TA11

DATA RELIABILITY

MD-11-DZTAE-B

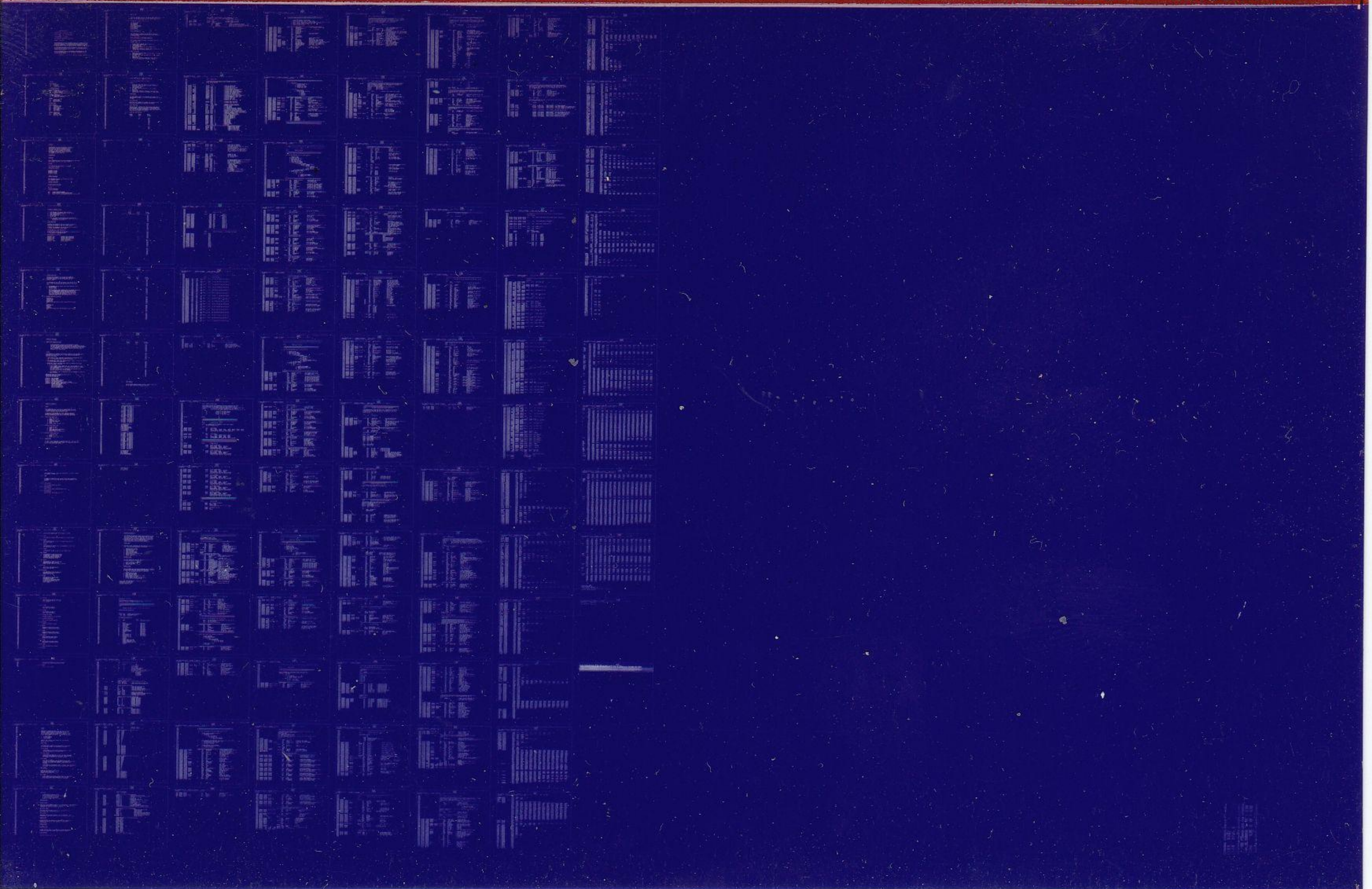
EP-DZTAE-B-DL-A

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA



```
                                .REM    %
```

IDENTIFICATION

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

PRODUCT NAME: TA11 DATA RELIABILITY

DATE CREATED: 21 JUNE 76

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?
```

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	314 063 314 063 146 231 146 231
1	512	314 231 063 146

```
    674                                                     314
    675                                                     231
    676                                                     063
    677                                                     146
    678
```

8.5.3 (CONT.)

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

8.5.3 (CONT.)

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
7	64	000
		222
		044
		266
		111
		333
		155
		377
8	1024	001
		003
		007
		017
		037
		077
		177
		377
9	32	376
		374
		370
		360
		340
		300
		200
		000
10	128	001
		376
		002
		375
		004
		373
		010
		367
11	256	020
		357
		040
		337
		100
		277
		200
		177

8.5.3 (CONT.)

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
12	512	000
		000
		000
		000
		000
		000
		000
		000
13	1024	377
		377
		377
		377
		377
		377
		377
		377
14	32	000
		377
		000
		377
		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
```

```
892                                TAPE PASSES=18
893                                FILES/PASS=12
894                                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-B
;*COPYRIGHT (C) 1973,1976
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
```

```
                        ;*
                        ;*PROGRAM BY JIM LACEY
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-CO),MAR 21, 1976.
                        ;*
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;;*****************************************************************
                        .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
                        ;*        12            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
                        .EQUIV  R6,SP           ;;STACK POINTER
                        .EQUIV  R7,PC           ;;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
```

```
1058            000300                  PR6=    300             ;;PRIORITY LEVEL 6
1059            000340                  PR7=    340             ;;PRIORITY LEVEL 7
1060
1061                                    ;*"SWITCH REGISTER" SWITCH DEFINITIONS
1062            100000                  SW15=   100000
1063            040000                  SW14=   40000
1064            020000                  SW13=   20000
1065            010000                  SW12=   10000
1066            004000                  SW11=   4000
1067            002000                  SW10=   2000
1068            001000                  SW09=   1000
1069            000400                  SW08=   400
1070            000200                  SW07=   200
1071            000100                  SW06=   100
1072            000040                  SW05=   40
1073            000020                  SW04=   20
1074            000010                  SW03=   10
1075            000004                  SW02=   4
1076            000002                  SW01=   2
1077            000001                  SW00=   1
1078                                    .EQUIV  SW09,SW9
1079                                    .EQUIV  SW08,SW8
1080                                    .EQUIV  SW07,SW7
1081                                    .EQUIV  SW06,SW6
1082                                    .EQUIV  SW05,SW5
1083                                    .EQUIV  SW04,SW4
1084                                    .EQUIV  SW03,SW3
1085                                    .EQUIV  SW02,SW2
1086                                    .EQUIV  SW01,SW1
1087                                    .EQUIV  SW00,SW0
1088
1089                                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1090            100000                  BIT15=  100000
1091            040000                  BIT14=  40000
1092            020000                  BIT13=  20000
1093            010000                  BIT12=  10000
1094            004000                  BIT11=  4000
1095            002000                  BIT10=  2000
1096            001000                  BIT09=  1000
1097            000400                  BIT08=  400
1098            000200                  BIT07=  200
1099            000100                  BIT06=  100
1100            000040                  BIT05=  40
1101            000020                  BIT04=  20
1102            000010                  BIT03=  10
1103            000004                  BIT02=  4
1104            000002                  BIT01=  2
1105            000001                  BIT00=  1
1106                                    .EQUIV  BIT09,BIT9
1107                                    .EQUIV  BIT08,BIT8
1108                                    .EQUIV  BIT07,BIT7
1109                                    .EQUIV  BIT06,BIT6
1110                                    .EQUIV  BIT05,BIT5
1111                                    .EQUIV  BIT04,BIT4
1112                                    .EQUIV  BIT03,BIT3
1113                                    .EQUIV  BIT02,BIT2
```

```
1114                            .EQUIV  BIT01,BIT1
1115                            .EQUIV  BIT00,BIT0
1116
1117                            ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1118    000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
1119    000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
1120    000014                  TBITVEC=14              ;;"T" BIT
1121    000014                  TRTVEC= 14              ;;TRACE TRAP
1122    000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
1123    000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1124    000024                  PWRVEC= 24              ;;POWER FAIL
1125    000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
1126    000034                  TRAPVEC=34              ;;"TRAP" TRAP
1127    000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
1128    000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
1129    000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1130                            ;;**************************************************************
1131
1132                            ;##########TA11 FUNCTIONS##########
1133    000000                  XWFG=   0                       ;WRITE FILE GAP FUNCTION
1134    000002                  XWRITE= 2                       ;WRITE FUNCTION
1135    000004                  XREAD=  4                       ;READ FUNCTION
1136    000006                  XBSFG=  6                       ;BACK SPACE FILE GAP FUNCTION
1137    000010                  XBSBG=  10                      ;BACK SPACE BLOCK GAP FUNCTION
1138    000012                  XSFFG=  12                      ;SPACE FWD FILE GAP FUNCTION
1139    000014                  XSFBG=  14                      ;SPACE FWD BLOCK GAP FUNCTION
1140    000016                  XRWND=  16                      ;REWIND FUNCTION
1141
1142                            ;##########TA11 BIT ASSIGNMENT##########
1143    100000                  ERROR=  BIT15
1144    040000                  CRCERR= BIT14
1145    020000                  LEADER= BIT13
1146    010000                  WRTLOCK=BIT12
1147    004000                  FGAP=   BIT11
1148    002000                  TIMERR= BIT10
1149    001000                  OFFLINE=BIT09
1150    000400                  UNIT=   BIT08
1151    000200                  TR.REQ= BIT07
1152    000100                  INT.EN= BIT06
1153    000040                  READY=  BIT05
1154    000020                  ILBS=   BIT04
1155    000010                  FUNC2=  BIT03
1156    000004                  FUNC1=  BIT02
1157    000002                  FUNC0=  BIT01
1158    000001                  GO=     BIT00
1159    000016                  FUNCTION=       FUNC2+FUNC1+FUNC0
1160
```

```
                                .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
```

```
1232  001224  000000                RDTRYS: .WORD   0               ;KEEPS COUNT OF REREADS
1233  001226  000000                WRTRYS: .WORD   0               ;KEEPS COUNT OF REWRITES
1234  001230  000000  000000        BYTNUM: .WORD   0,0             ;THE NUMBER OF BYTES "READ" OR "WRITTEN
1235                                                                ;DURING AN OPERATION
1236  001234  000017                LASTBK: .WORD   15.             ;WILL CONTAIN THE # OF THE LAST BLOCK
1237                                                                ;AFTER A "FORMAT" OR "WRITE" PASS
1238  001236  000011                LASTFL: .WORD   9.              ;WILL CONTAIN THE # OF THE LAST FILE
1239                                                                ;AFTER A "FORMAT" OR "WRITE" PASS
1240
1241
1242  001240  177500                TACSL:  177500                  ;ADDRESS OF TACS
1243  001242  177502                TADBL:  177502                  ;ADDRESS OF TADB
1244  001244  000260  000262        TAVEC:  260,262                 ;TA11 VECTOR ADDRESS
1245  001250  000300                TAPRIO: 300                     ;TA11 BR LEVEL 6
1246  001252  000000  000000        DRVKEY: 0,0
1247  001256  001252                DRVPNT: DRVKEY
1248  001260  000003                MAXRDS: .WORD   3               ;MAX REREADS BEFORE CALLING IT A HARD ERROR
1249  001262  000003                MAXERR: .WORD   3               ;MAX HARD ERRORS ALLOWED
1250  001264  000022                MAXEOT: .WORD   18.             ;NUMBER OF TAPE PASSES BEFORE END-OF-TEST
1251  001266  000000                PASCNT: .WORD   0               ;COUNT # OF TAPE PASSES
1252  001270  001274  000000        PSCNTL: .WORD   FORPAS,0        ;CONTROLS THE TYPE OF PASS
1253  001274  000001                FORPAS: .WORD   1               ;  1 FORMAT PASS
1254  001276  000003                RD1PAS: .WORD   3               ;  3 READONLY PASSES
1255  001300  000001                WRTPAS: .WORD   1               ;  1 WRITEONLY PASS
1256  001302  000004                RD2PAS: .WORD   4               ;  4 READONLY PASSES
1257  001304     000                FILE:   .BYTE   0               ;FILE NUMBER
1258  001305     377                        .BYTE   377             ;1'S COMPLEMENT ON FILE NUMBER
1259  001306     000                BLOCK:  .BYTE   0               ;BLOCK NUMBER
1260  001307     377                        .BYTE   377             ;1'S COMPLEMENT OF BLOCK NUMBER
1261  001310  000020                FILESZ: .WORD   16.             ;NUMBER OF BLOCKS PER FILE
```

```
                                ;////////////////////////////////////////////////////////////////////////////////
                                .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
```

```
1305
1306                                    ;//////////////////////////////////////////////////////////////////////////////
1307
1308                                    .SBTTL          DATA PATTERNS
1309
1310  001412  314  063  314  PAT0:   .BYTE   314,063,314,063,146,231,146,231
1311  001415  063  146  231
1312  001420  146  231
1313  001422  314  231  063  PAT1:   .BYTE   314,231,063,146,314,231,063,146
1314  001425  146  314  231
1315  001430  063  146
1316  001432  001  002  004  PAT2:   .BYTE   001,002,004,010,020,040,100,200
1317  001435  010  020  040
1318  001440  100  200
1319  001442  177  277  337  PAT3:   .BYTE   177,277,337,357,367,373,375,376
1320  001445  357  367  373
1321  001450  375  376
1322  001452  252  125  252  PAT4:   .BYTE   252,125,252,125,252,125,252,125
1323  001455  125  252  125
1324  001460  252  125
1325  001462  000  111  222  PAT5:   .BYTE   000,111,222,333,044,155,266,377
1326  001465  333  044  155
1327  001470  266  377
1328  001472  000  044  111  PAT6:   .BYTE   000,044,111,155,222,266,333,377
1329  001475  155  222  266
1330  001500  333  377
1331  001502  000  222  044  PAT7:   .BYTE   000,222,044,266,111,333,155,377
1332  001505  266  111  333
1333  001510  155  377
1334  001512  001  003  007  PAT8:   .BYTE   001,003,007,017,037,077,177,377
1335  001515  017  037  077
1336  001520  177  377
1337  001522  376  374  370  PAT9:   .BYTE   376,374,370,360,340,300,200,000
1338  001525  360  340  300
1339  001530  200  000
1340  001532  001  376  002  PAT10:  .BYTE   001,376,002,375,004,373,010,367
1341  001535  375  004  373
1342  001540  010  367
1343  001542  020  357  040  PAT11:  .BYTE   020,357,040,337,100,277,200,177
1344  001545  337  100  277
1345  001550  200  177
1346  001552  000  000  000  PAT12:  .BYTE   000,000,000,000,000,000,000,000
1347  001555  000  000  000
1348  001560  000  000
1349  001562  377  377  377  PAT13:  .BYTE   377,377,377,377,377,377,377,377
1350  001565  377  377  377
1351  001570  377  377
1352  001572  000  377  000  PAT14:  .BYTE   000,377,000,377,000,000,377,377
1353  001575  377  000  000
1354  001600  377  377
1355  001602  017  360  207  PAT15:  .BYTE   017,360,207,170,303,074,341,036
1356  001605  170  303  074
1357  001610  341  036
1358
```

```
1359
1360                                    ;////////////////////////////////////////////////////////////////////////////
1361
1362                                    .SBTTL          PARAMETER BLOCK USED WITH ALL FUNCTIONS
1363
1364   001612     000                   PARMBK: .BYTE   0                       ;USED FOR STATUS/ERROR
1365   001613     000                           .BYTE   0                       ;DRIVE # (DRIVE A=0, B=1)
1366   001614  001612                           .WORD   PARMBK                  ;POINTS TO STATUS/ERROR BYTE
1367   001616  013440                           .WORD   BUFFER                  ;FIRST ADDRESS OF DATA BUFFER
1368   001620  000000                           .WORD   0                       ;USED FOR BYTE COUNT.
1369
```

```
                                .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  012220                          EM1     ;DATA ERROR
001624  012511                          DH1     ;PC     FILE    BLOCK   BYTE    GDDAT   BDDAT   GDADR   BDADR
001626  011744                          DT1     ;$ERRPC $REG0   $REG2   BYTNUM  $GDDAT  $BDDAT  $GDADR  $BDADR
001630  012026                          DF1     ;FILE,BLOCK AND BYTE ARE TYPED IN DECIMAL

                                ;ITEM   2
001632  012233                          EM2     ;SYNC ERROR
001634  012607                          DH2     ;       EXPT'D  EXPT'D  RCV'D   RCV'D
                                                ;PC     FILE    BLOCK   FILE    BLOCK
001636  011766                          DT2     ;$ERRPC $REG0   $REG2   $TMP0   $TMP2
001640  012036                          DF2     ;ALL NUMBERS EXCEPT $ERRPC ARE TYPED IN DECIMAL

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

001642                          ITEMS1:         ;ITEMS 101-107

001642  012246                          EM101   ;DRIVE IS OFF-LINE
001644  012724                          DH101   ;PC     FILE    BLOCK   FUNCTION
001646  012002                          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  012270                          EM102   ;DRIVE IS WRITE-LOCK
001654  012724                          DH101   ;PC     FILE    BLOCK   FUNCTION
001656  012002                          DT101   ;$ERRPC $REG0   $REG2   $TMP0
001660  000000                          0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
                                                ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
```

```
1426  001662  012314                EM103   ;CLEAR LEADER ERROR
1427  001664  012724                DH101   ;PC      FILE    BLOCK   FUNCTION
1428  001666  012002                DT101   ;$ERRPC $REG0   $REG2   $TMP0
1429  001670  000000                0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
1430                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
1431
1432  001672  012337                EM104   ;FILE GAP ERROR
1433  001674  012724                DH101   ;PC      FILE    BLOCK   FUNCTION
1434  001676  012002                DT101   ;$ERRPC $REG0   $REG2   $TMP0
1435  001700  000000                0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
1436                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
1437
1438  001702  012356                EM105   ;TIMING ERROR
1439  001704  012724                DH101   ;PC      FILE    BLOCK   FUNCTION
1440  001706  012002                DT101   ;$ERRPC $REG0   $REG2   $TMP0
1441  001710  000000                0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
1442                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
1443
1444  001712  012373                EM106   ;BLOCK CHECK ERROR
1445  001714  012724                DH101   ;PC      FILE    BLOCK   FUNCTION
1446  001716  012002                DT101   ;$ERRPC $REG0   $REG2   $TMP0
1447  001720  000000                0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
1448                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
1449
1450  001722  012415                EM107   ;UNKNOWN INTERRUPT
1451  001724  012724                DH101   ;PC      FILE    BLOCK   FUNCTION
1452  001726  012002                DT101   ;$ERRPC $REG0   $REG2   $TMP0
1453  001730  000000                0       ;FILE AND BLOCK WILL BE TYPED IN DECIMAL
1454                                        ;$TMP0 WILL BE TYPED AS A FUNCTION NAME
1455
1456                        ;;**************************************************************
1457                        ;;**************************************************************
1458                        ;;**************************************************************
1459
1460  001732                ITEMS2:         ;ITEMS 201-202
1461
1462  001732  012437                EM201   ;TA11 FAILED TO RESPOND
1463  001734  012765                DH201   ;PC      TACS
1464  001736  012014                DT201   ;$ERRPC TACS
1465  001740  000000                0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
1466
1467  001742  012466                EM202   ;NO DRIVES AVAILABLE
1468  001744  013002                DH202   ;PC
1469  001746  012022                DT202   ;$ERRPC
1470  001750  000000                0       ;
1471
```

```
1472                                    ;////////////////////////////////////////////////////////////////////////
1473                                    ;////////////////////////////////////////////////////////////////////////
1474                                    ;;**********************************************************************
1475
1476                                    ;BEGIN1 IS FOR NORMAL START
1477                                    ;BEGIN2 IS FOR DRIVE SELECTION
1478                                    ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
1479
1480                                    ;;**********************************************************************
1481
1482  001752  005005                   BEGIN1: CLR     R5                      ;NORMAL START
1483  001754  012737  041101  001252           MOV     #"AB,@#DRVKEY
1484  001762  122737  000005  000041           CMPB    #5,@#41                 ;CASSETTE DDP?
1485  001770  001012                           BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
1486  001772  022737  000260  001244           CMP     #260,@#TAVEC            ;STANDARD VECTOR?
1487  002000  001006                           BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
1488  002002  000403                           BR      BEGIN3                  ;GET DRIVES AND ADDRESSES
1489  002004  012705  000001          BEGIN2: MOV     #1,R5                   ;ASK FOR DRIVES FLAG
1490  002010  000402                           BR      BGNCMN                  ;BEGIN COMMON CODE
1491  002012  012705  000002          BEGIN3: MOV     #2,R5                   ;ASK FOR DRIVES AND ADDRESSES
1492  002016                           BGNCMN:
1493                                   .SBTTL  INITIALIZE THE COMMON TAGS
1494                                   ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1495  002016  012706  001100                   MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1496  002022  005026                           CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1497  002024  022706  001140                   CMP     #SWR,R6 ;;DONE?
1498  002030  001374                           BNE     .-6             ;;LOOP BACK IF NO
1499  002032  012706  001100                   MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1500                                   ;;INITIALIZE A FEW VECTORS
1501  002036  012737  004750  000020           MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1502  002044  012737  000340  000022           MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1503  002052  012737  005050  000030           MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1504  002060  012737  000340  000032           MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1505  002066  012737  011512  000034           MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1506  002074  012737  000340  000036           MOV     #340,@#TRAPVEC+2;LEVEL 7
1507  002102  012737  011562  000024           MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1508  002110  012737  000340  000026           MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1509  002116  016767  002314  002304           MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
1510  002124  012767  002124  176754           MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1511                                   ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1512                                   ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1513  002132  013746  000004                   MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1514  002136  012737  002172  000004           MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
1515  002144  012767  177570  176766           MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1516  002152  012767  177570  176762           MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1517  002160  022777  177777  176752           CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1518  002166  001012                           BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1519                                                                   ;;AND  THE HARDWARE SWR IS NOT = -1
1520  002170  000403                           BR      65$             ;;BRANCH IF NO TIMEOUT
1521  002172  012716  002200          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
1522  002176  000002                           RTI
1523  002200  012767  000176  176732  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1524  002206  012767  000174  176726           MOV     #DISPREG,DISPLAY
1525  002214  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1526
1527                                   .SBTTL  TYPE PROGRAM NAME
```

```
1528                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1529  002220  005227  177777                        INC     #-1                 ;;FIRST TIME?
1530  002224  001036                            BNE     67$                 ;;BRANCH IF NO
1531  002226  022737  004716  000042                CMP     #$ENDAD,@#42        ;;ACT-11?
1532  002234  001432                            BEQ     67$                 ;;BRANCH IF YES
1533  002236  104400  002274                        TYPE    ,68$                ;;TYPE ASCIZ STRING
1534                                      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1535  002242  005737  000042                        TST     @#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
1536  002246  001006                            BNE     69$                 ;;BRANCH IF YES
1537  002250  026727  176664  000176                CMP     SWR,#SWREG          ;;SOFTWARE SWITCH REG SELECTED?
1538  002256  001005                            BNE     70$                 ;;BRANCH IF NO
1539  002260  104404                            GTSWR                       ;;GET SOFT-SWR SETTINGS
1540  002262  000403                            BR      70$
1541  002264  112767  000001  176642  69$:     MOVB    #1,$AUTOB           ;;SET AUTO-MODE INDICATOR
1542  002272                                    70$:
1543  002272  000413                            BR      67$                 ;;GET OVER THE ASCIZ
1544                                      ;;68$:  .ASCIZ  <CRLF>/MAINDEC-11-DZTAE-B/<CRLF>
1545  002322                                    67$:
1546                                      ;;**************************************************************
1547                                      ;;**************************************************************
1548
1549                                      ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
1550                                      ;
1551                                      ;       R5=2    ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
1552                                      ;       R5=1    ASK FOR DRIVE(S)
1553                                      ;       R5=0    DON'T ASK FOR ANYTHING
1554                                      ;
1555                                      ;;**************************************************************
1556  002322  010504                    BEGINX: MOV     R5,R4                   ;COPY R5
1557  002324  005305                            DEC     R5                      ;ASK FOR DRIVES?
1558  002326  002406                            BLT     CHKADR                  ;BR IF NO
1559  002330  004737  007610                    JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
1560  002334  005305                            DEC     R5                      ;ASK FOR ADDRESSES?
1561  002336  002402                            BLT     CHKADR                  ;BR IF NO
1562  002340  004737  007720                    JSR     PC,@#ASKADR             ;GO GET TA11 ADDRESSES
1563                                      ;;**************************************************************
1564                                      ;;**************************************************************
1565
1566                                      ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
1567                                      ;
1568                                      ;I.     TIMEOUT OCCURRED
1569                                      ;       A. TYPE ERROR MESSAGE
1570                                      ;       B. EXAMINE R4
1571                                      ;               1. R4>0 GOTO BEGINX
1572                                      ;               2. R4=0 EXAMINE (42)
1573                                      ;                       A. (42)=0 GOTO BEGINX
1574                                      ;                       B. (42)>0 GOTO $ENDAD
1575                                      ;
1576                                      ;II.    TIMEOUT DIDN'T OCCUR
1577                                      ;       A. CONTINUE
1578                                      ;
1579                                      ;;**************************************************************
1580  002344  012737  002362  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE OF TIMEOUTS
1581  002352  005000                            CLR     R0                      ;USE AS A SWITCH
1582  002354  005777  176660                    TST     @TACSL                  ;SEE IF TA11 RESPONDS
1583  002360  000402                            BR      2$                      ;BR IF NO TIMEOUT
```

```
1584  002362  005200                    1$:   INC     R0                      ;COME HERE ON TIMEOUT
1585  002364  022626                          CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
1586  002366  012737  000006  000004    2$:   MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
1587  002374  005700                          TST     R0                      ;DID A TIMEOUT OCCUR?
1588  002376  001412                          BEQ     3$                      ;BR IF NO
1589  002400  104201                          ERROR   201                     ;TA11 FAILED TO RESPOND
1590  002402  012705  000002                  MOV     #2,R5                   ;DRIVES & ADDRESSES
1591  002406  005704                          TST     R4                      ;OPERATOR INPUTS?
1592  002410  001344                          BNE     BEGINX                  ;BR IF YES
1593  002412  013700  000042                  MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
1594  002416  001741                          BEQ     BEGINX                  ;BR IF NO MONITOR
1595  002420  000137  004716                  JMP     @#$ENDAD                ;GO TO END
1596  002424  012777  007046  176612    3$:   MOV     #CASINT,@TAVEC          ;SETUP CASSETTE INTERRUPT VECTOR
1597  002432  005077  176610                  CLR     @TAVEC+2
```

```
1598                            ;;*****************************************************************
1599                            ;;*****************************************************************
1600
1601                            ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
1602                            ;
1603                            ;I.     DESIRED DRIVES CAN NOT BE TESTED
1604                            ;       A. TYPE ERROR MESSAGE
1605                            ;       B. EXAMINE R4
1606                            ;               1. R4>0 GOTO BEGINX
1607                            ;               2. R4=0 EXAMINE (42)
1608                            ;                       A. (42)=0 GOTO BEGINX
1609                            ;                       B. (42)>0 GOTO $ENDAD
1610                            ;
1611                            ;II.    BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
1612                            ;       A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
1613                            ;       B. TYPE THE DRIVE TO BE TESTED
1614                            ;       C. CONTINUE IN PROGRAM
1615                            ;
1616                            ;III.   DESIRED DRIVE(S) CAN BE TESTED
1617                            ;       A. TYPE THE DRIVE(S) TO BE TESTED
1618                            ;       B. CONTINUE IN PROGRAM
1619                            ;
1620                            ;;*****************************************************************
1621  002436  012700  001252   CHKDRV: MOV     #DRVKEY,R0              ;PICKUP ADDRESS OF ASCII DRIVE KEY
1622  002442  004737  010164           JSR     PC,@#EXAM               ;GO EXAMINE FIRST DRIVE
1623  002446  000410                   BR      1$                      ;OK TO TEST---GO CHECK NEXT
1624  002450  116010  000001           MOVB    1(R0),(R0)              ;REPLACE 1ST WITH 2ND
1625  002454  001412                   BEQ     2$                      ;BR IF NO 2ND DRIVE SELECTED
1626  002456  004737  010164           JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
1627  002462  000407                   BR      2$                      ;OK TO TEST
1628  002464  005010                   CLR     (R0)                    ;CLEAR DRIVE CODES
1629  002466  000405                   BR      2$
1630  002470  005200           1$:     INC     R0                      ;POINT TO 2ND
1631  002472  004737  010164           JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
1632  002476  000401                   BR      2$                      ;OK TO TEST
1633  002500  105010                   CLRB    (R0)                    ;CLEAR 2ND
1634  002502  012700  001252   2$:     MOV     #DRVKEY,R0              ;RESET ADDRESS POINTERS
1635  002506  010037  001256           MOV     R0,@#DRVPNT
1636  002512  121060  000001           CMPB    (R0),1(R0)              ;1ST = 2ND?
1637  002516  001002                   BNE     3$                      ;BR IF NO
1638  002520  105060  000001           CLRB    1(R0)                   ;YES---CLEAR 2ND
1639  002524  005710           3$:     TST     (R0)                    ;ANY DRIVES?
1640  002526  001424                   BEQ     5$                      ;BR IF NO
1641  002530  112037  001613           MOVB    (R0)+,@#PARMBK+1        ;SETUP TO TYPE THIS DRIVE
1642  002534  104400  001203           TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
1643  002540  106237  001613           ASRB    @#PARMBK+1              ;ADJUST FOR TYPING
1644  002544  004737  007554           JSR     PC,@#TPDRV              ;GO TYPE DRIVE
1645  002550  111037  001613           MOVB    (R0),@#PARMBK+1         ;GET 2ND
1646  002554  001406                   BEQ     4$                      ;BR IF NONE
1647  002556  104400  013174           TYPE    ,MSG9                   ;"AND"
1648  002562  106237  001613           ASRB    @#PARMBK+1              ;ADJUST FOR TYPING
1649  002566  004737  007554           JSR     PC,@#TPDRV              ;GO TYPE THE 2ND DRIVE
1650  002572  104400  013201   4$:     TYPE    ,MSG10                  ;"WILL BE TESTED"<CRLF>
1651  002576  000412                   BR      $START
1652  002600  104202           5$:     ERROR   202                     ;NO DRIVES AVAILABLE
1653  002602  012705  000002           MOV     #2,R5                   ;DRIVES & ADDRESS
```

```
 1654  002606  005704                    TST     R4              ;OPERATOR INPUTS?
 1655  002610  001244                    BNE     BEGINX          ;BR IF YES
 1656  002612  013700  000042            MOV     @#42,R0         ;GET MONITOR RETURN ADDRESS
 1657  002616  001641                    BEQ     BEGINX          ;NO MONITOR
 1658  002620  000137  004716            JMP     @#$ENDAD        ;GO TO END
```

```
1659                                    ;//////////////////////////////////////////////////////////////////////////
1660                                    ;//////////////////////////////////////////////////////////////////////////
1661                                    ;;**********************************************************************
1662                                    ;;**********************************************************************
1663                                    ;
1664                                    ;       1. CLEAR THE VARIABLE STORAGE AREA
1665                                    ;       2. SETUP FOR THE DRIVE THAT IS TO BE TESTED
1666                                    ;       3. SETUP "LASTFL" AND "LASTBK" INCASE OPERATOR SELECTS "READONLY" PASS
1667                                    ;       4. SETUP THE MAX. NUMBER OF RETRYS FOR READS AND WRITES
1668                                    ;       5. SETUP TO START WITH A "FORMAT" PASS
1669                                    ;       6. CLEAR THE PASS COUNTER
1670                                    ;
1671                                    ;;**********************************************************************
1672  002624  012700  001114   START:  MOV     #$ITEMB,R0              ;CLEAR VARIABLE STORAGE AREA
1673  002630  012701  000005           MOV     #$BDDAT-$ITEMB/2,R1
1674  002634  004737  002760           JSR     PC,@#CLEAR                      ;CLEAR AREA
1675  002640  012700  001206           MOV     #SOFTNM,R0
1676  002644  012701  000014           MOV     #LASTFL-SOFTNM/2,R1
1677  002650  004737  002760           JSR     PC,@#CLEAR
1678  002654  013701  001256           MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
1679  002660  112100                   MOVB    (R1)+,R0                ;SETUP THE DRIVE
1680  002662  006200                   ASR     R0
1681  002664  042700  177776           BIC     #^CBIT00,R0
1682  002670  110037  001613           MOVB    R0,@#PARMBK+1
1683  002674  105711                   TSTB    (R1)                    ;END OF DRIVE TABLE?
1684  002676  001002                   BNE     2$                      ;BR IF NO
1685  002700  012701  001252           MOV     #DRVKEY,R1              ;RESET DRIVE TABLE POINTER
1686  002704  010137  001256   2$:     MOV     R1,@#DRVPNT             ;RESTORE THE DRIVE POINTER
1687  002710  012737  000011  001236   MOV     #9.,@#LASTFL            ;SET "LAST FILE" & "LAST BLOCK" INCASE
1688  002716  012737  000017  001234   MOV     #15.,@#LASTBK           ;OPERATOR SELECTS "READONLY" PASS
1689  002724  013737  001260  001224   MOV     @#MAXRDS,@#RDTRYS       ;SETUP MAX. # OF REREAD TRYS
1690  002732  013737  001262  001226   MOV     @#MAXERR,@#WRTRYS       ;SETUP MAX. # OF REWRITE TRYS
1691  002740  012737  001274  001270   MOV     #FORPAS,@#PSCNTL        ;DO A FORMAT PASS
1692  002746  005037  001272           CLR     @#PSCNTL+2
1693  002752  005037  001266           CLR     @#PASCNT                ;CLEAR THE PASS COUNT
1694  002756  000404                   BR      LOOPER                  ;GET AROUND CLEAR
1695  002760  005020           CLEAR:  CLR     (R0)+
1696  002762  005301                   DEC     R1
1697  002764  003375                   BGT     CLEAR
1698  002766  000207                   RTS     PC
```

```
1699                                    ;;*****************************************************************
1700                                    ;;*****************************************************************
1701                                    ;
1702                                    ;I.     CHECK SWR<7>
1703                                    ;       A. SWR<7>=1 PERFORM AS PER SWR<1:0>
1704                                    ;               1. 00=FORMAT PASS
1705                                    ;               2. 01=READ ONLY PASS
1706                                    ;               3. 10=WRITE ONLY PASS
1707                                    ;               4. 11=READ ONLY PASS
1708                                    ;
1709                                    ;       B. SWR<7>=0
1710                                    ;               1. UPDATE PASCNT
1711                                    ;               2. PERFORM AS PER (PSCNTL+2)
1712                                    ;                       A. FORMAT
1713                                    ;                       B. READONLY
1714                                    ;                       C. WRITEONLY
1715                                    ;                       D. READONLY
1716                                    ;
1717                                    ;;*****************************************************************
1718  002770  117737  176144  001172  LOOPER: MOVB    @SWR,@#$TMP0            ;IF SWR<7>=1 ; DO PER SWR<1:0>
1719  002776  142737  177574  001172          BICB    #^C<BIT07!BIT01!BIT00>,@#$TMP0
1720  003004  100432                          BMI     3$                      ;BR IF SWR<7>=1
1721  003006  023777  001266  176254  1$:     CMP     @#PASCNT,@PSCNTL        ;IS THE COUNTER AT MAX.?
1722  003014  002421                          BLT     2$                      ;BR IF NO
1723  003016  005037  001266                  CLR     @#PASCNT                ;RESET THE COUNTER
1724  003022  062737  000002  001270          ADD     #2,@#PSCNTL             ;MOVE TO THE NEXT PASS TYPE
1725  003030  005237  001272                  INC     @#PSCNTL+2
1726  003034  023727  001270  001304          CMP     @#PSCNTL,#RD2PAS+2      ;TIME TO RESET PASS TYPE?
1727  003042  001361                          BNE     1$                      ;BR IF NO
1728  003044  012737  001274  001270          MOV     #FORPAS,@#PSCNTL        ;YES---RESET THE PASS CONTROL WORDS
1729  003052  005037  001272                  CLR     @#PSCNTL+2
1730  003056  000753                          BR      1$                      ;GO CHECK THE COUNT
1731  003060  005237  001266          2$:     INC     @#PASCNT                ;COUNT THIS PASS
1732  003064  013737  001272  001172          MOV     @#PSCNTL+2,@#$TMP0      ;PICKUP THE TYPE OF PASS TO DO NEXT
1733  003072                          3$:
1734  003072  005046                          CLR     -(SP)
1735  003074  113716  001172                  MOVB    @#$TMP0,(SP)            ;PICKUP THE DISPATCH INDEX
1736  003100  106316                          ASLB    (SP)                    ;POSITION THE INDEX
1737  003102  106316                          ASLB    (SP)                    ;    BEFORE USING IT
1738  003104  062607                          ADD     (SP)+,PC                ;GO TO THE ROUTINE
1739  003106  000137  003126                  JMP     @#FORMAT
1740  003112  000137  003556                  JMP     @#READONLY
1741  003116  000137  004132                  JMP     @#WRITEONLY
1742  003122  000137  003556                  JMP     @#READONLY
1743                                    ;////////////////////////////////////////////////////////////////
1744                                    ;////////////////////////////////////////////////////////////////
1745                                    ;;*****************************************************************
1746                                    ;;*****************************************************************
```

