# TA11

MOTION TEST

## MD-11-DZTAD-C

EP-DZTAD-C-DL-A



FICHE 1 OF 1

NOV 1976

digital

MADE IN U.S.A.

IDENTIFICATION

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

PRODUCT NAME: TAIL MOTION TEST

DATE REVISED: 21 JUNE 76

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: JIM LACEY

# C01



CONTENTS

# D01

SEQ 0003

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT CHECK
THE TU60 DRIVE FOR PROPER OPERATION.

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

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 |
| 214 | SETUP FOR MANUAL LOOPING |
| 220 | WRITE FILE GAP FROM BOT TO EOT |
| 224 | WRITE CONTINOUS BLOCKS OF DATA |
| 230 | READ CONTINOUS BLOCKS OF DATA |
| 234 | WRITE FILE GAP AND A BLOCK OF DATA |
| 240 | READ BLOCK OF DATA AND INTO A FILE GAP |
| 244 | SPACE FWD FILE GAP FROM BOT TO EOT |
| 250 | BACK SPACE FILE GAPS |

4.3 PROGRAM & OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A WRITE ENABLED CASSETTE IN BOTH DRIVES
3. REWIND BOTH DRIVES
4. LOAD ADDRESS 200.
5. SET SWITCHES (SEE SECTION 5.1)
6. PRESS START.
7. THE PROGRAM WILL LOOP & TTY BELL WILL RING ONCE EVERY PASS. IF SW(10)=0.

*** 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.
NOTE: IF LOAD MEDIUM IS CASSETTE WITH STANDARD VECTOR PROGRAM WILL RESPOND AS IF STARTED AT 210.

STARTING THE PROGRAM AT 204, 210, OR 214 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
DRIVE(S)? AB
DRIVE(S)? B,A
DRIVE(S)? B

4.3.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 210 OR 214 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?
```

4.3.3 SUBTEST LOOPING

THE SCOPE ROUTINE (REFER 5.2.1) PROVIDES A MEANS BY WHICH THE OPERATOR CAN SPECIFY THE FIRST ADDRESS OF A SCOPE LOOP.

THE OPERATOR TYPES A "CONTROL C" (↑C) AND WHEN THE NEXT SCOPE STATEMENT IS EXECUTED THE PROGRAM WILL ASK FOR:

1. TEST PC--- THE FIRST ADDRESS OF THE TEST (MUST BE A SCOPE)
2. LOOP PC--- THE ADDRESS TO LOOP BACK TO

4.3.3.1 SUBTEST LOOPING EXAMPLES

```
;OPERATOR TYPES "↑C"
TEST PC? 2242              ;OPERATOR TYPES "2242""CARRIAGE RETURN "(CR)"
LOOP PC?                   ;OPERATOR TYPES "CR" WHICH
                           ;IMPLIES "LOOP PC"="TEST PC"

;OPERATOR TYPES "↑C"
TEST PC? 3000              ;OPERATOR TYPES "3000""CR"
LOOP PC? 3020              ;OPERATOR TYPES "3020""CR"
                           ;PROGRAM USES THIS AS THE
                           ;FIRST ADDRESS OF THE SCOPE
                           ;LOOP

;OPERATOR TYPES "↑C"
TEST PC?                   ;OPERATOR TYPES "CR"
                           ;PROGRAM CONTINUES
                           ;FROM THIS POINT
```

# 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<15:08>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST. BELL WILL RING AT COMPLETION OF A PASS.
THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR
SW<14>=1...LOOP ON TEST
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<11>=1...INHIBIT ITERATIONS
SW<10>=1...RING BELL ON ERROR
SW<10>=0...RING BELL ON PASS COMPLETE
SW<09>=1...LOOP ON ERROR
SW<07>=1...LOCK ON CURRENT DRIVE
SW<06>=1...DELAY AT END OF EACH FUNCTION
SW<05>=1...RUN WITHOUT INTERRUPTS
SW<04>=1...IGNORE BLOCK CHECK ERRORS
SW<03>=1...INHIBIT DATA COMPARE

## 5.2 SUBROUTINE ABSTRACTS

### 5.2.1 SCOPE

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED BETWEEN EACH TEST IN THE INSTRUCTION SECTION. IT RECORDS THE STARTING ADDRESS OF EACH TEST IN LOCATION "$LPADR" AND "$LPERR" AS IT IS BEING ENTERED.

THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
NOTE: THIS ROUTINE CHECKS THE TTY INPUT BUFFER FOR A "CONTROL C" (REFER TO 4.3.3)

### 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.)

*** THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
*** IF THE PROCESSOR HALTS (BIT 15=1), OPERATOR CAN RESET S/W SWITCH REGISTER BY HITTING A "CONTROL G" (↑G) BEFORE HITTING CONTINUE.

### 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

#### 5.2.4.4 SETLOOP (T.SETLOOP)

SETUP TO LOOP AS PER THE TTY

5.2.5 THE FOLLOWING "TRAP" CALLS ARE WHAT ARE USED TO PERFORM THE TESTS.

THE ROUTINES THAT ARE CALLED IS WHAT MAKES UP THE HEART OF THIS PROGRAM.

5.2.5.1 WFG (T.WFG) WRITE A FILE GAP

5.2.5.2 WRITE (T.WRIT) WRITE A BLOCK OF DATA

5.2.5.3 READ (T.READ) READ A BLOCK OF DATA

5.2.5.4 BSFG (T.BSFG) BACK SPACE A FILE GAP

5.2.5.5 BSBG (T.BSBG) BACK SPACE A BLOCK GAP

5.2.5.6 SFFG (T.SFFG) SPACE FORWARD A FILE GAP

5.2.5.7 SFBG (T.SFBG) SPACE FORWARD A BLOCK GAP

5.2.5.8 REWIND (T.RWND) REWIND THE TAPE TO BOT

5.2.5.9 SELDRV (T.SELDRV) SELECT A DRIVE

5.2.5.10 BLKCMP (T.BLKCMP) COMPARE READ AND WRITE BUFFERS

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.6.1 DO.CMD

THIS ROUTINE WILL LOAD THE LOW BYTE OF THE "TACS" WITH THE FIRST BYTE FOLLOWING THE CALL.

WHEN THE FUNCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=1 THE TAII VECTOR IS SET TO "SERV". WHEN THE FUCTION IS TO BE PERFORMED WITH "INTERRUPT ENABLE"=0 THE VECTOR IS SET TO "BADINT"

NOTE SWR<5> PROVIDES OVERRIDE CAPABLITIES OF "INTERRUPT ENABLE"=1.

5.2.6.2 WAITFLAG

THIS ROUTINE IS CALLED AFTER A COMMAND HAS BEEN SENT TO THE TAII. IT WAITS A PREDETERMINED AMOUNT OF TIME A TAKES ONE OF THREE EXITS.

THE EXITS FOLLOW THE CALL AND ARE:

1. ERROR NO FLAGS OCCURRED
2. ERROR NO INTERRUPT OCCURRED
3. NORMAL RETURN

5.2.6.3 FLAGS

THIS ROUTINE IS CALLED TO DETERMINE WHAT FLAGS ARE UP. THIS ROUTINE WILL TAKE ONE OF FOUR RETURNS DEPENDING ON THE FLAGS.

THE RETURNS FOLLOW THE CALL AND ARE:

1. "TRANFER REQUEST"=0 AND "READY"=0
2. "TRANFER REQUEST"=1
3. "READY"=1 AND "ERROR"=0
4. "READY"=1 AND "ERROR"=1

5.2.6.3 DO.CRC

THIS ROUTINE IS USED TO CALCULATE "CRC". IT WORKS ON ONE BYTE AT A TIME WHICH MUST BE IN R0 WHEN CALLED.

5.2.6.4 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.5 GETDRV

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

5.2.6.6 ASKPDR

THIS ROUTINE IS USED TO INPUT THE ADDRESSES FOR THE "TACS",
"TADB" AND THE VECTOR AND THE PRIORITY LEVEL TO USE.

5.2.6.7 TYPERR

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

5.2.6.8 EXAM

THIS ROUTINE IS USED TO DETERMINE WHICH DRIVE(S) ARE AVAILABLE
FOR TESTING.

5.2.7 THE FOLLOW ROUTINES ARE USED TO MAKE ADJUSTMENTS TO
THE TU60. BEFORE USING ANY OF THEM LOAD AND START 214.

5.2.7.1 WFGSUB

WRITE FILE GAPS FROM "BOT" TO "EOT"
START AT 220
THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
THE "WRITE DELAY MONO".

5.2.7.2 WRTSUB

WRITE CONTINUOUS BLOCKS OF DATA
START AT 224
THE PROGRAM WILL HALT THREE(3) TIMES
AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
HALT 2 ---SWR<7:0> = PATTERN DESIRED
HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
** IF USING SOFTWARE SWITCH REGISTER, AFTER
EACH HALT OPERATOR WILL BE PROMPTED
FOR THE VALUE WITH "SWR=XXXXXX NEW="

5.2.7.3 RDSUB

READ CONTINUOUS BLOCKS OF DATA
START AT 230
THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
AND THE "THRESHOLD POT"

5.2.7.4 WGPBLK

WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
START AT 234
THE PROGRAM WILL HALT THREE (3) TIMES
AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
HALT 2 --- SWR<7:0> = PATTERN DESIRED
HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
AND THE "GAP TIME MONO".

** IF USING SOFTWARE SWITCH REGISTER, AFTER
EACH HALT OPERATOR WILL BE PROMPTED
FOR THE VALUE WITH "SWR=XXXXX, NEW="

5.2.7.5 RGBLK

READ A BLOCK OF DATA AND A FILE GAP
START AT 240
THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
IT CAN BE USED TO ADJUST THE "SIGNAL MON". THE THRESHOLD POT"
AND THE "TAPE BLANK MONO".

5.2.7.6 SFFGSB

SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
START AT 244
THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED). OR AFTER READ OR
WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
(SIGNAL MONO CAN BE CHECKED).

5.2.7.7 BSFGSB

BACK SPACE FILE GAP
START AT 250
THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".

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. THE FUNCTION BEING PERFORM
2. AN ERROR MESSAGE
3. A DATA HEADER
4. A DATA STRING

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

7. RESTRICTIONS

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

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS TAKES APPROXIMATELY 4 MINUTES ALL SUBSEQUENT PASSES TAKE APPOXIMATELY 8 MINUTES

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

A PROGRAM PASS THRU COUNT IS KEPT IN "$PASS".

8.4 ITERATIONS

THE FIRST PASS OF THE PROGRAM WILL AUTOMATICALLY INHIBIT ITERATIONS. ALL SUBSEQUENT PASSES WILL PERFORM FULL, (ONE PER DRIVE), ITERATIONS.

8.5 SPECIAL REGISTERS

R4 AND R5 ARE RESERVED FOR "TACS" AND "TADB" THROUGH OUT THE PROGRAM.

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF INDEPENDENT TESTS THAT
CHECK THE TAII FOR PROPER OPERATION.

EACH TESTS IS A SERIES OF "TRAP" CALLS TO ROUTINES THAT PERFORM
THE DESIRED FUNCTION.

THE PROGRAM STARTS WITH A SIMPLE SEQUENCE OF FUNCTION'S
AND BUILTS IN COMPLEXITY INTRODUCING ONE NEW FUNCTION
AT A TIME, UNTIL ALL LEGAL COMBINATIONS HAVE BEEN PERFORMED.
THEN, MULTI SPACING IS PERFORMED TO TRY AND GENERATE
NOISE THAT MIGHT CAUSE PROBLEMS DUE TO SPEED CHANGES AND
FREQUENT START STOPPING.

```
.TITLE  TA11 MOTION TEST        MAINDEC-11-DZTAD-C
:*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-C2), SEPT 14, 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

| BIT | NAME | INIT STATE | READ AND/OR WRITE? |
|---|---|---|---|
| 15 | ERROR | ? | READ ONLY |
| 14 | BLOCK CHECK ERROR | 0 | READ ONLY |
| 13 | CLEAR LEADER | ? | READ ONLY |
| 12 | WRITE LOCK | ? | READ ONLY |
| 11 | FILE GAP | 0 | READ ONLY |
| 10 | TIMING ERROR | 0 | READ ONLY |
| 09 | OFF LINE | ? | READ ONLY |
| 08 | UNIT SELECT | 0 | READ/WRITE |
| 07 | TRANSFER REQUEST | 0 | READ ONLY |
| 06 | INTERRUPT ENABLE | 0 | READ/WRITE |
| 05 | READY | 1 | READ ONLY |
| 04 | IL BS | 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

| | | | | |
|---|---|---|---|---|
| 57 | DC | GO BIT | 0 | WRITE ONLY! |

```
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH              USE
                        ;*      ------      --------------------
                        ;*        15        HALT ON ERROR
                        ;*        14        LOOP ON TEST
                        ;*        13        INHIBIT ERROR TYPEOUTS
                        ;*        11        INHIBIT ITERATIONS
                        ;*        10        BELL ON ERROR
                        ;*         9        LOOP ON ERROR
                        ;*         8        LOCK ON CURRENT DRIVE
                        ;*         6        DELAY AT END OF EACH FUNCTION
                        ;*         5        RUN WITHOUT INTERRUPTS
                        ;*         4        IGNORE BLOCK CHECK ERRORS
                        ;*         3        INHIBIT DATA COMPARE


                        .SBTTL  BASIC DEFINITIONS

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

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

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

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

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


                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
                        SW15=   100000
                        SW14=   40000
                        SW13=   20000
                        SW12=   10000
                        SW11=   4000
                        SW10=   2000
                        SW09=   1000
                        SW08=   400
                        SW07=   200
                        SW06=   100
                        SW05=   40
                        SW04=   20
                        SW03=   10
                        SW02=   4
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0


                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                        BIT15=  100000
                        BIT14=  40000
                        BIT13=  20000
                        BIT12=  10000
                        BIT11=  4000
                        BIT10=  2000
                        BIT09=  1000
                        BIT08=  400
                        BIT07=  200
                        BIT06=  100
                        BIT05=  40
                        BIT04=  20
                        BIT03=  10
                        BIT02=  4
                        BIT01=  2
                        BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
```

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

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

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

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

                        ;////////////////////////////////////////////////////////////////
                        ;////////////////////////////////////////////////////////////////
                        ;SPECIAL REGISTERS

000004                  TACS=   %4                      ;R4 IS USED AS A POINTER TO THE TACS REGISTER
000005                  TADB=   %5                      ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
                        ;////////////////////////////////////////////////////////////////
```

```
                        .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  001450          JMP     @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001502          JMP     @#BEGIN2                ;SELECT DRIVE(S) BEFORE STARTING TEST
000210  000137  001510          JMP     @#BEGIN3                ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
000214  000137  002052          JMP     @#BEGINX                ;SETUP FOR MANUAL LOOPING
000220  000137  012374          JMP     @#WFGSUB                ;WRITE FILE GAP FROM BOT TO EOT
000224  000137  012450          JMP     @#WRTSUB                ;WRITE CONTINUOUS BLOCKS OF DATA
000230  000137  012532          JMP     @#RDSUB                 ;READ CONTINUOUS BLOCKS OF DATA
000234  000137  012620          JMP     @#WGPBLK                ;WRITE FILE GAP AND A BLOCK OF DATA
000240  000137  012712          JMP     @#RGPBLK                ;READ BLOCK OF DATA AND INTO A FILE GAP
000244  000137  013010          JMP     @#SFFGSB                ;SPACE FWD FILE GAP FROM BOT TO EOT
000250  000137  013064          JMP     @#BSFGSB                ;BACK SPACE FILE GAPS
                        ;;**************************************************************
```

```
                        .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      $TMP0:  .WORD   0               ;;USER DEFINED
001170  000000      $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001172  000000      $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001174  177607  000377  $BELL:  .ASCIZ  <207><377><377>  ;;CODE FOR BELL
001200  077         $QUES:  .ASCII  /?/             ;;QUESTION MARK
001201  015         $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001202  000012      $LF:    .ASCIZ  <12>            ;;LINE FEED
                        ;;**********************************************************************
001204  000000      $SAVPC: .WORD   0                       ;STORAGE FOR THE PC
001206  000000      BYTNUM: .WORD   0                       ;THE NUMBER OF BYTES "READ" OR "WRITTEN
001210  000000      RCRC0:  .WORD   0                       ;CRC CALCULATED FOR "WRITE"
001212  000000      RCRC1:  .WORD   0                       ;CRC CALCULATED FOR "READ"
001214  000000      RCRC2:  .WORD   0                       ;CRC FROM THE TA11 FOR "READ"

001216  000001      $MXCNT: 1                               ;MAX. NUMBER OF ITERATIONS
```

```
  303  001220  177500           TACSL:  177500                ;LOW BYTE ADDRESS OF TACS
  304  001222  177501           TACSH:  177501                ;HIGH BYTE ADDRESS OF TACS
  305  001224  177502           TADBL:  177502                ;LOW BYTE ADDRESS OF TADB
  306  001226  177503           TADBH:  177503                ;HIGH BYTE ADDRESS OF TADB
  307  001230  000260  000262   TAVEC:  260,262               ;TA11 VECTOR ADDRESS
  308  001234  000300           TAPRIO: 300                   ;TA11 BR LEVEL 6
  309  001236  000000           DRIVE:  0                     ;NEXT DRIVE TO TEST
  310  001240  000000  000000   DRVKEY: 0,0                   ;DRIVE SELECT KEY:
  311  001244  001240           DRVPNT: DRVKEY
  312  001246  000000           ASKKEY: 0
  313  001250  177777  000000   CURDRV: -1,0                  ;CURRENT DRIVE BEING TESTED
  314  001254  000000           WAITKEY:        0             ;WAIT ON INTERRUPT KEY
  315  001256  001260           RWDFLG: RWDA                  ;REWIND FLAG POINTER
  316  001260  000000           RWDA:   0                     ;DRIVE "A" REWIND FLAG
  317  001262  000000           RWDB:   0                     ;DRIVE "B" REWIND FLAG
  318  001264  000000  000000   STALL:  0,0                   ;STALL TIME
```

```
                                .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 (SERRPC).
                                ;*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


001270                          $ERRTB:
                                ;NOTE:  ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS

                                ;ITEM   1
001270  015550                          EM1     ;IMPROPER FLAG SETTING
001272  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
001274  015054                          DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001276  000000                          0

                                ;ITEM   2
001300  015576                          EM2     ;IMPROPER FLAG OCCURRED
001302  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
001304  015054                          DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001306  000000                          0

                                ;ITEM   3
001310  015625                          EM3     ;MISSED A FLAG
001312  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
001314  015054                          DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001316  000000                          0

                                ;ITEM   4
001320  015643                          EM4     ;INTERRUPT FAILED
001322  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
001324  015054                          DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001326  000000                          0

                                ;ITEM   5
001330  015664                          EM5     ;PREMATURE READY OCCURRED
001332  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
001334  015054                          DT1     ;SAVPC  $ERRPC  $REG0   $REG1
001336  000000                          0

                                ;ITEM   6
001340  015715                          EM6     ;DIDN'T STOP IN A FILE GAP
001342  016112                          DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
```

```
                                        DT1     ;SAVPC  $ERRPC  $REGO   $REG1
                                        0

                        ;ITEM   7
                                        EM7     ;DIDN'T STOP ON CLEAR LEADER
                                        DH1     ;TEST   ERROR
                                                ;PC     PC      TACS    TADB
                                        DT1     ;SAVPC  $ERRPC  $REGO   $REG1
                                        0

                        ;ITEM   10
                                        EM10    ;BAD DATA READ
                                        DH2     ;TEST   ERROR           EXPT'D  RCV'D   BYTE
                                                ;PC     PC      TACS    DATA    DATA    NUMBER
                                        DT2     ;SAVPC  $ERRPC  $REGO   $GDDAT  $BDDAT  BYTNUM
                                        0

                        ;ITEM   11
                                        EM11    ;ILLEGAL BUFFER
                                        DH5     ;TEST   ERROR
                                                ;PC     PC      TACS
                                        DT5     ;SAVPC  $ERRPC  $REGO
                                        0

                        ;ITEM   12
                                        EM12    ;CRC ERROR
                                        DH3     ;TEST   ERROR           WRITE   READ    TU60
                                                ;PC     PC      TACS    CRC     CRC     CRC
                                        DT3     ;SAVPC  $ERRPC  $REGO   RCRC0   RCRC1   RCRC2
                                        0

                        ;ITEM   13
                                        EM13    ;SHORT RECORD
                                        DH4     ;TEST   ERROR                           BYTES
                                                ;PC     PC      TACS    TADB    LEFT
                                        DT4     ;SAVPC  $ERRPC  $REGO   $REG1   $TMPO
                                        0

                        ;ITEM   14
                                        EM14    ;BAD INTERRUPT
                                        DH5     ;TEST   ERROR
                                                ;PC     PC      TACS
                                        DT5     ;SAVPC  $ERRPC  $REGO
                                        0

                        ITEMS2:         ;ITEMS 201-202

                                        EM201   ;TA11 FAILED TO RESPOND
                                        DH201   ;PC     TACS
                                        DT201   ;$ERRPC TACS
                                        0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS

                                        EM202   ;NO DRIVES AVAILABLE
                                        DH202   ;PC
                                        DT202   ;$ERRPC
                                        0       ;
```

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

                                        :BEGIN1 IS FOR NORMAL START
                                        :BEGIN2 IS FOR DRIVE SELECTION
                                        :BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
                                        :BEGIN4 IS FOR MANUAL OPERATION

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

001450  005005                BEGIN1: CLR     R5                      ;NORMAL START
001452  012737  041101  001240        MOV     #"AB,@#DRVKEY
001460  122737  000005  000041        CMPB    #5,@#41                 ;CASSETTE EDP?
001466  001015                        BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001470  022737  000260  001230        CMP     #260,@#TAVEC            ;STANDARD VECTOR?
001476  001011                        BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001500  000403                        BR      BEGIN3                  ;GET DRIVES AND ADDRESSES
001502  012705  000001        BEGIN2: MOV     #1,R5                   ;ASK FOR DRIVES FLAG
001506  000405                        BR      BGNCMN                  ;BEGIN COMMON CODE
001510  012705  000002        BEGIN3: MOV     #2,R5                   ;ASK FOR DRIVES AND ADDRESSES
001514  000402                        BR      BGNCMN
001516  012705  000003        BEGIN4: MOV     #3,R5
001522                        BGNCMN:
                              .SBTTL  INITIALIZE THE COMMON TAGS
                              ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001522  012706  001100                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001526  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001530  022706  001140                CMP     #SWR,R6 ;;DONE?
001534  001374                        BNE     .-6             ;;LOOP BACK IF NO
001536  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                              ;;INITIALIZE A FEW VECTORS
001542  012737  006600  000020        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001550  012737  000340  000022        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001556  012737  007056  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001564  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001572  012737  014564  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001600  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
001606  012737  014672  000024        MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001614  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001622  016767  004574  004564        MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
001630  005067  177334                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001634  005067  177332                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001640  112767  000001  177247        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001646  012767  001646  177232        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001654  012767  001654  177226        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                              ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                              ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001662  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001666  012737  001722  000004        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001674  012767  177570  177236        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001702  012767  177570  177232        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001710  022777  177777  177222        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001716  001012                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                              ;;AND  THE HARDWARE SWR IS NOT = -1
001720  000403                        BR      65$             ;;BRANCH IF NO TIMEOUT
```

```
   487  001722  012716  001730          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
   488  001726  000002                          RTI
   489  001730  012767  000176  177202  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   490  001736  012767  000174  177175          MOV     #DISPREG,DISPLAY
   491  001744  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   492
   493                                  .SBTTL  TYPE PROGRAM NAME
   494                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
   495  001750  005227  177777                  INC     #-1             ;;FIRST TIME?
   496  001754  001036                          BNE     HERE            ;;BRANCH IF NO
   497  001756  022737  006442  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
   498  001764  001432                          BEQ     HERE            ;;BRANCH IF YES
   499  001766  104401  002024                  TYPE    ,MSGID          ;;TYPE ASCIZ STRING
   500                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
   501  001772  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
   502  001776  001006                          BNE     67$             ;;BRANCH IF YES
   503  002000  026727  177134  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
   504  002006  001005                          BNE     68$             ;;BRANCH IF NO
   505  002010  104405                          GTSWR                   ;;GET SOFT-SWR SETTINGS
   506  002012  000403                          BR      68$
   507  002014  112767  000001  177112  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
   508  002022                          68$:
   509  002022  000413                          BR      HERE            ;;GET OVER THE ASCIZ
   510                                  ;;MSGID:        .ASCIZ  <CRLF>/MAINDEC-11-DZTAD-C/<CRLF>
   511  002052                          HERE:
```

```
                        ;;*****************************************************************
                        ;;*****************************************************************
                        ;
                        ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
                        ;
                        ;       R5=3    MANUAL OPERATIONS
                        ;       R5=2    ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
                        ;       R5=1    ASK FOR DRIVE(S)
                        ;       R5=0    DON'T ASK FOR ANYTHING
                        ;
                        ;;*****************************************************************
002052  010504          BEGINX: MOV     R5,R4                   ;COPY R5
002054  005305                  DEC     R5                      ;ASK FOR DRIVES?
002056  002406                  BLT     CHKADR                  ;BR IF NO
002060  004737  007432          JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
002064  005305                  DEC     R5                      ;ASK FOR ADDRESSES?
002066  002402                  BLT     CHKADR                  ;BR IF NO
002070  004737  007542          JSR     PC,@#ASKADR             ;GO GET TA11 ADDRESSES
                        ;;*****************************************************************
                        ;;*****************************************************************

                        ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
                        ;
                        ;I.     TIMEOUT OCCURRED
                        ;       A. TYPE ERROR MESSAGE
                        ;       B. EXAMINE R4
                        ;               1. R4>0 GOTO BEGINX
                        ;               2. R4=0 EXAMINE (42)
                        ;                       A. (42)=0 GOTO BEGINX
                        ;                       B. (42)>0 GOTO $ENDAD
                        ;
                        ;II.    TIMEOUT DIDN'T OCCUR
                        ;       A. CONTINUE
                        ;
                        ;;*****************************************************************
002074  012737  002112  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE OF TIMEOUTS
002102  005000                  CLR     R0                      ;USE AS A SWITCH
002104  005777  177110          TST     @TACSL                  ;SEE IF TA11 RESPONDS
002110  000402                  BR      2$                      ;BR IF NO TIMEOUT
002112  005200          1$:     INC     R0                      ;COME HERE ON TIMEOUT
002114  022626                  CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
002116  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
002124  005700                  TST     R0                      ;DID A TIMEOUT OCCUR?
002126  001412                  BEQ     3$                      ;BR IF NO
002130  104201                  ERROR   201                     ;TA11 FAILED TO RESPOND
002132  012705  000002          MOV     #2,R5                   ;DRIVES & ADDRESSES
002136  005704                  TST     R4                      ;OPERATOR INPUTS?
002140  001344                  BNE     BEGINX                  ;BR IF YES
002142  013700  000042          MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002146  001741                  BEQ     BEGINX                  ;BR IF NO MONITOR
002150  000137  006442          JMP     @#$ENDAD                ;GO TO END
002154                  3$:
```

