IDENTIFICATION

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

PRODUCT NAME: TA11 LOGIC TEST (PART 1)

DATE REVISED: 21 JUNE 76

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: JIM LACEY

CONTENTS

1. ABSTRACT

2. REQUIREMENTS
   - 2.1 EQUIPMENT
   - 2.2 STORAGE
   - 2.3 PRELIMINARY PROGRAMS

3. LOADING PROCEDURE

4. STARTING PROCEDURE
   - 4.1 CONTROL SWITCH SETTINGS
   - 4.2 STARTING ADDRESS
   - 4.3 PROGRAM & OPERATOR ACTION

5. OPERATING PROCEDURE
   - 5.1 OPERATIONAL SWITCH SETTINGS
   - 5.2 SUBROUTINE ABSTRACTS

6. ERRORS

7. RESTRICTIONS

8. MISCELLANEOUS
   - 8.1 EXECUTION TIME
   - 8.2 STACK POINTER
   - 8.3 PASS COUNTER
   - 8.4 ITERATIONS
   - 8.5 SPECIAL REGISTERS

9. PROGRAM DESCRIPTION

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF BASIC LOGIC TESTS
THAT CHECK THE TA11 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

NONE

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
500	LOAD SWITCH REGISTER INTO THE TACS
600	WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
700	READ FROM BOT TO EOT

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
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?
```

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<08>=1...LOOP ON TEST AS PER SW<07:00>
SW<07>=1...LOCK ON CURRENT DRIVE (ONLY VALID
           FOR STARTING ADDRESSES 220 THRU 250).

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

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 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 LABEL OF THE STARTING ADDRESS OF THE SUBROUTINES.

5.2.4.1 TYPE ($TYPE)

ROUTINE TO TYPE AN ASCIZ STRING ON THE TTY

THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

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 WAITREADY (WAIT.ON.READY)

WAIT ON THE "TA11 READY" BIT TO SET

5.2.4.5 WAITXFER (WAIT.FOR.XFER.REQ)

WAIT ON THE "TA11 TRANSFER REQUEST" BIT TO SET

5.2.4.6 $TYPOC

CHANGE A BINARY NUMBER TO OCTAL ASCII AND TYPE IT.

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 $TYPEC

ROUTINE TO TYPE A SINGLE ASCII CHARACTER

5.2.5.2 TYPERR

THIS ROUTINE WILL TYPE THE ERROR MESSAGES

5.2.5.3 SELDRV

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

5.2.5.4 SELADR

THIS ROUTINE WILL ASK THE OPERATOR FOR THE ADDRESSES OF
THE "TACS","TADB" AND VECTOR AND THE PRIORITY TO USE.

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

5.2.6.1 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".

5.2.6.2 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 PROMPED
FOR THE VALUE WITH "SWR=XXX NEW="

5.2.6.3 RDSUB

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

5.2.6.4 WGPBLK

```
WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO ECT
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 PROMPED
        FOR THE VALUE WITH "SWR=XXX NEW="
```

5.2.6.5 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".
```

5.2.6.6 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).
```

5.2.6.7 BSFGSB

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

5.2.7 THE FOLLOWING SUBROUTINES ARE USED BY THE ADJUSTMENT ROUTINES

5.2.7.1 SETBUF

SETUP BLOCK SIZE AND PATTERN

5.2.7.2 WRTBLK

WRITES A BLOCK OF DATA

5.2.7.3 RDBLK

READS A BLOCK OF DATA

5.2.7.4 NXTDRV

CHANGE DRIVE

6. ERRORS

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

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

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

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 AND AT BEGINNING OF TAPE.

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS TAKES APPROXIMATELY 45 SECONDS ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 100 SECONDS.

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, (2000 DECIMAL UNLESS OTHERWISE SPECIFIED WITHIN A TEST), ITERATIONS.

8.5 SPECIAL REGISTERS

REGISTER R3,R4 AND R5 ARE RESERVED FOR THE THE FOLLOWING PURPOSES:
R3 = DRIVE
R4 = TACS
R5 = TADB

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF SMALL INDEPENDENT TESTS THAT CHECK THE TA11 FOR PROPER OPERATION WITH TAPE MOTION KEPT AT A MINIMUM.

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

1. VERIFY THAT THE "CONTROL AND STATUS" (TACS) AND THE "DATA BUFFER" (TADB) REGISTERS RESPOND TO ADDRESSING.
2. TEST THE "TACS" FOR PROPER OPERATION
3. TEST "READY"
4. TEST "CLEAR LEADER ERROR"
5. TEST "TRANSFER REQUEST"
6. TEST THE "TADB" FOR PROPER OPERATION.