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

                        .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 $EOP
                        ;               B. NO DATA ERROR --- GOTO 8.
                        ;       8. UPDATE THE BLOCK NUMBER
                        ;       9. END OF FILE?
                        ;               A. NO --- GOTO 3.
                        ;               B. YES
                        ;                       1.) UPDATE FILE NUMBER
                        ;                       2.) RESET BLOCK NUMBER TO 0
                        ;                       3.) GOTO 2.
                        ;
                        ;;**********************************************************************
                                                                                ,
003126  012737  003126  001106  FORMAT: MOV     #FORMAT,@#$LPADR        ;SETUP SCOPE LOOP ADDRESS
003134  104400  013222                  TYPE    ,MSG11                  ;"*** FORMAT ***"
003140  004737  007554                  JSR     PC,@#TPDRV              ;GO TYPE THE DRIVE
003144  104400  001203                  TYPE    ,$CRLF
003150  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
003152     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003154  001306                          .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
003156  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
003160     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003162  001304                          .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////
003164  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003170  004737  006746                  JSR     PC,@#REWIND             ;GO TO REWIND
003174  105737  001612          1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003200  001775                          BEQ     1$
003202  100022                          BPL     4$                      ;BR IF NO ERROR
003204  004037  005526                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
```

```
1803  003210  000137  004406                    JMP     @#$EOP                  ;GO TO END OF PROGRAM
1804                                    ;////////////////////////////////////////////////////////////
1805                                    ;       "WRITE-FILL-GAP"
1806                                    ;////////////////////////////////////////////////////////////
1807  003214                            2$:
1808  003214  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1809  003220  004737  006712                   JSR     PC,@#WFG                ;GO TO WFG
1810  003224  105737  001612            3$:    TSTB    @#PARMBK                ;WAIT ON FLAG
1811  003230  001775                           BEQ     3$
1812  003232  100006                           BPL     4$                      ;BR IF NO ERROR
1813  003234  004037  005526                   JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
1814  003240  000137  004406                   JMP     @#$EOP                  ;GO TO END OF PROGRAM
1815  003244  000137  004360                   JMP     @#CKEOTS
1816                                    ;////////////////////////////////////////////////////////////
1817                                    ;       "WRITE" A BLOCK OF DATA
1818                                    ;////////////////////////////////////////////////////////////
1819  003250  012737  003256  001106    4$:    MOV     #5$,@#$LPADR            ;SETUP SCOPE LOOP ADDRESS
1820  003256  004737  006062            5$:    JSR     PC,@#SETUPW             ;GO SETUP FOR A WRITE
1821  003262  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1822  003266  004737  006716                   JSR     PC,@#WRITE              ;GO TO WRITE
1823  003272  105737  001612            6$:    TSTB    @#PARMBK                ;WAIT ON FLAG
1824  003276  001775                           BEQ     6$
1825  003300  100006                           BPL     7$                      ;BR IF NO ERROR
1826  003302  004037  005526                   JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
1827  003306  000137  004406                   JMP     @#$EOP                  ;GO TO END OF PROGRAM
1828  003312  000137  004360                   JMP     @#CKEOTS
1829                                    ;////////////////////////////////////////////////////////////
1830                                    ;       "BACK-SPACE-BLOCK-GAP"
1831                                    ;////////////////////////////////////////////////////////////
1832  003316                            7$:
1833  003316  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1834  003322  004737  006732                   JSR     PC,@#BSBG               ;GO TO BSBG
1835  003326  105737  001612            8$:    TSTB    @#PARMBK                ;WAIT ON FLAG
1836  003332  001775                           BEQ     8$
1837  003334  100004                           BPL     9$                      ;BR IF NO ERROR
1838  003336  004037  005526                   JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
1839  003342  000137  004406                   JMP     @#$EOP                  ;GO TO END OF PROGRAM
1840                                    ;////////////////////////////////////////////////////////////
1841                                    ;       "READ" A BLOCK OF DATA
1842                                    ;////////////////////////////////////////////////////////////
1843  003346  004737  006210            9$:    JSR     PC,@#SETUPR             ;GO SETUP FOR A READ
1844  003352  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
1845  003356  004737  006722                   JSR     PC,@#READ               ;GO TO READ
1846  003362  105737  001612            10$:   TSTB    @#PARMBK                ;WAIT ON FLAG
1847  003366  001775                           BEQ     10$
1848  003370  100004                           BPL     11$                     ;BR IF NO ERROR
1849  003372  004037  005526                   JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
1850  003376  000137  004406                   JMP     @#$EOP
1851  003402  000004                    11$:   SCOPE
1852                                    ;////////////////////////////////////////////////////////////
1853                                    ;       CHECK FOR SYNC ERROR
1854                                    ;////////////////////////////////////////////////////////////
1855  003404  004737  006244                   JSR     PC,@#SYNCK              ;SYNC ERROR?
1856  003410  000403                           BR      12$                     ;RETURN HERE IF NO
1857  003412  104002                           ERROR   2                       ;SYNC ERROR
1858  003414  000137  004406                   JMP     @#$EOP
```

```
                                        ;//////////////////////////////////////////////////////////////////////////
                                        ;       CHECK FOR DATA ERROR IF SOFT REREAD
                                        ;//////////////////////////////////////////////////////////////////////////
003420  004737  006346        12$:    JSR     PC,@#DATCMP             ;CHECK THE DATA
003424  000430                        BR      15$                     ;RETURN HERE IF DATA IS GOOD
003426  104001                        ERROR   1                       ;DATA ERROR
003430  004037  006540                JSR     R0,@#CNTSFT             ;COUNT SOFT ERROR
003434  000730                        BR      7$                      ;GO REREAD
003436  004037  006576                JSR     R0,@#CNTHRD             ;COUNT HARD ERROR
003442  000402                        BR      13$                     ;REWRITE
003444  000137  004406                JMP     @#$EOP                  ;TO MANY HARD ERRORS
                                        ;//////////////////////////////////////////////////////////////////////////
                                        ;       IF HARD ERROR REWRITE
                                        ;//////////////////////////////////////////////////////////////////////////
003450  005337  001226        13$:    DEC     @#WRTRYS                ;TRY TO REWRITE THIS BLOCK?
003454  002414                        BLT     15$                     ;BR IF NO
003456  012700  001612                MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003462  004737  006732                JSR     PC,@#BSBG               ;GO TO BSBG
003466  105737  001612        14$:    TSTB    @#PARMBK                ;WAIT ON FLAG
003472  001775                        BEQ     14$
003474  100265                        BPL     4$                      ;GO START A WRITE IF NO ERROR
003476  004037  005526                JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003502  000137  004406                JMP     @#$EOP
003506  013737  001260  001224  15$:  MOV     @#MAXRDS,@#RDTRYS       ;RESET THE REREAD COUNT
003514  013737  001262  001226        MOV     @#MAXERR,@#WRTRYS       ;RESET THE REWRITE COUNT
                                        ;//////////////////////////////////////////////////////////////////////////
                                        ;       UPDATE BLOCK # & FILE #
                                        ;//////////////////////////////////////////////////////////////////////////
003522  062737                        ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
003524     001     377                .BYTE   1,-1
003526  001306                        .WORD   BLOCK
003530  123737  001310  001306        CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
003536  003244                        BGT     4$                      ;BR IF NO--GO START A WRITE
003540  062737                        ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
003542     001     377                .BYTE   1,-1
003544  001304                        .WORD   FILE
003546  012737                        MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
003550     000     377                .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003552  001306                        .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
003554  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 $EOP
                        ;               B. NO DATA ERROR --- GOTO 5.
                        ;       5. UPDATE THE BLOCK NUMBER
                        ;       6. END OF FILE?
                        ;               A. NO --- GOTO 2.
                        ;               B. YES
                        ;                       1.) UPDATE FILE NUMBER
                        ;                       2.) RESET BLOCK NUMBER TO 0
                        ;                       3.) SPACE FORWARD FILE GAP
                        ;                       4.) GOTO 2.
                        ;
                        ;;**********************************************************************
003556                  READONLY:
003556  012737  003556  001106         MOV     #READONLY,@#$LPADR      ;SETUP SCOPE LOOP ADDRESS
003564  104400  013241                 TYPE    ,MSG12                  ;"*** READ    ***"
003570  004737  007554                 JSR     PC,@#TPDRV              ;GO TYPE DRIVE TO BE TESTED
003574  104400  001203                 TYPE    $CRLF
003600  012737                         MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
003602     000     377                 .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003604  001306                         .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
003606  012737                         MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
003610     000     377                 .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
003612  001304                         .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////
003614  012700  001612                 MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003620  004737  006746                 JSR     PC,@#REWIND             ;GO TO REWIND
003624  105737  001612         1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003630  001775                         BEQ     1$
003632  100004                         BPL     2$                      ;BR IF NO ERROR
003634  004037  005526                 JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003640  000137  004406                 JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                        ;////////////////////////////////////////////////////////////////////////
```