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

                        ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
                        ;
                        ;I.     DESIRED DRIVES CAN NOT BE TESTED
                        ;       A. TYPE ERROR MESSAGE
                        ;       B. EXAMINE R4
                        ;               1. R4>0 GOTO BEGINX
                        ;               2. R4=0 EXAMINE (42)
                        ;                       A. (42)=0 GOTO BEGINX
                        ;                       B. (42)>0 GOTO $ENDAD
                        ;
                        ;II.    BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
                        ;       A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
                        ;       B. CONTINUE IN PROGRAM
                        ;
                        ;III.   DESIRED DRIVE(S) CAN BE TESTED
                        ;       A. CONTINUE IN PROGRAM
                        ;
                        ;**************************************************************
002154  012700  001240  CHKDRV: MOV     #DRVKEY,R0          ;PICKUP ADDRESS OF ASCII DRIVE KEY
002160  004737  006474          JSR     PC,@#EXAM           ;GO EXAMINE FIRST DRIVE
002164  000410                  BR      1$                  ;OK TO TEST---GO CHECK NEXT
002166  116010  000001          MOVB    1(R0),(R0)          ;REPLACE 1ST WITH 2ND
002172  001412                  BEQ     2$                  ;BR IF NO 2ND DRIVE SELECTED
002174  004737  006474          JSR     PC,@#EXAM           ;GO EXAMINE DRIVE
002200  000407                  BR      2$                  ;OK TO TEST
002202  005010                  CLR     (R0)                ;CLEAR DRIVE CODES
002204  000405                  BR      2$
002206  005200          1$:     INC     R0                  ;POINT TO 2ND
002210  004737  006474          JSR     PC,@#EXAM           ;GO EXAMINE DRIVE
002214  000401                  BR      2$                  ;OK TO TEST
002216  105010                  CLRB    (R0)                ;CLEAR 2ND
002220  012700  001240  2$:     MOV     #DRVKEY,R0          ;RESET ADDRESS POINTERS
002224  010037  001244          MOV     R0,@#DRVPNT
002230  121060  000001          CMPB    (R0),1(R0)          ;1ST = 2ND?
002234  001002                  BNE     3$                  ;BR IF NO
002236  105060  000001          CLRB    1(R0)               ;YES---CLEAR 2ND
002242  005710          3$:     TST     (R0)                ;ANY DRIVES?
002244  001401                  BEQ     5$                  ;BR IF NO
002246  000412                  BR      MANUAL
002250  104202          5$:     ERROR   202                 ;NO DRIVES AVAILABLE
002252  012705  000002          MOV     #2,R5               ;DRIVES & ADDRESS
002256  005704                  TST     R4                  ;OPERATOR INPUTS?
002260  001274                  BNE     BEGINX              ;BR IF YES
002262  013700  000042          MOV     @#42,R0             ;GET MONITOR RETURN ADDRESS
002266  001671                  BEQ     BEGINX              ;NO MONITOR
002270  000137  006442          JMP     @#$ENDAD            ;GO TO END
002274  020427  000003  MANUAL: CMP R4,#3
002300  001002                  BNE     OK
002302  016704  175471          MOV     -1,R4
002306  010437  001246  OK:     MOV     R4,@#ASKKEY
002312  000405                  BR      START
002314  104401  002024  PWRST:  TYPE    ,MSGID              ;POWER FAIL RESTART
002320  012737  001240  001244  MOV     #DRVKEY,@#DRVPNT
```

```
                                START:  MOV     #BADINT,@TAVEC          ;SETUP TA11 TRAP VECTOR
                                        MOV     @#TAPRIO,@TAVEC+2
                                        MOV     #340,@#PS               ;LOCKOUT ALL I/O INT
                                        MOV     @#TACSL,TACS            ;SETUP TACS
                                        MOV     @#TADBL,TADB            ;SETUP TADB
                                        CLR     @#$TSTNM                ;ZERO THE TEST NUMBER
                                        CLR     @#LOOPKEY               ;CLEAR THE LOOP UNDER TTY CONTROL KEY
                                        MOV     #1,@#$MXCNT             ;SET FOR ONE ITERATION PER TEST
                                        MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
                                        MOV     R1,DRIVE                ;SET DRIVE TO POINTER
                                ;///////////////////////////////////////////////////////////////////////
                                ;TYPE THE DRIVE(S) THAT WILL BE TESTED
                                        MOVB    (R1),@#CURDRV           ;GET CURRENT DRIVE FOR TYPE-OUT
                                        TYPE    ,MTSTDRV                ;TYPE "TESTING DRIVE"
                                        TYPE    ,CURDRV                 ;TYPE THE CURRENT DRIVE
                                        BIT     #SW07,@#SWR             ;LOCK ON ONE DRIVE?
                                        BNE     2$                      ;BR IF YES
                                        TSTB    @#DRVKEY+1              ;TWO DRIVES AVAILABLE?
                                        BEQ     2$                      ;BR IF NO
                                        INC     @#$MXCNT                ;SET FOR TWO ITERATIONS PER TEST (ONE PER DRIVE)
                                        INC     R1                      ;INCREMENT THE POINTER
                                        MOVB    (R1),@#CURDRV           ;SET SECOND DRIVE FOR TYPE-OUT
                                        BNE     1$                      ;GO TYPE IT
                                        MOV     #DRVKEY,R1              ;RESET POINTER
                                        MOVB    (R1),@#CURDRV           ;NOW GET THE SECOND DRIVE
                                1$:     TYPE    ,MANDRV                 ;TYPE "AND DRIVE"
                                        TYPE    ,CURDRV                 ;TYPE DRIVE
                                2$:     MOV     R1,@#DRVPNT             ;SAVE POINTER FOR NEXT TRIP THRU
                                        MOV     #5$,@#$LPADR            ;SETUP THE SCOPE LOOP ADDRESS
                                        TST     @#$PASS                 ;ONCE ONLY
                                        BNE     TST1
                                ;///////////////////////////////////////////////////////////////////////
                                ;ON THE FIRST PASS OF THE PROGRAM DETERMINE THE AMOUNT OF TIME TO
                                ;WAIT ON A FLAG ("READY"/"TRANSFER REQUEST") AND THE STALL TIME TO USE
                                ;WHEN STALLING AFTER A FUNCTION IS COMPLETED
                                        SELDRV                          ;SELECT DRIVE
                                        MOV     #FATAL,@#$ESCAPE        ;SETUP ESCAPE ON ERROR
                                        CLR     @RWDFLG                 ;SET FOR NO REWIND ERRORS
                                        MOV     #^CBIT15,@#MAXCNT       ;SETUP IMPOSSIBLE COUNT
                                        MOVB    #XRWND,@TACS            ;INSURE NO ERROR DUE TO "CLEAR LEADER"
                                        REWIND                          ;GO TO BOT
                                        WFG                             ;NOW TIME A WFG
                                        SUB     @#HICNT,@#MAXCNT        ;GET THE TIME IT TOOK TO WFG
                                        INC     @#MAXCNT                ;MAKE IT BIGGER
                                        ROL     @#MAXCNT
                                        ROL     @#MAXCNT
                                        CLR     R0
                                        CLR     R1
                                        MOV     #4$,@TAVEC              ;SETUP TA11 VECTOR
                                        MOVB    #XWFG!INT.EN!GO,@TACS   ;START A "WFG"
                                        CLR     @#PS                    ;ALLOW INTERRUPTS
                                3$:     SUB     #1,R0                   ;START TIMING HOW LONG
                                        SBC     R1                      ;    IT TAKES TO WRITE A FILE GAP
                                        BNE     3$
                                        MOV     #340,@#PS               ;LOCK OUT INTERRUPTS
                                        BR      FATAL                   ;IT TOOK TO LONG
```

```
                                4$:     ASR     R1                      ;DIVIDE THE TIME BY 2
                                        ROR     R0
                                        MOV     R0,@#STALL              ;AND SAVE IT AS THE STALL TIME
                                        MOV     R1,@#STALL+2
                                        MOV     #BADINT,@TAVEC          ;SET TRAP CATCHER
                                        TSTB    @#DRVKEY+1              ;TWO DRIVES?
                                        BEQ     5$                      ;BR IF NO
                                        SELDRV                          ;SELECT THE OTHER DRIVE
                                        CLR     @RWDFLG                 ;SET REWIND FLAG
                                        MOVB    #XRWND,@TACS            ;AVOID "CLEAR LEADER" ERROR
                                        REWIND                          ;GO TO BOT
                                        WFG                             ;GO TO OXIDE
                                5$:     TST     @#ASKKEY                ;GO START TESTING IF NO MANUAL
                                        BGE     TST1                    ;  OPERATIONS REQUESTED
                                        CLR     R0
                                        ROL     @#MAXCNT                ;INCREASE THE WAIT TIME
                                        HALT                            ;GIVE CONTROL TO THE OPERATOR

                                FATAL:
                                        TYPE    ,65$                    ;;TYPE ASCIZ STRING
                                        BR      64$                     ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>*FATAL ERROR ON DRIVE *
                                64$:
                                        TYPE    ,CURDRV                 ;CURRENT DRIVE
                                        MOV     @#42,R0                 ;CHECK FOR A MONITOR
                                        BEQ     1$                      ;BR IF NONE
                                        JMP     @#$ENDAD                ;LEAVE
                                1$:     HALT                            ;ERROR OCCURRED BEFORE TESTING STARTED
                                        BR      1$                      ;HANGUP
```

```
                                ;;*****************************************************************
                                ;*TEST 1         REWIND,WRITE,REWIND,READ
                                ;;*****************************************************************
003000 012767 003032 176164     TST1:   SCOPE
                                        MOV     #TST2,$ESCAPE   ;;ESCAPE TO TEST 2 ON ERROR
003006 104422                           SELDRV                  ;SELECT DRIVE FOR TESTING
003010 104421                           REWIND                  ;GO TO "CLEAR LEADER"
003012 104413 017022                    WRITE   ,WLNK1          ;WRITE ON TAPE
003016 104421                           REWIND                  ;GO TO BEGINNING OF TAPE
003020 104414 016766                    READ    ,RLNK1          ;READ
003024 104423                           BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003026 016766                           RLNK1                   ;LINK TO THE READ BUFFER
003030 017022                           WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 2         TEST BSBG AFTER WRITE AND READ AFTER BSBG
                                ;;*****************************************************************
003032 000004                   TST2:   SCOPE
003034 012767 003066 176130             MOV     #TST3,$ESCAPE   ;;ESCAPE TO TEST 3 ON ERROR
003042 104422                           SELDRV                  ;SELECT DRIVE FOR TESTING
003044 104421                           REWIND                  ;GO TO "BOT"
003046 104413 017022                    WRITE   ,WLNK1          ;WRITE ONE BLOCK
003052 104416                           BSBG                    ;BACK OVER THE BLOCK
003054 104414 016766                    READ    ,RLNK1          ;NOW READ
003060 104423                           BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003062 016766                           RLNK1                   ;LINK TO THE READ BUFFER
003064 017022                           WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 3         TEST BSFG AFTER WRITE AND READ AFTER BSFG
                                ;;*****************************************************************
003066 000004                   TST3:   SCOPE
003070 012767 003122 176074             MOV     #TST4,$ESCAPE   ;;ESCAPE TO TEST 4 ON ERROR
003076 104422                           SELDRV                  ;SELECT DRIVE FOR TESTING
003100 104421                           REWIND                  ;GO TO "BOT"
003102 104413 017022                    WRITE   ,WLNK1          ;WRITE ONE BLOCK
003106 104415                           BSFG                    ;BACK OVER THE BLOCK
003110 104414 016766                    READ    ,RLNK1          ;NOW READ
003114 104423                           BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003116 016766                           RLNK1                   ;LINK TO THE READ BUFFER
003120 017022                           WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 4         TEST BSBG AFTER READ
                                ;;*****************************************************************
003122 000004                   TST4:   SCOPE
003124 012767 003172 176040             MOV     #TST5,$ESCAPE   ;;ESCAPE TO TEST 5 ON ERROR
003132 104422                           SELDRV                  ;SELECT DRIVE FOR TESTING
003134 104421                           REWIND                  ;GO TO "BOT"
003136 104413 017022                    WRITE   ,WLNK1          ;WRITE ONE BLOCK
003142 104421                           REWIND                  ;GO TO "BOT"
003144 104414 016766                    READ    ,RLNK1          ;READ
003150 104423                           BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003152 016766                           RLNK1                   ;LINK TO THE READ BUFFER
003154 017022                           WLNK1                   ;LINK TO THE WRITE BUFFER
003156 104416                           BSBG                    ;BACK UP
003160 104414 016766                    READ    ,RLNK1          ;READ
003164 104423                           BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003166 016766                           RLNK1                   ;LINK TO THE READ BUFFER
```

```
                                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 5         TEST BSFG AFTER READ
                                ;;**************************************************************
                                TST5:   SCOPE
003172  000004
003174  012767  003242  175770         MOV     #TST6,$ESCAPE   ;;ESCAPE TO TEST 6 ON ERROR
003202  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
003204  104421                        REWIND                          ;GO TO "BOT"
003206  104413  017022                WRITE   ,WLNK1                  ;WRITE A BLOCK
003212  104421                        REWIND                          ;GO TO "BOT"
003214  104414  016766                READ    ,RLNK1                  ;READ A BLOCK
003220  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
003222  016766                        RLNK1                           ;LINK TO THE READ BUFFER
003224  017022                        WLNK1                           ;LINK TO THE WRITE BUFFER
003226  104415                        BSFG                            ;BACK OVER THE DATA
003230  104414  016766                READ    ,RLNK1                  ;NOW READ THE DATA
003234  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
003236  016766                        RLNK1                           ;LINK TO THE READ BUFFER
003240  017022                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 6         TEST SFBG AFTER REWIND AND BSFG AFTER SFBG
                                ;;**************************************************************
                                TST6:   SCOPE
003242  000004
003244  012767  003302  175720         MOV     #TST7,$ESCAPE   ;;ESCAPE TO TEST 7 ON ERROR
003252  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
003254  104421                        REWIND                          ;GO TO "BOT"
003256  104413  017022                WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
003262  104421                        REWIND                          ;GO TO "BOT"
003264  104420                        SFBG                            ;SPACE OVER THE DATA
003266  104415                        BSFG                            ;BACK OVER THE DATA
003270  104414  016766                READ    ,RLNK1                  ;READ THE BLOCK
003274  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
003276  016766                        RLNK1                           ;LINK TO THE READ BUFFER
003300  017022                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 7         TEST WRITE AFTER READ
                                ;;**************************************************************
                                TST7:   SCOPE
003302  000004
003304  012767  003356  175660         MOV     #TST10,$ESCAPE  ;;ESCAPE TO TEST 10 ON ERROR
003312  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
003314  104421                        REWIND                          ;GO TO "CLEAR LEADER"
003316  104413  017022                WRITE   ,WLNK1                  ;WRITE ONE BLOCK
003322  104416                        BSBG                            ;BACK OVER THE DATA BLOCK
003324  104414  016766                READ    ,RLNK1                  ;       AND READ IT
003330  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
003332  016766                        RLNK1                           ;LINK TO THE READ BUFFER
003334  017022                        WLNK1                           ;LINK TO THE WRITE BUFFER
003336  104413  017026                WRITE   ,WLNK2                  ;WRITE A SECOND BLOCK
003342  104416                        BSBG                            ;BACK OVER IT
003344  104414  016772                READ    ,RLNK2                  ;AND READ IT
003350  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
003352  016772                        RLNK2                           ;LINK TO THE READ BUFFER
003354  017026                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 10        TEST WRITE AFTER WRITE AND READ AFTER READ
                                ;;**************************************************************
```

```
                                TST10:  SCOPE
003356  000004
003360  012767  003422  175604            MOV     #TST11,$ESCAPE  ;;ESCAPE TO TEST 11 ON ERROR
003366  104422                            SELDRV                  ;SELECT DRIVE FOR TESTING
003370  104421                            REWIND                  ;GO TO "CLEAR LEADER"
003372  104413  017022                    WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
003376  104413  017026                    WRITE   ,WLNK2
003402  104421                            REWIND                  ;GO TO BOT
003404  104414  016766                    READ    ,RLNK1          ;READ BOTH BLOCKS
003410  104414  016772                    READ    ,RLNK2
003414  104423                            BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003416  016772                            RLNK2                   ;LINK TO THE READ BUFFER
003420  017026                            WLNK2                   ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 11        TEST BSBG AFTER BSBG
                                ;;**************************************************************
003422  000004                  TST11:  SCOPE
003424  012767  003464  175540            MOV     #TST12,$ESCAPE  ;;ESCAPE TO TEST 12 ON ERROR
003432  104422                            SELDRV                  ;SELECT DRIVE FOR TESTING
003434  104421                            REWIND                  ;GO TO "CLEAR LEADER"
003436  104413  017022                    WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
003442  104413  017026                    WRITE   ,WLNK2
003446  104416                            BSBG                    ;BACK OVER THE 2ND BLOCK
003450  104416                            BSBG                    ;BACK OVER THE 1ST BLOCK
003452  104414  016766                    READ    ,RLNK1          ;READ THE 1ST BLOCK
003456  104423                            BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003460  016766                            RLNK1                   ;LINK TO THE READ BUFFER
003462  017022                            WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 12        TEST READ AFTER SFBG
                                ;;**************************************************************
003464  000004                  TST12:  SCOPE
003466  012767  003526  175476            MOV     #TST13,$ESCAPE  ;;ESCAPE TO TEST 13 ON ERROR
003474  104422                            SELDRV                  ;SELECT DRIVE FOR TESTING
003476  104421                            REWIND                  ;GO TO "CLEAR LEADER"
003500  104413  017022                    WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
003504  104413  017026                    WRITE   ,WLNK2
003510  104421                            REWIND                  ;BOT
003512  104420                            SFBG                    ;SPACE OVER THE 1ST BLOCK
003514  104414  016772                    READ    ,RLNK2          ;READ THE 2ND BLOCK
003520  104423                            BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
003522  016772                            RLNK2                   ;LINK TO THE READ BUFFER
003524  017026                            WLNK2                   ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 13        TEST SFBG AFTER BSBG
                                ;;**************************************************************
003526  000004                  TST13:  SCOPE
003530  012767  003572  175434            MOV     #TST14,$ESCAPE  ;;ESCAPE TO TEST 14 ON ERROR
003536  104422                            SELDRV                  ;SELECT DRIVE FOR TESTING
003540  104421                            REWIND                  ;GO TO "CLEAR LEADER"
003542  104413  017022                    WRITE   ,WLNK1          ;WRITE TWO(2) BLOCKS
003546  104413  017026                    WRITE   ,WLNK2
003552  104416                            BSBG                    ;BACK SPACE OVER BOTH BLOCKS
003554  104416                            BSBG
003556  104420                            SFBG                    ;SPACE FWD OVER 1ST BLOCK
003560  104414  016772                    READ    ,RLNK2          ;READ 2ND BLOCK
003564  104423                            BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
```

```
                                        RLNK2                           ;LINK TO THE READ BUFFER
                                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 14        TEST SFBG AFTER BSFG
                                ;;**************************************************************
                                TST14:  SCOPE
                                        MOV     #TST15,$ESCAPE  ;;ESCAPE TO TEST 15 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "CLEAR LEADER"
                                        WRITE   ,WLNK1                  ;WRITE TWO BLOCKS
                                        WRITE   ,WLNK2
                                        BSFG                            ;SPACE REV FILE
                                        SFBG                            ;SPACE FWD BLOCK
                                        READ    ,RLNK2                  ;READ THE 2ND BLOCK
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK2                           ;LINK TO THE READ BUFFER
                                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 15        REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
                                ;;**************************************************************
                                TST15:  SCOPE
                                        MOV     #TST16,$ESCAPE  ;;ESCAPE TO TEST 16 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "CLEAR LEADER"
                                        WRITE   ,WLNK1                  ;WRITE 2 BLOCKS
                                        WRITE   ,WLNK2
                                        BSBG                            ;BACK OVER BOTH OF THEM
                                        BSBG
                                        READ    ,RLNK1                  ;READ THE FIRST ONE
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK1                           ;LINK TO THE READ BUFFER
                                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                        READ    ,RLNK2                  ;NOW READ THE 2ND ONE
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK2                           ;LINK TO THE READ BUFFER
                                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                        BSBG                            ;BACK OVER THE 2ND ONE
                                        WRITE   ,WLNK3                  ;WRITE OVER THE 2ND BLOCK
                                        BSBG                            ;NOW BACK UP AND
                                        READ    ,RLNK3                  ;READ IT
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK3                           ;LINK TO THE READ BUFFER
                                        WLNK3                           ;LINK TO THE WRITE BUFFER
                                        BSBG                            ;BACK OVER IT AGAIN
                                        WRITE   ,WLNK4                  ;AND WRITE OVER IT AGAIN
                                        BSBG                            ;BACK UP AND
                                        READ    ,RLNK4                  ;READ THE BLOCK
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK4                           ;LINK TO THE READ BUFFER
                                        WLNK4                           ;LINK TO THE WRITE BUFFER
                                        BSBG                            ;BACK OVER THE BLOCK AGAIN
                                        WRITE   ,WLNK2                  ;AND WRITE OVER IT AGAIN
                                        BSBG                            ;NOW BACK UP
                                        READ    ,RLNK2                  ;AND READ IT
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK2                           ;LINK TO THE READ BUFFER
```

```
                                WLNK2                           ;LINK TO THE WRITE BUFFER
                                BSBG                            ;BACK OVER BOTH BLOCKS
                                BSBG
                                READ    ,RLNK1                  ;AND READ THE 1ST ONE
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK1                           ;LINK TO THE READ BUFFER
                                WLNK1                           ;LINK TO THE WRITE BUFFER
                                READ    ,RLNK2                  ;AND THE 2ND ONE
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK2                           ;LINK TO THE READ BUFFER
                                WLNK2                           ;LINK TO THE WRITE BUFFER
                        ;;**************************************************************
                        ;*TEST 16       REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
                        ;;**************************************************************
                        T15T16: SCOPE
                                MOV     #TST17,$ESCAPE  ;;ESCAPE TO TEST 17 ON ERROR
                                SELDRV                          ;SELECT DRIVE FOR TESTING
                                REWIND                          ;GO TO "CLEAR LEADER"
                                WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
                                BSBG                            ;BACK OVER IT
                                READ    ,RLNK1                  ;AND READ IT
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK1                           ;LINK TO THE READ BUFFER
                                WLNK1                           ;LINK TO THE WRITE BUFFER
                                WRITE   ,WLNK2                  ;WRITE 2ND BLOCK
                                BSBG                            ;BACK OVER IT
                                READ    ,RLNK2                  ;AND READ IT
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK2                           ;LINK TO THE READ BUFFER
                                WLNK2                           ;LINK TO THE WRITE BUFFER
                                BSBG                            ;BACK OVER IT AND
                                WRITE   ,WLNK4                  ;WRITE OVER 2ND BLOCK
                                BSBG                            ;BACK UP AND
                                READ    ,RLNK4                  ;READ IT
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK4                           ;LINK TO THE READ BUFFER
                                WLNK4                           ;LINK TO THE WRITE BUFFER
                                BSBG                            ;BACK OVER IT AGAIN
                                WRITE   ,WLNK5                  ;REWRITE 2ND BLOCK
                                BSBG
                                READ    ,RLNK5                  ;READ 2ND BLOCK
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK5                           ;LINK TO THE READ BUFFER
                                WLNK5                           ;LINK TO THE WRITE BUFFER
                                BSBG                            ;BACK OVER IT AGAIN
                                WRITE   ,WLNK3                  ;WRITE ANOTHER BLOCK OVER IT
                                BSBG                            ;BACK OVER 2 BLOCKS
                                BSBG
                                READ    ,RLNK1                  ;READ THE 1ST BLOCK
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK1                           ;LINK TO THE READ BUFFER
                                WLNK1                           ;LINK TO THE WRITE BUFFER
                                READ    ,RLNK3                  ;READ 2ND BLOCK
                                BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                RLNK3                           ;LINK TO THE READ BUFFER
                                WLNK3                           ;LINK TO THE WRITE BUFFER
```