```
   12              GENERAL INFORMATION
   60          OPERATIONAL SWITCH SETTINGS
   73          BASIC DEFINITIONS
  183              TA11 DEFINITIONS
  224              STARTING ADDRESSES
  225          TRAP CATCHER
  234          STARTING ADDRESS(ES)
  246              TOGGLE IN ROUTINES
  293              LOAD SWITCH REGISTER INTO TACS
  301              WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
  327              READ FROM BOT TO EOT
  344          COMMON TAGS
  404          ERROR POINTER TABLE
  482              START OF TEST
  507          INITIALIZE THE COMMON TAGS
  545          TYPE PROGRAM NAME
  552          GET VALUE FOR SOFTWARE SWITCH REGISTER
  729          T1      TEST IF TACS EXIST
  753          T2      TEST IF TADB EXIST
  777          T3      TEST FOR LOW BYTE OF TACS
  801          T4      TEST FOR LOW BYTE OF TADB
  825          T5      TEST FOR HIGH BYTE OF TACS
  849          T6      TEST FOR HIGH BYTE OF TADB
  875          T7      TEST INITIAL CONDITION OF TACS BIT14 = 0
  885          T10     TEST INITIAL CONDITION OF TACS BIT11 = 0
  895          T11     TEST INITIAL CONDITION OF TACS BIT10 = 0
  905          T12     TEST INITIAL CONDITION OF TACS BIT08 = 0
  915          T13     TEST INITIAL CONDITION OF TACS BIT07 = 0
  925          T14     TEST INITIAL CONDITION OF TACS BIT06 = 0
  935          T15     TEST INITIAL CONDITION OF TACS BIT04 = 0
  945          T16     TEST INITIAL CONDITION OF TACS BIT03 = 0
  955          T17     TEST INITIAL CONDITION OF TACS BIT02 = 0
  965          T20     TEST INITIAL CONDITION OF TACS BIT01 = 0
  975          T21     TEST INITIAL CONDITION OF TACS BIT00 = 0
  992          T22     TEST BIT08 OF TACS MAY BE "SET" AND "CLEARED"
 1006          T23     TEST BIT08 OF TACS INITIALIZE TO ZERO
 1021          T24     TEST BIT06 OF TACS MAY BE "SET" AND "CLEARED"
 1035          T25     TEST BIT06 OF TACS INITIALIZE TO ZERO
 1050          T26     TEST BIT04 OF TACS MAY BE "SET" AND "CLEARED"
 1064          T27     TEST BIT04 OF TACS INITIALIZE TO ZERO
 1079          T30     TEST BIT03 OF TACS MAY BE "SET" AND "CLEARED"
 1093          T31     TEST BIT03 OF TACS INITIALIZE TO ZERO
 1108          T32     TEST BIT02 OF TACS MAY BE "SET" AND "CLEARED"
 1122          T33     TEST BIT02 OF TACS INITIALIZE TO ZERO
 1137          T34     TEST BIT01 OF TACS MAY BE "SET" AND "CLEARED"
 1151          T35     TEST BIT01 OF TACS INITIALIZE TO ZERO
 1166          T36     COUNT PATTERN TO TACS<04:01>
 1193          T37     TEST INITIAL CONDITION OF TACS BIT05 = 1
 1204          T40     TEST INITIAL CONDITION OF TACS BIT09 = 0
 1215          T41     TEST INITIAL CONDITION OF TACS BIT12 = 0
 1227          T42     TEST INITIAL CONDITION OF TACS BIT13 = 1
 1238          T43     TEST INITIAL CONDITION OF TACS BIT15 = 1
 1257          T44     TEST TACS BIT15 IS READ ONLY
 1278          T45     TEST TACS BIT14 IS READ ONLY
 1299          T46     TEST TACS BIT13 IS READ ONLY
 1320          T47     TEST TACS BIT12 IS READ ONLY
```

1341 T50 TEST TACS BIT11 IS READ ONLY
1362 T51 TEST TACS BIT10 IS READ ONLY
1383 T52 TEST TACS BIT09 IS READ ONLY
1404 T53 TEST TACS BIT07 IS READ ONLY
1425 T54 TEST TACS BIT05 IS READ ONLY
1449 T55 TEST HIGH BYTE SELECTION OF TACS