```
                                        ;       SETUP FOR "READ"
                                        ;////////////////////////////////////////////////////////////////////////
003644  004737  006210                2$:     JSR     PC,@#SETUPR             ;SETUP FOR "READ"
003650  000417                                BR      5$                      ;GO START A READ
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       "BSBG"--IF SCOPE OR SOFT DATA ERROR
                                        ;////////////////////////////////////////////////////////////////////////
003652  012737  003710  001106          3$:     MOV     #5$,@#$LPADR            ;SETUP THE SCOPE LOOP ADDRESS
003660  012700  001612                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003664  004737  006732                        JSR     PC,@#BSBG               ;GO TO BSBG
003670  105737  001612                4$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003674  001775                                BEQ     4$
003676  100004                                BPL     5$                      ;GO TRY TO REREAD
003700  004037  005526                        JSR     R0,@#CSRERR             ;GO CHECK CSR ERROR
003704  000137  004406                        JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       "READ" BLOCK OF DATA
                                        ;////////////////////////////////////////////////////////////////////////
003710  012737  003652  001106          5$:     MOV     #3$,@#$LPADR            ;SET THE LOOP ADDRESS
003716  012700  001612                        MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
003722  004737  006722                        JSR     PC,@#READ               ;GO TO READ
003726  105737  001612                6$:     TSTB    @#PARMBK                ;WAIT ON FLAG
003732  001775                                BEQ     6$
003734  100004                                BPL     7$                      ;BR IF NO ERROR
003736  004037  005526                        JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
003742  000137  004406                        JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       CHECK FOR SYNC ERROR
                                        ;////////////////////////////////////////////////////////////////////////
003746  004737  006244                7$:     JSR     PC,@#SYNCK              ;GO CHECK FOR SYNC ERROR
003752  000403                                BR      8$                      ;RETURN HERE IF NO ERROR
003754  104002                                ERROR   2                       ;SYNC ERROR
003756  000137  004406                        JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       CHECK FOR DATA ERROR
                                        ;////////////////////////////////////////////////////////////////////////
003762  004737  006346                8$:     JSR     PC,@#DATCMP             ;GO CHECK THE DATA
003766  000411                                BR      9$                      ;RETURN HERE IF NO ERROR
003770  104001                                ERROR   1                       ;DATA ERROR
003772  004037  006540                        JSR     R0,@#CNTSFT             ;COUNT SOFT ERROR
003776  000725                                BR      3$                      ;TRY TO REREAD
004000  004037  006576                        JSR     R0,@#CNTHRD             ;COUNT HARD ERROR
004004  000402                                BR      9$                      ;MOVE TO NEXT BLOCK
004006  000137  004406                        JMP     @#$EOP                  ;TO MANY HARD ERRORS
004012  000004                        9$:     SCOPE
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       IF LAST FILE OF LAST BLOCK GOTO CKEOTS
                                        ;////////////////////////////////////////////////////////////////////////
004014  013737  001260  001224                MOV     @#MAXRDS,@#RDTRYS       ;RESET THE REREAD TRYS
004022  123737  001236  001304                CMPB    @#LASTFL,@#FILE         ;LAST FILE
004030  003007                                BGT     10$                     ;BR IF NO
004032  123737  001234  001306                CMPB    @#LASTBK,@#BLOCK        ;LAST BLOCK
004040  003003                                BGT     10$                     ;BR IF NO
004042  005237  001212                        INC     @#EOTS                  ;COUNT END-OF-TAPE
004046  000544                                BR      CKEOTS                  ;CHECK FOR END-OF-PASS
                                        ;////////////////////////////////////////////////////////////////////////
```