```
                                ;;*****************************************************************
                                ;*TEST 17       TEST WFG AFTER WRITE AND BSFG AFTER WFG
                                ;;*****************************************************************
004204  000004                  TST17:  SCOPE
004206  012767  004242  174756          MOV     #TST20,$ESCAPE  ;;ESCAPE TO TEST 20 ON ERROR
004214  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
004216  104421                          REWIND                  ;GO TO "BOT"
004220  104413  017022                  WRITE   ,WLNK1          ;WRITE ONE BLOCK
004224  104412                          WFG                     ;FOLLOWED BY A FILE GAP
004226  104415                          BSFG                    ;BACK OVER THE DATA BLOCK
004230  104414  016766                  READ    ,RLNK1          ;READ THE DATA
004234  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004236  016766                          RLNK1                   ;LINK TO THE READ BUFFER
004240  017022                          WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 20       TEST BSBG AFTER WFG
                                ;;*****************************************************************
004242  000004                  TST20:  SCOPE
004244  012767  004300  174720          MOV     #TST21,$ESCAPE  ;;ESCAPE TO TEST 21 ON ERROR
004252  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
004254  104421                          REWIND                  ;GO TO BOT
004256  104413  017022                  WRITE   ,WLNK1          ;WRITE ONE BLOCK OF DATA
004262  104412                          WFG                     ;WRITE A FILE GAP
004264  104416                          BSBG                    ;BACK OVER THE DATA
004266  104414  016766                  READ    ,RLNK1          ;READ ONE BLOCK
004272  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004274  016766                          RLNK1                   ;LINK TO THE READ BUFFER
004276  017022                          WLNK1                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 21       TEST WRITE AFTER WFG
                                ;;*****************************************************************
004300  000004                  TST21:  SCOPE
004302  012767  004356  174662          MOV     #TST22,$ESCAPE  ;;ESCAPE TO TEST 22 ON ERROR
004310  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
004312  104421                          REWIND                  ;GO TO BOT
004314  104413  017022                  WRITE   ,WLNK1          ;WRITE ONE BOCK
004320  104421                          REWIND                  ;GO TO BOT
004322  104414  016766                  READ    ,RLNK1          ;READ ONE BLOCK
004326  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004330  016766                          RLNK1                   ;LINK TO THE READ BUFFER
004332  017022                          WLNK1                   ;LINK TO THE WRITE BUFFER
004334  104412                          WFG                     ;WRITE A FILE GAP
004336  104413  017026                  WRITE   ,WLNK2          ;WRITE ANOTHER BLOCK
004342  104416                          BSBG                    ;BACK OVER THE LAST BLOCK
004344  104414  016772                  READ    ,RLNK2          ;READ THE SECOND BLOCK
004350  104423                          BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
004352  016772                          RLNK2                   ;LINK TO THE READ BUFFER
004354  017026                          WLNK2                   ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 22       TEST WFG AFTER READ AND BSFG AFTER WRITE
                                ;;*****************************************************************
004356  000004                  TST22:  SCOPE
004360  012767  004434  174604          MOV     #TST23,$ESCAPE  ;;ESCAPE TO TEST 23 ON ERROR
004366  104422                          SELDRV                  ;SELECT DRIVE FOR TESTING
004370  104421                          REWIND                  ;GO TO "BOT"
004372  104413  017022                  WRITE   ,WLNK1          ;WRITE ONE BLOCK
```

```
1041  004376  104421                          REWIND                          ;GO TO "BOT"
1042  004400  104414  016766                  READ    ,RLNK1                  ;READ ONE BLOCK
1043  004404  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1044  004406  016766                          RLNK1                           ;LINK TO THE READ BUFFER
1045  004410  017022                          WLNK1                           ;LINK TO THE WRITE BUFFER
1046  004412  104412                          WFG                             ;WRITE A FILE GAP
1047  004414  104413  017026                  WRITE   ,WLNK2                  ;WRITE A BLOCK
1048  004420  104415                          BSFG                            ;BACK OVER THE DATA
1049  004422  104414  016772                  READ    ,RLNK2                  ;READ SECOND BLOCK
1050  004426  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1051  004430  016772                          RLNK2                           ;LINK TO THE READ BUFFER
1052  004432  017026                          WLNK2                           ;LINK TO THE WRITE BUFFER
1053                                ;;*****************************************************************
1054                                ;*TEST 23       TEST SFFG AFTER REWIND AND READ AFTER SFFG
1055                                ;;*****************************************************************
1056  004434  000004                TST23:  SCOPE
1057  004436  012767  004500  174526          MOV     #TST24,$ESCAPE  ;;ESCAPE TO TEST 24 ON ERROR
1058  004444  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1059  004446  104421                          REWIND                          ;GO TO "BOT"
1060  004450  104413  017022                  WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
1061  004454  104412                          WFG                             ;WRITE A FILE GAP
1062  004456  104413  017026                  WRITE   ,WLNK2                  ;WRITE A BLOCK OF DATA
1063  004462  104421                          REWIND                          ;GO TO "BOT"
1064  004464  104417                          SFFG                            ;SPACE OVER FIRST BLOCK
1065  004466  104414  016772                  READ    ,RLNK2                  ;READ THE 2ND BLOCK
1066  004472  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1067  004474  016772                          RLNK2                           ;LINK TO THE READ BUFFER
1068  004476  017026                          WLNK2                           ;LINK TO THE WRITE BUFFER
1069                                ;;*****************************************************************
1070                                ;*TEST 24       TEST BSBG AFTER SFFG
1071                                ;;*****************************************************************
1072  004500  000004                TST24:  SCOPE
1073  004502  012767  004546  174462          MOV     #TST25,$ESCAPE  ;;ESCAPE TO TEST 25 ON ERROR
1074  004510  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1075  004512  104421                          REWIND                          ;GO TO "BOT"
1076  004514  104413  017022                  WRITE   ,WLNK1                  ;WRITE DATA
1077  004520  104412                          WFG                             ;WRITE FILE GAP
1078  004522  104413  017026                  WRITE   ,WLNK2                  ;WRITE DATA
1079  004526  104421                          REWIND                          ;GO TO "BOT"
1080  004530  104417                          SFFG                            ;SPACE OVER FIRST FILE
1081  004532  104416                          BSBG                            ;BACKUP TO FIRST BLOCK
1082  004534  104414  016766                  READ    ,RLNK1                  ;READ THE BLOCK
1083  004540  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1084  004542  016766                          RLNK1                           ;LINK TO THE READ BUFFER
1085  004544  017022                          WLNK1                           ;LINK TO THE WRITE BUFFER
1086                                ;;*****************************************************************
1087                                ;*TEST 25       TEST SFBG AFTER SFFG
1088                                ;;*****************************************************************
1089  004546  000004                TST25:  SCOPE
1090  004550  012767  004622  174414          MOV     #TST26,$ESCAPE  ;;ESCAPE TO TEST 26 ON ERROR
1091  004556  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1092  004560  104421                          REWIND                          ;GO TO "BOT"
1093  004562  104413  017022                  WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
1094  004566  104412                          WFG                             ;FOLLOWED BY A FILE GAP
1095  004570  104413  017026                  WRITE   ,WLNK2                  ;FOLLOWED BY 2 BLOCKS OF DATA
1096  004574  104413  017032                  WRITE   ,WLNK3
```

```
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER 1ST FILE
                                        SFBG                            ;SPACE OVER BLOCK
                                        BSBG                            ;BACK OVER THE BLOCK
                                        READ    ,RLNK2                  ;READ THE LAST BLOCK
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK2                           ;LINK TO THE READ BUFFER
                                        WLNK2                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 26        TEST SFBG AFTER SFBG
                                ;;**************************************************************
                                TST26:  SCOPE
                                        MOV     #TST27,$ESCAPE  ;;ESCAPE TO TEST 27 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE A BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
                                        WRITE   ,WLNK2                  ;WRITE 3 BLOCKS OF DATA
                                        WRITE   ,WLNK3
                                        WRITE   ,WLNK4
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER THE 1ST FILE
                                        SFBG                            ;SPACE OVER TWO BLOCK
                                        SFBG
                                        READ    ,RLNK4                  ;READ LAST BLOCK OF DATA
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK4                           ;LINK TO THE READ BUFFER
                                        WLNK4                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 27        TEST BSFG AFTER SFBG AND BSFG AFTER BSFG
                                ;;**************************************************************
                                TST27:  SCOPE
                                        MOV     #TST30,$ESCAPE  ;;ESCAPE TO TEST 30 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
                                        WRITE   ,WLNK2                  ;WRITE A BLOCK OF DATA
                                        REWIND                          ;GO TO "BOT"
                                        SFFG                            ;SPACE OVER 1ST FILE
                                        SFBG                            ;SPACE OVER 1ST BLOCK OF 2ND FILE
                                        BSFG                            ;BACK TO BEGINNNG OF 2ND FILE
                                        BSFG                            ;BACK TO BEGINNING OF 1ST FILE
                                        READ    ,RLNK1                  ;READ 1ST BLOCK OF 1ST FILE
                                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
                                        RLNK1                           ;LINK TO THE READ BUFFER
                                        WLNK1                           ;LINK TO THE WRITE BUFFER
                                ;;**************************************************************
                                ;*TEST 30        TEST BSBG THRU A FILE GAP
                                ;;**************************************************************
                                TST30:  SCOPE
                                        MOV     #TST31,$ESCAPE  ;;ESCAPE TO TEST 31 ON ERROR
                                        SELDRV                          ;SELECT DRIVE FOR TESTING
                                        REWIND                          ;GO TO "BOT"
                                        WRITE   ,WLNK1                  ;WRITE ONE BLOCK OF DATA
                                        WFG                             ;WRITE A FILE GAP
```

```
                                ;WRITE TWO BLOCKS OF DATA
                                ;GO TO "BOT"
                                ;SPACE OVER 1ST FILE
                                ;SPACE OVER 1ST BLOCK OF 2ND FILE
                                ;BACK OVER 1 BLOCK
                                ;BACK OVER 1 BLOCK
                                ;READ
                                ;COMPARE THE READ AND WRITE BUFFERS
                                ;LINK TO THE READ BUFFER
                                ;LINK TO THE WRITE BUFFER
                        ;;**********************************************************************
                        ;*TEST 31        TEST BSFG AFTER BSBG
                        ;;**********************************************************************
                        TST31:  SCOPE
                                ;;ESCAPE TO TEST 32 ON ERROR
                                ;SELECT DRIVE FOR TESTING
                                ;GO TO "BOT"
                                ;WRITE ONE BLOCK OF DATA
                                ;WRITE A FILE GAP
                                ;WRITE A BLOCK OF DATA
                                ;GO TO "BOT"
                                ;SPACE OVER FILE
                                ;SPACE OVER 1ST BLOCK OF 2ND FILE
                                ;BACK OVER BLOCK
                                ;BACK OVER FILE
                                ;READ 1ST BLOCK ON TAPE
                                ;COMPARE THE READ AND WRITE BUFFERS
                                ;LINK TO THE READ BUFFER
                                ;LINK TO THE WRITE BUFFER
                        ;;**********************************************************************
                        ;*TEST 32        TEST BSBG AFTER BSFG
                        ;;**********************************************************************
                        TST32:  SCOPE
                                ;;ESCAPE TO TEST 33 ON ERROR
                                ;SELECT DRIVE FOR TESTING
                                ;GO TO "CLEAR LEADER"
                                ;WRITE BLOCK
                                ;WRITE A FILE GAP
                                ;FOLLOW WITH ANOTHER BLOCK
                                ;BACK UP 1 FILE
                                ;BACK UP 1 BLOCK
                                ;READ 1ST BLOCK ON TAPE
                                ;COMPARE THE READ AND WRITE BUFFERS
                                ;LINK TO THE READ BUFFER
                                ;LINK TO THE WRITE BUFFER
                        ;;**********************************************************************
                        ;*TEST 33        TEST BSFG AFTER SFFG AND SFFG AFTER READ
                        ;;**********************************************************************
                        TST33:  SCOPE
                                ;;ESCAPE TO TEST 34 ON ERROR
                                ;SELECT DRIVE FOR TESTING
                                ;GO TO "BOT"
                                ;WRITE A BLOCK OF DATA
                                ;WRITE A FILE GAP
                                ;WRITE A BLOCK OF DATA
```

```
1209  005176  104421                          REWIND                          ;GO TO "BOT"
1210  005200  104417                          SFFG                            ;SPACE OVER 1ST FILE
1211  005202  104415                          BSFG                            ;BACK TO BEGINNING OF 1ST FILE
1212  005204  104414  016766                  READ    ,RLNK1                  ;READ FIRST DATA BLOCK
1213  005210  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1214  005212  016766                          RLNK1                           ;LINK TO THE READ BUFFER
1215  005214  017022                          WLNK1                           ;LINK TO THE WRITE BUFFER
1216  005216  104417                          SFFG                            ;SPACE TO BEGINNING ON 2ND FILE
1217  005220  104414  016772                  READ    ,RLNK2                  ;READ 1ST BLOCK OF 2ND FILE
1218  005224  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1219  005226  016772                          RLNK2                           ;LINK TO THE READ BUFFER
1220  005230  017026                          WLNK2                           ;LINK TO THE WRITE BUFFER
1221                                  ;;**********************************************************************
1222                                  ;*TEST 34        TEST SFFG AFTER SFFG
1223                                  ;;**********************************************************************
1224  005232  000004                  TST34:  SCOPE
1225  005234  012767  005310  173730          MOV     #TST35,$ESCAPE  ;;ESCAPE TO TEST 35 ON ERROR
1226  005242  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1227  005244  104421                          REWIND                          ;GO TO BOT
1228  005246  104413  017022                  WRITE   ,WLNK1                  ;WRITE A BLOCK
1229  005252  104412                          WFG                             ;WRITE FILE GAP
1230  005254  104413  017026                  WRITE   ,WLNK2                  ;WRITE A BLOCK
1231  005260  104412                          WFG                             ;WRITE A FILE GAP
1232  005262  104413  017032                  WRITE   ,WLNK3                  ;WRITE A BLOCK
1233  005266  104412                          WFG                             ;WRITE A FILE GAP
1234  005270  104421                          REWIND                          ;GO TO "BOT"
1235  005272  104417                          SFFG                            ;SPACE OVER 1ST FILE
1236  005274  104417                          SFFG                            ;SPACE OVER 2ND FILE
1237  005276  104414  016776                  READ    ,RLNK3                  ;READ BLOCK OF 3RD FILE
1238  005302  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1239  005304  016776                          RLNK3                           ;LINK TO THE RE D BUFFER
1240  005306  017032                          WLNK3                           ;LINK TO THE WRITE BUFFER
1241                                  ;;**********************************************************************
1242                                  ;*TEST 35        TEST SFFG AFTER SFBG
1243                                  ;;**********************************************************************
1244  005310  000004                  TST35:  SCOPE
1245  005312  012767  005364  173652          MOV     #TST36,$ESCAPE  ;;ESCAPE TO TEST 36 ON ERROR
1246  005320  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
1247  005322  104421                          REWIND                          ;GO TO "BOT"
1248  005324  104413  017022                  WRITE   ,WLNK1                  ;WRITE A BLOCK
1249  005330  104413  017026                  WRITE   ,WLNK2                  ;WRITE A BLOCK
1250  005334  104412                          WFG                             ;WRITE A GAP
1251  005336  104413  017032                  WRITE   ,WLNK3                  ;WRITE A BLOCK
1252  005342  104412                          WFG                             ;WRITE A GAP
1253  005344  104421                          REWIND                          ;GO TO "BOT"
1254  005346  104420                          SFBG                            ;SPACE OVER THE 1ST BLOCK(AND 1ST FILE)
1255  005350  104417                          SFFG                            ;SPACE TO BEGINNING OF 2ND FILE
1256  005352  104414  016776                  READ    ,RLNK3                  ;READ 1ST BLOCK OF 2RD FILE
1257  005356  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
1258  005360  016776                          RLNK3                           ;LINK TO THE READ BUFFER
1259  005362  017032                          WLNK3                           ;LINK TO THE WRITE BUFFER
1260                                  ;;**********************************************************************
1261                                  ;*TEST 36        TEST WFG AFTER SFBG
1262                                  ;;**********************************************************************
1263  005364  000004                  TST36:  SCOPE
1264  005366  012767  005436  173576          MOV     #TST37,$ESCAPE  ;;ESCAPE TO TEST 37 ON ERROR
```

```
 1265  005374  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
 1266  005376  104421                        REWIND
 1267  005400  104413  017022                WRITE   ,WLNK1                  ;WRITE 2 BLOCKS
 1268  005404  104413  017026                WRITE   ,WLNK2
 1269  005410  104421                        REWIND                          ;GO TO "BOT"
 1270  005412  104420                        SFBG                            ;GET OVER THE 1ST BLOCK
 1271  005414  104412                        WFG                             ;WRITE A GAP OVER 2ND BLOCK
 1272  005416  104413  017032                WRITE   ,WLNK3                  ;WRITE A BLOCK
 1273  005422  104415                        BSFG                            ;BACK OVER THE BLOCK
 1274  005424  104414  016776                READ    ,RLNK3                  ;AND READ IT
 1275  005430  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
 1276  005432  016776                        RLNK3                           ;LINK TO THE READ BUFFER
 1277  005434  017032                        WLNK3                           ;LINK TO THE WRITE BUFFER
 1278                                ;;*************************************************************
 1279                                ;*TEST 37        TEST WRITE AFTER SFBG
 1280                                ;;*************************************************************
 1281  005436  000004                ↑ST37:  SCOPE
 1282  005440  012767  005510  173524        MOV     #TST40,$ESCAPE  ;;ESCAPE TO TEST 40 ON ERROR
 1283  005446  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
 1284  005450  104421                        REWIND                          ;GO TO BOT
 1285  005452  104413  017022                WRITE   ,WLNK1                  ;WRITE 2 BLOCK
 1286  005456  104413  017026                WRITE   ,WLNK2
 1287  005462  104421                        REWIND                          ;GO TO BOT
 1288  005464  104420                        SFBG                            ;SPACE OVER 1ST BLOCK
 1289  005466  104413  017032                WRITE   ,WLNK3                  ;WRITE OVER 2ND BLOCK
 1290  005472  104415                        BSFG                            ;GO TO BEGINNING OF FILE
 1291  005474  104420                        SFBG                            ;SPACE OVER 1ST BLOCK
 1292  005476  104414  016776                READ    ,RLNK3                  ;READ THE NEW 2ND BLOCK
 1293  005502  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
 1294  005504  016776                        RLNK3                           ;LINK TO THE READ BUFFER
 1295  005506  017032                        WLNK3                           ;LINK TO THE WRITE BUFFER
 1296                                ;;*************************************************************
 1297                                ;*TEST 40        3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
 1298                                ;;*************************************************************
 1299  005510  000004                ↑ST40:  SCOPE
 1300  005512  012767  005600  173452        MOV     #TST41,$ESCAPE  ;;ESCAPE TO TEST 41 ON ERROR
 1301  005520  104422                        SELDRV                          ;SELECT DRIVE FOR TESTING
 1302  005522  104421                        REWIND                          ;GO TO "BOT"
 1303  005524  104413  017022                WRITE   ,WLNK1                  ;WRITE A BLOCK
 1304  005530  104412                        WFG                             ;FILE GAP
 1305  005532  104413  017026                WRITE   ,WLNK2                  ;BLOCK
 1306  005536  104412                        WFG                             ;FILE GAP
 1307  005540  104413  017032                WRITE   ,WLNK3                  ;BLOCK
 1308  005544  104412                        WFG                             ;FILE GAP
 1309  005546  104421                        REWIND                          ;GO TO "BOT"
 1310  005550  104417                        SFFG                            ;SPACE OVER 1ST FILE
 1311  005552  104417                        SFFG                            ;SPACE OVER 2ND FILE
 1312  005554  104417                        SFFG                            ;SPACE OVER 3RD FILE
 1313  005556  104415                        BSFG                            ;BACK OVER 3RD FILE
 1314  005560  104415                        BSFG                            ;BACK OVER 2ND FILE
 1315  005562  104415                        BSFG                            ;BACK OVER 1ST FILE
 1316  005564  104417                        SFFG                            ;GO TO BEGINNING OF 2ND FILE
 1317  005566  104414  016772                READ    ,RLNK2                  ;READ THE BLOCK
 1318  005572  104423                        BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
 1319  005574  016772                        RLNK2                           ;LINK TO THE READ BUFFER
 1320  005576  017026                        WLNK2                           ;LINK TO THE WRITE BUFFER
```

```
                                ;;*****************************************************************
                                ;*TEST 41        TWO BLOCK FILES---MULTI SPACING FUNCTIONS
                                ;;*****************************************************************
005600  000004                  TST41:  SCOPE
005602  012767  005706  173362          MOV     #TST42,$ESCAPE  ;;ESCAPE TO TEST 42 ON ERROR
005610  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
005612  104421                          REWIND                          ;GO TO "BOT"
005614  104413  017022                  WRITE   ,WLNK1                  ;WRITE A 2 BLOCK FILE
005620  104413  017026                  WRITE   ,WLNK2
005624  104412                          WFG
005626  104413  017032                  WRITE   ,WLNK3                  ;WRITE A 2 BLOCK FILE
005632  104413  017036                  WRITE   ,WLNK4
005636  104412                          WFG
005640  104413  017042                  WRITE   ,WLNK5                  ;WRITE A 2 BLOCK FILE
005644  104413  017046                  WRITE   ,WLNK6
005650  104412                          WFG
005652  104421                          REWIND                          ;GO TO "BOT"
005654  104417                          SFFG                            ;SPACE OVER 1ST FILE
005656  104417                          SFFG                            ;SPACE OVER 2ND FILE
005660  104414  017006                  READ    ,RLNK5                  ;READ 1ST BLOCK OF 3RD FILE
005664  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
005666  017006                          RLNK5                           ;LINK TO THE READ BUFFER
005670  017042                          WLNK5                           ;LINK TO THE WRITE BUFFER
005672  104415                          BSFG                            ;BACK TO BEGINNING OF 3RD FILE
005674  104414  017006                  READ    ,RLNK5                  ;READ 1ST BLOCK OF 3RD FILE
005700  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
005702  017006                          RLNK5                           ;LINK TO THE READ BUFFER
005704  017042                          WLNK5                           ;LINK TO THE WRITE BUFFER
                                ;;*****************************************************************
                                ;*TEST 42        MULTI SPACING & TEST SFBG AFTER READ
                                ;;*****************************************************************
005706  000004                  TST42:  SCOPE
005710  012767  006056  173254          MOV     #TST43,$ESCAPE  ;;ESCAPE TO TEST 43 ON ERROR
005716  104422                          SELDRV                          ;SELECT DRIVE FOR TESTING
005720  104421                          REWIND                          ;GO TO "BOT"
005722  104413  017022                  WRITE   ,WLNK1                  ;WRITE A 3 BLOCK FILE
005726  104413  017026                  WRITE   ,WLNK2
005732  104413  017032                  WRITE   ,WLNK3
005736  104412                          WFG
005740  104413  017036                  WRITE   ,WLNK4                  ;WRITE A 2 BLOCK FILE
005744  104413  017042                  WRITE   ,WLNK5
005750  104412                          WFG
005752  104421                          REWIND                          ;GO TO "BOT"
005754  104417                          SFFG                            ;SPACE OVER THE 1ST FILE
005756  104417                          SFFG                            ;SPACE OVER THE 2ND FLE
005760  104415                          BSFG                            ;BACK TO BEGINNING OF 2 FILE
005762  104414  017002                  READ    ,RLNK4                  ;READ 1ST BLOCK OF 2ND FILE
005766  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
005770  017002                          RLNK4                           ;LINK TO THE READ BUFFER
005772  017036                          WLNK4                           ;LINK TO THE WRITE BUFFER
005774  104415                          BSFG                            ;BACK TO BEGINNING OF 2ND FILE
005776  104415                          BSFG                            ;BACK OVER 1ST FILE
006000  104414  016766                  READ    ,RLNK1                  ;READ 1ND BLOCK OF 1ST FILE
006004  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
006006  016766                          RLNK1                           ;LINK TO THE READ BUFFER
006010  017022                          WLNK1                           ;LINK TO THE WRITE BUFFER
```