1461 T56 TEST LOW BYTE SELECTION OF TACS
1473 T57 TEST "OUT LOW" SIGNAL
1487 T60 TEST "OUT HIGH" SIGNAL
1501 T61 TEST "SELECT 00" ISN'T STUCK
1520 T62 TEST "ERROR" FOR "CLEAR LEADER"
1566 T63 TEST THAT "READY" SETS WHEN "REWIND" IS COMPLETED
1592 T64 TEST THAT "READY" SETS WHEN "WRITE FILE GAP" IS COMPLETED
1618 T65 TEST "READY" CLEARS WHEN STARTING A "REWIND" FUNCTION
1634 T66 TEST "READY" CLEARS WHEN STARTING A "WRITE FILE GAP" FUNCTION
1650 T67 TEST THAT "RESET" CANNOT ABORT A "REWIND"
1679 T70 TEST "RESET" WILL ABORT A "WRITE FILE GAP"
1703 T71 TEST "WFG" AND "REWIND" FOR NO ERRORS
1731 T72 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
1753 T73 TEST THAT "CLEAR LEADER" GOES FALSE ON "REWIND"
1780 T74 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FILE GAP FUNCTION"
1810 T75 TEST "READY" CLEARS WHEN STARTING A "WRITE" FUNCTION
1826 T76 TEST "READY" CLEARS WHEN STARTING A "READ" FUNCTION
1842 T77 TEST "READY" CLEARS WHEN STARTING A "BACK SPACE FILE GAP" FUNCTION
1858 T100 TEST "READY" CLEARS WHEN STARTING A "BACK SPACE BLOCK GAP" FUNCTION
1874 T101 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD FILE GAP" FUNCTION
1890 T102 TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD BLOCK GAP" FUNCTION
1906 T103 TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FUNCTION"
1942 T104 TEST FOR "CLEAR LEADER ERROR" FOR "READ FUNCTION"
1972 T105 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE FILE GAP FUNCTION"
2002 T106 TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE BLOCK GAP FUNCTION"
2032 T107 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"
2062 T110 TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD BLOCK GAP FUNCTION"
2092 T111 TEST "GO" BIT IS WRITE ONLY
2106 **********TRANSFER REQUEST**********
2113 T112 TEST TRANSFER REQUEST & ILBS FOR "WRITE"
2157 T113 TEST "TRANSFER REQ" AND "ILBS" FOR "READ"
2194 **********TADB**********
2195 T114 TEST THAT TADB (READ BUFFER) INITIALIZES TO 0'S
2205 T115 TEST TADB (WRITE BUFFER) FOR WRITE ONLY
2214 T116 FLOAT A "1" THROUGH TADB
2241 T117 FLOAT A "0" THROUGH TADB
2268 T120 COUNT PATTERN TO TADB
2295 T121 END OF TEST CODE
2305 END OF PASS ROUTINE
2341 SCOPE HANDLER ROUTINE
2405 ERROR HANDLER ROUTINE
2457 ERROR TYPEOUT ROUTINE
2493 ROUTINE TO WAIT ON THE READY BIT TO SET
2522 ROUTINE TO WAIT ON TRANSFER REQUEST
2551 ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2585 CHANGE TA11 ADDRESSES
2587 ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2651 ********** MANUAL ADJUSTMENT ROUTINES **********
2661 WRITE FILE GAPS FROM "BOT" TO "EOT"
2686 WRITE CONTINUOUS BLOCKS OF DATA

2719 READ CONTINUOUS BLOCKS OF DATA
2742 WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2780 READ A BLOCK OF DATA AND A FILE GAP
2808 SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2835 BACK SPACE FILE GAP
2852 SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2886 WRITE ROUTINE FOR THE MANUAL OPERATIONS
2904 READ ROUTINE FOR THE MANUAL OPERATIONS
2929 ROUTINE TO CHANGE DRIVES
2945 ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
2975 TYPE ROUTINE
3045 READ AN OCTAL NUMBER FROM THE TTY
3083 TTY INPUT ROUTINE
3222 BINARY TO OCTAL (ASCII) AND TYPE
3299 TRAP DECODER
3322 TRAP TABLE
3343 POWER DOWN AND UP ROUTINES