```
2011                                    ;       UPDATE BLOCK # & FILE #
2012                                    ;///////////////////////////////////////////////////////////////////////
2013   004050                           10$:
2014   004050  062737                           ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
2015   004052     001     377                   .BYTE   1,-1
2016   004054  001306                           .WORD   BLOCK
2017   004056  123737  001310  001306           CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
2018   004064  003267                           BGT     2$                      ;BR IF NO
2019   004066  062737                           ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
2020   004070     001     377                   .BYTE   1,-1
2021   004072  001304                           .WORD   FILE
2022   004074  012737                           MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
2023   004076     000     377                   .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
2024   004100  001306                           .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
2025                                   ;///////////////////////////////////////////////////////////////////////
2026                                   ;       "SPACE-FORWARD-FILE-GAP"
2027                                   ;///////////////////////////////////////////////////////////////////////
2028   004102  012700  001612                   MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
2029   004106  004737  006736                   JSR     PC,@#SFFG               ;GO TO SFFG
2030   004112  105737  001612           11$:    TSTB    @#PARMBK                ;WAIT ON FLAG
2031   004116  001775                           BEQ     11$
2032   004120  100251                           BPL     2$                      ;GO READ
2033   004122  004037  005526                   JSR     R0,@#CSRERR             ;GO CHECK CSR ERROR
2034   004126  000137  004406                   JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
2035
```

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

                        .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.
                        ;
                        ;;**********************************************************************

004132                  WRITEONLY:
004132  012737  004132  001106          MOV     #WRITEONLY,@#$LPADR     ;SETUP SCOPE LOOP ADDRESS
004140  104400  013260                  TYPE    ,MSG13                  ;"*** WRITE  ***"
004144  004737  007554                  JSR     PC,@#TPDRV              ;TYPE DRIVE TO BE TESTED
004150  104400  001203                  TYPE    ,$CRLF
004154  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
004156     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
004160  001306                          .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
004162  012737                          MOV     (PC)+,@(PC)+            ;INITIALIZE THE FILE NUMBER
004164     000     377                  .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
004166  001304                          .WORD   FILE                    ;TO THE 1'S COMP. OF BYTE 0
                        ;////////////////////////////////////////////////////////////////////////
                        ;       "REWIND" TO "BOT"
                        ;////////////////////////////////////////////////////////////////////////
004170  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
004174  004737  006746                  JSR     PC,@#REWIND             ;GO TO REWIND
004200  105737  001612          1$:     TSTB    @#PARMBK                ;WAIT ON FLAG
004204  001775                          BEQ     1$
004206  100022                          BPL     4$                      ;BR IF NO ERROR
004210  004037  005526                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
004214  000137  004406                  JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
                        ;////////////////////////////////////////////////////////////////////////
                        ;       "WRITE-FILE-GAP"
                        ;////////////////////////////////////////////////////////////////////////
004220                  2$:
004220  012700  001612                  MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
004224  004737  006712                  JSR     PC,@#WFG                ;GO TO WFG
004230  105737  001612          3$:     TSTB    @#PARMBK                ;WAIT ON FLAG
004234  001775                          BEQ     3$
004236  100006                          BPL     4$                      ;BR IF NO ERROR
004240  004037  005526                  JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
```

```
2092  004244  000137  004406                JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
2093  004250  000137  004360                JMP     @#CKEOTS
2094                                ;////////////////////////////////////////////////////////////
2095                                ;       "WRITE" A BLOCK OF DATA
2096                                ;////////////////////////////////////////////////////////////
2097  004254  004737  006062          4$:   JSR     PC,@#SETUPW             ;GO SETUP FOR "WRITE"
2098  004260  012737  004266  001106        MOV     #5$,@#$LPADR            ;SETUP SCOPE LOOP ADDRESS
2099  004266                          5$:
2100  004266  012700  001612                MOV     #PARMBK,R0              ;R0=1ST ADDRESS OF PARAMETER BLOCK
2101  004272  004737  006716                JSR     PC,@#WRITE              ;GO TO WRITE
2102  004276  105737  001612          6$:   TSTB    @#PARMBK                ;WAIT ON FLAG
2103  004302  001775                       BEQ     6$
2104  004304  100006                       BPL     7$                      ;BR IF NO ERROR
2105  004306  004037  005526                JSR     R0,@#CSRERR             ;GO TO CSR ERROR CHECK
2106  004312  000137  004406                JMP     @#$EOP                  ;GO TO END-OF-PROGRAM
2107  004316  000137  004360                JMP     @#CKEOTS
2108  004322  000004                   7$:   SCOPE
2109                                ;////////////////////////////////////////////////////////////
2110                                ;       UPDATE BLOCK # & FILE #
2111                                ;////////////////////////////////////////////////////////////
2112  004324  062737                       ADD     (PC)+,@(PC)+            ;INCREMENT THE BLOCK NUMBER
2113  004326     001     377               .BYTE   1,-1
2114  004330  001306                       .WORD   BLOCK
2115  004332  123737  001310  001306        CMPB    @#FILESZ,@#BLOCK        ;TIME FOR A FILE GAP?
2116  004340  003345                       BGT     4$                      ;BR IF NO
2117  004342  062737                       ADD     (PC)+,@(PC)+            ;INCREMENT THE FILE NUMBER
2118  004344     001     377               .BYTE   1,-1
2119  004346  001304                       .WORD   FILE
2120  004350  012737                       MOV     (PC)+,@(PC)+            ;INITIALIZE THE BLOCK NUMBER
2121  004352     000     377               .BYTE   0,377                   ;SET BYTE 0 TO 0 AND BYTE 1
2122  004354  001306                       .WORD   BLOCK                   ;TO THE 1'S COMP. OF BYTE 0
2123  004356  000720                       BR      2$                      ;START A FILE GAP
```

```
2124                            ;////////////////////////////////////////////////////////////////////////////////
2125                            ;////////////////////////////////////////////////////////////////////////////////
2126
2127                            .SBTTL          CHECK "EOTS"
2128
2129                            ;;**************************************************************
2130                            ;;**************************************************************
2131
2132                            ;WHEN A "FORMAT" OR  "WRITEONLY" PASS HITS "EOT" OR A
2133                            ;"READONLY" PASS READS THE LAST BLOCK OF THE LAST FILE
2134                            ;THEY WILL COME HERE
2135                            ;
2136                            ;       1. (EOTS) IS CHECKED
2137                            ;               A. (EOTS)=(MAXEOT) GOTO $EOP
2138                            ;               B. (EOTS)<(MAXEOT) GOTO 2
2139                            ;       2. SWR<8> = 1?
2140                            ;               A. YES -- GOTO $EOP
2141                            ;               B. NO  -- GOTO LOOPER
2142                            ;
2143                            ;;**************************************************************
2144  004360  000004            CKEOTS: SCOPE
2145  004362  023737  001212  001264        CMP     @#EOTS,@#MAXEOT         ;MAX. EOTS OCCURRED?
2146  004370  002006                        BGE     $EOP                    ;BR IF YES
2147  004372  032777  000400  174540        BIT     #SW08,@SWR              ;IF SWR<08> = 1
2148  004400  001002                        BNE     $EOP                    ;GOTO END-OF-TEST
2149  004402  000137  002770                JMP     @#LOOPER                ;NO--LOOP
2150
```

```
                                        .SBTTL  END OF PASS ROUTINE

                                        ;;**********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS"
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO START
                                        ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
                                        ;*$ENDMG CAN BE CHANGED TO 7.

004406                                  $EOP:
004406  000004                                  SCOPE
004410  005067  174466                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
004414  005267  174460                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
004420  042767  100000  174452                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
004426  005327                                  DEC     (PC)+           ;;LOOP?
004430  000001                          $EOPCT: .WORD   1
004432  003135                                  BGT     $DOAGN          ;;YES
004434  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
004436  000001                          $ENDCT: .WORD   1
004440  004430                                  $EOPCT
004442  104400  004735                          TYPE    $ENDMG          ;;TYPE "END PASS"
004446  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
004452  001525                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
004454  000004                                  SCOPE
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       TYPE END-OF-TEST STATISTICS
                                        ;////////////////////////////////////////////////////////////////////////
004456  012700  001612                          MOV     #PARMBK,R0      ;R0=PARAMETER BLOCK ADDRESS
004462  004737  006746                          JSR     PC,@#REWIND     ;START A REWIND
004466  104400  013005                          TYPE    ,MSG1           ;<CRLF>"*** END-OF-TEST ***"

004472  104400  013032                          TYPE    ,MSG2           ;<CRLF>"SOFT ERRORS="
004476  013746  001206                          MOV     @#SOFTNM,-(SP)  ;PICKUP SINGLE PRECISION BINARY NUMBER
004502  004737  007356                          JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
004506  004737  007412                          JSR     PC,@#$SUPRS     ;TYPE  WITHOUT LEADING ZEROS

004512  104400  013050                          TYPE    ,MSG3           ;<CRLF>"HARD ERRORS="
004516  013746  001210                          MOV     @#HARDNM,-(SP)  ;PICKUP SINGLE PRECISION BINARY NUMBER
004522  004737  007356                          JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
004526  004737  007412                          JSR     PC,@#$SUPRS     ;TYPE  WITHOUT LEADING ZEROS

004532  104400  013066                          TYPE    ,MSG4           ;<CRLF>"BYTES READ="
004536  012746  001214                          MOV     #RBYTTL,-(SP)   ;GET ADDRESS OF DOUBLE PRECISION BINARY #
004542  004737  007162                          JSR     PC,@#$DB2D      ;CHANGE IT TO DECIMAL ASCIZ
004546  004737  007412                          JSR     PC,@#$SUPRS     ;TYPE IT WITHOUT LEADING ZERO

004552  104400  013103                          TYPE    ,MSG5           ;<CRLF>"BYTES WRITTEN="
004556  012746  001220                          MOV     #WBYTTL,-(SP)   ;GET ADDRESS OF DOUBLE PRECISION BINARY #
004562  004737  007162                          JSR     PC,@#$DB2D      ;CHANGE IT TO DECIMAL ASCIZ
004566  004737  007412                          JSR     PC,@#$SUPRS     ;TYPE IT WITHOUT LEADING ZERO

004572  104400  013123                          TYPE    ,MSG6           ;<CRLF>"TAPE PASSES="
004576  013746  001212                          MOV     @#EOTS,-(SP)    ;PICKUP SINGLE PRECISION BINARY NUMBER
004602  004737  007356                          JSR     PC,@#$SB2D      ;CHANGE IT TO DECIMAL ASCIZ
004606  004737  007412                          JSR     PC,@#$SUPRS     ;TYPE  WITHOUT LEADING ZEROS
```

```
2207
2208  004612  104400  013141                TYPE    ,MSG7                   ;<CRLF>"FILES/PASS="
2209  004616  013700  001236                MOV     @#LASTFL,R0             ;PICKUP THE LAST FILE NUMBER
2210  004622  105200                        INCB    R0                      ;ADD 1 TO MAKE IT # OF FILES
2211  004624  010046                        MOV     R0,-(SP)                ;PUT IT ON THE STACK
2212  004626  004737  007356                JSR     PC,@#$SB2D              ;CHANGE IT TO DECIMAL ASCIZ
2213  004632  004737  007412                JSR     PC,@#$SUPRS             ;TYPE IT WITHOUT LEADING ZEROS
2214
2215  004636  104400  013156                TYPE    ,MSG8                   ;<CRLF>"BLOCKS/FILE="
2216  004642  013746  001310                MOV     @#FILESZ,-(SP)          ;PICKUP SINGLE PRECISION BINARY NUMBER
2217  004646  004737  007356                JSR     PC,@#$SB2D              ;CHANGE IT TO DECIMAL ASCIZ
2218  004652  004737  007412                JSR     PC,@#$SUPRS             ;TYPE  WITHOUT LEADING ZEROS
2219
2220  004656  104400  001204                TYPE    ,$LF
2221                                ;////////////////////////////////////////////////////////////////
2222                                ;       CHECK HALT AT END-OF-TEST SWITCH
2223                                ;////////////////////////////////////////////////////////////////
2224  004662  032777  001000  174250        BIT     #SW09,@SWR              ;HALT AT END-OF-TEST?
2225  004670  001406                        BEQ     100$                    ;BR IF NO
2226  004672  000000                        HALT                            ;YES
2227
2228  004674  022767  000176  174236        CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
2229  004702  001001                        BNE     20$                     ;NO- GET OUT
2230  004704  104404                        GTSWR                           ;GET VALUE
2231  004706                        20$:                                    ;CONTINUE
2232- 004706                        100$:
2233  004706  013700  000042                MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
2234  004712  001405                        BEQ     $DOAGN          ;;RETURN ADDRESS
2235  004714  000005                        RESET                   ;;CLEAR THE WORLD
2236  004716  004710                $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
2237  004720  000240                        NOP                     ;;SAVE ROOM
2238  004722  000240                        NOP                     ;;FOR
2239  004724  000240                        NOP                     ;;ACT11
2240  004726                        $DOAGN:
2241  004726  000137                        JMP     @(PC)+          ;;RETURN
2242  004730  002624                $RTNAD: .WORD   START
2243  004732     377     377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
2244  004735     015  042412  042116 $ENDMG: .ASCIZ  <15><12>/END PASS/
2245  004742  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

004750                                  $SCOPE:
004750  104405                                 CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
004752  032777  040000  174160    1$:   BIT     #BIT14,@SWR             ;;LOOP ON PRESENT TEST?
004760  001025                          BNE     $OVER                   ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
004762  000416                  $XTSTR: BR      6$                      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
004764  013746  000004                  MOV     @#ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
004770  012737  005010  000004          MOV     #5$,@#ERRVEC            ;;SET FOR TIMEOUT
004776  005737  177060                  TST     @#177060                ;;TIME OUT ON XOR?
005002  012637  000004                  MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
005006  000404                          BR      $SVLAD                  ;;GO TO THE NEXT TEST
005010  022626                  5$:     CMP     (SP)+,(SP)+             ;;CLEAR THE STACK AFTER A TIME OUT
005012  012637  000004                  MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
005016  000406                          BR      $OVER                   ;;LOOP ON THE PRESENT TEST
005020                                  6$:;#####END OF CODE FOR THE XOR TESTER#####
005020  105267  174056          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
005024  011667  174056                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
005030  105067  174047                  CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
005034  016777  174042  174100  $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
005042  016716  174040                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
005046  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

005050                                  $ERROR:
005050  104405                                 CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
005052  105267  174025                  7$:    INCB    $ERFLG                  ;;SET THE ERROR FLAG
005056  001775                          BEQ     7$                              ;;DON'T LET THE FLAG GO TO ZERO
005060  016777  174016  174054          MOV     $TSTNM,@DISPLAY                 ;;DISPLAY TEST NUMBER AND ERROR FLAG
005066  032777  002000  174044          BIT     #BIT10,@SWR                     ;;BELL ON ERROR?
005074  001402                          BEQ     1$                              ;;NO - SKIP
005076  104400  005174                  TYPE    ,$BELL                          ;;RING BELL
005102  005267  174004                  1$:    INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
005106  011667  174004                  MOV     (SP),$ERRPC                     ;;GET ADDRESS OF ERROR INSTRUCTION
005112  162767  000002  173776          SUB     #2,$ERRPC
005120  117767  173772  173766          MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
005126  032777  020000  174004          BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
005134  001004                          BNE     20$                     ;;SKIP TYPEOUTS
005136  004767  000036                  JSR     PC,TYPERR               ;;GO TO USER ERROR ROUTINE
005142  104400  001203                  TYPE    ,$CRLF
005146                                  20$:
005146  005777  173766                  2$:    TST     @SWR                    ;;HALT ON ERROR
005152  100002                          BPL     3$                              ;;SKIP IF CONTINUE
005154  000000                          HALT                                    ;;HALT ON ERROR!
005156  104405                          CKSWR                                   ;;TEST FOR CHANGE IN SOFT-SWR
005160                                  3$:
005160  022737  004716  000042          CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
005166  001001                          BNE     6$                              ;;BRANCH IF NO
005170  000000                          HALT                                    ;;YES
005172                                  6$:
005172  000002                          RTI                                     ;;RETURN
005174  177607  000377                  $BELL:  .ASCIZ  <207><377><377>         ;;ASCII CODE FOR BELL
```

```
2320                                    ;;*****************************************************************
2321                                    ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2322  005200  104400  001203    TYPERR: TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
2323  005204  010046                    MOV     R0,-(SP)                ;SAVE R0
2324  005206  010146                    MOV     R1,-(SP)                ;SAVE R1
2325  005210  005046                    CLR     -(SP)                   ;PICKUP THE ITEM BYTE
2326  005212  113716  001114            MOVB    @#$ITEMB,(SP)
2327  005216  005316                    DEC     (SP)                    ;ADJUST THE INDEX SO IT
2328  005220  006316                    ASL     (SP)                    ;WILL WORK FOR THE
2329  005222  006316                    ASL     (SP)                    ;ERROR TABLE
2330  005224  006316                    ASL     (SP)
2331  005226  116601  000001            MOVB    1(SP),R1                ;GET THE ERROR TYPE
2332  005232  112600                    MOVB    (SP)+,R0                ;GET THE ERROR NUMBER
2333  005234  066100  005242            ADD     1$(R1),R0               ;FORM THE TABLE POINTER
2334  005240  000403                    BR      2$
2335  005242  001622          1$:       ITEMS0
2336  005244  001642                    ITEMS1
2337  005246  001732                    ITEMS2
2338
2339  005250  012067  000002    2$:     MOV     (R0)+,3$                ;PICKUP "ERROR MESSAGE" POINTER
2340  005254  104400                    TYPE                            ;TYPE "ERROR MESSAGE"
2341  005256  000000          3$:       0
2342  005260  104400  001203            TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
2343  005264  012067  000002            MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
2344  005270  104400                    TYPE                            ;TYPE "DATA HEADER"
2345  005272  000000          4$:       0
2346  005274  104400  001203            TYPE    ,$CRLF                  ;TYPE "CR" & "LF"
2347  005300  060107                    ADD     R1,PC                   ;GO TYPE THE DATA
2348  005302  000402                    BR      ERROR0
2349  005304  000430                    BR      ERROR1
2350  005306  000473                    BR      ERROR2
2351
2352                                    ;//////////////////////////////////////////////////////////////////
2353  005310  010246          ERROR0: MOV     R2,-(SP)                ;SAVE R2
2354  005312  012001                    MOV     (R0)+,R1                ;PICKUP THE "DATA POINTER"
2355  005314  012002                    MOV     (R0)+,R2                ;PICKUP "FORMAT" POINTER
2356  005316  000402                    BR      2$
2357  005320  104400  013277  1$:       TYPE    ,MSG14                  ;"  "
2358  005324  012100          2$:       MOV     (R1)+,R0                ;GET ADDRESS OF DATA WORD
2359  005326  001473                    BEQ     EREXT1                  ;GO TO EXIT IF 0
2360  005330  105722                    TSTB    (R2)+                   ;TYPE DECIMAL OR OCTAL?
2361  005332  001003                    BNE     3$                      ;BR IF DECIMAL
2362  005334  011046                    MOV     (R0),-(SP)      ;;SAVE (R0) FOR TYPEOUT
2363  005336  104401                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2364  005340  000767                    BR      1$                      ;GO GET NEXT DATA WORD
2365  005342  010046          3$:       MOV R0,-(SP)
2366  005344  004737  007162            JSR     PC,@#$DB2D              ;CHANGE NUMBER TO ASCIZ
2367  005350  062716  000004            ADD     #4,(SP)                 ;TYPE 6 DIGITS
2368  005354  012667  000002            MOV     (SP)+,4$                ;
2369  005360  104400                    TYPE                            ;CALL TYPE ASCIZ MESSAGE ROUTINE
2370  005362  000000          4$:       .WORD   0                       ;ADDRESS OF ASCIZ STRING
2371  005364  000755                    BR      1$                      ;GO GET NEXT DATA WORD
```

```
                        ;//////////////////////////////////////////////////////////////////////////////
005366  011001         ERROR1: MOV     (R0),R1                 ;PICKUP THE "DATA POINTER"
005370  013146                 MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
                                                        ;;$ERRPC
005372  104401                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
005374  012767  000003  000006        MOV     #3,2$                   ;PRINT 2 SPACES 3 TIMES
                                                                        ;AND 2 DECIMAL #'S 2 TIMES
005402  104400  013277         1$:     TYPE    ,MSG14                  ;" "
005406  005327                 DEC     (PC)+                   ;TYPE ANOTHER # ?
005410  000000                 2$:     .WORD   0
005412  001412                 BEQ     4$                      ;BR IF NO
005414  012146                 MOV     (R1)+,-(SP)             ;YES---PICKUP ADDRESS OF #
005416  004737  007162         JSR     PC,@#$DB2D              ;GO CHANGE TO DECIMAL ASCIZ
005422  062716  000004         ADD     #4,(SP)                 ;ONLY TYPE THE LAST 6 DIGITS
005426  012667  000002         MOV     (SP)+,3$                ;SAVE ADDRESS OF ASCIZ STRING
005432  104400                 TYPE                            ;CALL THE TYPE ROUTINE
005434  000000                 3$:     .WORD   0               ;POINTER GOES HERE
005436  000761                 BR      1$                      ;LOOP
005440  013101                 4$:     MOV     @(R1)+,R1               ;PICKUP FUNCTION INDEX
005442  016167  005456  000002        MOV     BADFUN(R1),5$           ;GET THE FUNCTION MESSAGE
005450  104400                 TYPE                            ;AND TYPE IT
005452  000000                 5$:     .WORD   0               ;MESSAGE POINTER GOES HERE
005454  000421                 BR      EREXT2                  ;GO TO EXIT
                        ;THIS TABLE CONTAINS THE POINTERS TO THE DIFFERENT ASCIZ MESSAGES
                        ;FOR THE FUNCTION BEING PERFORMED WHEN THE ERROR OCCURRED.
005456  012043         BADFUN: MXWFG           ;       WRITE-FILE-GAP
005460  012062                 MXWRIT          ;       WRITE
005462  012070                 MXREAD          ;       READ
005464  012075                 MXBSFG          ;       BACK-SPACE-FILE-GAP
005466  012121                 MXBSBG          ;       BACK-SPACE-BLK-GAP
005470  012144                 MXSFFG          ;       SPACE-FWD-FILE-GAP
005472  012167                 MXSFBG          ;       SPACE-FWD-BLK-GAP
005474  012211                 MXRWND          ;       REWIND

                        ;//////////////////////////////////////////////////////////////////////////////
005476  011000         ERROR2: MOV     (R0),R0                 ;PICKUP THE DATA POINTER
005500  005710         1$:     TST     (R0)                    ;ALL DATA TYPED?
005502  001406                 BEQ     EREXT2                  ;BR IF YES
005504  013046                 MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
005506  104401                 TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
005510  104400  013277         TYPE    ,MSG14                  ;" "
005514  000771                 BR      1$                      ;LOOP

                        ;//////////////////////////////////////////////////////////////////////////////
005516  012602         EREXT1: MOV     (SP)+,R2                ;RESTORE R2
005520  012601         EREXT2: MOV     (SP)+,R1                ;RESTORE R1
005522  012600                 MOV     (SP)+,R0                ;RESTORE R0
005524  000207                 RTS     PC                      ;RETURN
```

```
                                        ;//////////////////////////////////////////////////////////////////////////////
                                        ;//////////////////////////////////////////////////////////////////////////////
                                        .SBTTL          DETERMINE CONTROL AND STATUS ERROR
                                        
005526  113737  001304  001162  CSRERR: MOVB    @#FILE,@#$REG0          ;GET THE FILE NUMBER
005534  113737  001306  001166          MOVB    @#BLOCK,@#$REG2         ;GET THE BLOCK NUMBER
005542  013746  007160                  MOV     @#FUNLOC,-(SP)          ;GET THE LAST FUNCTION
005546  042716  177761                  BIC     #^CFUNCTION,(SP)        ;CLEAR AWAY THE JUNK
005552  011637  001172                  MOV     (SP),@#$TMP0            ;SAVE THE FUNCTION CODE
005556  113766  001612  000001          MOVB    @#PARMBK,1(SP)          ;COMBINE THE ST/ER WITH
005564  012601                          MOV     (SP)+,R1                ;THE FUNCTION
005566  005046                          CLR     -(SP)                   ;FIND OUT WHAT CAUSED THE ERROR
005570  032701  001000                  BIT     #OFFLINE,R1             ;OFF LINE?
005574  001025                          BNE     6$                      ;BR IF YES
005576  032701  010000                  BIT     #WRTLOCK,R1             ;WRITE LOCK?
005602  001021                          BNE     5$                      ;BR IF YES
005604  032701  020000                  BIT     #LEADER,R1              ;BOT/EOT?
005610  001015                          BNE     4$                      ;BR IF YES
005612  032701  004000                  BIT     #FGAP,R1                ;FILE GAP?
005616  001011                          BNE     3$                      ;BR IF YES
005620  032701  002000                  BIT     #TIMERR,R1              ;TIMING?
005624  001005                          BNE     2$                      ;BR IF YES
005626  032701  040000                  BIT     #CRCERR,R1              ;BLOCK CHECK?
005632  001001                          BNE     1$                      ;BR IF YES
005634  005216                          INC     (SP)                    ; 6 = UNKNOWN
005636  005216                  1$:     INC     (SP)                    ; 5 = CRCERR
005640  005216                  2$:     INC     (SP)                    ; 4 = TIMERR
005642  005216                  3$:     INC     (SP)                    ; 3 = FGAP
005644  005216                  4$:     INC     (SP)                    ; 2 = LEADER
005646  005216                  5$:     INC     (SP)                    ; 1 = WRTLOCK
005650                          6$:
005650  106316                          ASLB    (SP)                    ;POSITION INDEX
005652  062607                          ADD     (SP)+,PC                ;BR TO THE ROUTINE
005654  000406                          BR      OL.ERR
005656  000407                          BR      WL.ERR
005660  000410                          BR      CL.ERR
005662  000462                          BR      FG.ERR
005664  000463                          BR      TM.ERR
005666  000464                          BR      BC.ERR
005670  000472                          BR      X.ERR

005672  104101          OL.ERR: ERROR   101                             ;DRIVE IS OFFLINE
005674  000200                  RTS     R0

005676  104102          WL.ERR: ERROR   102                             ;DRIVE IS WRITE LOCK
005700  000200                  RTS     R0
```

```
2469  005702  022737  000000  001172  CL.ERR: CMP     #XWFG,@#$TMPO           ;FUNCTION "WRITE FILE GAP"?
2470  005710  001014                          BNE     2$                      ;BR IF NO
2471  005712                          1$:
2472  005712  162737                          SUB     (PC)+,@(PC)+            ;DECREMENT THE FILE NUMBER
2473  005714     001      377                 .BYTE   1,-1
2474  005716  001304                          .WORD   FILE
2475  005720  113737  001304  001236          MOVB    @#FILE,@#LASTFL         ;SAVE LAST FILE NUMBER
2476  005726  113737  001310  001234          MOVB    @#FILESZ,@#LASTBK       ;AND LAST BLOCK NUMBER
2477  005734  005337  001234                  DEC     @#LASTBK
2478  005740  000425                          BR      3$                      ;BR IF NO
2479  005742  022737  000002  001172  2$:     CMP     #XWRITE,@#$TMPO         ;FUNCTION "WRITE"?
2480  005750  001025                          BNE     4$                      ;BR IF NO
2481  005752  113737  001304  001236          MOVB    @#FILE,@#LASTFL         ;SAVE LAST FILE NUMBER
2482  005760  162737                          SUB     (PC)+,@(PC)+            ;DECREMENT THE BLOCK NUMBER
2483  005762     001      377                 .BYTE   1,-1
2484  005764  001306                          .WORD   BLOCK
2485  005766  113737  001306  001234          MOVB    @#BLOCK,@#LASTBK        ;SAVE LAST BLOCK NUMBER
2486  005774  163737  007154  001220          SUB     @#CBCNT,@#WBYTTL        ;SUBTRACT RESIDUE COUNT FROM
2487  006002  005637  001222                  SBC     @#WBYTTL+2              ;       THE TOTAL WRITE COUNT
2488  006006  005137  001306                  COM     @#BLOCK                 ;IF NO BLOCKS IN THIS FILE
2489  006012  001737                          BEQ     1$                      ;BACK TO LAST FILE
2490  006014  022020                  3$:     CMP     (R0)+,(R0)+             ;ADJUST FOR RETURN
2491  006016  005237  001212                  INC     @#EOTS                  ;COUNT END-OF-TAPE
2492  006022  000401                          BR      5$
2493  006024  104103                  4$:     ERROR   103                     ;"CLEAR LEADER" ERROR
2494  006026  000200                  5$:     RTS     R0                      ;RETURN
2495
2496  006030  104104                  FG.ERR: ERROR   104                     ;"FILE GAP ERROR"
2497  006032  000200                          RTS     R0
2498
2499  006034  104105                  TM.ERR: ERROR   105                     ;TIMING ERROR
2500  006036  000200                          RTS     R0
2501
2502  006040  104106                  BC.ERR: ERROR   106                     ;BLOCK CHECK ERROR
2503  006042  022737  000004  001172          CMP     #XREAD,@#$TMPO          ;IF FUNCTION ISN'T A READ
2504  006050  001001                          BNE     1$                      ;TAKE THE "EOP" EXIT
2505  006052  022020                          CMP     (R0)+,(R0)+             ;ADJUST FOR RETURN
2506  006054  000200                  1$:     RTS     R0
2507
2508  006056  104107                  X.ERR:  ERROR   107                     ;UNKNOWN INTERRUPT
2509  006060  000200                          RTS     R0
```

```
                                        ;;*****************************************************************
                                        ;;*****************************************************************
                                        .SBTTL          ROUTINE TO SETUP FOR A WRITE OPERATION

                                        ;THIS ROUTINE WILL SETUP THE BYTE COUNT, DETERMINE THE PATTERN TO
                                        ;BE WRITTEN ON TAPE, ADD THE SIZE OF THIS BLOCK TO THE TOTAL
                                        ;NUMBER OF BYTES TRANSFERRED AND CALL THE ROUTINE TO FILL THE
                                        ;WRITE BUFFER.

                                        ;CALL
                                        ;       JSR     PC,@#SETUPW

006062  013700  001306          SETUPW: MOV     @#BLOCK,R0              ;GET THE BLOCK NUMBER
006066  042700  177760                  BIC     #^C15.,R0               ;KEEP MAX PATTERN
006072  006300                          ASL     R0                      ;*2
006074  066037  001312  001220          ADD     BLKSZ(R0),@#WBYTTL      ;ADD THE BLOCK SIZE TO THE TOTAL
006102  005537  001222                  ADC     @#WBYTTL+2              ;NUMBER OF BYTES  WRITTEN
006106  016037  001312  001620          MOV     BLKSZ(R0),@#PARMBK+6    ;SET THE BYTE COUNT
006114  016000  001352                  MOV     PATS(R0),R0             ;GET PATTERN POINTER
006120  004737  006126                  JSR     PC,@#FILL               ;GO FILL WRITE BUFFER
006124  000207                          RTS     PC                      ;RETURN

                                        ;;*****************************************************************
                                        .SBTTL          ROUTINE TO FILL THE BUFFER BEFORE A WRITE
                                        ;CALL
                                        ;       MOV     #PATADR,R0              ;1ST ADDRESS OF 8 BYTE PATADR
                                        ;       JSR     PC,@#FILL
                                        ;THE BUFFER IS FILLED AS FOLLOWS:
                                        ;
                                        ;BYTE#  DATA
                                        ;
                                        ;0001   FILE NUMBER
                                        ;0002   FILE NUMBER COMPLEMENTED
                                        ;0003   BLOCK NUMBER
                                        ;0004   BLOCK NUMBER COMPLEMENTED
                                        ;0005   DATA PATTERN
                                        ;0006   DATA PATTERN
                                        ;  *      *       *
                                        ;  *      *       *
                                        ;  *      *       *
                                        ;1027.  DATA PATTERN
                                        ;1028.  DATA PATTERN

006126                          FILL:
006126  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006130  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006132  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006134  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006136  012701  000200                  MOV     #1024./8.,R1            ;FILL 1024. BYTES WITH PATTERN
006142  012702  013440                  MOV     #BUFFER,R2              ;1ST ADDRESS OF DATA BUFFER
006146  013722  001304                  MOV     @#FILE,(R2)+            ;FILE NUMBER AND ITS COMPLEMENT
006152  013722  001306                  MOV     @#BLOCK,(R2)+           ;BLOCK NUMBER AND ITS COMPLEMENT
006156  010003                          MOV     R0,R3                   ;SAVE FIRST ADDRESS OF PATTERN
006160  010300                  1$:     MOV     R3,R0                   ;RESET PATTERN POINTER
006162  012022                          MOV     (R0)+,(R2)+             ;MOVE THE PATTERN
```

```
2566  006164  012022                        MOV     (R0)+,(R2)+             ;INTO THE BUFFER AREA.
2567  006166  012022                        MOV     (R0)+,(R2)+
2568  006170  012022                        MOV     (R0)+,(R2)+
2569  006172  005301                        DEC     R1
2570  006174  001371                        BNE     1$
2571  006176  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
2572  006200  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
2573  006202  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
2574  006204  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
2575  006206  000207                        RTS     PC
2576
2577                                  ;;*****************************************************************
2578                                  ;;*****************************************************************
2579                                  .SBTTL          ROUTINE TO SETUP FOR A READ
2580
2581                                  ;THIS ROUTINE WILL SETUP THE BYTE COUNT AND ADD THE SIZE OF THIS
2582                                  ;BLOCK TO THE TOTAL NUMBER OF BYTES READ.
2583
2584                                  ;CALL
2585                                  ;       JSR     PC,SETUPR
2586
2587
2588  006210  013700  001306          SETUPR: MOV     @#BLOCK,R0              ;GET THE BLOCK NUMBER
2589  006214  042700  177760                  BIC     #^C15.,R0               ;KEEP MAX PATTERN
2590  006220  006300                          ASL     R0                      ;X2
2591  006222  066037  001312  001214          ADD     BLKSZ(R0),@#RBYTTL      ;ADD THE BLOCK SIZE TO THE TOTAL
2592  006230  005537  001216                  ADC     @#RBYTTL+2              ;NUMBER OF BYTES READ
2593  006234  016037  001312  001620          MOV     BLKSZ(R0),@#PARMBK+6    ;SET THE BYTE COUNT
2594  006242  000207                          RTS     PC                      ;RETURN
2595
2596                                  ;;*****************************************************************
2597                                  .SBTTL          ROUTINE TO CHECK FOR SYNC PROBLEMS
2598
2599                                  ;THIS ROUTINE CHECKS THE FIRST FOUR BYTES OF THE DATA
2600                                  ;BUFFER TO INSURE THE PROPER BLOCK WAS READ
2601                                  ;BYTE 1 = FILE NUMBER
2602                                  ;BYTE 2 = FILE NUMBER COMPLEMENT
2603                                  ;BYTE 3 = BLOCK NUMBER
2604                                  ;BYTE 4 = BLOCK NUMBER COMPLEMENT
2605
2606                                  ;CALL
2607                                  ;       JSR     PC,@#SYNCK
2608                                  ;       NORMAL RETURN
2609                                  ;       ERROR RETURN
2610
2611                                  ;
2612                                  ;
2613  006244  012700  013440          SYNCK:  MOV     #BUFFER,R0              ;ADDRESS OF FIRST WORD
2614  006250  011001                          MOV     (R0),R1                 ;CHECK FILE # WAS READ OK
2615  006252  000301                          SWAB    R1                      ;BYTE 1 SHOULD BE THE
2616  006254  062001                          ADD     (R0)+,R1                ;COMPLEMENT OF BYTE 0
2617  006256  005101                          COM     R1
2618  006260  001031                          BNE     2$                      ;BR IF FILE NUMBER READ WRONG
2619  006262  011001                          MOV     (R0),R1                 ;CHECK BLOCK # WAS READ OK
2620  006264  000301                          SWAB    R1                      ;BYTE 3 SHOULD BE THE
2621  006266  062001                          ADD     (R0)+,R1                ;COMPLEMENT OF BYTE 2
```

```
2622  006270  005101                          COM     R1
2623  006272  001024                          BNE     2$                      ;BR IF BLOCK NUMBER READ WRONG
2624  006274  023740  001306                  CMP     @#BLOCK,-(R0)           ;MAKE SURE PROPER BLOCK WAS READ
2625  006300  001003                          BNE     1$                      ;BR IF AT THE WRONG BLOCK
2626  006302  023740  001304                  CMP     @#FILE,-(R0)            ;MAKE SURE PROPER FILE
2627  006306  001416                          BEQ     2$                      ;BR IF IN THE PROPER FILE
2628  006310  113737  001304  001162  1$:     MOVB    @#FILE,@#$REG0
2629  006316  113737  001306  001166          MOVB    @#BLOCK,@#$REG2
2630  006324  113737  013440  001172          MOVB    @#BUFFER,@#$TMP0
2631  006332  113737  013442  001176          MOVB    @#BUFFER+2,@#$TMP2
2632  006340  062716  000002                  ADD     #2,(SP)                 ;TAKE ERROR RETURN
2633  006344  000207                2$:      RTS     PC                      ;RETURN
2634
2635                                ;;*****************************************************************
2636                                .SBTTL          ROUTINE TO CHECK THE READ DATA
2637
2638                                ;CALL
2639                                ;       JSR     PC,DATCMP
2640                                ;       NORMAL RETURN           ;ERROR FLAG=0 AND DATA IS GOOD
2641                                ;       ERROR1 RETURN           ;DATA IS BAD (ERROR FLAG=?)
2642                                ;       ERROR2 RETURN           ;DATA IS GOOD BUT ERROR FLAG=1
2643
2644  006346  013700  001306        DATCMP: MOV     @#BLOCK,R0              ;GET THE BLOCK NUMBER
2645  006352  042700  177760                BIC     #^C15.,R0               ;KEEP MAX PATTERN
2646  006356  006300                        ASL     R0                      ;X2
2647  006360  016001  001312                MOV     BLKSZ(R0),R1            ;PICKUP THE BLOCK SIZE
2648  006364  012702  013440                MOV     #BUFFER,R2              ;FIRST ADDRESS OF DATA
2649  006370  012703  000004                MOV     #4,R3                   ;SETUP TO CHECK FILE AND BLOCK
2650  006374  012704  001304                MOV     #FILE,R4
2651  006400  005301                1$:     DEC     R1                      ;COUNT THIS BYTE
2652  006402  122224                        CMPB    (R2)+,(R4)+             ;CHECK IT
2653  006404  001015                        BNE     4$                      ;BR IF BAD
2654  006406  005303                        DEC     R3                      ;AGAIN?
2655  006410  001373                        BNE     1$                      ;BR IF YES
2656  006412  012703  000010        2$:     MOV     #8.,R3                  ;GET NUMBER OF BYTES/PATTERN
2657  006416  016004  001352                MOV     PAT$(R0),R4             ;GET FIRST ADDRESS OF PATTERN
2658  006422  005301                3$:     DEC     R1                      ;LAST BYTE BEEN CHECKED?
2659  006424  002437                        BLT     5$                      ;BR IF YES
2660  006426  122224                        CMPB    (R2)+,(R4)+             ;NO--CHECK IT
2661  006430  001003                        BNE     4$                      ;BR IF BAD
2662  006432  005303                        DEC     R3                      ;END OF PATTERN?
2663  006434  003372                        BGT     3$                      ;BR IF NO
2664  006436  000765                        BR      2$                      ;YES
2665  006440  005037  001124        4$:     CLR     @#$GDDAT
2666  006444  005037  001126                CLR     @#$BDDAT
2667  006450  114437  001124                MOVB    -(R4),@#$GDDAT          ;SAVE GOOD DATA
2668  006454  010437  001120                MOV     R4,@#$GDADR             ;SAVE GOOD ADDRESS
2669  006460  114237  001126                MOVB    -(R2),@#$BDDAT          ;SAVE BAD DATA
2670  006464  010237  001122                MOV     R2,@#$BDADR             ;SAVE BAD ADDRESS
2671  006470  005401                        NEG     R1                      ;SAVE THE BYTE NUMBER
2672  006472  066001  001312                ADD     BLKSZ(R0),R1
2673  006476  010137  001230                MOV     R1,@#BYTNUM
2674  006502  113737  001304  001162        MOVB    @#FILE,@#$REG0
2675  006510  113737  001306  001166        MOVB    @#BLOCK,@#$REG2
2676  006516  062716  000002                ADD     #2,(SP)                 ;SETUP TO TAKE ERROR1 EXIT
2677  006522  000405                        BR      6$                      ;TAKE ERROR1 EXIT
```

```
2678  006524  105737  001612          5$:     TSTB    @#PARMBK                ;DATA IS GOOD BUT WHAT ABOUT
2679                                                                          ;THE ERROR FLAG
2680  006530  100002                          BPL     6$                      ;BR IF ERROR FLAG=0
2681  006532  062716  000004                  ADD     #4,(SP)                 ;SETUP TO TAKE ERROR2 EXIT
2682  006536  000207                  6$:     RTS     PC                      ;RETURN
2683
2684                                  ;;*****************************************************************
2685
2686                                  .SBTTL          COUNT SOFT DATA ERROR
2687
2688                                  ;CALL
2689                                  ;       JSR     R0,@#CNTSFT
2690                                  ;       RETURN  FOR SOFT RETRY
2691                                  ;       RETURN  FOR HARD
2692                                  ;
2693  006540  005237  001206          CNTSFT: INC     @#SOFTNM                ;COUNT THIS SOFT ERROR
2694  006544  005337  001224                  DEC     @#RDTRYS                ;REREAD THE RECORD?
2695  006550  002011                          BGE     1$                      ;BR IF YES
2696  006552  013737  001260  001224          MOV     @#MAXRDS,@#RDTRYS       ;RESET REREAD COUNT
2697  006560  163737  001260  001206          SUB     @#MAXRDS,@#SOFTNM       ;DON'T COUNT AS SOFT
2698  006566  005337  001206                  DEC     @#SOFTNM
2699  006572  005720                          TST     (R0)+                   ;STEP OVER THE REREAD RETURN
2700  006574  000200                  1$:     RTS     R0                      ;RETURN
2701                                  ;;*****************************************************************
2702
2703                                  .SBTTL          COUNT HARD ERROR
2704
2705                                  ;CAL
2706                                  ;       JSR     R0,@#CNTHRD
2707                                  ;       RETURN  MORE ERRORS ALLOWED
2708                                  ;       RETURN  MAX. ERRORS HAVE OCCURRED
2709                                  ;
2710  006576  005237  001210          CNTHRD: INC     @#HARDNM                ;COUNT THIS HARD ERROR
2711  006602  023737  001210  001262          CMP     @#HARDNM,@#MAXERR       ;MAX. HARD ERRORS OCCURRED
2712  006610  002401                          BLT     1$                      ;BR IF NO
2713  006612  005720                          TST     (R0)+                   ;TAKE FATAL EXIT
2714  006614  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

006616                          $SAVREG:
006616  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006620  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006622  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006624  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006626  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
006630  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006632  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
006636  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
006642  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
006646  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
006652  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
006654                          $RESREG:
006654  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
006660  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
006664  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
006670  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
006674  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
006676  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
006700  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
006702  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
006704  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
006706  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
006710  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
                                        ;
006712  010746          WFG:    MOV     PC,-(SP)        ; WRITE A FILE GAP
006714  000415          FUNTAB: BR      IOCOM
006716  010746          WRITE:  MOV     PC,-(SP)        ; WRITE A BLOCK OF DATA
006720  000413                  BR      IOCOM
006722  010746          READ:   MOV     PC,-(SP)        ; READ A BLOCK OF DATA
006724  000411                  BR      IOCOM
006726  010746          BSFG:   MOV     PC,-(SP)        ; BACK SPACE A FILE GAP
006730  000407                  BR      IOCOM
006732  010746          BSBG:   MOV     PC,-(SP)        ; BACK SPACE A BLOCK GAP
006734  000405                  BR      IOCOM
006736  010746          SFFG:   MOV     PC,-(SP)        ; SPACE FORWARD A FILE GAP
006740  000403                  BR      IOCOM
006742  010746          SFBG:   MOV     PC,-(SP)        ; SPACE FORWARD A BLOCK GAP
006744  000401                  BR      IOCOM
006746  010746          REWIND: MOV     PC,-(SP)        ; REWIND TO BEGINNING OF TAPE
006750  162716  006714  IOCOM:  SUB     #FUNTAB,(SP)    ;PC ON STACK IS USED TO DETERMINE
006754  006216                  ASR     (SP)            ;  FUNCTION
006756  032777  000040  172254  CHKRDY: BIT     #40,@TACSL      ;READY BIT UP?
006764  001774                  BEQ     CHKRDY          ;WAIT ON IT
006766  012667  000166          MOV     (SP)+,FUNLOC    ;PUT FUNC. IN BITS 1-3 OF FUNLOC
006772  010146                  MOV     R1,-(SP)
006774  010046                  MOV     R0,-(SP)
006776  012701  007152          MOV     #CASPAR,R1      ;PT. TO INT. HANDLER PARAM. BLK.
007002  016011  000002          MOV     2(R0),(R1)      ;PTR. TO ST/ERR WD. INTO P.BLK.
007006  105031                  CLRB    @(R1)+          ;ZERO ST/ERR BYTE
007010  005200                  INC     R0
007012  152067  000143  CHANOK: BISB    (R0)+,FUNLOC+1  ;COPY CHAN. NUM. TO FUNC. WORD
007016  005720                  TST     (R0)+
007020  012061  000002          MOV     (R0)+,2(R1)     ;BUFF. ADR.
007024  011021                  MOV     (R0),(R1)+      ;BYTE COUNT
007026  005721                  TST     (R1)+
007030  152711  000101          BISB    #101,(R1)       ;ADD INT. ENABLE AND GO BITS
                                                        ;TO THE FUNCTION
007034  011177  172200  DOFUN:  MOV     (R1),@TACSL     ;START THE FUNCTION
007040  012600                  MOV     (SP)+,R0
007042  012601                  MOV     (SP)+,R1
007044  000207                  RTS     PC
```

```
                        ;;*****************************************************************
                        ;
                        .SBTTL          CASSETTE INTERRUPT HANDLER
                        ;
                        ;
007046                  CASINT:
007046  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
007050  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007052  016702  172162          MOV     TACSL,R2
007056  012701  007154          MOV     #CBCNT,R1               ;PARAM. BLOCK
007062  105712                  TSTB    (R2)                    ;TRANSFER REQUEST?
007064  100411                  BMI     TREQ                    ;YES
007066  011204                  MOV     (R2),R4
007070  000304                  SWAB    R4
007072  052704  000001          BIS     #1,R4                   ;SET THE DONE BIT
007076  110451                  MOVB    R4,@-(R1)               ;LOAD ST/ERR BYTE
007100  105012                  CLRB    (R2)                    ;TURN OFF INTERS.
007102  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007104  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007106  000002                  RTI
                        ;
007110  005321          TREQ:   DEC     (R1)+
007112  100412                  BMI     RWILBS                  ;WHEN COUNT NEG., DONE
007114  032712  000004          BIT     #4,(R2)                 ;CHECK READ OR WRITE FUNC.
007120  001003                  BNE     READR
007122  113177  172114          MOVB    @(R1)+,@TADBL           ;WRITE BYTE
007126  000402                  BR      RWDONE
007130  117731  172106  READR:  MOVB    @TADBL,@(R1)+           ;READ BYTE
007134  005241          RWDONE: INC     -(R1)                   ;BUMP   ADDR.
007136  000402                  BR BACK                         ;GET BACK
                        ;
007140  052712  000020  RWILBS: BIS     #20,(R2)                ;INIT. LAST BYTE SEQ.
007144                  BACK:
007144  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007146  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007150  000002                  RTI
                        ;
                        ;
                        ; CASSETTE INTER. HANDLER PARAM. BLOCK
                        ;
007152  000000          CASPAR: .WORD   0                       ;STATUS/ERROR POINTER
007154  000000          CBCNT:  .WORD   0                       ;WORKING BYTE COUNT
007156  000000                  .WORD   0                       ;WORKING BUFFER ADDRESS
007160  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


007162  104411                  $DB2D:  SAVREG                          ;;SAVE REGISTERS
007164  016602  000002                  MOV     2(SP),R2                ;;PICKUP THE DATA POINTER
007170  012700  007342                  MOV     #$DECVL,R0              ;;GET ADDRESS OF "$DECVL" STRING
007174  010066  000002                  MOV     R0,2(SP)                ;;PUT ADDRESS OF ASCIZ STRING ON STACK
007200  012201                          MOV     (R2)+,R1                ;;PICKUP THE BINARY NUMBER
007202  012202                          MOV     (R2)+,R2
007204  012767  000012  000046          MOV     #10.,4$                 ;;SET UP TO DO 10 CONVERSIONS
007212  012704  007272                  MOV     #$TNPWR,R4              ;;ADDRESS OF TEN POWER
007216  012705  007274                  MOV     #$TNPWR+2,R5
007222  005003                  1$:     CLR     R3                      ;;CLEAR PARTIAL
007224  161401                  2$:     SUB     (R4),R1                 ;;SUBTRACT TEN POWER
007226  005602                          SBC     R2
007230  161502                          SUB     (R5),R2
007232  002402                          BLT     3$                      ;;BR IF TEN POWER TO LARGE
007234  005203                          INC     R3                      ;;ADD 1 TO PARTIAL
007236  000772                          BR      2$                      ;;LOOP
007240  062401                  3$:     ADD     (R4)+,R1                ;;RESTORE SUBTRACTED VALUE
007242  005502                          ADC     R2
007244  062402                          ADD     (R4)+,R2
007246  022525                          CMP     (R5)+,(R5)+             ;;MOVE TO NEXT TEN POWER
007250  052703  000060                  BIS     #'0,R3                  ;;CHANGE PARTIAL TO ASCII
007254  110320                          MOVB    R3,(R0)+                ;;SAVE IT
007256  005327                          DEC     (PC)+                   ;;DONE?
007260  000000                  4$:     .WORD   0
007262  001357                          BNE     1$                      ;;BR IF NO
007264  105020                          CLRB    (R0)+                   ;;TERMINATOR
007266  104412                          RESREG                          ;;RESTORE REGISTERS
007270  000207                          RTS     PC                      ;;RETURN
007272  145000                  $TNPWR: 145000                          ;;1.0E09
007274  035632                          35632
007276  160400                          160400                          ;;1.0E08
007300  002765                          2765
007302  113200                          113200                          ;;1.0E07
007304  000230                          230
007306  041100                          041100                          ;;1.0E06
007310  000017                          17
007312  103240                          103240                          ;;1.0E05
007314  000001                          1
007316  023420                          23420                           ;;1.0E04
007320  000000                          0
007322  001750                          1750                            ;;1.0E03
007324  000000                          0
```

```
2916  007326  000144                          144                     ;;1.0E02
2917  007330  000000                          0
2918  007332  000012                          12                      ;;1.0E01
2919  007334  000000                          0
2920  007336  000001                          1                       ;;1.0E00
2921  007340  000000                          0
2922  007342  000014                $DECVL:   .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
2923                                .SBTTL    SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
2924
2925                                ;;*****************************************************************
2926                                ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2927                                ;*UNSIGNED DECIMAL ASCIZ NUMBER.
2928                                ;*CALL
2929                                ;*      MOV     NUMBER,-(SP)    ;;PUT BINARY NUMBER ON THE STACK
2930                                ;*      JSR     PC,@#$SB2D      ;;CALL
2931                                ;*      RETURN                  ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
2932
2933
2934  007356  016667  000002  000022 $SB2D:   MOV     2(SP),1$        ;;SAVE BINARY NUMBER
2935  007364  012746  007406                  MOV     #1$,-(SP)       ;;SET POINTER
2936  007370  004737  007162                  JSR     PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
2937  007374  062716  000005                  ADD     #5,(SP)         ;;ONLY ALLOW FIVE CHARACTERS
2938  007400  012666  000002                  MOV     (SP)+,2(SP)     ;;PICKUP POINTER
2939  007404  000207                          RTS     PC              ;;RETURN
2940  007406  000000  000000          1$:     .WORD   0,0
2941                                .SBTTL    TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
2942
2943                                ;;*****************************************************************
2944                                ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
2945                                ;*LEADING NUMBERS.
2946                                ;*CALL
2947                                ;*      MOV     #NUMADR,-(SP)   ;;FIRST ADDRESS OF ASCIZ STRING
2948                                ;*      JSR     PC,@#$SUPRS
2949
2950
2951  007412  010046                $SUPRS: MOV     R0,-(SP)        ;;SAVE R0
2952  007414  016600  000004                  MOV     4(SP),R0        ;;PICKUP THE POINTER
2953  007420  105710                1$:     TSTB    (R0)            ;;TERMINATECR?
2954  007422  001403                          BEQ     2$              ;;BR IF YES
2955  007424  122720  000060                  CMPB    #'0,(R0)+       ;;IS THIS AN ASCII "0" ?
2956  007430  001773                          BEQ     1$              ;;BR IF YES
2957  007432  005300                2$:     DEC     R0              ;;BACKUP BY "1"
2958  007434  010067  000002                  MOV     R0,3$           ;;SAVE FOR TYPING
2959  007440  104400                          TYPE                    ;;GO TYPE
2960  007442  000000                3$:     .WORD   0               ;;ASCIZ POINTER GOES HERE
2961  007444  012600                          MOV     (SP)+,R0        ;;RESTORE R0
2962  007446  012616                          MOV     (SP)+,(SP)      ;;RESTORE THE STACK
2963  007450  000207                          RTS     PC              ;;RETURN
2964                                .SBTTL    READ AN OCTAL NUMBER FROM THE TTY
2965
2966                                ;;*****************************************************************
2967                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2968                                ;*CHANGE IT TO BINARY.
2969                                ;*CALL:
2970                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
2971                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
```

```
2972                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
2973
2974  007452  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2975  007454  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
2976  007462  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2977  007464  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2978  007466  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2979  007470  104407                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
2980  007472  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2981  007474  005001                            CLR     R1              ;;CLEAR DATA WORD
2982  007476  005002                            CLR     R2
2983  007500  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2984  007502  001412                            BEQ     3$              ;;IF ZERO GET OUT
2985  007504  006301                            ASL     R1              ;;*2
2986  007506  006102                            ROL     R2
2987  007510  006301                            ASL     R1              ;;*4
2988  007512  006102                            ROL     R2
2989  007514  006301                            ASL     R1              ;;*8
2990  007516  006102                            ROL     R2
2991  007520  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2992  007524  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2993  007526  000764                            BR      2$              ;;LOOP
2994  007530  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2995  007532  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
2996  007536  010267  000010                    MOV     R2,$HIOCT
2997  007542  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
2998  007544  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
2999  007546  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
3000  007550  000002                            RTI                     ;;RETURN
3001  007552  000000                    $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
3002                                    ;;*****************************************************************
3003                                    .SBTTL          ROUTINE TO TYPE DRIVE
3004
3005                                    ;CALL
3006                                    ;       JSR     PC,@#TPDRV
3007                                    ;
3008
3009
3010  007554  104400  013302    TPDRV:  TYPE    ,MSG15                  ;" DRIVE "
3011  007560  113746  001613            MOVB    @#PARMBK+1,-(SP)        ;PICKUP THE DRIVE NUMBER
3012  007564  042716  177776            BIC     #^C1,(SP)               ;STRIP ANY JUNK
3013  007570  062716  000101            ADD     #'A,(SP)                ;MAKE IT ASCIZ
3014  007574  012667  000006            MOV     (SP)+,1$                ;SAVE IT FOR TYPEOUT
3015  007600  104400                    TYPE                            ;TYPE IT
3016  007602  007606                    1$
3017  007604  000207                    RTS     PC
3018  007606  000000    1$:     .WORD   0
```

```
3019                                    ;;*****************************************************************
3020
3021                                    .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
3022
3023                                    ;CALL
3024                                    ;       JSR     PC,@#ASKDRV
3025                                    ;       RETURN                  ;NOTE: R0 AND R1 ARE DESTROYED
3026
3027  007610  104400  013312    ASKDRV: TYPE    ,MSGDRV                 ;<CRLF>"DRIVE(S)? "
3028  007614  005067  171432            CLR     DRVKEY
3029  007620  104407                    RDLIN                           ;GO GET A DRIVE
3030  007622  012600                    MOV     (SP)+,R0                ;SETUP TO CHECK FOR VALID DRIVE(S)
3031  007624  105710                    TSTB    @R0                     ;WAS A DRIVE SELECTED?
3032  007626  001425                    BEQ     NOTLGL                  ;BR IF NO
3033  007630  012701  001252            MOV     #DRVKEY,R1
3034  007634  122710  000101    LOOP:   CMPB    #'A,@R0                 ;WAS DRIVE "A" SELECTED?
3035  007640  001002                    BNE     NOTA                    ;BR IF NO
3036  007642  112021                    MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "A"
3037  007644  000411                    BR      NEXT
3038  007646  122710  000102    NOTA:   CMPB    #'B,@R0                 ;WAS DRIVE "B" SELECTED?
3039  007652  001002                    BNE     NOTB                    ;BR IF NO
3040  007654  112021                    MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "B"
3041  007656  000404                    BR      NEXT
3042  007660  122710  000054    NOTB:   CMPB    #54,@R0                 ;WAS A COMMA TYPED?
3043  007664  001006                    BNE     NOTLGL                  ;BR IF NO
3044  007666  105720                    TSTB    (R0)+                   ;DUMP THE COMMA
3045  007670  105710            NEXT:   TSTB    @R0                     ;TERMINATOR?
3046  007672  001406                    BEQ     EXIT                    ;BR IF YES
3047  007674  022701  001254            CMP     #DRVKEY+2,R1            ;TWO DRIVES SELECTED?
3048  007700  101355                    BHI     LOOP                    ;BR IF NO
3049  007702  104400  001202    NOTLGL: TYPE    ,$QUES                  ;ILLEGAL INPUT DETECTED
3050  007706  000740                    BR      ASKDRV                  ;GO TRY AGAIN
3051  007710  005767  171336    EXIT:   TST     DRVKEY                  ;ANY DRIVE SELECTED?
3052  007714  001772                    BEQ     NOTLGL                  ;BR IF NO
3053  007716  000207                    RTS     PC
3054
```