```
 1377  006012  104415                          BSFG                    ;GO TO BEGINNING OF 1ST FILE
 1378  006014  104414  016766                 READ    .RLNK1          ;READ 1ST BLOCK
 1379  006020  104423                         BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1380  006022  016766                         RLNK1                   ;LINK TO THE READ BUFFER
 1381  006024  017022                         WLNK1                   ;LINK TO THE WRITE BUFFER
 1382  006026  104420                         SFBG                    ;GET OVER 2ND BLOCK
 1383  006030  104414  016776                 READ    .RLNK3          ;READ 3RD BLOCK
 1384  006034  104423                         BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1385  006036  016776                         RLNK3                   ;LINK TO THE READ BUFFER
 1386  006040  017032                         WLNK3                   ;LINK TO THE WRITE BUFFER
 1387  006042  104417                         SFFG                    ;GO TO BEGINNING OF 2ND FILE
 1388  006044  104414  017002                 READ    ,RLNK4          ;READ 1ST BLOCK OF 2ND FILE
 1389  006050  104423                         BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1390  006052  017002                         RLNK4                   ;LINK TO THE READ BUFFER
 1391  006054  017036                         WLNK4                   ;LINK TO THE WRITE BUFFER
 1392                                  ;;*****************************************************************
 1393                                  ;*TEST 43       TWO BLOCK FILES--MULTI SPACING & TEST SFFG AFTER BSBG
 1394                                  ;;*****************************************************************
 1395  006056  000004                 TST43:  SCOPE
 1396  006060  012767  006176  173104         MOV     #TST44,$ESCAPE  ;;ESCAPE TO TEST 44 ON ERROR
 1397  006066  104422                         SELDRV                  ;SELECT DRIVE FOR TESTING
 1398  006070  104421                         REWIND                  ;GO TO BOT
 1399  006072  104413  017022                 WRITE   .WLNK1          ;WRITE A 2 BLOCK FILE
 1400  006076  104413  017026                 WRITE   .WLNK2
 1401  006102  104412                         WFG
 1402  006104  104413  017032                 WRITE   ,WLNK3          ;WRITE A 2 BLOCK FILE
 1403  006110  104413  017036                 WRITE   ,WLNK4
 1404  006114  104412                         WFG
 1405  006116  104413  017042                 WRITE   .WLNK5          ;WRITE A 2 BLOCK FILE
 1406  006122  104413  017046                 WRITE   ,WLNK6
 1407  006126  104412                         WFG
 1408  006130  104421                         REWIND                  ;BOT
 1409  006132  104417                         SFFG                    ;SPACE OVER 3 FILES
 1410  006134  104417                         SFFG
 1411  006136  104417                         SFFG
 1412  006140  104415                         BSFG                    ;BACK OVER 2 FILES
 1413  006142  104415                         BSFG
 1414  006144  104416                         BSBG                    ;BACK TO 2ND BLOCK OF 1ST FILE
 1415  006146  104417                         SFFG                    ;GO TO BEGINNING OF 2ND FILE
 1416  006150  104414  016776                 READ    ,RLNK3          ;READ 1ST RECORD OF 2ND FILE
 1417  006154  104423                         BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1418  006156  016776                         RLNK3                   ;LINK TO THE READ BUFFER
 1419  006160  017032                         WLNK3                   ;LINK TO THE WRITE BUFFER
 1420  006162  104417                         SFFG                    ;GO TO 3RD FILE
 1421  006164  104414  017006                 READ    ,RLNK5          ;READ 1ST RECORD OF 3RD FILE
 1422  006170  104423                         BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1423  006172  017006                         RLNK5                   ;LINK TO THE READ BUFFER
 1424  006174  017042                         WLNK5                   ;LINK TO THE WRITE BUFFER
 1425                                  ;;*****************************************************************
 1426                                  ;*TEST 44       6 ONE BLOCK FILES---MULTI SPACING
 1427                                  ;;*****************************************************************
 1428  006176  000004                 TST44:  SCOPE
 1429  006200  012767  006366  172764         MOV     #$EOP,$ESCAPE   ;;ESCAPE TO $EOP ON ERROR
 1430  006206  104422                         SELDRV                  ;SELECT DRIVE FOR TESTING
 1431  006210  104421                         REWIND                  ;GO TO BOT
 1432  006212  104413  017022                 WRITE   ,WLNK1          ;WRITE A 1 RECORD FILE
```

```
 1433  006216  104412
 1434  006220  104413  017026          WFG
 1435  006224  104412                  WRITE   ,WLNK2          ;WRITE A 1 BLOCK FILE
 1436  006226  104413  017032          WFG
 1437  006232  104412                  WRITE   ,WLNK3          ;WRITE A 1 BLOCK FILE
 1438  006234  104413  017036          WFG
 1439  006240  104412                  WRITE   ,WLNK4          ;WRITE A 1 BLOCK FILE
 1440  006242  104413  017042          WFG
 1441  006246  104412                  WRITE   ,WLNK5          ;WRITE A 1 BLOCK FILE
 1442  006250  104413  017046          WFG
 1443  006254  104412                  WRITE   ,WLNK6          ;WRITE A 1 BLOCK FILE
 1444  006256  104421                  WFG
 1445  006260  104417                  REWIND                  ;BOT
 1446  006262  104417                  SFFG                    ;SPACE OVER 1ST FILE
 1447  006264  104417                  SFFG                    ;SPACE OVER 2ND FILE
 1448  006266  104417                  SFFG                    ;SPACE OVER 3RD FILE
 1449  006270  104417                  SFFG                    ;SPACE OVER 4TH FILE
 1450  006272  104417                  SFFG                    ;SPACE OVER 5TH FILE
 1451  006274  104415                  SFFG                    ;SPACE OVER 6TH FILE
 1452  006276  104414  017012          BSFG                    ;NOW BACK UP AND READ IT
 1453  006302  104423                  READ    ,RLNK6
 1454  006304  017012                  BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1455  006306  017046                  RLNK6                   ;LINK TO THE READ BUFFER
 1456  006310  104415                  WLNK6                   ;LINK TO THE WRITE BUFFER
 1457  006312  104415                  BSFG                    ;BACK UP TO 5TH FILE
 1458  006314  104414  017006          BSFG
 1459  006320  104423                  READ    ,RLNK5          ;AND READ IT
 1460  006322  017006                  BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1461  006324  017042                  RLNK5                   ;LINK TO THE READ BUFFER
 1462  006326  104415                  WLNK5                   ;LINK TO THE WRITE BUFFER
 1463  006330  104415                  BSFG                    ;BACK UP TO 3RD FILE
 1464  006332  104415                  BSFG
 1465  006334  104414  016776          BSFG
 1466  006340  104423                  READ    ,RLNK3          ;AND READ IT
 1467  006342  016776                  BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1468  006344  017032                  RLNK3                   ;LINK TO THE READ BUFFER
 1469  006346  104415                  WLNK3                   ;LINK TO THE WRITE BUFFER
 1470  006350  104415                  BSFG                    ;BACK UP TO 1ST FILE
 1471  006352  104415                  BSFG
 1472  006354  104414  016766          BSFG
 1473  006360  104423                  READ    ,RLNK1          ;AND READ IT
 1474  006362  016766                  BLKCMP                  ;COMPARE THE READ AND WRITE BUFFERS
 1475  006364  017022                  RLNK1                   ;LINK TO THE READ BUFFER
                                        WLNK1                   ;LINK TO THE WRITE BUFFER
```

```
                        .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.

006366                  $EOP:
006366  000004                  SCOPE
006370  005067  172506          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
006374  005067  172570          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
006400  005267  172474          INC     $PASS           ;;INCREMENT THE PASS NUMBER
006404  042767  100000  172466  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
006412  005327                  DEC     (PC)+           ;;LOOP?
006414  000001          $EOPCT: .WORD   1
006416  003015                  BGT     $DOAGN          ;;YES
006420  012737                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
006422  000001          $ENDCT: .WORD   1
006424  006414                  $EOPCT
006426  104401  006461          TYPE    ,$ENDMG         ;;TYPE "END PASS"
006432  013700  000042  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
006436  001405                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
006440  000005                  RESET                   ;;CLEAR THE WORLD
006442  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
006444  000240                  NOP                     ;;SAVE ROOM
006446  000240                  NOP                     ;;FOR
006450  000240                  NOP                     ;;ACT11
006452                  $DOAGN:
006452  000137                  JMP     @(PC)+          ;;RETURN
006454  002326          $RTNAD: .WORD   START
006456     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
006461     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS/
006466  050040  051501  000123
```

```
1512                                    ;;*****************************************************************
1513                                    ;
1514                                    .SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
1515
1516                                    ;CALL:
1517                                    ;       MOV     #DRVKEY,R0
1518                                    ;       JSR     PC,@#EXAM               ;R1 IS DESTROYED
1519                                    ;       NORMAL RETURN
1520                                    ;       ERROR RETURN
1521
1522  006474  013701  001220       EXAM:   MOV     @#TACSL,R1              ;PICKUP THE "CONTROL & STATUS" REG. ADR.
1523  006500  005011                       CLR     (R1)                    ;DRIVE="A", FUNCTION="WFG"
1524  006502  122710  000101               CMPB    #'A,(R0)                ;EXAMINE DRIVE "A"?
1525  006506  001402                       BEQ     1$                      ;BR IF YES
1526  006510  052711  000400               BIS     #UNIT,(R1)              ;SELECT DRIVE "B"
1527  006514  032711  000040       1$:     BIT     #READY,(R1)             ;WAIT ON READY
1528  006520  001775                       BEQ     1$
1529  006522  005711                       TST     (R1)                    ;ANY ERROR?
1530  006524  100024                       BPL     4$                      ;BR IF NO
1531  006526  032711  001000               BIT     #OFFLINE,(R1)           ;ERROR DUE TO "OFF LINE"?
1532  006532  001017                       BNE     3$                      ;BR IF YES
1533  006534  032711  010000               BIT     #WRTLOCK,(R1)           ;ERROR DUE TO "WRITE LOCK"?
1534  006540  001411                       BEQ     2$                      ;BR IF NO
1535  006542  122777  000201  172370       CMPB    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
1536  006550  001412                       BEQ     4$                      ;BR IF YES
1537  006552  122777  000203  172360       CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
1538  006560  001406                       BEQ     4$                      ;BR IF YES
1539  006562  000403                       BR      3$                      ;TAKE THE ERROR EXIT
1540  006564  03271.  0[illegible]  2$:     BIT     #LEADER,(R1)            ;ERROR DUE TO "CLEAR LEADER"?
1541  006570  001002                       BNE     4$                      ;BR IF YES
1542  006572  062716  000002       3$:     ADD     #2,(SP)                 ;TAKE ERROR RETURN
1543  006576  000207               4$:     RTS     PC                      ;RETURN
```

```
                        .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
                        ;*SW11=1         INHIBIT ITERATIONS
                        ;*SW09=1         LOOP ON ERROR
                        ;*CALL
                        ;*      SCOPE           ;;SCOPE=IOT

006600                  $SCOPE:
006600  104406                 CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
006602  105777  172336         TSTB    @$TKS           ;TTY FLAG UP?
006606  001405                 BEQ     100$            ;BR IF NO
006610  104407                 RDCHR                   ;GO GET ONE CHARACTER FROM TTY
006612  122627  000003         CMPB    (SP)+,#3        ;IS IT A CONTROL/C ?
006616  001001                 BNE     100$            ;BR IF NO
006620  104424                 SETLOOP                 ;GO GET "TEST" AND "SCOPE LOOP" ADDRESSES
006622  005727         100$:   TST     (PC)+           ;LOOP ON TEST?
006624  000000         LOOPKEY:        0
006626  001105                 BNE     $OVER           ;BR IF YES
006630  032777  040000  172302 1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
006636  001101                 BNE     $OVER           ;;YES IF SW14=1
                        ;*****START OF CODE FOR THE XOR TESTER*****
006640  000416         $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
006642  013746  000004         MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
006646  012737  006666  000004 MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
006654  005737  177060         TST     @#177060        ;;TIME OUT ON XOR?
006660  012637  000004         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
006664  000453                 BR      $SVLAD          ;;GO TO THE NEXT TEST
006666  022626         5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
006670  012637  000004         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
006674  000413                 BR      7$              ;;LOOP ON THE PRESENT TEST
006676                  6$:;*****END OF CODE FOR THE XOR TESTER*****
006676  105767  172201 2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
006702  001421                 BEQ     3$              ;;BR IF NO
006704  126767  172205  172171 CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
006712  101015                 BHI     3$              ;;BR IF NO
006714  032777  001000  172216 BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
006722  001404                 BEQ     4$              ;;BR IF NO
006724  016767  172160  172154 7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
006732  000443                 BR      $OVER
006734  105067  172143 4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
006740  005067  172224         CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
006744  000415                 BR      1$              ;;ESCAPE TO THE NEXT TEST
006746  032777  004000  172164 3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
006754  001011                 BNE     1$              ;;BR IF YES
006756  005767  172116         TST     $PASS           ;;IF FIRST PASS OF PROGRAM
006762  001406                 BEQ     1$              ;;      INHIBIT ITERATIONS
006764  005267  172114         INC     $ICNT           ;;INCREMENT ITERATION COUNT
006770  026767  172174  172106 CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
006776  002021                 BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
```

```
 1600  007000  012767  000001  172076  1$:     MOV     #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
 1601  007006  016767  172204  172154          MOV     $MYCNT,$TIMES     ;;SET NUMBER OF ITERATIONS TO DO
 1602  007014  105267  172062          $SVLAD: INCB    $TSTNM            ;;COUNT TEST NUMBERS
 1603  007020  011567  172062                  MOV     (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
 1604  007024  011667  172060                  MOV     (SP),$LPERR       ;;SAVE ERROR LOOP ADDRESS
 1605  007030  005067  172136                  CLR     $ESCAPE           ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 1606  007034  112767  000001  172053          MOVB    #1,$ERMAX         ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 1607  007042  016777  172034  172072  $OVER:  MOV     $TSTNM,@DISPLAY   ;;DISPLAY TEST NUMBER
 1608  007050  016716  172032                  MOV     $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
 1609  007054  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
                                ;*SW09=1         LOOP ON ERROR
                                ;*CALL
                                ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

