$ERRO      2#   1336
.$POWE      2#   2534
.$RDOC      2#   2021
.$READ      2#   2274
.$SAVE      2#   1774
.$SB2D      2#   1980
.$SCOP      2#   1303
.$SPAC      2#
.$SUPR      2#   1998
.$TRAP      2#   2490
.$TYPE      2#   2204
.$TYPO      2#   2413

. ABS.  015470     000

ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:DZTAEC.BIN,DSKZ:DZTAEC.LST/CRF/SOL=DSKM:DZTAEC.P11
RUN-TIME: 15 8 .9 SECONDS
RUN-TIME RATIO: 108/25=4.3
CORE USED:  25K  (50 PAGES)
```