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

007720  010046                  ASKADR: MOV     R0,-(SP)                ;SAVE R0
007722  104400  013326          1$:     TYPE    ,MSGASK                 ;"TACS?"
007726  104410                          RDOCT                           ;GET VALUE
007730  012600                          MOV     (SP)+,R0                ;PICK UP THE OCTAL NUMBER
007732  001411                          BEQ     3$                      ;IF "0" USE OLD VALUES
007734  020027  160000                  CMP     R0,#160000              ;MAKE SURE IT IS A BUS ADDRESS
007740  103770                          BLO     1$
007742  010037  001240                  MOV     R0,@#TACSL              ;SAVE THE TACS
007746  062700  000002                  ADD     #2,R0                   ;STEP TO TADB ADDRESS
007752  010037  001242                  MOV     R0,@#TADBL              ;AND SAVE IT
007756  104400  013335          3$:     TYPE    ,MSGVEC                 ;"VECTOR?"
007762  104410                          RDOCT
007764  012600                          MOV     (SP)+,R0
007766  001411                          BEQ     5$
007770  020027  001000                  CMP     R0,#1000                ;MAKE SURE ADDRESS IS IN VECTOR AREA
007774  103370                          BHIS    3$
007776  010037  001244                  MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
010002  062700  000002                  ADD     #2,R0
010006  010037  001246                  MOV     R0,@#TAVEC+2
010012  104400  013346          5$:     TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
010016  104410                          RDOCT
010020  012600                          MOV     (SP)+,R0
010022  001413                          BEQ     6$                      ;IF "0" USE OLD VALUE
010024  020027  000007                  CMP     R0,#7                   ;MAKE SURE ITS VALID
010030  101370                          BHI     5$
010032  000300                          SWAB    R0                      ;PUT INTO HIGH BYTE
010034  006200                          ASR     R0                      ;AND SHIFT
010036  006200                          ASR     R0                      ;INTO PROPER
010040  006200                          ASR     R0                      ;POSITION
010042  042700  177437                  BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
010046  010037  001250                  MOV     R0,@#TAPRIO             ;STORE IT AWAY
010052  104400  013361          6$:     TYPE    ,MTACS                  ;TACS="
010056  016746  171156                  MOV     TACSL,-(SP)     ;;SAVE TACSL FOR TYPEOUT
010062  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010064  104400  013370                  TYPE    ,MTADB                  ;"TADB="
010070  016746  171146                  MOV     TADBL,-(SP)     ;;SAVE TADBL FOR TYPEOUT
010074  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010076  104400  013377                  TYPE    ,MTAVEC                 ;"VECTOR="
010102  016746  171136                  MOV     TAVEC,-(SP)     ;;SAVE TAVEC FOR TYPEOUT
010106  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010110  104400  013411                  TYPE    ,MTAPRI                 ;"PRIORITY="
010114  016746  171130                  MOV     TAPRIO,-(SP)    ;;SAVE TAPRIO FOR TYPEOUT
010120  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010122  104400  013425                  TYPE    ,MSGOK                  ;"OK?"
010126  104406                          RDCHR                   ;GO READ ONE CHARACTER
010130  012600                          MOV     (SP)+,R0                ;GET IT
010132  022700  000015                  CMP     #15,R0                  ;IS IT "CR"?
010136  001406                          BEQ     7$                      ;BRANCH IF YES
010140  022700  000131                  CMP     #'Y,R0                  ;IS IT "Y"?
010144  001403                          BEQ     7$                      ;IT WAS
010146  104400  001202                  TYPE    ,$QUES                  ;TYPE "?"
```

```
3111  010152  000663                    BR      1$                      ;AND LET HIM CORRECT THEM
3112  010154  104400  013432    7$:     TYPE    ,MYES                   ;TYPE OUT "YES"
3113  010160  012600                    MOV     (SP)+,R0                ;RESTORE R0
3114  010162  000207                    RTS     PC                      ;AND RETURN
```

```
3115                                    ;;*****************************************************************
3116                                    ;
3117                                    .SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
3118
3119                                    ;CALL:
3120                                    ;       MOV     #DRVKEY,R0
3121                                    ;       JSR     PC,@#EXAM               ;R1 IS DESTROYED
3122                                    ;       NORMAL RETURN
3123                                    ;       ERROR RETURN
3124
3125  010164  013701  001240        EXAM:   MOV     @#TACSL,R1              ;PICKUP THE "CONTROL & STATUS" REG. ADR.
3126  010170  005011                        CLR     (R1)                    ;DRIVE="A", FUNCTION="WFG"
3127  010172  122710  000101                CMPB    #'A,(R0)                ;EXAMINE DRIVE "A"?
3128  010176  001402                        BEQ     1$                      ;BR IF YES
3129  010200  052711  000400                BIS     #UNIT,(R1)              ;SELECT DRIVE "B"
3130  010204  032711  000040        1$:     BIT     #READY,(R1)             ;WAIT ON READY
3131  010210  001775                        BEQ     1$
3132  010212  005711                        TST     (R1)                    ;ANY ERROR?
3133  010214  100024                        BPL     4$                      ;BR IF NO
3134  010216  032711  001000                BIT     #OFFLINE,(R1)           ;ERROR DUE TO "OFF LINE"?
3135  010222  001017                        BNE     3$                      ;BR IF YES
3136  010224  032711  010000                BIT     #WRTLOCK,(R1)           ;ERROR DUE TO "WRITE LOCK"?
3137  010230  001411                        BEQ     2$                      ;BR IF NO
3138  010232  122777  000201  170700        CMPB    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
3139  010240  001412                        BEQ     4$                      ;BR IF YES
3140  010242  122777  000203  170670        CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
3141  010250  001406                        BEQ     4$                      ;BR IF YES
3142  010252  000403                        BR      3$                      ;TAKE THE ERROR EXIT
3143  010254  032711  020000        2$:     BIT     #LEADER,(R1)            ;ERROR DUE TO "CLEAR LEADER"?
3144  010260  001002                        BNE     4$                      ;BR IF YES
3145  010262  062716  000002        3$:     ADD     #2,(SP)                 ;TAKE ERROR RETURN
3146  010266  000207                4$:     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
                                ;*