007056                          $ERROR:
007056  104406                          CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
007060  011437  001162                  MOV     @TACS,@#$REG0           ;SAVE THE TA11 STATUS
007064  011537  001164                  MOV     @TADB,@#$REG1           ;SAVE THE DATA BUFFER
007070  105714                          TSTB    @TACS                   ;IF "XFER REQ" IS UP KNOCK IT DOWN
007072  100003                          BPL     100$
007074  112714  000020                  MOVB    #ILBS,@TACS
007100  000406                          BR      101$
007102  032714  000040          100$:   BIT     #READY,@TACS            ;IS READY SET?
007106  001003                          BNE     101$                    ;BR IF YES
007110  000005                          RESET                           ;REGAIN CONTROL
007112  013714  001162                  MOV     @#$REG0,@TACS           ;RESTORE THE TACS
007116  022626                  101$:   CMP     (SP)+,(SP)+             ;POP THE STACK
007120  005767  172122                  TST     ASKKEY                  ;MANUAL LOOPING?
007124  002002                          BGE     102$                    ;BR IF NO
007126  011667  172040                  MOV     (SP),$ESCAPE            ;SETUP FOR MANUAL ESCAPE
007132  016616  177774          102$:   MOV     -4(SP),(SP)             ;GET PC+2 OF ERROR
007136  105267  171741          7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
007142  001775                          BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
007144  016777  171732  171770          MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
007152  032777  002000  171760          BIT     #BIT10,@SWR             ;;BELL ON ERROR?
007160  001402                          BEQ     1$                      ;;NO - SKIP
007162  104401  001174                  TYPE    ,$BELL                  ;;RING BELL
007166  005267  171720          1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
007172  011667  171720                  MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
007176  162767  000002  171712          SUB     #2,$ERRPC
007204  117767  171706  171702          MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
007212  032777  020000  171720          BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
007220  001004                          BNE     20$                     ;;SKIP TYPEOUTS
007222  004767  000060                  JSR     PC,TYPERR               ;;GO TO USEF ERROR ROUTINE
007226  104401  001201                  TYPE    ,$CRLF
007232                          20$:
007232  005777  171702          2$:     TST     @SWR                    ;;HALT ON ERROR
007236  100002                          BPL     3$                      ;;SKIP IF CONTINUE
007240  000000                          HALT                            ;;HALT ON ERROR!
007242  104406                          CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
007244  032777  001000  171666  3$:     BIT     #BIT09,@SWR             ;;LOOP ON ERROR SWITCH SET?
007252  001402                          BEQ     4$                      ;;BR IF NO
007254  016716  171630                  MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
007260  005767  171706          4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
007264  001402                          BEQ     5$                      ;;BR IF NONE
007266  016716  171700                  MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

```
 1666  007272                        5$:
 1667  007272  022737  006442  000042        CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
 1668  007300  001001                        BNE     6$              ;;BRANCH IF NO
 1669  007302  000000                        HALT                    ;;YES
 1670  007304                        6$:
 1671  007304  000002                        RTI                     ;;RETURN
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
001234  001204  TYPERR: TYPE    .$CRLF                  ;TYPE A CARRIAGE RETURN & LINE FEED
                        TYPE                            ;TYPE THE FUNCTION BEING PERFORMED
000000          MFUNC:  0                               ;MESSAGE POINTER GOES HERE
001205                  TYPE    .$CRLF
000002  001234          SUB     #2,@#SAVPC              ;ADJUST THE MAIN FLOW PC
                        MOV     R0,-(SP)                ;SAVE R0
001114                  MOVB    @#$ITEMB,R0             ;PICKUP THE ITEM INDEX
                        DEC     R0                      ;ADJUST THE INDEX
                        ASL     R0                      ;SO IT WILL WORK FOR
                        ASL     R0                      ;THE ERROR TABLE
                        ASL     R0
001270                  ADD     #$ERRTB,R0              ;FORM THE TABLE POINTER
000002                  MOV     (R0)+,1$                ;PICKUP "ERROR MESSAGE" POINTER
                        TYPE                            ;TYPE "ERROR MESSAGE"
000000          1$:     0                               ;"ERROR MESSAGE POINTER" GOES HERE
000004                  MOV     (R0)+,2$                ;PICKUP "DATA HEADER" POINTER
                        BEQ     3$                      ;IF "0" DON'T TYPE
                        TYPE                            ;TYPE "DATA HEADER
000000          2$:     0                               ;"DATA HEADER" POINTER GOES HERE
012000          3$:     MOV     (R0)+,R0                ;PICKUP "DATA POINTER"
001004                  BNE     5$                      ;IF THERE IS DATA TO TYPE GO DO IT
012600          4$:     MOV     (SP)+,R0                ;RESTORE R0
001201                  TYPE    .$CRLF                  ;TYPE A CAPRAGE RETURN&LINE FEED
000207                  RTS     PC                      ;RETURN
                5$:
013046                  MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                                                ;;TYPE DATA
104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
005710                  TST     (R0)            ;TERMINATOR?
001770                  BEQ     4$              ;BR IF YES
104401  007426          TYPE    .6$             ;TYPE 2 SPACES
000771                  BR      5$              ;LOOP
020040  000     6$:     .ASCIZ  /  /            ;ASCII STRING OF 2 SPACES
007432                  .EVEN
```

```
                        ;;*****************************************************************
                        .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST

                        ;CALL
                        ;       JSR     PC,@#ASKDRV
                        ;       RETURN                  ;NOTE: R0 AND R1 ARE DESTROYED

007432  104401  015146  ASKDRV: TYPE    ,MSGDRV         ;<CRLF>"DRIVE(S)? "
007436  005067  171576          CLR     DRVKEY
007442  104410                  RDLIN                   ;GO GET A DRIVE
007444  012600                  MOV     (SP)+,R0        ;SETUP TO CHECK FOR VALID DRIVE(S)
007446  105710                  TSTB    @R0             ;WAS A DRIVE SELECTED?
007450  001425                  BEQ     NOTLGL          ;BR IF NO
007452  012701  001240          MOV     #DRVKEY,R1
007456  122710  000101  LOOP:   CMPB    #'A,@R0         ;WAS DRIVE "A" SELECTED?
007462  001002                  BNE     NOTA            ;BR IF NO
007464  112021                  MOVB    (R0)+,(R1)+     ;SET KEY FOR DRIVE "A"
007466  000411                  BR      NEXT
007470  122710  000102  NOTA:   CMPB    #'B,@R0         ;WAS DRIVE "B" SELECTED?
007474  001002                  BNE     NOTB            ;BR IF NO
007476  112021                  MOVB    (R0)+,(R1)+     ;SET KEY FOR DRIVE "B"
007500  000404                  BR      NEXT
007502  122710  000054  NOTB:   CMPB    #54,@R0         ;WAS A COMMA TYPED?
007506  001006                  BNE     NOTLGL          ;BR IF NO
007510  105720                  TSTB    (R0)+           ;DUMP THE COMMA
007512  105710  NEXT:   TSTB    @R0             ;TERMINATOR?
007514  001406                  BEQ     EXIT            ;BR IF YES
007516  023701  001242          CMP     #DRVKEY+2,R1    ;TWO DRIVES SELECTED?
007522  101355                  BHI     LOOP            ;BR IF NO
007524  104401  001200  NOTLGL: TYPE    ,$QUES          ;ILLEGAL INPUT DETECTED
007530  000740                  BR      ASKDRV          ;GO TRY AGAIN
007532  005767  171502  EXIT:   TST     DRVKEY          ;ANY DRIVE SELECTED?
007536  001772                  BEQ     NOTLGL          ;BR IF NO
007540  000207                  RTS     PC

                        ;;*****************************************************************
                        ;CALL
                        ;JSR     PC,@#ASKADR

007542  010046          ASKADR: MOV     R0,-(SP)        ;SAVE R0
007544  104401  015162  1$:     TYPE    ,MSGASK         ;"TACS?"
007550  104411                  RDOCT                   ;GET VALUE
007552  012600                  MOV     (SP)+,R0        ;PICK UP THE OCTAL NUMBER
007554  001423                  BEQ     3$              ;IF "0" USE OLD VALUES
007556  020027  160000          CMP     R0,#160000      ;MAKE SURE IT IS A BUS ADDRESS
007562  103770                  BLO     1$
007564  010037  001220          MOV     R0,@#TACSL      ;SAVE TOE TACS
007570  062700  000002          ADD     #2,R0           ;STEP TO TADB ADDRESS
007574  010037  001224          MOV     R0,@#TADBL      ;AND SAVE IT
007600  013737  001220  001222  MOV     @#TACSL,@#TACSH ;SET UP TACS UPPER
007606  005237  001222          INC     @#TACSH         ;BYTE POINTER

007612  013737  001224  001226  MOV     @#TADBL,@#TADBH ;SET UP TADB UPPER
007620  005237  001226          INC     @#TADBH         ;BYTE POINTER
007624  104401  015172  3$:     TYPE    ,MSGVEC         ;"VECTOR?"
```

```
 1764  007630  104411                  RDOCT
 1765  007632  012600                  MOV     (SP)+,R0
 1766  007634  001411                  BEQ     5$
 1767  007636  020027  001000          CMP     R0,#1000                ;MAKE SURE ADDRESS IS IN VECTOR AREA
 1768  007642  103370                  BHIS    3$
 1769  007644  010037  001230          MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
 1770  007650  062700  000002          ADD     #2,R0
 1771  007654  010037  001232          MOV     R0,@#TAVEC+2
 1772  007660  104401  015202    5$:   TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
 1773  007664  104411                  RDOCT
 1774  007666  012600                  MOV     (SP)+,R0
 1775  007670  001413                  BEQ     6$                      ;IF "0" USE OLD VALUE
 1776  007672  020027  000007          CMP     R0,#7                   ;MAKE SURE ITS VALID
 1777  007676  101370                  BHI     5$
 1778  007700  000300                  SWAB    R0                      ;PUT INTO HIGH BYTE
 1779  007702  006200                  ASR     R0                      ;AND SHIFT
 1780  007704  006200                  ASR     R0                      ;INTO PROPER
 1781  007706  006200                  ASR     R0                      ;POSITION
 1782  007710  042700  177437          BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
 1783  007714  010037  001234          MOV     R0,@#TAPRIO             ;STORE IT AWAY
 1784  007720  104401  015214    6$:   TYPE    ,MTACS                  ;TACS="
 1785  007724  016746  171270          MOV     TACSL,-(SP)     ;;SAVE TACSL FOR TYPEOUT
 1786  007730  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1787  007732  104401  015223          TYPE    ,MTADB                  ;"TADB="
 1788  007736  016746  171262          MOV     TADBL,-(SP)     ;;SAVE TADBL FOR TYPEOUT
 1789  007742  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1790  007744  104401  015232          TYPE    ,MTAVEC                 ;"VECTOR="
 1791  007750  016746  171254          MOV     TAVEC,-(SP)     ;;SAVE TAVEC FOR TYPEOUT
 1792  007754  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1793  007756  104401  015243          TYPE    ,MTAPRI                 ;"PRIORITY="
 1794  007762  016746  171246          MOV     TAPRIO,-(SP)    ;;SAVE TAPRIO FOR TYPEOUT
 1795  007766  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 1796  007770  104401  015256          TYPE    ,MSGOK                  ;"OK?"
 1797  007774  104407                  RDCHR                   ;GO READ ONE CHARACTER
 1798  007776  012600                  MOV     (SP)+,R0                ;GET IT
 1799  010000  022700  000015          CMP     #15,R0                  ;IS IT "CR"?
 1800  010004  001406                  BEQ     7$                      ;BRANCH IF YES
 1801  010006  022700  000131          CMP     #'Y,R0                  ;IS IT "Y"?
 1802  010012  001403                  BEQ     7$                      ;IT WAS
 1803  010014  104401  001200          TYPE    ,$QUES                  ;TYPE "?"
 1804  010020  000651                  BR      1$                      ;AND LET HIM CORRECT THEM
 1805  010022  104401  015264    7$:   TYPE    ,MYES                   ;TYPE OUT "YES"
 1806  010026  012600                  MOV     (SP)+,R0                ;RESTORE R0
 1807  010030  000207                  RTS     PC                      ;AND RETURN
 1808
 1809
 1810                          ;*****************************************************************
 1811                          ;THIS ROUTINE WILL CHANGE AN .ASCII STRING TO AN OCTAL NUMBER
 1812                          ;
 1813                          ;REGISTER R1 MUST BE INITILIZED BEFORE ENTERING
 1814                          ;
 1815                          ;CALL:
 1816                          ;       JSR     R3,@#A2OCT              ;CALL THE ROUTINE
 1817                          ;       POINTER                         ;ADDRESS OF THE FIRST CHARACTER IN THE STRING
 1818                          ;                                       ;STRING MUST BE TERMINATED BY A BYTE OF ALL 1'S
 1819                          ;       ERROR RETURN                    ;RETURN HERE IF ILLEGAL CHARACTER
```

```
1820                                    ;                               ;OR STRING IS TO LONG
1821                                    ;       NORMAL RETURN           ;OCTAL NUMBER IS IN R1
1822
1823  010032  010046                    A2OCT:  MOV     R0,-(SP)        ;SAVE R0
1824  010034  012300                            MOV     (R3)+,R0        ;PICK UP THE POINTER
1825  010036  012767  000006  000006           MOV     #6,2$           ;MAX. NUMBER OF CHAR.'S ALLOWED
1826  010044  105710                    1$:     TSTB    (R0)            ;TERMINATOR?
1827  010046  001420                            BEQ     3$              ;BR IF YES
1828  010050  005327                            DEC     (PC)+           ;COUNT THIS CHARACTER
1829  010052  000000                    2$:     0
1830  010054  002416                            BLT     4$              ;BR IF TO MANY CHARACTERS IN THE STRING
1831  010056  121027  000060                   CMPB    (R0),#'0        ;CHECK THIS CHAR. IS BETWEEN
1832  010062  002413                            BLT     4$              ;0 AND 7
1833  010064  121027  000067                   CMPB    (R0),#'7
1834  010070  003010                            BGT     4$
1835  010072  142710  177770                   BICB    #^C7,(R0)       ;STRIP AWAY THE ASCII CODE
1836  010076  006301                            ASL     R1              ;POSITION THE FOR THIS CHAR
1837  010100  006301                            ASL     R1
1838  010102  006301                            ASL     R1
1839  010104  152001                            BISB    (R0)+,R1        ;SET IN THIS DIGIT
1840  010106  000756                            BR      1$              ;GO GET THE NEXT CHARACTER
1841  010110  005723                    3$:     TST     (R3)+           ;INCREMENT FOR NORMAL RETURN
1842  010112  012600                    4$:     MOV     (SP)+,R0        ;RESTORE R0
1843  010114  000203                            RTS     R3              ;RETURN TO USER
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL INPUT THE ADDRESSES OF "TEST" AND "SCOPE LOOP" FROM THE TTY

010116                          T.SETLOOP:
010116  104401  015326                  TYPE    .MTEST                  ;TYPE "TEST PC?"
010122  104410                          RDLIN                           ;GET AN ASCII STRING
010124  012667  000022                  MOV     (SP)+,11$               ;GET FIRST ADDRESS OF INPUT BUFFER
010130  105777  000016                  TSTB    @11$                    ;FIRST CHAR. A "CR" ?
010134  001442                          BEQ     3$                      ;BR IF YES---GO CLEAR THE LOOP KEY
010136  022626                          CMP     (SP)+,(SP)+             ;POP POP
010140  010167  000110                  MOV     R1,SAVR1                ;SAVE R1
010144  005001                          CLR     R1                      ;SET PARTIAL TO ZERO
010146  004337  010032                  JSR     R3,@#A2OCT              ;GO CHANGE ASCII TO OCTAL
010152  000000                  11$:    0                               ;POINTER TO STRING
010154  000760                          BR      T.SETLOOP               ;ERROR RETURN--GO TRY AGAIN
010156  005721                          TST     (R1)+                   ;JUST INCREMENTING
010160  010116                          MOV     R1,(SP)                 ;PUT "TEST PC"; ON THE STACK
010162  104401  015341          5$:     TYPE    ,MLOOP                  ;TYPE "LOOP PC'?"
010166  104410                          RDLIN                           ;GET A STRING OF ASCII
010170  012[illegible]  000014          MOV     (SP)+,12$               ;GET FIRST ADDRESS OF INPUT BUFFER
010174  105777  000010                  TSTB    @12$                    ;JUST A "CR" ?
010200  001410                          BEQ     1$                      ;BR IF YES
010202  005001                          CLR     R1                      ;CLEAR PARTIAL
010204  004337  010032                  JSR     R3,@#A2OCT              ;CHANGE TO OCTAL
010210  000000                  12$:    0
010212  000763                          BR      5$
010214  010137  001106                  MOV     R1,@#$LPADR             ;SAVE SCOPE LOOP ADDRESS
010220  000402                          BR      2$                      ;GO SET LOOP KEY
010222  011637  001106          1$:     MOV     (SP),@#$LPADR           ;SAVE SCOPE LOOP ADDRESS
010226  011637  001110          2$:     MOV     (SP),@#$LPERR           ;ERROR LOOP SAME AS THE TEST ADDRESS
010232  012737  177777  006624          MOV     #-1,@#LOOPKEY           ;SET THE LOOP KEY
010240  000402                          BR      4$
010242  005037  006624          3$:     CLR     @#LOOPKEY               ;CLEAR THE LOOP KEY
010246  016701  000002          4$:     MOV     SAVR1,R1                ;RESTORE R1
010252  000002                          RTI                             ;BYE BYE
010254  000000                  SAVR1:  0                               ;HOLD R1 HERE
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL SELECT THE NEXT DRIVE TO TEST

010256                          T.SELDRV:
010256  032777  000200  170654          BIT     #SW07,@SWR              ;LOCK ON CURRENT DRIVE?
010264  001041                          BNE     5$                      ;BR IF YES
010266  010146                          MOV     R1,-(SP)                ;SAVE R1 ON THE STACK
010270  016701  170742                  MOV     DRIVE,R1                ;GET THE DRIVE
010274  111137  001250                  MOVB    (R1),@#CURDRV           ;SAVE CURRENT DRIVE
010300  121127  000101                  CMPB    (R1),#'A                ;IS IT DRIVE "A"?
010304  001406                          BEQ     1$                      ;BR IF YES
010306  052714  000400                  BIS     #UNIT,@TACS             ;SELECT DRIVE "B"
010312  012737  001262  001256          MOV     #RWDB,@#RWDFLG          ;SET REWIND POINTER TO "B"
010320  000405                          BR      2$
010322  042714  000400          1$:     BIC     #UNIT,@TACS             ;SELECT DRIVE "A"
010326  012737  001260  001256          MOV     #RWDA,@#RWDFLG          ;SET REWIND POINTER TO "A"
010334  005201                  2$:     INC     R1                      ;INCREMENT TO THE NEXT DRIVE
010336  105711                          TSTB    (R1)                    ;OUT OF DRIVES TO TEST?
010340  001002                          BNE     3$                      ;BR IF NO
010342  012701  001240                  MOV     #DRVKEY,R1              ;RESET TO THE FIRST DRIVE
010346  010167  170664          3$:     MOV     R1,DRIVE                ;SAVE FOR NEXT TIME
010352  032777  001000  170560          BIT     #SW09,@SWR              ;IS SWITCH 9=1?
010360  001002                          BNE     4$                      ;BR IF YES
010362  105037  001103                  CLRB    @#SERFLG                ;CLEAR THE ERROR INDICATOR
010366  012601                  4$:     MOV     (SP)+,R1                ;RESTORE R1
010370  000002                  5$:     RTI
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL COMPARE THE READ AND WRITE BUFFERS
010372                          T.BLKCMP:
010372  011600                          MOV     (SP),R0                 ;GET THE PC OF THE CALL
010374  010037  001204                  MOV     R0,@#SAVPC              ;SAVE THE PC OF THE CALL
010400  012737  015531  007314          MOV     #MBUFCK,@#MFUNC         ;POINTER FOR .ASCII MESSAGE
010406  012003                          MOV     (R0)+,R3                ;PICKUP THE READ LINK ADDRESS
010410  012301                          MOV     (R3)+,R1                ;PICKUP THE READ BUFFER ADDRESS
010412  011303                          MOV     (R3),R3                 ;PICKUP THE READ BUFFER SIZE
010414  013002                          MOV     @(R0)+,R2               ;PICKUP THE WRITE BUFFER ADDRESS
010416  005067  000034                  CLR     2$+2                    ;ZERO THE OFFSET
010422  010016                          MOV     R0,(SP)                 ;SET THE RETURN
010424  005037  001206                  CLR     @#BYTNUM                ;ZERO THE BYTE NUMBER
010430  032777  000010  170502          BIT     #SW03,@SWR              ;COMPARE DATA BUFFERS?
010436  001401                          BEQ     1$                      ;BR IF YES
010440  000002                          RTI                             ;NO---GO BACK
010442  005303                  1$:     DEC     R3                      ;CHECK FOR THE TERMINATION
010444  002001                          BGE     4$                      ;BR IF MORE BYTES TO COMPARE
010446  000002                          RTI                             ;RETURN
010450  005237  001206          4$:     INC     @#BYTNUM                ;COUNT THIS BYTE
010454  122162  000000          2$:     CMPB    (R1)+,0(R2)             ;COMPARE READ AND WRITE BUFFER
010460  001006                          BNE     3$                      ;BR IF NOT EQUAL
010462  005267  177770                  INC     2$+2                    ;INCREMENT WRITE BUFFER INDEX
010466  042767  177774  177762          BIC     #^C3,2$+2               ;DON'T LET IT GET BIGGER THAN 3
010474  000762                          BR      1$
010476  114137  001126          3$:     MOVB    -(R1),@#$BDDAT          ;GET THE RECEIVED BYTE
010502  066702  177750                  ADD     2$+2,R2                 ;FORM ADDRESS
010506  111237  001124                  MOVB    (R2),@#$GDDAT           ;GET THE EXPECTED BYTE
010512  104010                          ERROR   10                      ;RECEIVED AND EXPECTED DATA NOT EQUAL
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL WRITE A FILE GAP ON THE TAPE
010514  011637  001204          T.WFG:  MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
010520  012737  015512  007314          MOV     #MXWFG,@#MFUNC          ;SET MESSAGE POINTER FOR "WFG"
010526  004037  012234          1$:     JSR     R0,@#DO.CMD             ;GO STAR" A "WFG"
010532  000101                          XWFG!INT.EN!GO
010534  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
010540  004037  011726                  JSR     R0,@#WAITFLAG           ;GO WAIT FOR A FLAG
010544  104003                          ERROR   3                       ;NO FLAGS OCCURRED
010546  104004                          ERROR   4                       ;NO INTERRUPT
010550  004037  012024                  JSR     R0,@#FLAGS              ;NORMAL RETURN---FIND OUT WHAT  FLAG
010554  104002                          ERROR   2                       ;RETURN HERE IF "XFER REQ" AND "READY" =0
010556  104002                          ERROR   2                       ;RETURN HERE IF "XFER REQ" =1
010560  000002                          RTI                             ;"RETURN HERE IF "READY"=1 AND "ERROR" =0
010562  104002                          ERROR   2                       ;"READY"=1 AND "ERROR"=1
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL WRITE A RECORD ON THE TAPE
010564 011637 001204            .WRIT:  MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
010570 012737 015363 007314             MOV     #MXWRIT,@#MFUNC         ;POINTER FOR .ASCIZ OF "WRITE"
010576 017602 000000                    MOV     @(SP),R2                ;GET THE WRITE LINK POINTER
010602 062716 000002                    ADD     #2,(SP)                 ;ADJUST THE RETURN
010606 012201                           MOV     (R2)+,R1                ;GET FIRST ADDRESS OF WRITE BUFFER
010610 005037 012232                    CLR     @#CRC.WD                ;CLEAR THE CRC WORD
010614 005067 000066                    CLR     6$+2                    ;ZERO THE INDEX
010620 012202                           MOV     (R2)+,R2                ;GET NUMBER OF BYTE TO WRITE
010622 003001                           BGT     2$                      ;CHECK FOR LEGAL NUMBER
010624 104011                           ERROR   11                      ;BAD NUMBER OF BYTES
010626 004037 012234            2$:     JSR     R0,@#DO.CMD             ;GO START A "WRITE"
010632 000103                           XWRITE!INT.EN!GO
010634 005037 177776            3$:     CLR     @#PS                    ;ALLOW INTERRUPTS
010640 004037 011726                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
010644 104003                           ERROR   3                       ;NO FLAGS OCCURRED
010646 104004                           ERROR   4                       ;INTERRUPT FAILED
010650 005037 001254                    CLR     @#WAITKEY               ;CLEAR THE WAIT FOR INTERRUPT KEY
010654 012737 000340 177776             MOV     #340,@#PS               ;LOCK OUT THE WORLD
010662 004037 012024                    JSR     R0,@#FLAGS              ;WAIT FLAG OCCURRED?
010666 104002                           ERROR   2                       ;"READY"=0 & "XFER REQ" =0
010670 000402                           BR      4$                      ;"XFER REQ"=1
010672 104005                           ERROR   5                       ;"READY"=1 & "ERROR"=0
010674 104002                           ERROR   2                       ;"READY"=1 & "ERROR"=1
010676 005302                   4$:     DEC     R2                      ;NEED TO WRITE A BYTE ON TAPE
010700 002417                           BLT     5$                      ;BR IF NO
010702 005000                           CLR     R0
010704 156100 000000            6$:     BISB    0(R1),R0                ;PICKUP BYTE FOR TRANSFER
010710 110015                           MOVB    R0,@TADB                ;TRANSFER A BYTE TO THE TA11
010712 004737 012112                    JSR     PC,@#DO.CRC             ;CALCULATE CRC FOR THIS BYTE
010716 005267 177764                    INC     6$+2                    ;INCREASE THE INDEX
010722 042767 177774 177756             BIC     #^C3,6$+2               ;KEEP IT LESS THAN 4
010730 012777 012350 170272             MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
010736 000736                           BR      3$                      ;LOOP
010740 013761 012232 000004     5$:     MOV     @#CRC.WD,4(R1)          ;SAVE THE CRC WORD
010746 004037 012234                    JSR     R0,@#DO.CMD             ;START A "ILBS"
010752 000122                           XWRITE!INT.EN!ILBS
010754 005037 177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
010760 004037 011726                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
010764 104003                           ERROR   3                       ;NO FLAG
010766 104004                           ERROR   4                       ;NO INTERRUPT
010770 004037 012024                    JSR     R0,@#FLAGS              ;WHAT FLAG IS SET?
010774 104002                           ERROR   2                       ;NONE ("XFER REQ"=0 & "READY"=0)
010776 104002                           ERROR   2                       ;"XFER REQ"
011000 000002                           RTI                             ;"READY"
011002 104002                           ERROR   2                       ;"READY" & "ERROR"
```

```
 1998                                    ;;*****************************************************************
 1999                                    ;THIS ROUTINE WILL READ A RECORD FROM THE TAPE
 2000  011004  011637  001204            T.READ: MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
 2001  011010  012737  015371  007314            MOV     #MXREAD,@#MFUNC         ;SET POINTER OF .ASCIZ MESSAGE
 2002  011016  017602  000000                    MOV     @(SP),R2                ;GET THE READ LINK POINTER
 2003  011022  062716  00000_                    ADD     #2,(SP)                 ;ADJUST FOR RETURN
 2004  011026  010200                            MOV     R2,R0                   ;PICKUP AND SAVE THE "CRC" THAT
 2005  011030  016000  000034                    MOV     WLNK1-RLNK1(R0),R0      ;    WAS CALCULATED ON THE WRITE
 2006  011034  016067  000004  170146            MOV     4(R0),RCRC0             ;         DATA FOR THIS BLOCK
 2007  011042  012201                            MOV     (R2)+,R1                ;GET ADDRESS OF BUFFER
 2008  011044  012202                            MOV     (R2)+,R2                ;GET NUMBER OF BYTES
 2009  011046  003001                            BGT     2$                      ;MAKE SURE NUMBER IS LEGAL
 2010  011050  104011                            ERROR   11                      ;ILLEGAL NUMBER OF BYTES
 2011  011052  005037  012232            2$:     CLR     @#CRC.WD                ;CLEAR CRC WORD
 2012  011056  004037  012234                    JSR     R0,@#DO.CMD             ;GO START A "READ"
 2013  011062  000105                            XREAD!INT.EN!GO
 2014  011064  005037  177776            3$:     CLR     @#PS                    ;ALLOW INTERRUPTS
 2015  011070  105011                            CLRB    (R1)                    ;SET TERMINATOR IF LAST BYTE
 2016  011072  004037  011726                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
 2017  011076  104003                            ERROR   3                       ;NONE OCCURRED
 2018  011100  104004                            ERROR   4                       ;NO INTERRUPT
 2019  011102  005037  001254                    CLR     @#WAITKEY               ;CLEAR THE WAIT ON INTERRUPT KEY
 2020  011106  012737  000340  177776            MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
 2021  011114  004037  012024                    JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
 2022  011120  104002                            ERROR   2                       ;?UNKNOWN FLAG
 2023  011122  000402                            BR      4$                      ;"XFER REQ"
 2024  011124  104005                            ERROR   5                       ;"READY"
 2025  011126  000413                            BR      8$                      ;"READY" WITH "ERROR"
 2026  011130  005302                    4$:     DEC     R2                      ;READ A BYTE?
 2027  011132  002431                            BLT     5$                      ;BR IF NO
 2028  011134  005000                            CLR     R0
 2029  011136  151500                            BISB    @TADB,R0                ;READ ONE BYTE
 2030  011140  110021                            MOVB    R0,(R1)+                ;AND PUT IT IN THE BUFFER
 2031  011142  004737  012112                    JSR     PC,@#DO.CRC             ;CALCULATE CRC FOR THIS BYTE
 2032  011146  012777  012350  170054            MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
 2033  011154  000743                            BR      3$                      ;LOOP
 2034  011156  010237  001166            8$:     MOV     R2,@#$TMP0              ;GET NUMBER OF BYTES NOT READ
 2035  011162  001414                            BEQ     9$                      ;SKIP OUT IF 0
 2036  011164  013746  001172                    MOV     @#$ESCAPE,-(SP)         ;SAVE THE ESCAPE ADDRESS
 2037  011170  005037  001172                    CLR     @#$ESCAPE               ;NO ESCAPE DESIRED
 2038  011174  016646  000004                    MOV     4(SP),-(SP)             ;COPY THE STACK
 2039  011200  016646  000004                    MOV     4(SP),-(SP)
 2040  011204  104013                            ERROR   13                      ;BLOCK READ SHORT
 2041  011206  012637  001172                    MOV     (SP)+,@#$ESCAPE         ;RESTORE ESCAPE ADDRESS
 2042  011212  000002                            RTI                             ;LEAVE
 2043  011214  104002                    9$:     ERROR   2                       ;ERROR OCCURRED DURING READ
 2044  011216  013737  012232  001212    5$:     MOV     @#CRC.WD,@#RCRC1        ;SAVE THE CALCULATED CRC
 2045  011224  004037  012234                    JSR     R0,@#DO.CMD             ;DO AN ILBS
 2046  011230  000124                            XREAD!INT.EN!ILBS
 2047  011232  111537  001214                    MOVB    @TADB,@#RCRC2           ;SAVE 1ST BYTE OF CRC FROM TA11
 2048  011236  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
 2049  011242  004037  011726                    JSR     R0,@#WAITFLAG           ;WAIT ON FLAG
 2050  011246  104003                            ERROR   3                       ;NONE OCCURRED
 2051  011250  104004                            ERROR   4                       ;INTERRUPT FAILED
 2052  011252  111567  167737                    MOVB    @TADB,RCRC2+1           ;SAVE 2ND BYTE OF CRC FROM TA11
 2053  011256  004037  012024                    JSR     R0,@#FLAGS              ;CHECK FLAGS
```

```
 2054  011262  104002                          ERROR   2                       ;UNKNOWN FLAG
 2055  011264  104002                          ERROR   2                       ;"XFER REQ"
 2056  011266  000002                          RTI                             ;"READY"
 2057  011270  032714  037000                  BIT     #LEADER!WRTLOCK!FGAP!TIMERR!OFFLINE,@TACS
 2058  011274  001401                          BEQ     6$                      ;BR IF CRC ERROR
 2059  011276  104002                          ERROR   2                       ;ERROR OCCURRED
 2060  011300  032777  000020  167632  6$:     BIT     #SW04,@SWR              ;IGNORE CRC ERRORS?
 2061  011306  001001                          BNE     7$                      ;BR IF YES
 2062  011310  104012                          ERROR   12                      ;"READY" AND "CRC ERROR
 2063  011312  000002                  7$:     RTI                             ;IT WAS A CRC ERROR AND WE ARE IGNORING THEM
 2064                                                                          ;     SO RETURN TO CALLER
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL BACK SPACE ONE FILE GAP
011314  011637  001204          T.BSFG: MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
011320  012737  015443  007314          MOV     #MXBSFG,@#MFUNC         ;SAVE POINTER OF .ASCIZ NAME
011326  004037  012234          1$:     JSR     R0,@#DO.CMD             ;START A "BSFG"
011332  000107                          XBSFG!INT.EN!GO
011334  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
011340  004037  011726                  JSR     R0,@#WAITFLAG           ;WAIT ON FLAG
011344  104003                          ERROR   3                       ;NO FLAG
011346  104004                          ERROR   4                       ;FLAG DIDN'T INTERRUPT
011350  004037  012024                  JSR     R0,@#FLAGS              ;FIND OUT WHAT FLAG OCCURRED
011354  104002                          ERROR   2                       ;UNKNOWN
011356  104002                          ERROR   2                       ;"XFER REQ"
011360  000401                          BR      2$                      ;"READY"
011362  104002                          ERROR   2                       ;"READY" & "ERROR"
011364  032714  004000          2$:     BIT     #FGAP,@TACS             ;IN A FILE GAP?
011370  001401                          BEQ     3$                      ;BR IN NO
011372  000002                          RTI                             ;GO BACK
011374  104006                  3$:     ERROR   6                       ;"BSFG" DIDN'T STOP IN A FILE  GAP

                                ;;*****************************************************************
                                ;THIS ROUTINE WILL BACK SPACE ONE BLOCK GAP
011376  011637  001204          T.BSBG: MOV     (SP),@#SAVPC            ;SAVE THE PC OF THE CALL
011402  012737  015467  007314          MOV     #MXBSBG,@#MFUNC         ;MESSAGE POINTER
011410  004037  012234          1$:     JSR     R0,@#DO.CMD             ;START A "BSBG"
011414  000111                          XBSBG!INT.EN!GO
011416  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
011422  004037  011726                  JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
011426  104003                          ERROR   3                       ;NO FLAG OCCURRED
011430  104004                          ERROR   4                       ;FLAG DIDN'T INTERRUPT
011432  004037  012024                  JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
011436  104002                          ERROR   2                       ;UNKNOWN
011440  104002                          ERROR   2                       ;"XFER REQ"
011442  000002                          RTI                             ;"READY"-- GO BACK TO USER
011444  104002                          ERROR   2                       ;"READY" &"ERROR"
```

```
2101                                    ;;*****************************************************************
2102                                    ;THIS ROUTINE WILL SPACE FORWARD FOR ONE FILE GAP
2103  011446  011637  001204            T.SFFG: MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2104  011452  012737  015376  007314            MOV     #MXSFFG,@#MFUNC         ;POINTER TO .ASCIZ MESSAGE
2105  011460  004037  012234            1$:     JSR     R0,@#DO.CMD             ;START A "SFFG"
2106  011464  000113                            XSFFG!INT.EN!GO
2107  011466  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2108  011472  004037  011726                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2109  011476  104003                            ERROR   3                       ;FLAG FAILED TO OCCUR
2110  011500  104004                            ERROR   4                       ;FLAG DIDN'T INTERRUPT
2111  011502  004037  012024                    JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
2112  011506  104002                            ERROR   2                       ;UNKNOWN
2113  011510  104002                            ERROR   2                       ;"XFER REQ"
2114  011512  000401                            BR      2$                      ;"READY"
2115  011514  104002                            ERROR   2                       ;"READY" AND "ERROR"
2116  011516  032714  004000            2$:     BIT     #FGAP,@TACS             ;DID "SFFG" STOP IN A "FILE GAP"?
2117  011522  001401                            BEQ     3$                      ;BR IF NO
2118  011524  000002                            RTI                             ;RETURN TO CALLER
2119  011526  104006            3$:     ERROR   6                       ;"SFFG" DIDN'T STOP IN A FILE  GAP
2120
2121
2122                                    ;;*****************************************************************
2123                                    ;THIS ROUTINE WILL SPACE FORWARD ONE BLOCK GAP
2124  011530  011637  001204            T.SFBG: MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2125  011534  012737  015421  007314            MOV     #MXSFBG,@#MFUNC         ;POINTER TO .ASCIZ MESSAGE
2126  011542  004037  012234            1$:     JSR     R0,@#DO.CMD             ;START A "SFBG"
2127  011546  000115                            XSFBG!INT.EN!GO
2128  011550  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2129  011554  004037  011726                    JSR     R0,@#WAITFLAG           ;WAIT FOR A FLAG
2130  011560  104003                            ERROR   3                       ;NO FLAG
2131  011562  104004                            ERROR   4                       ;NO INTERRUPT
2132  011564  004037  012024                    JSR     R0,@#FLAGS              ;FIND OUT WHAT FLAG
2133  011570  104002                            ERROR   2                       ;UNKNOWN
2134  011572  104002                            ERROR   2                       ;"XFER REQ"
2135  011574  000002                            RTI                             ;"READY"
2136  011576  005737  001246                    TST     @#ASKKEY                ;RUNNING UNDER MANUAL LOOPING?
2137  011602  002001                            BGE     2$                      ;BR IF NO
2138  011604  000002                            RTI                             ;RETURN
2139  011606  104002            2$:     ERROR   2                       ;"READY" AND "ERROR"
```

```
2140                                    ;;*****************************************************************
2141                                    ;THIS ROUTINE WILL REWIND THE TAPE
2142  011610  011637  001204            .RWND:  MOV     (SP),@#SAVPC            ;SAVE PC OF CALL
2143  011614  012737  015354  007314            MOV     #MXRWND,@#MFUNC         ;POINTER TO .ASCIZ MESSAGE
2144  011622  004037  012024                    JSR     R0,@#FLAGS              ;CHECK THE FLAGS
2145  011626  104001                            ERROR   1                       ;NO FLAGS
2146  011630  104001                            ERROR   1                       ;"XFER REQ"
2147  011632  000411                            BR      1$                      ;NORMAL RETURN
2148  011634  005777  167416                    TST     @RWDFLG                 ;WAS LAST FUNCTION A REWIND?
2149  011640  001406                            BEQ     1$                      ;BR IF YES
2150  011642  032714  020000                    BIT     #LEADER,@TACS           ;ON CLEAR LEADER?
2151  011646  001403                            BEQ     1$                      ;BR IF NO
2152  011650  005077  167402                    CLR     @RWDFLG                 ;ALLOW THIS ERFOR ONLY ONE TIME
2153  011654  104001                            ERROR   1                       ;ON CLEAR LEADER BUT SHOULDN'T BE
2154  011656  004037  012234            1$:     JSR     R0,@#DO.CMD             ;GO START A "REWIND"
2155  011662  000117                            XRWND!INT.EN!GO
2156  011664  005037  177776                    CLR     @#PS                    ;ALLOW INTERRUPTS
2157  011670  004037  011726                    JSR     R0,@#WAITFLAG           ;WAIT ON FLAG
2158  011674  104003                            ERROR   3                       ;NO FLAG OCCURRED
2159  011676  104004                            ERROR   4                       ;NO INTERRUPT
2160  011700  004037  012024                    JSR     R0,@#FLAGS              ;WHAT FLAG OCCURRED
2161  011704  104002                            ERROR   2                       ;UNKNOWN
2162  011706  104002                            ERROR   2                       ;"XFER REQ"
2163  011710  000401                            BR      2$                      ;"READY"
2164  011712  104002                            ERROR   2                       ;"READY" & "ERROR"
2165  011714  032714  020000            2$:     BIT     #LEADER,@TACS           ;ON "CLEAR LEADER"?
2166  011720  001401                            BEQ     3$                      ;BR IF NO
2167  011722  000002                            RTI                             ;YES---RETURN
2168  011724  104007            3$:     ERROR   7                               ;"REWIND" FAILED TO GO TO "CLEAR LEADER"
```

```
                        ;*****************************************************************
                        ;THIS ROUTINE IS USED TO WAIT ON A FLAG
                        ;
                        ;CALL:
                        ;       JSR     R0,@#WAITFLAG
                        ;       RETURN  FOR NO FLAGS
                        ;       RETURN  FOR NO INTERRUPT
                        ;       NORMAL  RETURN
                        ;
011726                  WAITFLAG:
011726  005067  000064         CLR     LOCNT                   ;ZERO THE LOW COUNTER
011732  016767  000064  000060 MOV     MAXCNT,HICNT            ;SET THE HIGH COUNTER TO MAX.COUNT
011740  032777  000040  167172 BIT     #SWO5,@SWR              ;RUNNING WITH INTERRUPTS?
011746  001403                 BEQ     1$                      ;BR IF YES
011750  012767  177777  167276 MOV     #-1,WAITKEY             ;SET THE INTERRUPT OCCURRED FLAG
011756  032714  000240         1$:     BIT     #TR:REQ!READY,@TACS     ;TEST FOR FLAGS
011762  001007                 BNE     2$                      ;BR IF A FLAG IS UP
011764  005267  000026         INC     LOCNT                   ;COUNT THE LOW COUNTER
011770  001362                 BNE     1$                      ;LOOP IF NOT ZERO
011772  005367  000022         DEC     HICNT                   ;COUT THE HIGH COUNTER
011776  100367                 BPL     1$                      ;LOOP IF NOT NEG.
012000  000405                 BR      3$                      ;LEAVE
012002  005720         2$:     TST     (R0)+                   ;INCREMENT THE RETURN ADDRESS
012004  005767  167244         TST     WAITKEY                 ;LOOK AT THE WAIT ON INT. KEY
012010  001401                 BEQ     3$                      ;BR IF NOT SET
012012  005720                 TST     (R0)+                   ;INCREMENT THE RETURN ADDRESS
012014  000200         3$:     RTS     R0                      ;RETURN TO USER
012016  000000         LOCNT:  0
012020  000000         HICNT:  0
012022  000000         MAXCNT: 0
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE IS USED TO DETERMINE WHAT FLAG IS SET
                                ;
                                ;CALL:
                                ;       JSR     R0,@#FLAGS
                                ;       RETURN  FOR UNKNOWN FLAG
                                ;       RETURN  FOR "TRANSFER REQUEST"
                                ;       RETURN  FOR "READY"
                                ;       RETURN  FOR "READY" & "ERROR"
                                ;
012024  03271   000240          FLAGS:  BIT     #TR.REQ!READY,@TACS     ;CHECK FOR "XFER REQ" OR "READY"
012030  001427                         BEQ     LVFLAG                  ;BR IF NO FLAGS
012032  005720                         TST     (R0)+                   ;INCREMENT RETURN ADDRESS
012034  105714                         TSTB    @TACS                   ;CHECK FOR "XFER REQ"
012036  100424                         BMI     LVFLAG                  ;BR IF SET
012040  005720                         TST     (R0)+                   ;INCREMENT RETURN ADDRESS
012042  032777  000100  167D7D        BIT     #SW06,@SWR              ;STALL?
012050  001414                         BEQ     3$                      ;BR IF NO
012052  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012054  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012056  013700  001264                 MOV     @#STALL,R0              ;GET STALL TIME
012062  013701  001266                 MOV     @#STALL+2,R1
012066  062700  000001          1$:    ADD     #1,R0                   ;STALL
012072  005501                         ADC     R1
012074  001374                         BNE     1$                      ;LOOP
012076  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
012100  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
012102  005714                 3$:     TST     @TACS                   ;IS "ERROR" SET
012104  100001                         BPL     LVFLAG                  ;BR IN NO "ERROR"
012106  005720                         TST     (R0)+                   ;INCREMENT RETURN ADDRESS
012110  000200                 LVFLAG: RTS     R0
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL CALCULATE THE CRC
                                ;CALL:
                                :       JSR     PC,@#DO.CRC             ;R0=1 BYTE OF DATA

012112                          DO.CRC:
012112  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012114  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012116  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
012120  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
012122  010446                         MOV     R4,-(SP)        ;;PUSH R4 ON STACK
012124  012767  000010  000054         MOV     #8.,3$                  ;MAKE EIGHT ITERATIONS
012132  016703  000074                 MOV     CRC.WD,R3               ;PICKUP THE CRC WORD
012136  005001                  1$:    CLR     R1
012140  010302                         MOV     R3,R2                   ;GET THE PARTIAL CRC WORD
012142  042702  177776                 BIC     #^CBIT00,R2             ;STRIP AWAY EVERYTHING BUT BIT00
012146  006000                         ROR     R0                      ;PULL OFF THIS BIT
012150  006101                         ROL     R1                      ;AND SETUP TO XOR IT
012152  010104                         MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R2"
012154  040204                         BIC     R2,R4
012156  040102                         BIC     R1,R2
012160  050402                         BIS     R4,R2                   ;RESULTS TO "R2"
012162  006002                         ROR     R2
012164  103006                         BCC     2$
012166  012701  040002                 MOV     #BIT14!BIT01,R1
012172  010104                         MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R3"
012174  040304                         BIC     R3,R4
012176  040103                         BIC     R1,R3
012200  050403                         BIS     R4,R3                   ;RESULTS TO "R3"
012202  006003                  2$:    ROR     R3
012204  005327                         DEC     (PC)+
012206  000000                  3$:    0
012210  003352                         BGT     1$                      ;BR IF MORE BITS TO DO
012212  010367  000014                 MOV     R3,CRC.WD
012216  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
012220  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
012222  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
012224  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
012226  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
012230  000207                         RTS     PC
012232  000000                  CRC.WD: 0                               ;CRC WORD
```

```
                                ;*****************************************************
                                ;THIS ROUTINE WILL LOAD THE LOW BYTE OF TACS WITH THE DESIRED FUNCTION
                                ;
                                ;CALL:
                                ;       JSR     R0,@#DO.CMD
                                ;       DESIRED FUNCTIN
                                ;       RETURN
                                ;
012234  010146          DO.CMD: MOV     R1,-(SP)                ;SAVE R1
012236  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INERRUPTS
012244  012001                        MOV     (R0)+,R1                ;GET THE COMMAND
012246  042701  177640                BIC     #^C<INT.EN!ILBS!FUNCTION!GO>,R1 ;STRIP AWAY JUNK
012252  032777  000040  166660        BIT     #SW05,@SWR              ;WITH OR WITHOUT INTERRUPTS
012260  001405                        BEQ     4$                      ;BR IF WITH
012262  042701  000100                BIC     #INT.EN,R1              ;CLEAR INTERRUPT ENABLE IF SET
012266  012777  012366  166734        MOV     #BADINT,@TAVEC          ;SETUP TO CATCH BAD INTERRUPT
012274  005067  166754          4$:   CLR     WAITKEY                 ;ZERO THE WAIT ON INTERRUPT KEY
012300  032701  000100                BIT     #INT.EN,R1              ;RUNNING WITH INTERRUPTS?
012304  001403                        BEQ     3$                      ;BR IF NO
012306  012777  012350  166714        MOV     #SERV,@TAVEC            ;SETUP TO SERVICE THE INTERRUPT
012314  110114                  3$:   MOVB    R1,@TACS                ;LOAD THE COMMAND
012316  032701  000001                BIT     #GO,R1                  ;IS GO BIT ON A "1"?
012322  001403                        BEQ     2$                      ;BR IF GO =0
012324  032714  000040          1$:   BIT     #READY,@TACS            ;WAIT FOR "READY" TO CLEAR
012330  001375                        BNE     1$
012332  005101                  2$:   COM     R1                      ;SET OR CLEAR REWIND FLAG
012334  042701  177761                BIC     #^C<FUNCTION>,R1        ;EXTRACT FUNCTION BITS
012340  010177  166712                MOV     R1,@RWDFLG              ;AND SAVE AS THE REWIND FLAG
012344  012601                        MOV     (SP)+,R1                ;GET R1 BACK
012346  000200                        RTS     R0                      ;RETURN TO USER

                                ;*****************************************************
                                ;THIS ROUTINE WILL SERVICE ALL LEGAL INTERRUPTS

012350  012737  177777  001254  SERV:   MOV     #-1,@#WAITKEY           ;SET THE WAIT KEY
012356  012777  012366  166644          MOV     #BADINT,@TAVEC          ;SET TO CATCH ILLEGAL INTERRUPTS
012364  000002                        RTI

                                ;*****************************************************
                                ;THIS ROUTINE WILL CATCH ILLEGAL INTERRUPTS FROM THE TA11

012366  011637  001204          BADINT: MOV     (SP),@#SAVPC            ;SAVE WHERE WE WERE AT
012372  104014                        ERROR   14                      ;ILLEGAL INTERRUPT OCCURRED
```

```
2317                    ;;*****************************************************************
2318                    ;;*****************************************************************
2319                    ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60
2320                    ;NOTE:  *** BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ***
```

```
                                        :** NOTE: IF  USING  SOFTWARE  SWITCH  REGISTER THE
                                        :       PROGRAM  MUST  HAVE  BEEN  STARTED  AT  A
                                        :       NORMAL  STARTING  ADDRESS  FIRST  SO  THAT
                                        :       THE AUTO-SIZING ROUTINE CAN SET UP ADDRESS
                                        :       176  AS  THE  SOFTWARE  SWITCH  REGISTER
                                        :       PLACE VALUE INTO 176
                                    ;;**********************************************************************



                                    ;;**********************************************************************
                                    :       WRITE FILE GAPS FROM "BOT" TO "EOT"
                                            ;START AT 220
                                            ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
                                            ;THE "WRITE DELAY MONO".
                                    ;;**********************************************************************
012374  012706  001100          WFGSUB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012400  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012404  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012410  000005                          RESET                           ;RESET THE WORLD
012412  012737  012374  001110          MOV     #WFGSUB,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012420  104422                          SELDRV                          ;SELECT THE DRIVE
012422  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012426  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012432  001775                          BEQ     100$
012434  104412                  1$:     WFG                             ;WRITE A FILE GAP
012436  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012442  001774                          BEQ     1$                      ;BR IF NO
012444  000000                          HALT                            ;STOP IF YES
012446  000752                          BR      WFGSUB                  ;LOOP ON CONT.


                                    ;;**********************************************************************
                                    :       WRITE CONTINUOUS BLOCKS OF DATA
                                            ;START AT 224
                                            ;THE PROGRAM WILL HALT THREE(3) TIMES
                                            ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
                                            ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
                                            ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
                                            ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
                                            ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
```

```
                                        ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
                                ;;**************************************************************
012450  004737  013116          WRTSUB: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
012454  012706  001100          WLOOP:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012460  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012464  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012470  000005                          RESET                           ;RESET THE WORLD
012472  012737  012454  001110          MOV     #WLOOP,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
012500  104422                          SELDRV                          ;SELECT THE DRIVE
012502  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012506  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012512  001775                          BEQ     100$
012514  104413  017052          1$:     WRITE   ,WLNKX                  ;WRITE A BLOCK
012520  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012524  001773                          BEQ     1$                      ;BR IF NO
012526  000000                          HALT                            ;STOP IF "EOT"
012530  000751                          BR      WLOOP                   ;LOOP IF CONT.


                                ;;**************************************************************
                                ;       READ AND COMPARE CONTINUOUS BLOCKS OF DATA
                                        ;START AT 230
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
                                        ;AND THE "THRESHOLD POT".
                                ;;**************************************************************
012532  012706  001100          RDSUB:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012536  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012542  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012546  000005                          RESET                           ;RESET THE WORLD
012550  012737  012532  001110          MOV     #RDSUB,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
012556  104422                          SELDRV                          ;SELECT THE DRIVE
012560  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012564  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012570  001775                          BEQ     100$
012572  104414  017016          1$:     READ    ,RLNKX                  ;READ A BLOCK
012576  012706  001100                  MOV     #STACK,SP               ;INIT. IN CASE OF ERROR
012602  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012606  001351                          BNE     RDSUB                   ;BR IF YES---LOOP
012610  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
012612  017016                          RLNKX                           ;LINK TO THE READ BUFFER
012614  017052                          WLNKX                           ;LINK TO THE WRITE BUFFER
012616  000765                          BR      1$


                                ;;**************************************************************
                                ;       WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
                                        ;START AT 234
                                        ;THE PROGRAM WILL HALT THREE(3) TIMES
                                        ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
                                        ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
                                        ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
                                        ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
                                        ;AND THE "GAP TIME MONO".
```

```
                                        ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
                                ;;**********************************************************
012620  004737  013116          WGPBLK: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
012624  012706  001100          WGBLOP: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012630  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012634  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012640  000005                          RESET                           ;RESET THE WORLD
012642  012737  012624  001110          MOV     #WGBLOP,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012650  104422                          SELDRV                          ;SELECT THE DRIVE
012652  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012656  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012662  001775                          BEQ     100$
012664  104412                  1$:     WFG                             ;WRITE A FILE GAP
012666  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012672  001005                          BNE     2$                      ;BR IF YES
012674  104413  017052                  WRITE   ,WLNKX                  ;WRITE A BLOCK
012700  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012704  001767                          BEQ     1$                      ;BR IF NO
012706  000000                  2$:     HALT                            ;STOP AT "EOT"
012710  000745                          BR      WGBLOP                  ;START OVER ON CONT.


                                ;;**********************************************************
                                ;       READ AND COMPARE A BLOCK OF DATA AND A FILE GAP
                                        ;START AT 240
                                        ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
                                        ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
                                        ;AND THE "TAPE BLANK MONO".
                                ;;**********************************************************
012712  012706  001100          RGPBLK: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
012716  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
012722  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
012726  000005                          RESET                           ;RESET THE WORLD
012730  012737  012712  001110          MOV     #RGPBLK,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
012736  104422                          SELDRV                          ;SELECT THE DRIVE
012740  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
012744  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
012750  001775                          BEQ     100$
012752  104414  017016          1$:     READ    ,RLNKX                  ;READ A BLOCK OF DATA
012756  012706  001100                  MOV     #STACK,SP               ;KEEP OUT OF THE WAY
012762  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
012766  001351                          BNE     RGPBLK                  ;BR IF YES
012770  104423                          BLKCMP                          ;COMPARE THE READ AND WRITE BUFFERS
012772  017016                          RLNKX                           ;LINK TO THE READ BUFFER
012774  017052                          WLNKX                           ;LINK TO THE WRITE BUFFER
012776  104420                          SFBG                            ;GET INTO FILE GAP
013000  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
013004  001342                          BNE     RGPBLK                  ;BR IF YES
013006  000761                          BR      1$                      ;LOOP


                                ;;**********************************************************
                                ;       SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
                                        ;START AT 244
                                        ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
                                        ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
```

```
                                        ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
                                        ;(SIGNAL MONO CAN BE CHECKED).
                                ;;**************************************************************
013010  012706  001100          SFFGSB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
013014  013704  001220                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
013020  013705  001224                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
013024  000005                          RESET                           ;RESET THE WORLD
013026  012737  013010  001110          MOV     #SFFGSB,@#$LPERR        ;SETUP THE LOOP ON ERROR ADDRESS
013034  104422                          SELDRV                          ;SELECT THE DRIVE
013036  112714  000017                  MOVB    #XRWND!GO,@TACS         ;SEND TAPE TO "BOT"
013042  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
013046  001775                          BEQ     100$
013050  104417                  1$:     SFFG                            ;SPACE INTO A FILE GAP
013052  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
013056  001774                          BEQ     1$                      ;BR IF NO
013060  000000                          HALT                            ;STOP AT "EOT"
013062  000752                          BR      SFFGSB                  ;LOOP ON CONT.


                                ;;**************************************************************
                                ;       BACK SPACE FILE GAP
                                        ;START AT 250
                                        ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
                                ;;**************************************************************
013064  000005                  BSFGSB: RESET                           ;RESET THE WORLD
013066  012737  013064  001110          MOV     #BSFGSB,@#$LPERR        ;LOOP ON ERROR ADDRESS
013074  005067  166072                  CLR     $ESCAPE                 ;DON'T ESCAPE ON ERROR
013100  104422                          SELDRV                          ;SELECT THE DRIVE
013102  104415                  1$:     BSFG                            ;BACK SPACE A FILE GAP
013104  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
013110  001774                          BEQ     1$                      ;BR IF NO
013112  000000                          HALT                            ;STOP AT BOT
013114  000763                          BR      BSFGSB                  ;START OVER ON CONT.


                                ;;**************************************************************
                                ;       SETUP READ AND WRITE LINKS FOR SUBROUTINES
013116  005000                  SETBUF: CLR     R0
013120  000000                          HALT                            ;OPERATOR PUTS BYTE COUNT IN THE SWR
```

```
 2511  013122  022767  000176  166010          CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
 2512  013130  001001                          BNE     20$                     ;NO- GET OUT
 2513  013132  104405                          GTSWR                           ;GET VALUE
 2514  013134                          20$:                                    ;CONTINUE
 2515  013134  157700  166000                  BISB    @SWR,R0         ;PICKUP THE BYTE COUNT
 2516  013140  001006                          BNE     2$                      ;BR IF NON-ZERO
 2517  013142  105777  165773                  TSTB    @SWR+1                  ;CHECK IF GREATER THAN 377
 2518  013146  001402                          BEQ     1$                      ;BR IF NO
 2519  013150  012700  000376                  MOV     #376,R0                 ;SET FOR MAX ALLOWED
 2520  013154  005200                  1$:     INC     R0                      ;MAKE IT 377 OR 1
 2521  013156  010037  017020          2$:     MOV     R0,@#RLNKX+2            ;SETUP READ LINK
 2522  013162  010037  017054                  MOV     R0,@#WLNKX+2            ;SETUP WRITE LINK
 2523  013166  000000                          HALT                            ;SET PATTERN INTO THE SWR
```

```
 2524  013170  022767  000176  165742          CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
 2525  013176  001001                          BNE     21$                     ;NO- GET OUT
 2526  013200  104405                          GTSWR                           ;GET VALUE
 2527  013202                          21$:                                    ;CONTINUE
 2528  013202  017700  165732                  MOV     @SWR,R0         ;PICKUP THE PATTERN
 2529  013206  012701  017122                  MOV     #WBUFX,R1               ;PICKUP FIRST ADDRESS OF BUFFER
 2530  013212  110021                          MOVB    R0,(R1)+                ;FILL THE BUFFER
 2531  013214  110021                          MOVB    R0,(R1)+
 2532  013216  110021                          MOVB    R0,(R1)+
 2533  013220  110021                          MOVB    R0,(R1)+
 2534  013222  000000                          HALT                            ;SET OPERATIONAL SWITCHES
```

```
 2535  013224  022767  000176  165706          CMP     #SWREG,SWR              ;USING S/W SWITCH REG?
 2536  013232  001001                          BNE     22$                     ;NO- GET OUT
 2537  013234  104405                          GTSWR                           ;GET VALUE
 2538  013236                          22$:                                    ;CONTINUE
 2539  013236  000207                          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
                                ;*

013240  105767  165713  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
013244  100002                 BPL     1$              ;;BR IF YES
013246  000000                 HALT                    ;;HALT HERE IF NO TERMINAL
013250  000407                 BR      3$              ;;LEAVE
013252  010046          1$:    MOV     R0,-(SP)        ;;SAVE R0
013254  017600  000002         MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
013260  112046          2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
013262  001005                 BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
013264  005726                 TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
013266  012600          60$:   MOV     (SP)+,R0        ;;RESTORE R0
013270  062716  000002         3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
013274  000002                 RTI                     ;;RETURN
013276  122716  000011         4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
013302  001430                 BEQ     8$
013304  122716  000200         CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
013310  001006                 BNE     5$
013312  005726                 TST     (SP)+           ;;POP  <CR><LF> EQUIV
013314  104401                 TYPE                    ;;TYPE A CR AND LF
013316  001201                 $CRLF
013320  105067  000130         CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
013324  000755                 BR      2$              ;;GET NEXT CHARACTER
013326  004767  000056  5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
013332  126726  165620  6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
013336  001350                 BNE     2$              ;;IF NO GO GET NEXT CHAR.
013340  016746  165610         MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                       ;;AND THE NULL CHAR.
013344  105366  000001  7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
013350  002770                 BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
013352  004767  000032         JSR     PC,$TYPEC       ;;GO TYPE A NULL
013356  105367  000072         DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
013362  000770                 BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

013364  112716  000040  8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
013370  004767  000014  9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
013374  132767  000007  000052 BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
013402  001372                 BNE     9$              ;;TAB STOP
013404  005726                 TST     (SP)+           ;;POP SPACE OFF STACK
```

```
 2596  013406  000724                        BR      2$              ;;GET NEXT CHARACTER
 2597  013410  105777  165534        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 2598  013414  100375                        BPL     $TYPEC
 2599  013416  116677  000002  165526        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 2600  013424  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 2601  013432  001003                        BNE     1$              ;;BRANCH IF NO
 2602  013434  105067  000014                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 2603  013440  000406                        BR      $TYPEX          ;;EXIT
 2604  013442  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 2605  013450  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
 2606  013452  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
 2607  013454  000000        $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 2608  013456  000207        $TYPEX: RTS     PC
 2609
 2610                                        .SBTTL  TTY INPUT ROUTINE
 2611
 2612                                        ;;**************************************************************
 2613                                        .ENABL  LSB
 2614
 2615                                        ;;**************************************************************
 2616                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 2617                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 2618                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 2619                                        ;*WHEN OPERATING IN TTY FLAG MODE.
 2620  013460  022767  000176  165452  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 2621  013466  001074                        BNE     15$             ;;BRANCH IF NO
 2622  013470  105777  165450                TSTB    @$TKS           ;;CHAR THERE?
 2623  013474  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
 2624  013476  117746  165444                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
 2625  013502  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
 2626  013506  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 2627  013512  001062                        BNE     15$             ;;NO, RETURN TO USER
 2628  013514  126727  165414  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
 2629  013522  001456                        BEQ     15$             ;;BRANCH IF YES
 2630
 2631  013524  104401  014205                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
 2632  013530  104401  014212        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
 2633  013534  016746  164436                MOV     $SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT
 2634  013540  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2635  013542  104401  014223                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
 2636  013546  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
 2637  013550  005046                        CLR     -(SP)           ;;THE NEW SWR
 2638  013552  105777  165366        7$:     TSTB    @$TKS           ;;CHAR THERE?
 2639  013556  100375                        BPL     7$              ;;IF NOT TRY AGAIN
 2640
 2641  013560  117746  165362                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
 2642  013564  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 2643
 2644
 2645
 2646  013570  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
 2647  013574  001005                        BNE     10$             ;;BRANCH IF NOT
 2648  013576  104401  014200                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
 2649  013602  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOLS INPUT
 2650  013606  000757                        BR      19$             ;;LET'S TRY IT AGAIN
 2651
```

```
                                
                                
013610  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
013614  001022                          BNE     16$             ;;BRANCH IF NO
013616  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
013622  001403                          BEQ     11$             ;;BRANCH IF YES
013624  016700  000002  165306          MOV     2(SP),@SWR      ;;SAVE NEW SWR
013632  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
013636  104401  001201          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
013642  126727  165267  000001          CMPB    @$INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
013650  001003                          BNE     15$             ;;BRANCH IF NOT
013652  012777  000100  165264          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERPUPTS
013660  000002          15$:    RTI                             ;;RETURN
013662  004767  177522          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
013666  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
013672  002420                          BLT     18$             ;;BRANCH IF YES
013674  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
013700  003015                          BGT     18$             ;;BRANCH IF YES
013702  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
013706  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
013712  001403                          BEQ     17$             ;;BRANCH IF YES
013714  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
013716  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
013720  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
013722  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
013726  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
013732  000707                          BR      7$              ;;GET THE NEXT ONE
013734  104401  001200          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
013740  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                
                                
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
                                
013742  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
013744  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
013752  105777  165166          1$:     TSTB    @$TKS           ;;WAIT FOR
013756  100375                          BPL     1$              ;;A CHARACTER
013760  117766  165162  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
013766  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
013774  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
014002  001013                          BNE     3$              ;;BRANCH IF NO
014004  105777  165134          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
014010  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
014012  117746  165130                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
014016  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
014022  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
014026  001366                          BNE     2$              ;;IF NOT DISCARD IT
014030  000750                          BR      1$              ;;YES, RESUME
014032  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
014040  002407                          BLT     4$              ;;BRANCH IF YES
```

```
2708  014042  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
2709  014050  003003                          BGT     4$              ;;BRANCH IF YES
2710  014052  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
2711  014060  000002                  4$:     RTI                     ;;GO BACK TO USER
2712                                          ;;**************************************************************
2713                                          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2714                                          ;*CALL:
2715                                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
2716                                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2717                                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2718
2719  014062  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
2720  014064  012703  014170          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
2721  014070  022703  014200          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
2722  014074  101405                          BLOS    4$              ;;BR IF YES
2723  014076  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
2724  014100  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
2725  014102  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
2726  014106  001003                          BNE     3$              ;;SKIP IF NOT
2727  014110  104401  001200          4$:     TYPE    ,$QUES          ;;TYPE A '?'
2728  014114  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
2729  014116  111367  000044          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
2730  014122  104401  014166                  TYPE    ,9$
2731  014126  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
2732  014132  001356                          BNE     2$              ;;LOOP IF NOT RETURN
2733  014134  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
2734  014140  104401  001202                  TYPE    ,$LF            ;;TYPE A LINE FEED
2735  014144  012603                          MOV     (SP)+,R3        ;;RESTORE R3
2736  014146  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2737  014150  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
2738  014156  012766  014170  000004          MOV     #$TTYIN,4(SP)
2739  014164  000002                          RTI                     ;;RETURN
2740  014166     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
2741  014167     000                          .BYTE   0               ;;TERMINATOR
2742  014170  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
2743  014200  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
2744  014205     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
2745  014212  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
2746  014220  020075     000
2747  014223     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
2748  014230  036440  000040
2749                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2750
2751                                  ;;**************************************************************
2752                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2753                                  ;*CHANGE IT TO BINARY.
2754                                  ;*CALL:
2755                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
2756                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2757                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
2758
2759  014234  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2760  014236  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
2761  014244  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2762  014246  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2763  014250  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

# E07