010270  105767  170663          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
010274  100002                         BPL     1$              ;;BR IF YES
010276  000000                         HALT                    ;;HALT HERE IF NO TERMINAL
010300  000407                         BR      3$              ;;LEAVE
010302  010046                  1$:    MOV     R0,-(SP)        ;;SAVE R0
010304  017600  000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
010310  112046                  2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
010312  001005                         BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
010314  005726                         TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
010316  012600                  60$:   MOV     (SP)+,R0        ;;RESTORE R0
010320  062716  000002          3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
010324  000002                         RTI                     ;;RETURN
010326  122716  000011          4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
010332  001430                         BEQ     8$
010334  122716  000200                 CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
010340  001006                         BNE     5$
010342  005726                         TST     (SP)+           ;;POP  <CR><LF> EQUIV
010344  104400                         TYPE                    ;;TYPE A CR AND LF
010346  001203                         $CRLF
010350  105067  000130                 CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
010354  000755                         BR      2$              ;;GET NEXT CHARACTER
010356  004767  000056          5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
010362  126726  170570          6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
010366  001350                         BNE     2$              ;;IF NO GO GET NEXT CHAR.
010370  016746  170560                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                               ;;AND THE NULL CHAR.
010374  105366  000001          7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
010400  002770                         BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
010402  004767  000032                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
010406  105367  000072                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
010412  000770                         BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

010414  112716  000040          8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
010420  004767  000014          9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
010424  132767  000007  000052         BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
010432  001372                         BNE     9$              ;;TAB STOP
010434  005726                         TST     (SP)+           ;;POP SPACE OFF STACK
```

```
3203  010436  000724                          BR      2$              ;;GET NEXT CHARACTER
3204  010440  105777  170504          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3205  010444  100375                          BPL     $TYPEC
3206  010446  116677  000002  170476          MOVB    2(SP),@$TPB     ;;LOAD CHAR TC BE TYPED INTO DATA REG.
3207  010454  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3208  010462  001003                          BNE     1$              ;;BRANCH IF NO
3209  010464  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3210  010470  000406                          BR      $TYPEX          ;;EXIT
3211  010472  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3212  010500  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
3213  010502  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
3214  010504  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3215  010506  000207          $TYPEX: RTS     PC
3216
3217                          .SBTTL  TTY INPUT ROUTINE
3218
3219                          ;;**************************************************************
3220                          .ENABL  LSB
3221
3222                          ;;**************************************************************
3223                          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3224                          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3225                          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3226                          ;*WHEN OPERATING IN TTY FLAG MODE.
3227  010510  022767  000176  170422  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
3228  010516  001074                          BNE     15$             ;;BRANCH IF NO
3229  010520  105777  170420                  TSTB    @$TKS           ;;CHAR THERE?
3230  010524  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
3231  010526  117746  170414                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
3232  010532  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
3233  010536  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
3234  010542  001062                          BNE     15$             ;;NO, RETURN TO USER
3235  010544  126727  170364  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
3236  010552  001456                          BEQ     15$             ;;BRANCH IF YES
3237
3238  010554  104400  011235                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
3239  010560  104400  011242          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
3240  010564  016746  167406                  MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
3241  010570  104401                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3242  010572  104400  011253                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
3243  010576  005046                  19$:     CLR     -(SP)           ;;CLEAR COUNTER
3244  010600  005046                          CLR     -(SP)           ;;THE NEW SWR
3245  010602  105777  170336          7$:     TSTB    @$TKS           ;;CHAR THERE?
3246  010606  100375                          BPL     7$              ;;IF NOT TRY AGAIN
3247
3248  010610  117746  170332                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
3249  010614  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3250
3251
3252
3253  010620  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
3254  010624  001005                          BNE     10$             ;;BRANCH IF NOT
3255  010626  104400  011230                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
3256  010632  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
3257  010636  000757                          BR      19$             ;;LET'S TRY IT AGAIN
3258
```

```
3259
3260  010640  021627  000015          10$:  CMP   (SP),#15        ;;IS IT A <CR>?
3261  010644  001022                        BNE   16$             ;;BRANCH IF NO
3262  010646  005766  000004                TST   4(SP)           ;;YES, IS IT THE FIRST CHAR?
3263  010652  001403                        BEQ   11$             ;;BRANCH IF YES
3264  010654  016677  000002  170256        MOV   2(SP),@SWR      ;;SAVE NEW SWR
3265  010662  062706  000006          11$:  ADD   #6,SP           ;;CLEAR UP STACK
3266  010666  104400  001203          14$:  TYPE  ,$CRLF          ;;ECHO <CR> AND <LF>
3267  010672  126727  170237  000001        CMPB  $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
3268  010700  001003                        BNE   15$             ;;BRANCH IF NOT
3269  010702  012777  000100  170234        MOV   #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
3270  010710  000002          15$:  RTI                           ;;RETURN
3271  010712  004767  177522          16$:  JSR   PC,$TYPEC       ;;ECHO CHAR
3272  010716  021627  000060                CMP   (SP),#60        ;;CHAR < 0?
3273  010722  002420                        BLT   18$             ;;BRANCH IF YES
3274  010724  021627  000067                CMP   (SP),#67        ;;CHAR > 7?
3275  010730  003015                        BGT   18$             ;;BRANCH IF YES
3276  010732  042726  000060                BIC   #60,(SP)+       ;;STRIP-OFF ASCII
3277  010736  005766  000002                TST   2(SP)           ;;IS THIS THE FIRST CHAR
3278  010742  001403                        BEQ   17$             ;;BRANCH IF YES
3279  010744  006316                        ASL   (SP)            ;;NO, SHIFT PRESENT
3280  010746  006316                        ASL   (SP)            ;;   CHAR OVER TO MAKE
3281  010750  006316                        ASL   (SP)            ;;   ROOM FOR NEW ONE.
3282  010752  005266  000002          17$:  INC   2(SP)           ;;KEEP COUNT OF CHAR
3283  010756  056616  177776                BIS   -2(SP),(SP)     ;;SET IN NEW CHAR
3284  010762  000707                        BR    7$              ;;GET THE NEXT ONE
3285  010764  104400  001202          18$:  TYPE  ,$QUES          ;;TYPE ?<CR><LF>
3286  010770  000720                        BR    20$             ;;SIMULATE CONTROL-U
3287                                .DSABL  LSB
3288
3289
3290                                ;;*****************************************************************
3291                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3292                                ;*CALL:
3293                                ;*      RDCHR                  ;;INPUT A SINGLE CHARACTER FROM THE TTY
3294                                ;*      RETURN HERE            ;;CHARACTER IS ON THE STACK
3295                                ;*                             ;;WITH PARITY BIT STRIPPED OFF
3296                                ;
3297
3298  010772  011646          $RDCHR: MOV   (SP),-(SP)      ;;PUSH DOWN THE PC
3299  010774  016666  000004  000002  MOV   4(SP),2(SP)      ;;SAVE THE PS
3300  011002  105777  170136          1$:  TSTB  @$TKS             ;;WAIT FOR
3301  011006  100375                       BPL   1$              ;;A CHARACTER
3302  011010  117766  170132  000004       MOVB  @$TKB,4(SP)     ;;READ THE TTY
3303  011016  042766  177600  000004       BIC   #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3304  011024  026627  000004  000023       CMP   4(SP),#23       ;;IS IT A CONTROL-S?
3305  011032  001013                       BNE   3$              ;;BRANCH IF NO
3306  011034  105777  170104          2$:  TSTB  @$TKS           ;;WAIT FOR A CHARACTER
3307  011040  100375                       BPL   2$              ;;LOOP UNTIL ITS THERE
3308  011042  117746  170100               MOVB  @$TKB,-(SP)     ;;GET CHARACTER
3309  011046  042716  177600               BIC   #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3310  011052  022627  000021               CMP   (SP)+,#21       ;;IS IT A CONTROL-Q?
3311  011056  001366                       BNE   2$              ;;IF NOT DISCARD IT
3312  011060  000750                       BR    1$              ;;YES, RESUME
3313  011062  026627  000004  000140  3$:  CMP   4(SP),#140      ;;IS IT UPPER CASE?
3314  011070  002407                       BLT   4$              ;;BRANCH IF YES
```

```
3315  011072  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
3316  011100  003003                          BGT     4$              ;;BRANCH IF YES
3317  011102  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
3318  011110  000002                  4$:     RTI                     ;;GO BACK TO USER
3319                                  ;;**************************************************************
3320                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3321                                  ;*CALL:
3322                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3323                                  ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3324                                  ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3325
3326  011112  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3327  011114  012703  011220          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
3328  011120  022703  011230          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
3329  011124  101405                          BLOS    4$              ;;BR IF YES
3330  011126  104406                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
3331  011130  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
3332  011132  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
3333  011136  001003                          BNE     3$              ;;SKIP IF NOT
3334  011140  104400  001202          4$:     TYPE    ,$QUES          ;;TYPE A '?'
3335  011144  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
3336  011146  111367  000044          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
3337  011152  104400  011216                  TYPE    ,9$
3338  011156  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
3339  011162  001356                          BNE     2$              ;;LOOP IF NOT RETURN
3340  011164  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3341  011170  104400  001204                  TYPE    ,$LF            ;;TYPE A LINE FEED
3342  011174  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3343  011176  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3344  011200  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3345  011206  012766  011220  000004          MOV     #$TTYIN,4(SP)
3346  011214  000002                          RTI                     ;;RETURN
3347  011216     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
3348  011217     000                  .BYTE   0                       ;;TERMINATOR
3349  011220  000010          $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
3350  011230  052536  005015  000     $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
3351  011235     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
3352  011242  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
3353  011250  020075     000
3354  011253     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
3355  011260  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

011264  017546  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
011270  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
011276  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
011302  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
011306  000406                          BR      $TYPON
011310  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
011316  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
011324  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
011332  010346                          MOV     R3,-(SP)        ;;SAVE R3
011334  010446                          MOV     R4,-(SP)        ;;SAVE R4
011336  010546                          MOV     R5,-(SP)        ;;SAVE R5
011340  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
011344  005404                          NEG     R4
011346  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
011352  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
011356  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
011362  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
011366  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
011370  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
011372  000404                          BR      3$              ;;GO DO MSB
011374  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
011376  006105                          ROL     R5
011400  006105                          ROL     R5
011402  010503                          MOV     R5,R3
011404  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
011406  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
011412  100016                          BPL     7$              ;;BR IF NO
011414  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
011420  001002                          BNE     4$              ;;TEST FOR 0
011422  005704                          TST     R4              ;;SUPPRESS THIS 0?
011424  001403                          BEQ     5$              ;;BR IF YES
```

```
 3412  011426  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 3413  011430  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 3414  011434  052703  000040              5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 3415  011440  110367  000040                    MOVB    R3,8$           ;;SAVE FOR TYPING
 3416  011444  104400  011504                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
 3417  011450  105367  000032              7$:   DECB    $OCNT           ;;COUNT BY 1
 3418  011454  003347                          BGT     2$              ;;BR IF MORE TO DO
 3419  011456  002402                          BLT     6$              ;;BR IF DONE
 3420  011460  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 3421  011462  000744                          BR      2$              ;;GO DO THE LAST DIGIT
 3422  011464  012605                    6$:   MOV     (SP)+,R5        ;;RESTORE R5
 3423  011466  012604                          MOV     (SP)+,R4        ;;RESTORE R4
 3424  011470  012603                          MOV     (SP)+,R3        ;;RESTORE R3
 3425  011472  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 3426  011500  012616                          MOV     (SP)+,(SP)
 3427  011502  000002                          RTI                     ;;RETURN
 3428  011504     000                    8$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 3429  011505     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 3430  011506     000                 $OCNT:   .BYTE   0               ;;OCTAL DIGIT COUNTER
 3431  011507     000                 $OFILL:  .BYTE   0               ;;ZERO FILL SWITCH
 3432  011510  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.

011512  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
011514  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
011520  005740                          TST     -(R0)           ;;BACKUP BY 2
011522  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
011524  006300                          ASL     R0              ;;POSITION FOR INDEXING
011526  016000  011534                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
011532  000200                          RTS     R0              ;;GO TO ROUTINE

                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
011534                          $TRPAD:
011534  010270                          $TYPE   ;;CALL=TYPE     TRAP+0(104400)  TTY TYPEOUT ROUTINE
011536  011310                          $TYPOC  ;;CALL=TYPOC    TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
011540  011264                          $TYPOS  ;;CALL=TYPOS    TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
011542  011324                          $TYPON  ;;CALL=TYPON    TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)

011544  010560                          $GTSWR  ;;CALL=GTSWR    TRAP+4(104404)  GET SOFT-SWR SETTING

011546  010510                          $CKSWR  ;;CALL=CKSWR    TRAP+5(104405)  TEST FOR CHANGE IN SOFT-SWR
011550  010772                          $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
011552  011112                          $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
011554  007452                          $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
011556  006616                          $SAVREG ;;CALL=SAVREG   TRAP+11(104411) SAVE R0-R5 ROUTINE
011560  006654                          $RESREG ;;CALL=RESREG   TRAP+12(104412) RESTORE R0-R5 ROUTINE
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;**********************************************************************
                                        ;POWER DOWN ROUTINE
011562  012737  011726  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
011570  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
011576  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011600  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011602  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011604  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011606  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
011610  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011612  017746  167322                  MOV     @$WR,-(SP)      ;;PUSH @SWR ON STACK
011616  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
011622  012737  011634  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
011630  000000                          HALT
011632  000776                          BR      .-2             ;;HANG UP
                                        ;;**********************************************************************
                                        ;POWER UP ROUTINE
011634  012737  011726  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
011642  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
011646  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
011652  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
011656  001375                          BNE     1$              ;;OF  WORD
011660  012677  167254                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
011664  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
011666  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
011670  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
011672  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
011674  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
011676  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
011700  012737  011562  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
011706  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
011714  104400                          TYPE                    ;REPORT THE POWER FAILURE
011716  011734          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
011720  012716                  MOV     (PC)+,(SP)              ;;RESTART AT $EOP
011722  004406          $PWRAD: .WORD   $EOP                    ;;RESTART ADDRESS
011724  000002                  RTI
011726  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
011730  000776                  BR      .-2                     ;; BEFORE THE POWER DOWN WAS COMPLETE
011732  000000          $SAVR6: 0                               ;;PUT THE SP HERE
011734  005015  047520  042527 $POWER: .ASCIZ  <15><12>"POWER"
011742  000122
                                        .EVEN
```

```
                                        ;;*****************************************************************
                                        ;DATA POINTERS
                                        
011744  001116  001162  001166  DT1:    .WORD   $ERRPC,$REG0,$REG2,BYTNUM,$GDDAT,$BDDAT,$GDADR,$BDADR,0
011752  001230  001124  001126
011760  001120  001122  000000
                                        
011766  001116  001162  001166  DT2:    .WORD   $ERRPC,$REG0,$REG2,$TMP0,$TMP2,0
011774  001172  001176  000000
                                        
012002  001116  001162  001166  DT101:  .WORD   $ERRPC,$REG0,$REG2,$TMP0,0
012010  001172  000000
                                        
012014  001116  001240  000000  DT201:  .WORD   $ERRPC,TACSL,0
                                        
012022  001116  000000          DT202:  .WORD   $ERRPC,0
                                        
                                        ;DATA FORMATS
        000000                  ON=0    ;OCTAL NUMBER
        000001                  DN=1    ;DECIMAL NUMBER
                                        
012026      000                 DF1:    .BYTE   ON      ;$ERRPC
012027      001                         .BYTE   DN      ;$REG0
012030      001                         .BYTE   DN      ;$REG2
012031      001                         .BYTE   DN      ;BYTNUM
012032      000                         .BYTE   ON      ;$GDDAT
012033      000                         .BYTE   ON      ;$BDDAT
012034      000                         .BYTE   ON      ;$GDADR
012035      000                         .BYTE   ON      ;$BDADR
                                        
012036      000                 DF2:    .BYTE   ON      ;$ERRPC
012037      001                         .BYTE   DN      ;$REG0
012040      001                         .BYTE   DN      ;$REG2
012041      001                         .BYTE   DN      ;$TMP0
012042      001                         .BYTE   DN      ;$TMP1
```

```
3552                                    ;;**********************************************************************
3553
3554                                    ;MESSAGES
3555
3556  012043    127  044522  042524  MXWFG:  .ASCIZ  /WRITE-FILE-GAP/
3557  012050 043055  046111  026505
3558  012056 040507  000120
3559  012062 051127  052111  000105  MXWRIT: .ASCIZ  /WRITE/
3560  012070 042522  042101     000  MXREAD: .ASCIZ  /READ/
3561  012075    102  041501  026513  MXBSFG: .ASCIZ  /BACK-SPACE-FILE-GAP/
3562  012102 050123  041501  026505
3563  012110 044506  042514  043455
3564  012116 050101     000
3565  012121    102  041501  026513  MXBSBG: .ASCIZ  /BACK-SPACE-BLK-GAP/
3566  012126 050123  041501  026505
3567  012134 046102  026513  040507
3568  012142 000120
3569  012144 050123  041501  026505  MXSFFG: .ASCIZ  /SPACE-FWD-FILE-GAP/
3570  012152 053506  026504  044506
3571  012160 042514  043455  050101
3572  012166    000
3573  012167    123  040520  042503  MXSFBG: .ASCIZ  /SPACE-FWD-BLK-GAP/
3574  012174 043055  042127  041055
3575  012202 045514  043455  050101
3576  012210    000
3577  012211    122  053505  047111  MXRWND: .ASCIZ  /REWIND/
3578  012216 000104
3579  012220 040504  040524  042440  EM1:    .ASCIZ  "DATA ERROR"
3580  012226 051122  051117     000
3581  012233    123  047131  020103  EM2:    .ASCIZ  "SYNC ERROR"
3582  012240 051105  047522  000122
3583  012246 051104  053111  020105  EM101:  .ASCIZ  "DRIVE IS OFF-LINE"
3584  012254 051511  047440  043106
3585  012262 046055  047111  000105
3586  012270 051104  053111  020105  EM102:  .ASCIZ  "DRIVE IS WRITE-LOCK"
3587  012276 051511  053440  044522
3588  012304 042524  046055  041517
3589  012312 000113
3590  012314 046103  040505  020122  EM103:  .ASCIZ  "CLEAR LEADER ERROR"
3591  012322 042514  042101  051105
3592  012330 042440  051122  051117
3593  012336    000
3594  012337    106  046111  020105  EM104:  .ASCIZ  "FILE GAP ERROR"
3595  012344 040507  020120  051105
3596  012352 047522  000122
3597  012356 044524  044515  043516  EM105:  .ASCIZ  "TIMING ERROR"
3598  012364 042440  051122  051117
3599  012372    000
3600  012373    102  047514  045503  EM106:  .ASCIZ  "BLOCK CHECK ERROR"
3601  012400 041440  042510  045503
3602  012406 042440  051122  051117
3603  012414    000
3604  012415    125  045516  047516  EM107:  .ASCIZ  "UNKNOWN INTERRUPT"
3605  012422 047127  044440  052116
3606  012430 051105  052522  052120
3607  012436    000
```

```
3608  012437  124     030501  020061  EM201:  .ASCIZ  "TA11 FAILED TO RESPOND"
3609  012444  040506  046111  042105
3610  012452  052040  020117  042522
3611  012460  050123  047117  000104
3612  012466  047516  042040  044522  EM202:  .ASCIZ  "NO DRIVE AVAILABLE"
3613  012474  042526  040440  040526
3614  012502  046111  041101  042514
3615  012510  000
3616  012511  120     020103  020040  DH1:    .ASCIZ  /PC      FILE    BLOCK   BYTE    GDDAT   BDDAT   GDADR   BDADR/
3617  012516  020040  043040  046111
3618  012524  020105  020040  041040
3619  012532  047514  045503  020040
3620  012540  041040  052131  020105
3621  012546  020040  043440  042104
3622  012554  052101  020040  041040
3623  012562  042104  052101  020040
3624  012570  043440  040504  051104
3625  012576  020040  041040  040504
3626  012604  051104  000
3627  012607  040     020040  020040  DH2:    .ASCII  /       EXPT'D  EXPT'D  RCV'D   RCV'D/<15><12>
3628  012614  020040  042440  050130
3629  012622  023524  020104  042440
3630  012630  050130  023524  020104
3631  012636  051040  053103  042047
3632  012644  020040  051040  053103
3633  012652  042047  005015
3634  012656  041520  020040  020040          .ASCIZ  /PC      FILE    BLOCK   FILE    BLOCK/
3635  012664  020040  044506  042514
3636  012672  020040  020040  046102
3637  012700  041517  020113  020040
3638  012706  044506  042514  020040
3639  012714  020040  046102  041517
3640  012722  000113
3641  012724  041520  020040  020040  DH101:  .ASCIZ  /PC      FILE    BLOCK   FUNCTION/
3642  012732  020040  044506  042514
3643  012740  020040  020040  046102
3644  012746  041517  020113  020040
3645  012754  052506  041516  044524
3646  012762  047117  000
3647  012765  120     020103  020040  DH201:  .ASCIZ  /PC      TACS/
3648  012772  020040  052040  041501
3649  013000  000123
3650  013002  041520  000             DH202:  .ASCIZ  /PC/
3651  013005  200     025052  020052  MSG1:   .ASCIZ  <CRLF>"*** END-OF-TEST ***"
3652  013012  047105  026504  043117
3653  013020  052055  051505  020124
3654  013026  025052  000052
3655  013032  051600  043117  020124  MSG2:   .ASCIZ  <CRLF>"SOFT ERRORS="
3656  013040  051105  047522  051522
3657  013046  000075
3658  013050  044200  051101  020104  MSG3:   .ASCIZ  <CRLF>"HARD ERRORS="
3659  013056  051105  047522  051522
3660  013064  000075
3661  013066  041200  052131  051505  MSG4:   .ASCIZ  <CRLF>"BYTES READ="
3662  013074  051040  040505  036504
3663  013102  000
```