```
2764  014252  104410                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
2765  014254  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2766  014256  005001                          CLR     R1              ;;CLEAR DATA WORD
2767  014260  005002                          CLR     R2
2768  014262  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2769  014264  001412                          BEQ     3$              ;;IF ZERO GET OUT
2770  014266  006301                          ASL     R1              ;;*2
2771  014270  006102                          ROL     R2
2772  014272  006301                          ASL     R1              ;;*4
2773  014274  006102                          ROL     R2
2774  014276  006301                          ASL     R1              ;;*8
2775  014300  006102                          ROL     R2
2776  014302  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2777  014306  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2778  014310  000764                          BR      2$              ;;LOOP
2779  014312  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2780  014314  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2781  014320  010267  000010                  MOV     R2,$HIOCT
2782  014324  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2783  014326  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2784  014330  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2785  014332  000002                          RTI                     ;;RETURN
2786  014334  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 15-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

014336  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
014342  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
014350  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
014354  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
014360  000406                          BR      $TYPON
014362  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
014370  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
014376  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
014404  010346                          MOV     R3,-(SP)        ;;SAVE R3
014406  010446                          MOV     R4,-(SP)        ;;SAVE R4
014410  010546                          MOV     R5,-(SP)        ;;SAVE R5
014412  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
014416  005404                          NEG     R4
014420  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
014424  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
014430  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
014434  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
014440  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
014442  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
014444  000404                          BR      3$              ;;GO DO MSB
014446  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
014450  006105                          ROL     R5
014452  006105                          ROL     R5
014454  010503                          MOV     R5,R3
014456  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
014460  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
014464  100016                          BPL     2$              ;;BR IF NO
014466  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
014472  001002                          BNE     4$              ;;TEST FOR 0
014474  005704                          TST     R4              ;;SUPPRESS THIS 0?
014476  001403                          BEQ     5$              ;;BR IF YES
```

```
 2843  014500  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 2844  014502  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 2845  014506  052703  000040            5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 2846  014512  110367  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
 2847  014516  104401  014556                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
 2848  014522  105367  000032            7$:   DECB    $OCNT           ;;COUNT BY 1
 2849  014526  003347                        BGT     2$              ;;BR IF MORE TO DO
 2850  014530  002402                        BLT     6$              ;;BR IF DONE
 2851  014532  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 2852  014534  000744                        BR      2$              ;;GO DO THE LAST DIGIT
 2853  014536  012605                  6$:   MOV     (SP)+,R5        ;;RESTORE R5
 2854  014540  012604                        MOV     (SP)+,R4        ;;RESTORE R4
 2855  014542  012603                        MOV     (SP)+,R3        ;;RESTORE R3
 2856  014544  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 2857  014552  012616                        MOV     (SP)+,(SP)
 2858  014554  000002                        RTI                     ;;RETURN
 2859  014556     000                  9$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 2860  014557     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 2861  014560     000                  $OCNT:.BYTE   0               ;;OCTAL DIGIT COUNTER
 2862  014561     000                  $OFILL:.BYTE  0               ;;ZERO FILL SWITCH
 2863  014562  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.

014564  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
014566  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
014572  005740                          TST     -(R0)           ;;BACKUP BY 2
014574  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
014576  006300                          ASL     R0              ;;POSITION FOR INDEXING
014600  016000  014620                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
014604  000200                          RTS     R0              ;;GO TO ROUTINE


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

014606  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
014610  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
014616  000002                          RTI                     ;;RESTORE THE PSW

                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------

014620  014606                  $TRPAD: .WORD   $TRAP2
014622  013240                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
014624  014362                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
014626  014336                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
014630  014376                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

014632  013530                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

014634  013460                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
014636  013742                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
014640  014062                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
014642  014234                          $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
014644  010514                          T.WFG   ;;CALL=WFG      TRAP+12(104412) ROUTINE TO WRITE A FILE GAP
014646  010564                          T.WRIT  ;;CALL=WRITE    TRAP+13(104413) ROUTINE TO WRITE A BLOCK OF DATA
014650  011004                          T.READ  ;;CALL=READ     TRAP+14(104414) ROUTINE TO READ A BLOCK OF DATA
014652  011314                          T.BSFG  ;;CALL=BSFG     TRAP+15(104415) ROUTINE TO BACK SPACE A FILE GAP
014654  011376                          T.BSBG  ;;CALL=BSBG     TRAP+16(104416) ROUTINE TO BACK SPACE A BLOCK GAP
014656  011446                          T.SFFG  ;;CALL=SFFG     TRAP+17(104417) ROUTINE TO SPACE FWD A FILE GAP
014660  011530                          T.SFBG  ;;CALL=SFBG     TRAP+20(104420) ROUTINE TO SPACE FWD A BLOCK GAP
014662  011610                          T.RWND  ;;CALL=REWIND   TRAP+21(104421) ROUTINE TO REWIND THE TAPE
014664  010256                          T.SELDRV        ;;CALL=SELDRV   TRAP+22(104422) ROUTINE TO SELECT THE NEXT DRIVE
014666  010372                          T.BLKCMP        ;;CALL=BLKCMP   TRAP+23(104423) COMPARE READ AND WRITE BUFFERS
014670  010116                          T.SETLOOP       ;;CALL=SETLOOP  TRAP+24(104424) SETUP TO LOOP AS PER THE TTY
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES
                                
                                        ;;**********************************************************************
                                        ;POWER DOWN ROUTINE
014672  012737  015036  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
014700  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
014706  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
014710  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
014712  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
014714  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
014716  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
014720  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
014722  017746  164212                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
014726  010667  000110                MOV     SP,$SAVR6       ;;SAVE SP
014732  012737  014744  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
014740  000000                        HALT
014742  000776                        BR      .-2             ;;HANG UP
                                
                                        ;;**********************************************************************
                                        ;POWER UP ROUTINE
014744  012737  015036  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
014752  016706  000064                MOV     $SAVR6,SP       ;;GET SP
014756  005067  000060                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
014762  005267  000054          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
014766  001375                        BNE     1$              ;;OF  WORD
014770  012677  164144                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
014774  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
014776  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
015000  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
015002  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
015004  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
015006  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
015010  012737  014672  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
015016  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
015024  104401                        TYPE                    ;REPORT THE POWER FAILURE
015026  015044          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
015030  012716                        MOV     (PC)+,(SP)      ;;RESTART AT PWRST
015032  002314          $PWRAD: .WORD   PWRST                   ;;RESTART ADDRESS
015034  000002                        RTI
015036  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
015040  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMP_ETE
015042  000000          $SAVR6: 0                               ;;PUT THE SP HERE
015044  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
015052  000122
                                        .EVEN
```

```
                                        ;;*****************************************************************
                                        ;DATA POINTERS
015054  001204  001116  001162  DT1:    .WORD   SAVPC,$ERRPC,$REG0,$REG1,0
015062  001164  000000
015066  001204  001116  001162  DT2:    .WORD   SAVPC,$ERRPC,$REG0,$GDDAT,$BDDAT,BYTNUM,0
015074  001124  001126  001206
015102  000000
015104  001204  001116  001162  DT3:    .WORD   SAVPC,$ERRPC,$REG0,RCRC0,RCRC1,RCRC2,0
015112  001210  001212  001214
015120  000000
015122  001204  001116  001162  DT4:    .WORD   SAVPC,$ERRPC,$REG0,$REG1,$TMP0,0
015130  001164  001166  000000
015136  001204  001116  001162  DT5:    .WORD   SAVPC,$ERRPC,$REG0,0
015144  000000
```

```
2983                                    ;;**********************************************************************
2984
2985                                    ;MESSAGES
2986
2987  015146  042200  044522  042526   MSGDRV: .ASCIZ  <CRLF>"DRIVE(S)? "
2988  015154  051450  037451  000040
2989  015162  005015  040524  051503   MSGASK: .ASCIZ  <15><12>/TACS?/
2990  015170  000077
2991  015172  042526  052103  051117   MSGVEC: .ASCIZ  /VECTOR?/
2992  015200  000077
2993  015202  051120  047511  044522   MSGPRI: .ASCIZ  /PRIORITY?/
2994  015210  054524  000077
2995  015214  052040  041501  036523   MTACS:  .ASCIZ  / TACS=/
2996  015222     000
2997  015223     040  040524  041104   MTADB:  .ASCIZ  / TADB=/
2998  015230  000075
2999  015232  053040  041505  047524   MTAVEC: .ASCIZ  / VECTOR=/
3000  015240  036522     000
3001  015243     040  051120  047511   MTAPRI: .ASCIZ  / PRIORITY=/
3002  015250  044522  054524  000075
3003  015256  005015  045517  000077   MSGOK:  .ASCIZ  <15><12>/OK?/
3004  015264  042531  100123     000   MYES:   .ASCIZ  /YES/<CRLF>
3005  015271     015  052012  051505   MTSTDRV:        .ASCIZ  <15><12>/TESTING DRIVE /
3006  015276  044524  043516  042040
3007  015304  044522  042526  000040
3008  015312  040440  042116  042040   MANDRV: .ASCIZ  / AND DRIVE /
3009  015320  044522  042526  000040
3010  015326  005015  042524  052123   MTEST:  .ASCIZ  <15><12>/TEST PC?/
3011  015334  050040  037503     000
3012  015341     015  046012  047517   MLOOP:  .ASCIZ  <15><12>/LOOP PC?/
3013  015346  020120  041520  000077
3014  015354  042522  044527  042116   MXRWND: .ASCIZ  /REWIND/
3015  015362     000
3016  015363     127  044522  042524   MXWRIT: .ASCIZ  /WRITE/
3017  015370     000
3018  015371     122  040505  000104   MXREAD: .ASCIZ  /READ/
3019  015376  050123  041501  026505   MXSFFG: .ASCIZ  /SPACE-FWD-FILE-GAP/
3020  015404  053506  026504  044506
3021  015412  042514  043455  050101
3022  015420     000
3023  015421     123  040520  042503   MXSFBG: .ASCIZ  /SPACE-FWD-BLK-GAP/
3024  015426  043055  042127  041055
3025  015434  045514  043455  050101
3026  015442     000
3027  015443     102  041501  026513   MXBSFG: .ASCIZ  /BACK-SPACE-FILE-GAP/
3028  015450  050123  041501  026505
3029  015456  044506  042514  043455
3030  015464  050101     000
3031  015467     102  041501  026513   MXBSBG: .ASCIZ  /BACK-SPACE-BLK-GAP/
3032  015474  050123  041501  026505
3033  015502  046102  026513  040507
3034  015510  000120
3035  015512  051127  052111  026505   MXWFG:  .ASCIZ  /WRITE-FILE-GAP/
3036  015520  044506  042514  043455
3037  015526  050101     000
3038  015531     102  043125  042506   MBUFCK: .ASCIZ  /BUFFER COMPARE/
```

```
 3039  015536  020122  047503  050115
 3040  015544  051101  000105
 3041  015550  046511  051120  050117  EM1:    .ASCIZ  /IMPROPER FLAG SETTING/
 3042  015556  051105  043040  040514
 3043  015564  020107  042523  052124
 3044  015572  047111  000107
 3045  015576  046511  051120  050117  EM2:    .ASCIZ  /IMPROPER FLAG OCCURRED/
 3046  015604  051105  043040  040514
 3047  015612  020107  041517  052503
 3048  015620  051122  042105     000
 3049  015625     115  051511  042523  EM3:    .ASCIZ  /MISSED A FLAG/
 3050  015632  020104  020101  046106
 3051  015640  043501     000
 3052  015643     111  052116  051105  EM4:    .ASCIZ  /INTERRUPT FAILED/
 3053  015650  052522  052120  043040
 3054  015656  044501  042514  000104
 3055  015664  051120  046505  052101  EM5:    .ASCIZ  /PREMATURE READY OCCURRED/
 3056  015672  051125  020105  042522
 3057  015700  042101  020131  041517
 3058  015706  052503  051122  042105
 3059  015714     000
 3060  015715     104  042111  023516  EM6:    .ASCIZ  /DIDN'T STOP IN A FILE GAP/
 3061  015722  020124  052123  050117
 3062  015730  044440  020116  020101
 3063  015736  044506  042514  043440
 3064  015744  050101     000
 3065  015747     104  042111  023516  EM7:    .ASCIZ  /DIDN'T STOP ON CLEAR LEADER/
 3066  015754  020124  052123  050117
 3067  015762  047440  020116  046103
 3068  015770  040505  020122  042514
 3069  015776  042101  051105     000
 3070  016003     102  042101  042040  EM10:   .ASCIZ  /BAD DATA READ/
 3071  016010  052101  020101  042522
 3072  016016  042101     000
 3073  016021     111  046114  043505  EM11:   .ASCIZ  /ILLEGAL BUFFER SIZE/
 3074  016026  046101  041040  043125
 3075  016034  042506  020122  044523
 3076  016042  042532     000
 3077  016045     103  041522  042440  EM12:   .ASCIZ  /CRC ERROR/
 3078  016052  051122  051117     000
 3079  016057     123  047510  052122  EM13:   .ASCIZ  /SHORT RECORD/
 3080  016064  051040  041505  051117
 3081  016072  000104
 3082  016074  040502  020104  047111  EM14:   .ASCIZ  /BAD INTERRUPT/
 3083  016102  042524  051122  050125
 3084  016110  000124
 3085  016112  005015                  DH1:    .ASCII  <15><12>
 3086  016114  042524  052123  020040          .ASCII  /TEST    ERROR/<15><12>
 3087  016122  020040  051105  047522
 3088  016130  006522     012
 3089  016133     120  020103  020040          .ASCIZ  /PC      PC      TACS    TADB/<15><12>
 3090  016140  020040  050040  020103
 3091  016146  020040  020040  052040
 3092  016154  041501  020123  020040
 3093  016162  052040  042101  006502
 3094  016170  000012
```

```
3095  016172  005015                    DH2:    .ASCII  <15><12>
3096  016174  042524  052123  020040            .ASCII  /TEST   ERROR          EXPT'D  RCV'D   BYTE/<15><12>
3097  016202  020040  051105  047522
3098  016210  020122  020040  020040
3099  016216  020040  020040  020040
3100  016224  054105  052120  042047
3101  016232  020040  041522  023526
3102  016240  020104  020040  054502
3103  016246  042524  005015
3104  016252  041520  020040  020040          .ASCIZ  /PC     PC      TACS    DATA    DATA    NUMBER/<15><12>
3105  016260  020040  041520  020040
3106  016266  020040  020040  040524
3107  016274  051503  020040  020040
3108  016302  040504  040524  020040
3109  016310  020040  040504  040524
3110  016316  020040  020040  052516
3111  016324  041115  051105  005015
3112  016332     000
3113  016333     015     012            DH3:    .ASCII  <15><12>
3114  016335     124  051505  020124            .ASCII  /TEST   ERROR          WRITE   READ    TU60/<15><12>
3115  016342  020040  042440  051122
3116  016350  051117  020040  020040
3117  016356  020040  020040  020040
3118  016364  053440  044522  042524
3119  016372  020040  051040  040505
3120  016400  020104  020040  052040
3121  016406  033125  006460     012
3122  016413     120  020103  020040          .ASCIZ  /PC     PC      TACS    CRC     CRC     CRC/<15><12>
3123  016420  020040  050040  020103
3124  016426  020040  020040  052040
3125  016434  041501  020123  020040
3126  016442  041440  041522  020040
3127  016450  020040  041440  041522
3128  016456  020040  020040  041440
3129  016464  041522  005015     000
3130  016471     015     012            DH4:    .ASCII  <15><12>
3131  016473     124  051505  020124            .ASCII  /TEST   ERROR                          BYTES/<15><12>
3132  016500  020040  042440  051122
3133  016506  051117  020040  020040
3134  016514  020040  020040  020040
3135  016522  020040  020040  020040
3136  016530  020040  041040  052131
3137  016536  051505  005015
3138  016542  041520  020040  020040          .ASCIZ  /PC     PC      TACS    TADB    LEFT/<15><12>
3139  016550  020040  041520  020040
3140  016556  020040  020040  040524
3141  016564  051503  020040  020040
3142  016572  040524  041104  020040
3143  016600  020040  042514  052106
3144  016606  005015     000
3145  016611     015     012            DH5:    .ASCII  <15><12>
3146  016613     124  051505  020124            .ASCII  /TEST   ERROR/<15><12>
3147  016620  020040  042440  051122
3148  016626  051117  005015
3149  016632  041520  020040  020040          .ASCIZ  /PC     PC      TACS/<15><12>
3150  016640  020040  041520  020040
```

```
 3151  016646  020040  020040  040524
 3152  016654  051503  005015     000
 3153          016662                    .EVEN
 3154  016662  001116  001220  000000  DT201:  .WORD   $ERRPC,TACSL,0
 3155
 3156  016670  001116  000000          DT202:  .WORD   $ERRPC,0
 3157
 3158  016674  040524  030461  043040  EM201:  .ASCI~  "TA11 FAILED TO RESPOND"
 3159  016702  044501  042514  020104
 3160  016710  047524  051040  051505
 3161  016716  047520  042116     000
 3162  016723     116  020117  051104  EM202:  .ASCIZ  "NO DRIVE AVAILABLE"
 3163  016730  053111  020105  053101
 3164  016736  044501  040514  046102
 3165  016744  000105
 3166  016746  041520  020040  020040  DH201:  .ASCIZ  /FC      TACS/
 3167  016754  020040  040524  051503
 3168  016762     000
 3169  016763     120  000103          DH202:  .ASCIZ  /PC/
 3170                                          .EVEN
```

```
                                        ;BUFFER LINKS
                                        ;THESE LINKS POINT TO THE STARTING ADDRESS OF THE BUFFERS
                                        ;AND ALSO INDICATE HOW MANY BYTES TO READ OR WRITE

017766  017130  000200        RLNK1:  RBUF1,1=8.
017772  017332  000200        RLNK2:  RBUF2,128.
017776  017534  000010        RLNK3:  RBUF3,8.
020002  017736  000040        RLNK4:  RBUF4,32.
020006  020140  000020        RLNK5:  RBUF5,16.
020012  020342  000100        RLNK6:  RBUF6,64.
020016  017130  000000        RLNKX:  RBUF1,0
017032  017056  000200        WLNK1:  WBUF1,128.
017026  017064  000200        WLNK2:  WBUF2,128.
017032  017072  000010        WLNK3:  WBUF3,8.
017036  017100  000040        WLNK4:  WBUF4,32.
017042  017106  000020        WLNK5:  WBUF5,16.
017046  017114  000100        WLNK6:  WBUF6,64.
017052  017122  000000        WLNKX:  WBUFX,0

                                        :BUFFERS

017056  000002                WBUF1:
017056     314                        .BYTE   ^B11001100
017057     063                        .BYTE   ^B00110011
017060     314                        .BYTE   ^B11001100
017061     063                        .BYTE   ^B00110011
017062  000000                WCRC1:  .WORD   0                       ;CRC STORAGE FOR WBUF1

017064  000002                WBUF2:
017064     377                        .BYTE   ^B11111111
017065     000                        .BYTE   ^B00000000
017066     377                        .BYTE   ^B11111111
017067     000                        .BYTE   ^B00000000
017070  000000                WCRC2:  .WORD   0                       ;CRC STORAGE FOR WBUF2

017072  000002                WBUF3:
017072     231                        .BYTE   ^B10011001
017073     146                        .BYTE   ^B01100110
017074     231                        .BYTE   ^B10011001
017075     146                        .BYTE   ^B01100110
017076  000000                WCRC3:  .WORD   0                       ;CRC STORAGE FOR WBUF3

017100  000002                WBUF4:
017100     360                        .BYTE   ^B11110000
017101     017                        .BYTE   ^B00001111
017102     360                        .BYTE   ^B11110000
017103     017                        .BYTE   ^B00001111
017104  000000                WCRC4:  .WORD   0                       ;CRC STORAGE FOR WBUF4

017106  000002                WBUF5:
017106     252                        .BYTE   ^B10101010
017107     125                        .BYTE   ^B01010101
017110     252                        .BYTE   ^B10101010
017111     125                        .BYTE   ^B01010101
017112  000000                WCRC5:  .WORD   0                       ;CRC STORAGE FOR WBUF5
```

```
                                WBUF6:
017114  000001                          .BYTE   ^B11111111
017114  000377                          .BYTE   ^B11111111
017115  000377                          .BYTE   ^B00000000
017116  000000                          .BYTE   ^B00000000
017117  000000                          .BYTE   ^B00000000
017120  000000                  WCRC6:  .WORD   0                        ;CRC STORAGE FOR WBUF6

017122  000004                  WBUFX:
017122  000000                          .BYTE   0
017123  000000                          .BYTE   0
017124  000000                          .BYTE   0
017125  000000                          .BYTE   0
017126  000000                  WCRCX:  .WORD   0

017130  000202                  RBUF1:  .BLKB   130.
017332  000202                  RBUF2:  .BLKB   130.
017534  000202                  RBUF3:  .BLKB   130.
```

```
3246  017736  000202          RBUF4:  .BLKB   130.
3247  020140  000202          RBUF5:  .BLKB   130.
3248  020342  000202          RBUF6:  .BLKB   130.
3249
3250          020544          LASTADDRESS=    .
3251
3252          000001                  .END
```

```
ASKADR  007542            529    1748*
ASKDRV  007432            526    1716*   1739
ASKKEY  001246            312*    616*    688    1637    2136
A20CT   010032           1823*   1856    1867
BADINT  012366            620     680    2288    2308    2314*
BEGINX  002052            237     523*    559     561     609     611
BEGIN1  001450            234     442*
BEGIN2  001502            235     449*
BEGIN3  001510            236     448     451*
BEGIN4  001516            453*
BGNCMN  001522            445     447     450     452     454*
BIT0   = 000001            171*
BIT00  = 000001            161*    171     214    1535    1537    2247
BIT01  = 000002            160*    170     213    1537    2256
BIT02  = 000004            159*    169     212
BIT03  = 000010            158*    168     211
BIT04  = 000020            157*    167     210
BIT05  = 000040            156*    166     209
BIT06  = 000100            155*    165     208
BIT07  = 000200            154*    164     207    1535    1537
BIT08  = 000400            153*    163     206
BIT09  = 001000            152*    162     205    1586    1660
BIT1   = 000002            170*
BIT10  = 002000            151*    204    1644
BIT11  = 004000            150*    203    1593
BIT12  = 010000            149*    202
BIT13  = 020000            148*    201    1651
BIT14  = 040000            147*    200    1568    2256
BIT15  = 100000            146*    199     658
BIT2   = 000004            169*
BIT3   = 000010            168*
BIT4   = 000020            167*
BIT5   = 000040            166*
BIT6   = 000100            165*
BIT7   = 000200            164*
BIT8   = 000400            163*
BIT9   = 001000            162*
BLKCMP= 104423             715     728     741     754     759     772     777     792     805     811     826     841     856
                           872     887     902     908     913     920     927     933     937     950     956     963     970
                           978     982     996    1010    1023    1030    1043    1050    1066    1083    1102    1122    1141
                          1161    1180    1196    1213    1218    1238    1257    1275    1293    1319    1341    1346    1368
                          1374    1379    1384    1389    1417    1422    1453    1459    1466    1473    2400    2458    2915*
BPTVEC= 000014             178*
BSBG   = 104416            726     757     903     809     838     839     868     869     899     900     909     911     916
                           918     923     925     930     931     948     954     959     961     966     969     973     975
                           976    1008    1028    1081    1100    1158    1159    1177    1194    1414    2910*
BSFG   = 104415            739     775     790     884     994    1048    1138    1139    1178    1193    1211    1273    1290
                          1313    1314    1315    1344    1366    1371    1372    1377    1412    1413    1451    1456    1457
                          1462    1463    1464    1469    1470    1471    2500    2909*
BSFGSB  013064             244    2496*   2497    2504
BYTNUM  001206             297*   1919*   1926*   2969
CHKADR  002074             525     528     547*
CHKDRV  002154             585*
CKSWR  = 104406           1558    1625    1659    2902*
CR     = 000015             86*   2600    2610
CRCERR= 040000             200*
```

```
CRC.WD  012232          1958*   1986    2011*   2044    2244    2265*   2272#
CRLF  = 000200           87#     511    2571    2610    2987    3004
CURDRV  001250          313#    632*    634     641*    644*    646     699     1889*
DDISP = 177570           93#     275     482
DH1     016112          338     345     352     359     366     373     380     3085#
DH2     016172          387    3095#
DH201   016746          423    3166#
DH202   016763          428    3169#
DH3     016333          401    3113#
DH4     016471          408    3130#
DH5     016611          394     415    3145#
DISPLA  001142          275#    482*    490*    1607*   1643*
DISPRE  000174          231#    490
DO.CMD  012234         1940    1963    1987    2012    2045    2069    2090    2105    2126    2154    2281#
DO.CRC  012112         1981    2031    2237#
DRIVE   001236          309#    629*   1888    1901*
DRVKEY  001240          310#    311     443*    585     598     619     637     643     681     1717*   1722    1736    1740
                       1900
DRVPNT  001244          311#    599*    619*    628     647*
DSWR  = 177570           92#     274     481
DT1     015054          340     347     354     361     368     375     382     2966#
DT2     015066          389    2969#
DT201   016662          424    3154#
DT202   016670          429    3156#
DT3     015104          403    2973#
DT4     015122          410    2977#
DT5     015136          396     417    2980#
EMTVEC= 000030          191#    465*    466*
EM1     015550          337    3041#
EM10    016003          386    3070#
EM11    016021          393    3073#
EM12    016045          400    3077#
EM13    016057          407    3079#
EM14    016074          414    3082#
EM2     015576          344    3045#
EM201   016674          422    3158#
EM202   016723          427    3162#
EM3     015625          351    3049#
EM4     015643          358    3052#
EM5     015664          365    3055#
EM6     015715          372    3060#
EM7     015747          379    3065#
ERROR = 100000          199#
ERRVEC= 000004          174#     479     480*    491*    547*    553*    1573    1574*   1576*   1579*
EXAM    006474          586      590     595    1522#
EXIT    007532         1735    1740#
FATAL   002716          656      675     694#
FGAP  = 004000          203#    2057    2080    2116
FLAGS   012024         1946    1971    1993    2021    2053    2075    2096    2111    2132    2144    2160    2311#
FUNCTI= 000016          215#    2284    2299
FUNC0 = 000002          213#     215
FUNC1 = 000004          212#     215
FUNC2 = 000010          211#     215
GNS   = ****** U        230      510     697    2895    2896    2897    2898    2900    2902    2903    2904    2905    2906
                       2907    2908    2909    2910    2911    2912    2913    2914    2915    2916
GO    = 000001          214#     669    1941    1964    2013    2070    2091    2106    2127    2155    2284    2294    2343
```

```
                    2371    2393    2425    2451    2481
GTSWR = 104405       505    2513    2526    2537    2900#
HERE    002052       496     498     509     511#
HICNT   012020       662    2180*   2188*   2197#
HT    = 000011        84#   2569    2610
ILBS  = 000020       210#   1630    1988    2046    2284
INT.EN= 000100       208#    669    1941    1964    1988    2013    2046    2070    2091    2106    2127    2155    2284
                    2287    2290
IOTVEC= 000020       179#    463*    464*
ITEMS2  001430       420#
LASTAD= 020544      3250#
LEADER= 020000       201#   1540    2057    2150    2165    2347    2375    2398    2429    2432    2456    2462    2485
                    2501
LF    = 000012        85#   2604    2610
LOCNT   012016      2179*   2186*   2196#
LOOP    007456      1723#   1737
LOOPKE  006624       626*   1566#   1874*   1876*
LVFLAG  012110      2212    2215    2229    2231#
MANDRV  015312       645    3008#
MANUAL  002274       605     613#
MAXCNT  012022       658*    662*    663*    664*    665*    691*   2180    2198#
MBUFCK  015531      1912    3038#
MFUNC   007314      1676#   1912*   1939*   1954*   2001*   2068*   2089*   2104*   2125*   2143*
MLOOP   015341      1861    3012#
MSGASK  015162      1749    2989#
MSGDRV  015146      1716    2987#
MSGID   002024       499     511#    618
MSGOK   015256      1796    3003#
MSGPRI  015202      1772    2993#
MSGVEC  015172      1763    2991#
MTACS   015214      1784    2995#
MTADB   015223      1787    2997#
MTAPRI  015243      1793    3001#
MTAVEC  015232      1790    2999#
MTEST   015326      1848    3010#
MTSTDR  015271       633    3005#
MXBSBG  015467      2089    3031#
MXBSFG  015443      2068    3027#
MXREAD  015371      2001    3018#
MXRWND  015354      2143    3014#
MXSFBG  015421      2125    3023#
MXSFFG  015376      2104    3019#
MXWFG   015512      1939    3035#
MXWRIT  015363      1954    3016#
MYES    015264      1805    3004#
NEXT    007512      1726    1730    1734#
NOTA    007470      1724    1727#
NOTB    007502      1728    1731#
NOTLGL  007524      1721    1732    1738#   1741
OFFLIN= 001000       205#   1531    2057
OK      002306       614     616#
PIRQ  = 177772        91#
PIRQVE= 000240       185#
PR0   = 000000       108#
PR1   = 000040       109#
PR2   = 000100       110#
```

```
PR3    = 000140            111*
PR4    = 000200            112*
PR5    = 000240            113*
PR6    = 000300            114*
PR7    = 000340            115*
PS     = 177776             88*    89    622*   670*   674*   1942*  1965*  1970*  1989*  2014*  2020*  2048*  2071*
                           2092*  2107*  2128*  2156*  2282*
PSW    = 177776             89*
PWRST    002314            618*  2954
PWRVEC= 000024            180*   469*   470*  2921*  2922*  2931*  2937*  2949*  2950*
RBUF1    017130           3176   3182   3243*
RBUF2    017332           3177   3244*
RBUF3    017534           3178   3245*
RBUF4    017736           3179   3246*
RBUF5    020140           3180   3247*
RBUF6    020342           3181   3248*
RCRC0    001210            298*  2006*  2973
RCRC1    001212            299*  2044*  2973
RCRC2    001214            300*  2047*  2052*  2973
RDCHR  = 104407           1561   1797   2723   2903*
RDLIN  = 104410           1718   1849   1862   2764   2904*
RDOCT  = 104411           1750   1764   1773   2905*
RDSUB    012532            240   2387*  2391   2399
READ   = 104414            714    727    740    753    758    771    776    791    804    810    824    825    840
                           855    871    886    901    905    912    919    926    932    936    949    955    962
                           969    977    981    995   1009   1022   1029   1042   1049   1065   1082   1101   1121
                          1140   1160   1179   1195   1212   1217   1237   1256   1274   1292   1317   1340   1345
                          1367   1373   1378   1383   1388   1416   1421   1452   1458   1465   1472   2396   2454
                          2908*
READY  = 000040            209*  1527   1632   2184   2211   2296   2344   2372   2394   2426   2452   2482
RESVEC= 000010            175*
REWIND= 104421             660    696    711    713    724    737    753    752    768    770    786    788    801
                           820    823    835    850    853    865    881    896    946    991   1005   1019   1021
                          1039   1041   1059   1063   1075   1079   1092   1097   1111   1117   1131   1135   1150
                          1155   1170   1174   1189   1205   1209   1227   1234   1247   1253   1266   1269   1284
                          1287   1302   1309   1327   1337   1355   1363   1398   1408   1431   1444   2913*
RGPBLK   012712            242   2445*  2449   2457   2463
RLNKX    017016           2396   2401   2454   2459   2521*  3182*
RLNK1    016766            714    716    727    729    740    742    753    755    758    760    771    773    776
                           778    791    793    804    806    824    840    842    901    903    932    934    949
                           951    977    979    995    997   1009   1011   1022   1024   1042   1044   1082   1084
                          1140   1142   1160   1162   1179   1181   1195   1197   1212   1214   1373   1375   1379
                          1380   1472   1474   2005   3176*
RLNK2    016772            810    812    825    827    855    857    871    873    886    888    905    907    926
                           928    936    938    955    957   1029   1031   1049   1051   1065   1067   1101   1103
                          1217   1219   1317   1319   3177*
RLNK3    016776            912    914    981    983   1237   1239   1256   1258   1274   1276   1292   1294   1383
                          1385   1416   1418   1465   1467   3178*
RLNK4    017002            919    921    962    964   1121   1123   1367   1369   1388   1390   3179*
RLNK5    017006            969    971   1340   1342   1345   1347   1421   1423   1458   1460   3180*
RLNK6    017012           1452   1454   3181*
RWDA     001260            315    316*   1896
RWDB     001262            317*  1893
RWDFLG   001256            315*   657*   684*  1893*  1896*  2148   2152*  2300*
SAVPC    001204            296*  1678*  1911*  1938*  1953*  2000*  2067*  2089*  2103*  2124*  2142*  2314*  2966
                           2969   2973   2977   2980
```

```
SAVR1   010254          1854*   1877    1879*
SELDRV= 104422           655     683     710     723     736     749     767     785     800     819     834     849     864
                         880     895     945     990    1004    1018    1038    1058    1074    1091    1110    1130    1149
                        1169    1188    1204    1226    1246    1265    1283    1301    1326    1254    1397    1430    2342
                        2370    2392    2424    2450    2480    2499    2914*
SERV    012350          1984    2032    2292    2307*
SETBUF  013116          2364    2418    2509*
SETLOO= 104424          1564    2916*
SFBG  = 104420           789     854     870     885    1099    1119    1120    1137    1157    1176    1254    1270    1288
                        1291    1382    2461    2912*
SFFG  = 104417          1064    1080    1098    1118    1136    1156    1175    1210    1216    1235    1236    1255    1310
                        1311    1312    1316    1338    1339    1364    1365    1387    1409    1410    1411    1415    1420
                        1445    1446    1447    1448    1449    1450    2484    2911*
SFFGSB  013010           243    2475*   2479    2488
STACK = 001100            79*    461    2337    2365    2387    2397    2419    2445    2455    2475
STALL   001264           318*    679*    679*   2221    2222
START   002326           617     620*   1508
STKLMT= 177774            90*
SWR     001140           274*    459     481*    483     489*    503     635    1535    1537    1568    1586    1593    1644
                        1651    1656    1660    1885    1902    1920    2060    2181    2217    2285    2511    2515    2517
                        2524    2528    2535    2620    2657*   2929    2942*
SWREG   000176           232*    489     503    2511    2524    2535    2620    2633
SW0   = 000001           143*
SW00  = 000001           133*    143
SW01  = 000002           132*    142
SW02  = 000004           131*    141
SW03  = 000010           130*    140    1920
SW04  = 000020           129*    139    2060
SW05  = 000040           128*    138    2181    2285
SW06  = 000100           127*    137    2217
SW07  = 000200           126*    136     635    1685
SW08  = 000400           125*    135
SW09  = 001000           124*    134    1902
SW1   = 000002           142*
SW10  = 002000           123*
SW11  = 004000           122*
SW12  = 010000           121*
SW13  = 020000           120*
SW14  = 040000           119*
SW15  = 100000           118*
SW2   = 000004           141*
SW3   = 000010           140*
SW4   = 000020           139*
SW5   = 000040           138*
SW6   = 000100           137*
SW7   = 000200           136*
SW8   = 000400           135*
SW9   = 001000           134*
TACSH   001222           304*   1758*   1759*
TACSL   001220           303*    549     623    1522    1755*   1758    1785    2338    2366    2388    2420    2446    2476
                        3154
TADBH   001226           306*   1761*   1762*
TADBL   001224           305*    624    1757*   1761    1788    2339    2367    2389    2421    2447    2477
TAPRIO  001234           308*    621    1783*   1794
TAVEC   001230           307*    446     620*    621*    668*    680*   1769*   1771*   1791   1984*   2032*   2288*   2292*
                        2308*
```

```
TBITVE= 000014   176*
TIMERR= 002000   204*  2057
TKVEC = 000060   183*
TPVEC = 000064   184*
TRAPVE= 000034   182*   467*   468*
TRTVEC= 000014   177*
TR.REQ= 000200   207*  2184  2211
TST1    002776   650    689    708*
TST10   003356   799    817*
TST11   003422   818    832*
TST12   003464   833    847*
TST13   003526   848    862*
TST14   003572   863    878*
TST15   003634   879    893*
TST16   004026   994    943*
TST17   004204   944    988*
TST2    003032   709    721*
TST20   004242   989   1002*
TST21   004300  1003   1016*
TST22   004356  1017   1036*
TST23   004434  1037   1056*
TST24   004500  1057   1072*
TST25   004546  1073   1089*
TST26   004622  1090   1108*
TST27   004702  1109   1128*
TST3    003066   722    734*
TST30   004754  1129   1147*
TST31   005032  1148   1167*
TST32   005104  1168   1186*
TST33   005150  1187   1202*
TST34   005232  1203   1224*
TST35   005310  1225   1244*
TST36   005364  1245   1263*
TST37   005436  1264   1281*
TST4    003122   735    747*
TST40   005510  1282   1299*
TST41   005600  1300   1324*
TST42   005706  1325   1352*
TST43   006056  1353   1395*
TST44   006176  1396   1428*
TST5    003172   748    765*
TST6    003242   766    783*
TST7    003302   784    798*
TYPE  = 104401   499    618    633    634    645    646    695    699   1498   1646   1654   1674   1675
                1677   1687   1691   1696   1704   1716   1738   1749   1763   1772   1794   1787   1790
  .             1793   1796   1803   1805   1848   1861   2574   2631   2632   2635   2648   2659   2679
                2727   2730   2734   2847   2895*  2951
TYPERR  007306  1653   1674*
TYPOC = 104402  1701   1786   1789   1792   1795   2634   2896*
TYPON = 104404  2898*
TYPOS = 104403  2897*
T.BLKC  010372  1909*  2915
T.BSBG  011376  2088*  2910
T.BSFG  011314  2067*  2909
T.READ  011004  2000*  2908
T.RWND  011610  2142*  2913
```

```
T.SELD  010256          1884*   2914
T.SETL  010116          1847*   1858    2916
T.SFBG  011530          2124*   2912
T.SFFG  011446          2103*   2911
T.WFG   010E14          1938*   2906
T.WRIT  010564          1953*   2907
UNIT  = 000400           206*   1526    1892    1895
WAITFL  011726          1943    1966    1990    2016    2049    2072    2093    2108    2129    2157    2178*
WAITKE  001254           314*   196°*   2019*   2183*   2192    2289*   2307*
WBUFX   017122          2529    3189    3235*
WBUF1   017056          3183    3193*
WBUF2   017064          3184    3200*
WBUF3   017072          3185    3207*
WBUF4   017100          3186    3214*
WBUF5   017106          3187    3221*
WBUF6   017114          3188    3228*
WCRCX   017126          3240*
WCRC1   017062          3198*
WCRC2   017070          3205*
WCRC3   017076          3212*
WCRC4   017104          3219*
WCRC5   017112          3226*
WCRC6   017120          3233*
WFG   = 104412           661     687     993    1007    1026    1046    1061    1077    1094    1113    1133    1152    1172
                        1191    1207    1229    1231    1233    1250    1252    1271    1304    1306    1308    1330    1333
                        1336    1359    1362    1401    1404    1407    1433    1435    1437    1439    1441    1443    2346
                        2428    2906*
WFGSUB  012374           238    2337*   2341    2350
WGBLOP  012624          2419*   2423    2435
WGPBLK  012620           241    2418*
WLNKX   017052          2374    2402    2431    2460    2522*   3189*
WLNK1   017022           712     717     725     730     738     743     751     756     761     769     774     779     787
                         794     802     807     821     836     843     851     866     882     897     904     935     947
                         952     980     992     998    1006    1012    1020    1025    1040    1045    1060    1076    1085
                        1093    1112    1132    1143    1151    1163    1171    1182    1190    1198    1206    1215    1228
                        1248    1267    1285    1303    1328    1356    1376    1381    1399    1432    1475    2005    3183*
WLNK2   017026           808     813     822     828     837     852     858     867     874     893     889     898     908
                         924     929     939     953     958    1027    1032    1047    1052    1062    1068    1078    1095
                        1104    1114    1134    1153    1173    1192    1208    1220    1230    1249    1268    1286    1305
                        1320    1329    1357    1400    1434    3184*
WLNK3   017032           910     915     974     984    1096    1115    1154    1232    1240    1251    1259    1272    1277
                        1289    1295    1307    1331    1358    1386    1402    1419    1436    1468    3185*
WLNK4   017036           917     922     960     965    1116    1124    1332    1360    1370    1391    1403    1438    3186*
WLNK5   017042           967     972    1334    1343    1348    1361    1405    1424    1440    1461    3187*
WLNK6   017046          1335    1406    1442    1455    3188*
WLOOP   012454          2365*   2369    2378
WRITE = 104413           712     725     738     751     769     787     802     808     821     822     836     837     851
                         852     866     867     882     883     897     898     910     917     924     947     953     960
                         967     974     992    1006    1020    1027    1040    1047    1060    1062    1076    1078    1093
                        1095    1096    1112    1114    1115    1116    1132    1134    1151    1153    1154    1171    1173
                        1190    1192    1206    1208    1228    1230    1232    1248    1249    1251    1267    1268    1272
                        1285    1286    1289    1303    1305    1307    1328    1329    1331    1332    1334    1335    1356
                        1357    1358    1360    1361    1399    1400    1402    1403    1405    1406    1432    1434    1436
                        1438    1440    1442    2374    2431    2907*
WRTLOC= 010000           202*   1533    2057
WRTSUB  012450           239    2364*
```

```
XBSBG = 000010            193*   2091
XBSFG = 000006            192*   2070
XREAD = 000004            191*   2013   2046
XRWND = 000016            196*    659    685   2155   2343   2371   2393   2425   2451   2481
XSFBG = 000014            195*   2127
XSFFG = 000012            194*   2106
XWFG  = 000000            189*    669   1941
XWRITE= 000002            190*   1964   1988
$AUTOB  001134            271*    507*  2628   2749
$BDADR  001122            266*
$BDDAT  001126            268*   1932*  2969
$BELL   001174            291*   1646   1672
$CHARC  013454           2576*   2586*  2593   2602*  2607*
$CKSWR  013460           2620*   2902
$CMTAG  001100            254*    456    457    465    471    472    473
$CM1  = 000002            286*    287*   288*
$CM2  = 000004            286*    287*   288*
$CM3  = 000002            284*    286
$CM4  = 000001            288*    289*
$CNTLG  014205           2631    2744*
$CNTLU  014200           2648    2743*
$CRLF   001201            293*   1654   1672   1674   1677   1696   2575   2610   2659   2743
$DOAGN  006452           1494    1500   1506*
$ENDAD  006442            497     562    612    702   1502*  1667
$ENDCT  006422            471    1496*
$ENDMG  006461           1498    1510*
$ENULL  006456           1509*
$EOP    006366           1429    1486*
$EOPCT  006414            471*   1493*  1497
$ERFLG  001103            257*   1549   1582   1584   1590*  1610   1641*  1672   1904*
$ERMAX  001115            263*    474*  1584   1606*  1610
$ERROR  007056            465    1624*
$ERRPC  001116            264*   1648*  1649*  1650   1672   2966   2969   2973   2977   2980   3154   3156
$ERRTB  001270            333*   1685
$ERTTL  001112            261*   1647*  1672
$ESCAP  001172            290*    473*   656*   709*   722*   735*   748*   766*   784*   799*   818*   833*   848*
                          863*    879*   894*   944*   989*  1003*  1017*  1037*  1057*  1073*  1090*  1109*  1129*
                         1148*   1168*  1187*  1203*  1225*  1245*  1264*  1282*  1300*  1325*  1353*  1396*  1429*
                         1605*   1639*  1663   1665   1672   2036   2037*  2041*  2498*
$FILLC  001156            282*   2579   2610
$FILLS  001155            281*   2610
$GDADR  001120            265*
$GDDAT  001124            267*   1934*  2969
$GET42  006432           1499*
$GTSWR  013530           2632*   2900
$HD   = 000000             12
$HIOCT  014334           2781*   2786*
$ICNT   001104            258*   1597*  1598   1600*  1610
$ILLUP  015036           2921    2937   2956*
$INTAG  001135            272*   2660   2749
$ITEMB  001114            262*   1650*  1672   1680
$LF     001202            294*   1672   2610   2734   2743
$LPADR  001106            259*    475*   648*  1588*  1603*  1608   1610   1870*  1872*
$LPERR  001110            260*    476*  1588   1604*  1610   1662   1873*  2341*  2369*  2391*  2423*  2449*  2479*
                         2497*
$MAIL = ****** U           493     503   1603   1656   2563
```

```
$MNEW   014223          2635    2747#
$MSWR   014212          2632    2745#
$MXCNT  001216           302#    627#    639#   1601    1610
$NULL   001154           280#   2581    2610
$NWTST= 000001           705#    718#    731#    744#    762#    780#    795#    814#    829#    844#    859#    875#    890#
                         940#    985#    999#   1013#   1033#   1053#   1069#   1086#   1105#   1125#   1144#   1164#   1183#
                        1199#   1221#   1241#   1260#   1278#   1296#   1321#   1349#   1392#   1425#
$OCNT   014560          2819#   2848#   2861#
$OMODE  014562          2814#   2818#   2823    2826#   2837#   2863#
$OVER   007042          1567    1569    1589    1599    1607#
$PASS   001100           255#    649    1490#   1491#   1509    1595    1610
$PCWER  015044          2952    2959#
$PWRAD  015032          2954#
$PWRDN  014672           469    2921#   2949
$PWRMG  015026          2952#
$PWRUP  014744          2931    2937#
$QUES   001200           292#   1672    1738    1803    2610    2678    2727    2743
$RDCHR  013742          2691#   2903
$RDDEC= ****** U        2906
$RDLIN  014062          2719#   2904
$RDOCT  014234          2759#   2905
$RDSZ = 000010          2712#
$REGAD  001160           284#
$REG0   001162           286#   1626#   1635    2966    2969    2973    2977    2980
$REG1   001164           287#   1627#   2966    2977
$RTNAD  006454          1508#
$R2A  = ****** U        2906
$SAVRE= ****** U        2906
$SAVR6  015042          2930#   2938    2939#   2940#   2958#
$SCOPE  006600           463    1557#
$SETUP= 000137           442#    462     463     465     467     469     471     472     473     475     497     500    1488
                        1558    1625    1659    1667    2615    2749
$SS   = 000000           705#
$STUP = 177777           442#
$SVLAD  007014          1577    1602#
$SWR  = 167000             2#      12      62      63      64      65      66      67      68     289     290     291     472
                         473     475     476     709     722     735     748     766     784     799     818     833     848
                         863     879     894     944     989    1003    1017    1037    1057    1073    1090    1109    1129
                        1148    1168    1187    1203    1225    1245    1264    1282    1300    1325    1353    1396    1429
                        1481    1489    1501    1507    1509    1550    1551    1552    1553    1554    1568    1580    1592
                        1583    1584    1591    1592    1593    1604    1607    1610    1616    1617    1618    1619    1620
                        1644    1651    1656    1660    1672    1902    2955
$SWRMK= 000000          1554
$TIMES  001170           289#    472#   1489#   1591#   1598    1601#   1610
$TKB    001146           277#   2613    2624    2641    2695    2701
$TKS    001144           276#   1559    2613    2622    2638    2662#   2693    2699
$TMPO   001166           288#   2034#   2977
$TN   = 000045             2#      12     705     709#    718     722#    731     735#    744     748#    762     766#    780
                         784#    795     799#    814     818#    829     833#    844     948#    859     863#    875     879#
                         890     894#    940     944#    985     989#    999    1003#   1013    1017#   1033    1037#   1053
                        1057#   1069    1073#   1086    1090#   1105    1109#   1125    1129#   1144    1148#   1164    1168#
                        1183    1187#   1199    1203#   1221    1225#   1241    1245#   1260    1264#   1279    1282#   1296
                        1300#   1321    1325#   1349    1353#   1392    1396#   1425    1429#
$TPB    001152           279#   2599#   2610
$TPFLG  001157           283#   2557    2610
$TPS    001150           278#   2597    2610
```

# N08



```
$TRAP   014564            467    2872#
$TRAP2  014606           2883#   2894
$TRP  = 000025           2887#   2896#   2897#   2898#   2899#   2900    2901#   2902    2903#   2904#   2905#   2906#   2907#
                         2908#   2909#   2910#   2911#   2912#   2913#   2914#   2915#   2916#   2917#
$TRPAD  014620           2877    2894#
$TSTNM  001102            256#    625*   1488*   1549    1602*   1607    1610    1643    1672
$TTYIN  014170           2720    2721    2738    2742#
$TYPBN  ****** U         2899
$TYPDS=  ***** U         2899
$TYPE   013240           2557#   2887    2895
$TYPEC  013410           2578    2585    2592    2597#   2598    2664
$TYPEX  013456           2603    2605    2608#
$TYPOC  014362           2817#   2896
$TYPON  014376           2816    2819#   2898
$TYPOS  014336           2812#   2897
$XTSTR  006640           1571#
$$GET4= 000000           1501#
$OFILL  014561           2913*   2817*   2827    2862#
$40CAT= ****** U         1568    1653
.     = 020544            226#    230#    253#    295     460     475     476     511#   1509    1512    1610    1672    1707#
                         2610    2613    2742#   2743    2749    2933    2957    3153#   3243#   3244#   3245#   3246#   3247#
                         3248#   3250

. ABS.  020544     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 TADC,TADC/SOL/CRF:SYM=DZTADC.NEW
 RUN-TIME: 48 37 3 SECONDS
 RUN-TIME RATIO: 185/90=2.0
 CORE USED:  24K  (47 PAGES)
```

Spooler runtime 14 Seconds, 63 KCS, 388 disk reads, 4 disk writes, 104 pages