```
 3664  013103     200  054502  042524  MSG5:   .ASCIZ  <CRLF>"BYTES WRITTEN="
 3665  013110  020123  051127  052111
 3666  013116  042524  036516     000
 3667  013123     200  040524  042520  MSG6:   .ASCIZ  <CRLF>"TAPE PASSES="
 3668  013130  050040  051501  042523
 3669  013136  036523     000
 3670  013141     200  044506  042514  MSG7:   .ASCIZ  <CRLF>"FILES/PASS="
 3671  013146  027523  040520  051523
 3672  013154  000075
 3673  013156  041200  047514  045503  MSG8:   .ASCIZ  <CRLF>"BLOCKS/FILE="
 3674  013164  027523  044506  042514
 3675  013172  000075
 3676  013174  040440  042116     000  MSG9:   .ASCIZ  " AND"
 3677  013201     040  044527  046114  MSG10:  .ASCIZ  " WILL BE TESTED"<CRLF>
 3678  013206  041040  020105  042524
 3679  013214  052123  042105  000200
 3680  013222  025052  020052  047506  MSG11:  .ASCIZ  "*** FORMAT ***"
 3681  013230  046522  052101  025040
 3682  013236  025052     000
 3683  013241     052  025052  051040  MSG12:  .ASCIZ  "*** READ   ***"
 3684  013246  040505  020104  020040
 3685  013254  025052  000052
 3686  013260  025052  020052  051127  MSG13:  .ASCIZ  "*** WRITE  ***"
 3687  013266  052111  020105  025040
 3688  013274  025052     000
 3689  013277     040  000040          MSG14:  .ASCIZ  "  "
 3690  013302  042040  044522  042526  MSG15:  .ASCIZ  " DRIVE "
 3691  013310  000040
 3692  013312  042200  044522  042526  MSGDRV: .ASCIZ  <CRLF>"DRIVE(S)? "
 3693  013320  051450  037451  000040
 3694  013326  052200  041501  037523  MSGASK: .ASCIZ  <CRLF>"TACS?"
 3695  013334     000
 3696  013335     200  042526  052103  MSGVEC: .ASCIZ  <CRLF>"VECTOR?"
 3697  013342  051117  000077
 3698  013346  050200  044522  051117  MSGPRI: .ASCIZ  <CRLF>"PRIORITY?"
 3699  013354  052111  037531     000
 3700  013361     200  040524  051503  MTACS:  .ASCIZ  <CRLF>"TACS="
 3701  013366  000075
 3702  013370  052040  042101  036502  MTADB:  .ASCIZ  / TADB=/
 3703  013376     000
 3704  013377     040  053040  041505  MTAVEC: .ASCIZ  "  VECTOR="
 3705  013404  047524  036522     000
 3706  013411     040  050040  044522  MTAPRI: .ASCIZ  "  PRIORITY="
 3707  013416  051117  052111  036531
 3708  013424     000
 3709  013425     200  045517  000077  MSGOK:  .ASCIZ  <CRLF>"OK?"
 3710  013432  042531  100123     000  MYES:   .ASCIZ  "YES"<CRLF>
 3711          013440                  .EVEN
 3712
 3713                                  .SBTTL          READ AND WRITE BUFFER
 3714
 3715  013440  002004                  BUFFER: .BLKB   1028.
 3716          015444                  LASTADDRESS=    .
 3717
 3718          000001                          .END
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
ASKADR  007720          1562    3060#
ASKDRV  007610          1559    3027#   3050
BACK    007144          2843    2846#
BADFUN  005456          2391    2397#
BC.ERR  006040          2460    2502#
BEGINX  002322          1556#   1592    1594    1655    1657
BEGIN1  001752          1171    1482#
BEGIN2  002004          1172    1489#
BEGIN3  002012          1173    1488    1491#
BGNCMN  002016          1485    1487    1490    1492#
BIT0   =  000001        1115#
BIT00  =  000001        1105#   1115    1158    1681    1719    3138    3140
BIT01  =  000002        1104#   1114    1157    1719    3140
BIT02  =  000004        1103#   1113    1156
BIT03  =  000010        1102#   1112    1155
BIT04  =  000020        1101#   1111    1154
BIT05  =  000040        1100#   1110    1153
BIT06  =  000100        1099#   1109    1152
BIT07  =  000200        1098#   1108    1151    1719    3138    3140
BIT08  =  000400        1097#   1107    1150
BIT09  =  001000        1096#   1106    1149
BIT1   =  000002        1114#
BIT10  =  002000        1095#   1148    2297
BIT11  =  004000        1094#   1147
BIT12  =  010000        1093#   1146
BIT13  =  020000        1092#   1145    2304
BIT14  =  040000        1091#   1144    2259
BIT15  =  100000        1090#   1143
BIT2   =  000004        1113#
BIT3   =  000010        1112#
BIT4   =  000020        1111#
BIT5   =  000040        1110#
BIT6   =  000100        1109#
BIT7   =  000200        1108#
BIT8   =  000400        1107#
BIT9   =  001000        1106#
BLKSZ   001312          1267#   2526    2528    2591    2593    2647    2672
BLOCK   001306          1259#   1790    1889    1890    1897    1940    2006    2016    2017    2024    2068    2114    2115
                        2122    2427    2484    2485    2488*   2523    2562    2588    2624    2629    2644    2675
BPTVEC=  000014         1122#
BSBG    006732          1834    1876    1964    2785#
BSFG    006726          2783#
BUFFER  013440          1367    2560    2613    2630    2631    2648    3715#
BYTNUM  001230          1234#   2673*   3519
C      =  000016        1734#   1735    2453#
CASINT  007046          1596    2819#
CASPAR  007152          2799    2854#
CBCNT   007154          2486    2823    2855#
CHANOK  007012          2803#
CHKADR  002344          1558    1561    1580#
CHKDRV  002436          1621#
CHKRDY  006756          2794#   2795
CKEOTS  004360          1815    1828    2009    2093    2107    2144#
CKSWR  =  104405        2258    2293    2312    3464#
CLEAR   002760          1674    1677    1695#   1697
CL.ERR  005702          2457    2469#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
CNTHRD  006576          1867    1996    2710#
CNTSFT  006540          1865    1994    2693#
CR    = 000015          1030#   3207    3217
CRCERR= 040000          1144#   2444
CRLF  = 000200          1031#   1545    3178    3217    3651    3655    3658    3661    3664    3667    3670    3673    3677
                        3692    3694    3696    3698    3700    3709    3710
CSRERR  005526          1802    1813    1826    1838    1849    1880    1952    1968    1979    2033    2080    2091    2105
                        2426#
DATCMP  006346          1862    1991    2644#
DDISP = 177570          1037#   1204    1516
DF1     012026          1399    3537#
DF2     012036          1406    3546#
DH1     012511          1397    3616#
DH101   012724          1415    1421    1427    1433    1439    1445    1451    3641#
DH2     012607          1403    3627#
DH201   012765          1463    3647#
DH202   013002          1468    3650#
DISPLA  001142          1204#   1516*   1524*   2276*   2296*
DISPRE  000174          1168#   1524
DN    = 000001          3535#   3538    3539    3540    3547    3548    3549    3550
DOFUN   007034          2810#
DRVKEY  001252          1246#   1247    1483*   1621    1634    1685    3028*   3033    3047    3051
DRVPNT  001256          1247#   1635*   1678    1686*
DSWR  = 177570          1036#   1203    1515
DT1     011744          1398    3519#
DT101   012002          1416    1422    1428    1434    1440    1446    1452    3526#
DT2     011766          1405    3523#
DT201   012014          1464    3529#
DT202   012022          1469    3531#
EMTVEC= 000030          1125#   1503*   1504*
EM1     012220          1396    3579#
EM101   012246          1414    3583#
EM102   012270          1420    3586#
EM103   012314          1426    3590#
EM104   012337          1432    3594#
EM105   012356          1438    3597#
EM106   012373          1444    3600#
EM107   012415          1450    3604#
EM2     012233          1402    3581#
EM201   012437          1462    3608#
EM202   012466          1467    3612#
EOTS    001212          1229#   2008*   2145    2204    2491*
EREXT1  005516          2359    2416#
EREXT2  005520          2394    2409    2417#
ERROR = 100000          1143#
ERROR0  005310          2348    2353#
ERROR1  005366          2349    2373#
ERROR2  005476          2350    2407#
ERRVEC= 000004          1118#   1513    1514*   1525*   1580*   1586*   2264    2265*   2267*   2270*
EXAM    010164          1622    1626    1631    3125#
EXIT    007710          3046    3051#
FGAP  = 004000          1147#   2440
FG.ERR  006030          2458    2496#
FILE    001304          1257#   1793    1894    1943    2004    2021    2071    2119    2426    2474    2475    2481    2561
                        2626    2628    2650    2674
FILESZ  001310          1261#   1890    2017    2115    2216    2476
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
FILL    006126          2530    2554#
FORMAT  003126          1739    1784#
FORPAS  001274          1252    1253#   1691    1728
FUNCTI= 000016          1159#   2429
FUNCO = 000002          1157#   1159
FUNC1 = 000004          1156#   1159
FUNC2 = 000010          1155#   1159
FUNLOC  007160          2428    2796*   2803*   2857#
FUNTAB  006714          2778#   2792
GNS   = ****** U        1167    1544    3457    3458    3459    3460    3462    3464    3465    3466    3467    3468    3469
GO    = 000001          1158#
GTSWR = 104404          1539    2230    3462#
HARDNM  001210          1228#   2189    2710*   2711
HT    = 000011          1028#   3176    3217
ILBS  = 000020          1154#
INT.EN= 000100          1152#
IOCOM   006750          2778    2780    2782    2784    2786    2788    2790    2792#
IOTVEC= 000020          1123#   1501*   1502*
ITEMS0  001622          1390#   2335
ITEMS1  001642          1412#   2336
ITEMS2  001732          1460#   2337
LASTAD= 015444          3716#
LASTBK  001234          1236#   1688*   2006    2476*   2477*   2485*
LASTFL  001236          1238#   1676    1687*   2004    2209    2475*   2481*
LEADER= 020000          1145#   2438    3143
LF    = 000012          1029#   3211    3217
LOOP    007634          3034#   3048
LOOPER  002770          1694    1718#   2149
MAXEOT  001264          1250#   2145
MAXERR  001262          1249#   1690    1883    2711
MAXRDS  001260          1248#   1689    1882    2003    2696    2697
MSGASK  013326          3061    3694#
MSGDRV  013312          3027    3692#
MSGOK   013425          3103    3709#
MSGPRI  013346          3079    3698#
MSGVEC  013335          3070    3696#
MSG1    013005          2181    3651#
MSG10   013201          1650    3677#
MSG11   013222          1785    3680#
MSG12   013241          1935    3683#
MSG13   013260          2063    3686#
MSG14   013277          2357    2379    2412    3689#
MSG15   013302          3010    3690#
MSG2    013032          2183    3655#
MSG3    013050          2188    3658#
MSG4    013066          2193    3661#
MSG5    013103          2198    3664#
MSG6    013123          2203    3667#
MSG7    013141          2208    3670#
MSG8    013156          2215    3673#
MSG9    013174          1647    3676#
MTACS   013361          3091    3700#
MTADB   013370          3094    3702#
MTAPRI  013411          3100    3706#
MTAVEC  013377          3097    3704#
MXBSBG  012121          2401    3565#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
MXBSFG  012075          2400    3561#
MXREAD  012070          2399    3560#
MXRWND  012211          2404    3577#
MXSFBG  012167          2403    3573#
MXSFFG  012144          2402    3569#
MXWFG   012043          2397    3556#
MXWRIT  012062          2398    3559#
MYES    013432          3112    3710#
NEXT    007670          3037    3041    3045#
NOTA    007646          3035    3038#
NOTB    007660          3039    3042#
NOTLGL  007702          3032    3043    3049#   3052
OFFLIN= 001000          1149#   2434    3134
OL.ERR  005672          2455    2463#
ON    = 000000          3534#   3537    3541    3542    3543    3544    3546
PARMBK  001612          1364#   1366    1641*   1643*   1645*   1648*   1682*   1797    1799    1808    1810    1821    1823
                        1833    1835    1844    1846    1875    1877    1947    1949    1963    1965    1974    1976    2028
                        2030    2075    2077    2086    2088    2100    2102    2179    2431    2528*   2593*   2678    3011
PASCNT  001266          1251#   1693*   1721    1723*   1731*
PATS    001352          1288#   2529    2657
PAT0    001412          1288    1310#
PAT1    001422          1289    1313#
PAT10   001532          1298    1340#
PAT11   001542          1299    1343#
PAT12   001552          1300    1346#
PAT13   001562          1301    1349#
PAT14   001572          1302    1352#
PAT15   001602          1303    1355#
PAT2    001432          1290    1316#
PAT3    001442          1291    1319#
PAT4    001452          1292    1322#
PAT5    001462          1293    1325#
PAT6    001472          1294    1328#
PAT7    001502          1295    1331#
PAT8    001512          1296    1334#
PAT9    001522          1297    1337#
PC    =%000007          1049#   1559*   1562*   1622*   1626*   1631*   1644*   1649*   1674*   1677*   1698*   1738*   1786*
                        1788*   1791*   1798*   1809*   1820*   1822*   1834*   1843*   1845*   1855*   1862*   1876*   1887*
                        1892*   1895*   1936*   1938*   1941*   1948*   1957*   1964*   1975*   1984*   1991*   2014*   2019*
                        2022*   2029*   2064*   2066*   2069*   2076*   2087*   2097*   2101*   2112*   2117*   2120*   2166*
                        2169*   2180*   2185*   2186*   2190*   2191*   2195*   2196*   2200*   2201*   2205*   2206*   2212*
                        2213*   2217*   2218*   2236*   2241    2306*   2347*   2366*   2380*   2384*   2419*   2454*   2472*
                        2482*   2530*   2531*   2575*   2594*   2633*   2682*   2777    2779    2781    2783    2785    2787
                        2789    2791    2813*   2896*   2901*   2936*   2939*   2963*   3017*   3053*   3114*   3146*   3185*
                        3192*   3199*   3213*   3215*   3271*   3506
PIRQ  = 177772          1035#
PIRQVE= 000240          1129#
PR0   = 000000          1052#
PR1   = 000040          1053#
PR2   = 000100          1054#
PR3   = 000140          1055#
PR4   = 000200          1056#
PR5   = 000240          1057#
PR6   = 000300          1058#
PR7   = 000340          1059#
PS    = 177776          1032#   1033
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
FSCNTL  001270          1252#  1691*  1692*  1721   1724*  1725*  1726   1728*  1729*  1732
PSW    = 177776         1033#
PWRVEC= 000024          1124#  1507*  1508*  3474*  3475*  3484*  3490*  3502*  3503*
RBYTTL  001214          1230#  2194   2591*  2592*
RDCHR = 104406          3104   3330   3465#
RDLIN = 104407          2979   3029   3466#
RDOCT = 104410          3062   3071   3080   3467#
RDTRYS  001224          1232#  1689*  1882*  2003*  2694*  2696*
RD1PAS  001276          1254#
RD2PAS  001302          1256#  1726
READ    006722          1845   1975   2781#
READON  003556          1740   1742   1933#  1934
READR   007130          2838   2841#
READY = 000040          1153#  3130
RESREG= 104412          2900   3469#
RESVEC= 000010          1119#
REWIND  006746          1798   1948   2076   2180   2791#
RWDONE  007134          2840   2842#
RWILBS  007140          2836   2845#
R0     =%000000         1040#  1581*  1584*  1587   1593*  1621*  1624*  1628*  1630*  1633*  1634*  1635   1636
                        1638*  1639   1641   1645   1656*  1672*  1675*  1679*  1680*  1681*  1682   1695*  1797*
                        1802*  1808*  1813*  1821*  1826*  1833*  1838*  1844*  1849*  1865*  1867*  1875*  1880*
                        1947*  1952*  1963*  1968*  1974*  1979*  1994*  1996*  2028*  2033*  2075*  2080*  2086*
                        2091*  2100*  2105*  2173*  2179*  2209*  2210*  2211   2233*  2236   2323   2332*  2333*
                        2339   2343   2354   2355   2358*  2362   2365   2373   2407*  2408   2410   2418*  2464*
                        2467*  2490   2494*  2497*  2500*  2505   2506*  2509*  2523*  2524*  2525*  2526   2528
                        2529*  2555   2563   2564*  2565   2566   2567   2568   2574*  2588*  2589*  2590*  2591
                        2593   2613*  2614   2616   2619   2621   2624   2626   2644*  2645*  2646*  2647   2657
                        2672   2699   2700*  2713   2714*  2735   2760*  2798   2800   2802*  2803   2804   2805
                        2806   2811*  2876*  2877   2895*  2899*  2951   2952*  2953   2955   2957*  2958   2961*
                        2976   2980*  2983   2999*  3030*  3031   3034   3036   3038   3040   3042   3044   3045
                        3060   3063*  3065   3067   3068*  3069   3072*  3074   3076   3077*  3078   3081*  3083
                        3085*  3086*  3087*  3088*  3089*  3090   3105*  3106   3109   3113*  3127   3168   3169*
                        3170   3173*  3441   3442*  3443   3444*  3445*  3446*  3447*  3476   3501*
R1     =%000001         1041#  1673*  1676*  1678*  1679   1683   1685*  1686   1696*  2324   2331*  2333   2347
                        2354*  2358   2373*  2374   2383   2390*  2391   2417*  2432*  2434   2436   2438   2440
                        2442   2444   2556   2559*  2569*  2573*  2614*  2615*  2616*  2617*  2619*  2620*  2621*
                        2622*  2647*  2651*  2658*  2671*  2672*  2673   2736   2759*  2797   2799*  2800*  2801*
                        2805*  2806*  2807   2808*  2810   2812*  2820   2823*  2829*  2832*  2835*  2839   2841*
                        2842*  2848*  2878*  2884*  2890*  2977   2981*  2985*  2987*  2989*  2992*  2995   2998*
                        3033*  3036*  3040*  3047   3125*  3126*  3129*  3130   3132   3134   3136   3143   3477
                        3500*
R2     =%000002         1042#  2353   2355*  2360   2416*  2557   2560*  2561*  2562*  2565*  2566*  2567*  2568*
                        2572*  2648*  2652   2660   2669   2670   2737   2758*  2821   2822*  2824   2826   2830*
                        2831*  2837   2845*  2847*  2875*  2878   2879*  2885*  2886*  2891*  2892*  2978   2982*
                        2986*  2988*  2990*  2996   2997*  3478   3499*
R3     =%000003         1043#  2558   2563*  2564   2571*  2649*  2654*  2656*  2662*  2738   2757*  2883*  2888*
                        2894*  2895   3326   3327*  3328   3331*  3332   3336   3338   3340*  3342*  3389   3398*
                        3404*  3405*  3408*  3413*  3414*  3415   3424*  3479   3498*
R4     =%000004         1044#  1556*  1591   1654   2650*  2652   2657*  2660   2667   2668   2739   2756*  2826*
                        2827*  2828*  2829   2881*  2884   2890   2892   3390   3392*  3393*  3394*  3395   3396*
                        3410   3412*  3420*  3423*  3480   3497*
R5     =%000005         1045#  1482*  1489*  1491*  1556   1557*  1560*  1590*  1653*  2740   2755*  2882*  2886
                        2893   3391   3397*  3399*  3401*  3402*  3403*  3404   3422*  3481   3496*
R6     =%000006         1046#  1048   1495*  1496*  1497
R7     =%000007         1047#  1049
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
SAVREG= 104411          2874    3468#
SETUPR  006210          1843    1957    2588#
SETUPW  006062          1820    2097    2523#
SFBG    006742          2789#
SFFG    006736          2029    2787#
SOFTNM  001206          1227#   1675    1676    2184    2693*   2697*   2698*
SP    =%000006          1048#   1499*   1513*   1521*   1525    1585    1734*   1735*   1736*   1737*   1738    2184*   2189*
                        2194*   2199*   2204*   2211*   2216*   2264*   2267    2269    2270    2274    2277*   2301    2323*
                        2324*   2325*   2326*   2327*   2328*   2329*   2330*   2331    2332    2353*   2362*   2365*   2367*
                        2368    2374*   2383*   2385*   2386    2410*   2416    2417    2418    2428*   2429*   2430    2431*
                        2432    2433*   2446*   2447*   2448*   2449*   2450*   2451*   2453*   2454    2555*   2556*   2557*
                        2558*   2571    2572    2573    2574    2632*   2676*   2681*   2735*   2736*   2737*   2738*   2739*
                        2740*   2741*   2742*   2743*   2744*   2751*   2752*   2753*   2754*   2755    2756    2757    2758
                        2759    2760    2777*   2779*   2781*   2783*   2785*   2787*   2789*   2791*   2792*   2793*   2796
                        2797*   2798*   2811    2812    2820*   2821*   2831    2832    2847    2848    2875    2877*   2934
                        2935*   2937*   2938*   2951*   2952    2961    2962*   2974*   2975*   2976*   2977*   2978*   2980
                        2983*   2991*   2992    2994    2995*   2997    2998    2999    3011*   3012*   3013*   3014    3030
                        3060*   3063    3072    3081    3092*   3095*   3098*   3101*   3105    3113    3145*   3168*   3169
                        3170*   3172    3173    3174*   3176    3178    3180    3186    3188*   3190*   3198*   3202    3206
                        3207    3211    3231*   3232*   3233    3240*   3243*   3244*   3248*   3249*   3253    3256*   3260
                        3262    3264    3265*   3272    3274    3276*   3277    3279*   3280*   3281*   3282*   3283*   3298*
                        3299*   3302*   3303*   3304    3308*   3309*   3310    3313    3315    3317*   3326*   3331    3342
                        3343*   3344*   3345*   3381*   3382    3383    3384*   3389*   3390*   3391*   3397    3422    3423
                        3424    3425*   3426*   3441*   3442    3476*   3477*   3478*   3479*   3480*   3481*   3482*   3483
                        3491*   3495    3496    3497    3498    3499    3500    3501    3506*
STACK = 001100          1023#   1499
START   002624          1651    1672#   2242
STKLMT= 177774          1034#
SWR     001140          1203#   1497    1515*   1517    1523*   1537    1718    2147    2224    2228    2259    2297    2304
                        2309    3138    3140    3227    3264*   3482    3495*
SWREG   000176          1169#   1523    1537    2228    3227    3240
SW0   = 000001          1087#
SW00  = 000001          1077#   1087
SW01  = 000002          1076#   1086
SW02  = 000004          1075#   1085
SW03  = 000010          1074#   1084
SW04  = 000020          1073#   1083
SW05  = 000040          1072#   1082
SW06  = 000100          1071#   1081
SW07  = 000200          1070#   1080
SW08  = 000400          1069#   1079    2147
SW09  = 001000          1068#   1078    2224
SW1   = 000002          1086#
SW10  = 002000          1067#   2279
SW11  = 004000          1066#
SW12  = 010000          1065#
SW13  = 020000          1064#
SW14  = 040000          1063#
SW15  = 100000          1062#
SW2   = 000004          1085#
SW3   = 000010          1084#
SW4   = 000020          1083#
SW5   = 000040          1082#
SW6   = 000100          1081#
SW7   = 000200          1080#
SW8   = 000400          1079#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
SW9    = 001000          1078#
SYNCK    006244          1855    1984    2613#
TACSL    001240          1242#   1582    2794    2810*   2822    3067*   3092    3125    3529
TADBL    001242          1243#   2839*   2841    3069*   3095
TAPRIO   001250          1245#   3090*   3101
TAVEC    001244          1244#   1486    1596*   1597*   3076*   3078*   3098
TBITVE=  000014          1120#
TIMERR=  002000          1148#   2442
TKVEC  = 000060          1127#
TMPSWR=  160000          2279#   2320
TM.ERR   006034          2459    2499#
TPDRV    007554          1644    1649    1786    1936    2064    3010#
TPVEC  = 000064          1128#
TRAPVE=  000034          1126#   1505*   1506*
TREQ     007110          2825    2835#
TRTVEC=  000014          1121#
TR.REQ=  000200          1151#
TYPE   = 104400          1533    1642    1647    1650    1785    1787    1935    1937    2063    2065    2172    2181    2183
                         2188    2193    2198    2203    2208    2215    2220    2299    2307    2322    2340    2342    2344
                         2346    2357    2369    2379    2387    2392    2412    2959    3010    3015    3027    3049    3061
                         3070    3079    3091    3094    3097    3100    3103    3110    3112    3181    3238    3239    3242
                         3255    3266    3285    3334    3337    3341    3416    3457#   3504
TYPERR   005200          2306    2322#
TYPOC  = 104401          2363    2376    2411    3093    3096    3099    3102    3241    3458#
TYPON  = 104403          3460#
TYPOS  = 104402          3459#
UNIT   = 000400          1150#   3129
WBYTTL   001220          1231#   2199    2486*   2487*   2526*   2527*
WFG      006712          1809    2087    2777#
WL.ERR   005676          2456    2466#
WRITE    006716          1822    2101    2779#
WRITE0   004132          1741    2061#   2062
WRTLOC=  010000          1146#   2436    3136
WRTPAS   001300          1255#
WRTRYS   001226          1233#   1690*   1873*   1883*
XBSBG  = 000010          1137#
XBSFG  = 000006          1136#
XREAD  = 000004          1135#   2503
XRWND  = 000016          1140#
XSFBG  = 000014          1139#
XSFFG  = 000012          1138#
XWFG   = 000000          1133#   2469
XWRITE=  000002          1134#   2479
X.ERR    006056          2461    2508#
$AUTOB   001134          1200#   1541*   3235    3356
$BOADR   001122          1195#   2670*   3519
$BODAT   001126          1197#   1673    2666*   2669*   3519
$BELL    005174          2299    2319#
$CHARC   010504          3183*   3193*   3200    3209*   3214#
$CKSWR   010510          3227#   3464
$CMTAG   001100          1183#   1494    1495    1503    1509    1510
$CM1   = 000004          1215#   1216#   1217#   1218#   1219#
$CM2   = 000010          1215#   1216#   1217#   1218#   1219#
$CM3   = 000004          1213#   1215
$CM4   = 000004          1219#   1220#   1221#   1222#   1223#
$CNTLG   011235          3238    3351#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
$CNTLU  011230          3255    3350#
$CRLF   001203          1224#   1642    1787    1937    2065    2307    2320    2322    2342    2346    3182    3217    3266
                        3350
$DB2D   007162          2195    2200    2366    2384    2874#   2936
$DECVL  007342          2876    2922#
$DOAGN  004726          2168    2174    2234    2240#
$ENDAD  004716          1531    1595    1658    2236#   2314
$ENDCT  004436          1509    2170#
$ENDMG  004735          2172    2244#
$ENULL  004732          2243#
$EOP    004406          1803    1814    1827    1839    1850    1858    1869    1881    1953    1969    1980    1987    1998
                        2034    2081    2092    2106    2146    2148    2161#   3507
$EOPCT  004430          1509*   2167#   2171
$ERFLG  001103          1186#   2251    2275*   2279    2294*   2319
$ERMAX  001115          1192#   2279
$ERROR  005050          1503    2292#
$ERRPC  001116          1193#   2301*   2302*   2303    2319    3519    3523    3526    3529    3531
$ERRTB  001622          1384#
$ERTTL  001112          1190#   2300*   2319
$FILLC  001156          1211#   3186    3217
$FILLS  001155          1210#   3217
$GDADR  001120          1194#   2668*   3519
$GDDAT  001124          1196#   2665*   2667*   3519
$GET42  004446          2173#
$GTSWR  010560          3239#   3462
$HD   = 000000           957
$HIOCT  007552          2996*   3001#
$ICNT   001104          1187#
$ILLUP  011726          3474    3490    3509#
$INTAG  001135          1201#   3267    3356
$ITEMB  001114          1191#   1672    1673    2303*   2320    2326
$LF     001204          1225#   2220    2320    3217    3341    3350
$LPADR  001106          1188#   1510*   1784*   1819*   1934*   1962*   1973*   2062*   2098*   2274*   2277    2279
$LPERR  001110          1189#
$MAIL = ****** U        1527    1537    2274    2309    3170
$MNEW   011253          3242    3354#
$MSWR   011242          3239    3352#
$NULL   001154          1209#   3188    3217
$OCNT   011506          3388*   3417*   3430#
$OMODE  011510          3383*   3387*   3392    3395*   3406*   3432#
$OVER   005034          2260    2271    2276#
$PASS   001100          1184#   2164*   2165*   2243
$POWER  011734          3505    3512#
$PWRAD  011722          3507#
$PWRON  011562          1507    3474#   3502
$PWRMG  011716          3505#
$PWRUP  011634          3484    3490#
$QUES   001202          1223#   2320    3049    3110    3217    3285    3334    3350
$RDCHR  010772          3298#   3465
$RDDEC= ****** U        3468
$RDLIN  011112          3326#   3466
$RDOCT  007452          2974#   3467
$RDSZ = 000010          3319#
$REGAD  001160          1213#
$REGO   001162          1215#   2426*   2628*   2674*   3519    3523    3526
$REG1   001164          1216#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- USER SYMBOLS

```
$REG2   001166          1217#  2427*  2629*  2675*  3519   3523   3526
$REG3   001170          1218#
$RESRE  006654          2750#  3469
$RTNAD  004730          2242#
$R2A  = ****** U        3470
$SAVRE  006616          2734#  3468
$SAVR6  011732          3483*  3491   3492*  3493*  3511#
$SB2D   007356          2185   2190   2205   2212   2217   2934#
$SCOPE  004750          1501   2257#
$SETUP= 000137          1482#  1500   1501   1503   1505   1507   1509   1510   1531   1534   2163   2258   2293
                        2312   2314   3222   3356
$STUP = 177777          1482#
$SUPRS  007412          2186   2191   2196   2201   2206   2213   2218   2951#
$SVLAD  005020          2268   2273#
$SWR  = 160000           947#   957   1006   1007   1008   1009   1010   1011   1223   1510   1511   2156   2164
                        2175   2241   2243   2252   2253   2254   2259   2271   2273   2275   2279#  2285   2286
                        2287   2288   2289   2297   2304   2309   2313   2319   2320#  3508
$SWRMK= 000000          2254
$TKB    001146          1206#  3220   3231   3248   3302   3308
$TKS    001144          1205#  3220   3229   3245   3269*  3300   3306
$TMP0   001172          1219#  1718*  1719*  1732*  1734   1735   2430*  2469   2479   2503   2630*  3523   3526
$TMP1   001174          1220#
$TMP2   001176          1221#  2631*  3523
$TMP3   001200          1222#
$TN   = 000001           947#   957
$TNPWR  007272          2881   2882   2902#
$TPB    001152          1208#  3206*  3217
$TPFLG  001157          1212#  3164   3217
$TPS    001150          1207#  3204   3217
$TRAP   011512          1505   3441#
$TRP  = 000013          3449#  3458#  3459#  3460#  3461#  3462   3463#  3464   3465#  3466#  3467#  3468#  3469#
                        3470#
$TRPAD  011534          3446   3456#
$TSTNM  001102          1185#  2163*  2251   2273*  2276   2279   2296   2319
$TTYIN  011220          3327   3328   3345   3349#
$TYPBN= ****** U        3461
$TYPOS= ****** U        3461
$TYPE   010270          3164#  3449   3457
$TYPEC  010440          3185   3192   3199   3204#  3205   3271
$TYPEX  010506          3210   3212   3215#
$TYPOC  011310          3386#  3458
$TYPON  011324          3385   3388#  3460
$TYPOS  011264          3381#  3459
$XTSTR  004762          2262#
$$GET4= 000001          2175#  2233#
$OFILL  011507          3382*  3386*  3396   3431#
$40CAT= ****** U        2259   2306
.     = 015444          1163#  1167#  1182#  1226   1498   1510   1545#  2243   2246   2279   2319   2320   2922#
                        3217   3220   3349#  3350   3356   3486   3510   3711#  3715#  3716
```

```
BGOTO   1161#  2452
CALL    1161#  1797  1807  1821  1832  1844  1875  1947  1963  1974  2028  2075  2085  2099
COMMEN     1#  1130#
DECBLK  1161#  2482
DECFIL  1161#  2471
ENDCOM     1#  1130#
EOPCOD  1161#  2175
ERROR   1024#  1589  1652  1857  1864  1986  1993  2463  2466  2493  2496  2499  2502  2508
ESCAPE     1#  1130#
GETPRI     1#  1130#
GETSWR     1#  1130#  1534#
INCBLK  1161#  1887  2013  2112
INCFIL  1161#  1892  2019  2117
JGOTO   1161#  1733
MORETA  1161#  1227
MULT       1#  1130#
NEWTST     1#  1130#
OUTDBL  1161#  2194  2199
OUTSGL  1161#  2184  2189  2204  2216
POP        1#  1130#  2571  2755  2831  2846  2997  3495  3496
PUSH       1#  1130#  2554  2735  2819  2976  3476  3482
REMARK  1161#  1794  1804  1816  1829  1840  1852  1859  1870  1884  1944  1954  1959  1970  1981
        1988   2000  2010  2025  2072  2082  2094  2109  2176  2221
REPORT     1#  1130#
SCOPE   1025#  1851  1999  2108  2144  2162  2175
SETBLK  1161#  1788  1895  1938  2022  2066  2120
SETFIL  1161#  1791  1941  2069
SETPRI     1#  1130#
SETTRA  3449#  3458  3459  3460  3462  3464  3465  3466  3467  3468  3469
SETUP      1#  1130#  1492
SKIP       1#  1130#
SLASH      1#  1130#  1263  1284  1306  1360  1472  1659  1743  1747  1794  1796  1804  1806  1816
        1818   1829  1831  1840  1842  1852  1854  1859  1861  1870  1872  1884  1886  1899  1944
        1946   1954  1956  1959  1961  1970  1972  1981  1983  1988  1990  2000  2002  2010  2012
        2025   2027  2036  2072  2074  2082  2084  2094  2096  2109  2111  2124  2176  2178  2221
        2223   2352  2372  2406  2415  2422
SPACE   1130#
STARS      1#   957  1130#  1174  1178  1226  1386  1408  1456  1474  1480  1546  1555  1563  1579
        1598   1620  1661  1671  1699  1717  1745  1754  1782  1906  1932  2043  2059  2129  2143
        2153   2248  2281  2320  2510  2533  2577  2596  2635  2684  2701  2715  2719  2764  2814
        2860   2863  2925  2943  2966  3002  3019  3055  3115  3149  3219  3222  3290  3319  3358
        3435   3472  3488  3515  3552
SWRSU      1#  1130#  1511#
TRMTRP  3449#
TYPBIN     1#  1130#
TYPDEC     1#  1130#
TYPNAM     1#  1130#  1527
TYPNUM     1#  1130#
TYPOCS     1#  1130#
TYPOCT     1#  1130#  2362  2374  2410  3092  3095  3098  3101  3240
TYPTXT     1#  1130#
$$CMRE  1176#  1215  1216  1217  1218
$$CMTM  1176#  1219  1220  1221  1222
$$ESCA     1#  1130#
$$NEWT     1#  1130#
$$SET   3449#  3458  3459  3460  3462  3464  3465  3466  3467  3468  3469
```

DZTAEB.P11 CROSS REFERENCE TABLE -- MACRO NAMES

```
$$SKIP      1#   1130#
.EQUAT      1#    947#   1020
.HEADE      1#    947#
.KT11       1#
.SETUP      1#    947#   1482
.SWRHI      1#    947#   1002
.SWRLO   1012#
.$ACTI      1#
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#    947#   1161
.$CMTA      1#    947#   1176
.$DB2D      1#    947#   2861
.$DB2O      1#
.$DIV       1#
.$EOP       1#    947#   2151
.$ERRO      1#    947#   2279
.$ERRT      1#
.$MULT      1#
.$POWE      1#    947#   3470
.$RAND      1#
.$RDDE      1#
.$RDOC      1#    947#   2964
.$READ      1#    947#   3217
.$R2AZ      1#
.$SAVE      1#    947#   2717
.$SB2D      1#    947#   2923
.$SB2O      1#
.$SCOP      1#    947#   2246
.$SIZE      1#
.$SPAC    947#
.$SUPR      1#    947#   2941
.$TRAP      1#    947#   3433
.$TYPB      1#
.$TYPD      1#
.$TYPE      1#    947#   3147
.$TYPO      1#    947#   3356
.$4DCA      1#
.1170       1#
```

DZTAEB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

DZTAEB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
INCB    2210  2273  2294  3213
IOT     1025
JMP     1171  1172  1173  1595  1658  1739  1740  1741  1742  1803  1814  1815  1827  1828  1839
        1850  1858  1869  1881  1953  1969  1980  1987  1998  2034  2081  2092  2093  2106  2107
        2149  2241
JSR     1559  1562  1622  1626  1631  1644  1649  1674  1677  1786  1798  1602  1809  1813  1820
        1822  1826  1834  1838  1843  1845  1849  1855  1862  1865  1867  1876  1880  1936  1948
        1952  1957  1964  1968  1975  1979  1984  1991  1994  1996  2029  2033  2064  2076  2080
        2087  2091  2097  2101  2105  2180  2185  2186  2190  2191  2195  2196  2200  2201  2205
        2206  2212  2213  2217  2218  2236  2306  2366  2384  2530  2936  3185  3192  3199  3271
MOV     1483  1489  1491  1495  1499  1501  1502  1503  1504  1505  1506  1507  1508  1509  1510
        1513  1514  1515  1516  1521  1523  1524  1525  1556  1580  1586  1590  1593  1596  1621
        1634  1635  1653  1656  1672  1673  1675  1676  1678  1685  1686  1687  1688  1689  1690
        1691  1728  1732  1784  1788  1791  1797  1808  1819  1821  1833  1844  1875  1882  1883
        1895  1934  1938  1941  1947  1962  1963  1973  1974  2003  2022  2028  2062  2066  2069
        2075  2086  2098  2100  2120  2169  2173  2179  2184  2189  2194  2199  2204  2209  2211
        2216  2233  2264  2265  2267  2270  2274  2276  2277  2296  2301  2323  2324  2339  2343
        2353  2354  2355  2358  2362  2365  2368  2373  2374  2377  2383  2386  2390  2391  2407
        2410  2416  2417  2418  2428  2430  2432  2523  2528  2529  2555  2556  2557  2558  2559
        2560  2561  2562  2563  2564  2565  2566  2567  2568  2571  2572  2573  2574  2588  2593
        2613  2614  2619  2644  2647  2648  2649  2650  2656  2657  2668  2670  2673  2696  2735
        2736  2737  2738  2739  2740  2741  2742  2743  2744  2751  2752  2753  2754  2755  2756
        2757  2758  2759  2760  2777  2779  2781  2783  2785  2787  2789  2791  2796  2797  2798
        2799  2800  2805  2806  2810  2811  2812  2820  2821  2822  2823  2826  2831  2832  2847
        2848  2875  2876  2877  2878  2879  2880  2881  2882  2934  2935  2938  2951  2952  2958
        2961  2962  2974  2975  2976  2977  2978  2980  2995  2996  2997  2998  2999  3014  3030
        3033  3060  3063  3067  3069  3072  3076  3078  3081  3090  3092  3095  3098  3101  3105
        3113  3125  3168  3169  3173  3188  3240  3264  3269  3298  3299  3326  3327  3342  3343
        3344  3345  3381  3389  3390  3391  3397  3404  3422  3423  3424  3425  3426  3441  3442
        3446  3474  3475  3476  3477  3478  3479  3480  3481  3482  3483  3484  3490  3491  3495
        3496  3497  3498  3499  3500  3501  3502  3503  3506
MOVB    1541  1624  1641  1645  1679  1682  1718  1735  2303  2326  2331  2332  2426  2427  2431
        2475  2476  2481  2485  2628  2629  2630  2631  2667  2669  2674  2675  2829  2839  2841
        2895  2983  3011  3036  3040  3170  3198  3206  3231  3248  3302  3308  3331  3336  3382
        3383  3386  3387  3388  3392  3395  3396  3415  3444
NEG     2671  3393
NOP     2237  2238  2239
RESET   2235
ROL     2986  2988  2990  3399  3401  3402  3403  3405
RTI     1522  2278  2318  2745  2761  2833  2849  3000  3175  3270  3318  3346  3427  3508
RTS     1698  2419  2464  2467  2494  2497  2500  2506  2509  2531  2575  2594  2633  2682  2700
        2714  2813  2901  2939  2963  3017  3053  3114  3146  3215  3447
SBC     2487  2885
SUB     2302  2472  2482  2486  2697  2792  2884  2886
SWAB    2615  2620  2827  3085
TRAP    3449  3458  3459  3460  3462  3464  3465  3466  3467  3468  3469
TST     1535  1582  1587  1591  1639  1654  2266  2309  2408  2699  2713  2804  2807  2994  3051
        3132  3172  3180  3202  3262  3277  3410  3443
TSTB    1683  1799  1810  1823  1835  1846  1877  1949  1965  1976  2030  2077  2088  2102  2360
        2678  2824  2953  3031  3044  3045  3164  3204  3229  3245  3300  3306
.ASCII  1223  1224  3627
.ASCIZ  1225  1545  2244  2319  3350  3351  3352  3354  3512  3556  3559  3560  3561  3565  3569
        3573  3577  3579  3581  3583  3586  3590  3594  3597  3600  3604  3608  3612  3616  3634
        3641  3647  3650  3651  3655  3658  3661  3664  3667  3670  3673  3676  3677  3680  3683
        3686  3689  3690  3692  3694  3696  3698  3700  3702  3704  3706  3709  3710
.BLKB   2922  3349  3715
```

DZTAEB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

DZTAEB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

```
.IIF     947   952   957  1006  1007  1008  1009  1012  1013  1167  1226  1500  1503  1509  1510
        1511  1531  2155  2156  2163  2164  2243  2246  2252  2253  2254  2258  2275  2279  2285
        2286  2287  2288  2289  2293  2312  2314  2319  2320  2363  2375  2411  3093  3096  3099
        3102  3217  3220  3241  3342  3350  3356  3457  3458  3459  3460  3462  3464  3465  3466
        3467  3468  3469
.IRP    1227  1482  1739  2175  2455  2555  2571  2735  2755  2820  2831  2647  2976  2997  3476
        3482  3495  3496
.LIST      1   947   957   958   959   960  1130  1161  1167  1213  1215  1216  1217  1218  1219
        1220  1221  1222  1223  1386  1387  1388  1389  1408  1409  1410  1411  1456  1457  1458
        1459  1472  1473  1474  1482  1511  1531  1534  1545  1546  1547  1548  1563  1564  1565
        1598  1599  1600  1659  1660  1661  1662  1663  1699  1700  1701  1734  1743  1744  1745
        1746  1747  1748  1749  1750  1751  1754  1755  1756  1899  1900  1901  1902  1903  1906
        1907  1908  2036  2037  2038  2039  2040  2043  2044  2045  2124  2125  2126  2129  2130
        2131  2163  2175  2233  2254  2279  2314  2320  2422  2423  2424  2453  2510  2511  2512
        2577  2578  2579  3319  3449  3457  3458  3459  3460  3461  3462  3463  3464  3465  3466
        3467  3468  3469  3470  3515  3552
.MACRO     1  1012  1161  1176  3449
.MCALL   947  1130  1511  1534
.NLIST     1   947   957   958   959   960  1130  1161  1167  1213  1215  1216  1217  1218  1219
        1220  1221  1222  1223  1386  1387  1388  1389  1408  1409  1410  1411  1456  1457  1458
        1459  1472  1473  1474  1482  1511  1531  1534  1545  1546  1547  1548  1563  1564  1565
        1598  1599  1600  1659  1660  1661  1662  1663  1699  1700  1701  1734  1743  1744  1745
        1746  1747  1748  1749  1750  1751  1754  1755  1756  1899  1900  1901  1902  1903  1906
        1907  1908  2036  2037  2038  2039  2040  2043  2044  2045  2124  2125  2126  2129  2130
        2131  2163  2175  2233  2254  2279  2314  2320  2422  2423  2424  2453  2510  2511  2512
        2577  2578  2579  3319  3449  3457  3458  3459  3460  3461  3462  3463  3464  3465  3466
        3467  3468  3469  3470  3515  3552
.NTYPE  1734  2453
.PAGE   1002  1161  1176  1262  1305  1359  1370  1472  1598  1659  1699  1747  1899  2036  2124
        2151  2246  2279  2320  2372  2422  2469  2510  2715  2764  2814  2860  3002  3019  3055
        3115  3147  3356  3433  3470  3515  3552
.REM       1   960
.REPT    957  1167  1215  1219  1386  1408  1456  1472  1546  1563  1598  1659  1661  1699  1743
        1745  1747  1754  1899  1906  2036  2043  2124  2129  2422  2510  2577
.SBTTL   957  1002  1020  1161  1170  1176  1265  1286  1308  1362  1370  1472  1493  1527  1534
        1752  1904  2041  2127  2151  2246  2279  2320  2424  2512  2534  2579  2597  2636  2686
        2703  2717  2766  2816  2861  2923  2941  2964  3003  3021  3056  3117  3147  3217  3356
        3433  3449  3470  3515  3552  3713
.TITLE   947
.WORD   1167  1168  1169  1184  1187  1188  1189  1190  1193  1194  1195  1196  1197  1198  1199
        1202  1203  1204  1213  1215  1216  1217  1218  1219  1220  1221  1222  1227  1228  1229
        1230  1231  1232  1233  1234  1236  1238  1248  1249  1250  1251  1252  1253  1254  1255
        1256  1261  1267  1268  1269  1270  1271  1272  1273  1274  1275  1276  1277  1278  1279
        1280  1281  1282  1366  1367  1368  1790  1793  1889  1894  1897  1940  1943  2016  2021
        2024  2068  2071  2114  2119  2122  2167  2170  2242  2370  2381  2388  2393  2474  2484
        2854  2855  2856  2857  2897  2940  2960  3001  3018  3214  3432  3505  3507  3519  3523
        3526  3529  3531
```

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

*,DZTAEB.SEQ/SOL/CRF/PAGNUM/NL:TOC=SYSMAC.CO,DZTAEB.P11
RUN-TIME: 38 52 6 SECONDS

RUN-TIME RATIO: 188/97=1.9
CORE USED: 35K (69 PAGES)

Spooler runtime 13 Seconds, 57 KCS, 358 disk reads, 5 disk writes, 101 pages

[redacted] Date 08-Nov-76 16:28:30 Monitor IPC-0 [redacted]