```
                        .TITLE  TA11 BASIC LOGIC TEST (PART 1)  MAINDEC-11-DZTAA-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-C1),MAR 24, 1976.
                        ;*
                        ;;*************************************************************
                        ;;*************************************************************
                        ;;*************************************************************
                        .REM!

                                GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

                        ADDRESS MNEMONIC DESCRIPTION
                        ============================


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


                        TACS REGISTER DESCRIPTION
                        =========================
                                                        INIT
                        BIT     NAME                    STATE   READ AND/OR WRITE?
                        ===     ====                    =====   ==========_=====

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



```
                        ;;*************************************************************
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ======          ====================
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<7:0>
                        ;*         7            LOCK ON CURRENT DRIVE (ONLY VALID WITH MANUAL LOOPING)
                        ;;*************************************************************
                        .SBTTL  BASIC DEFINITIONS

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

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

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

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  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
```



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

```
                                ;TA11 FUNCTIONS
000000                          WFG=    0                       ;WRITE FILE GAP FUNCTION
000002                          WRITE=  2                       ;WRITE FUNCTION
000004                          READ=   4                       ;READ FUNCTION
000006                          BSFG=   6                       ;BACK SPACE FILE GAP FUNCTION
000010                          BSBG=   10                      ;BACK SPACE BLOCK GAP FUNCTION
000012                          SFFG=   12                      ;SPACE FWD FILE GAP FUNCTION
000014                          SFBG=   14                      ;SPACE FWD BLOCK GAP FUNCTION
000016                          REWIND= 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
000003                          DRIVE=  %3                      ;R3 CONTAINS THE DRIVE UNDER TEST
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  001356                JMP     @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  001410                JMP     @#BEGIN2                ;SELECT DRIVE(S) BEFORE STARTING TEST
000210  000137  001416                JMP     @#BEGIN3                ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
000214  000137  001424                JMP     @#BEGIN4                ;SETUP FOR MANUAL LOOPING
000220  000137  012622                JMP     @#WFGSUB                ;WRITE FILE GAP FROM BOT TO EOT
000224  000137  012706                JMP     @#WRTSUB                ;WRITE CONTINUOUS BLOCKS OF DATA
000230  000137  012774                JMP     @#RDSUB                 ;READ CONTINUOUS BLOCKS OF DATA
000234  000137  013054                JMP     @#WGPBLK                ;WRITE FILE GAP AND A BLOCK OF DATA
000240  000137  013156                JMP     @#RGPBLK                ;READ BLOCK OF DATA AND INTO A FILE GAP
000244  000137  013252                JMP     @#SFFGSB                ;SPACE FWD FILE GAP FROM BOT TO EOT
000250  000137  013336                JMP     @#BSFGSB                ;BACK SPACE FILE GAPS
```

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

                                ;THE FOLLOWING ROUTINES CAN BE TOGGLED IN.

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


                                        .REM    !


                                        THE FOLLOWING ROUTINES (LOOP1, LOOP2, & LOOP3) CAN BE TOGGLED
                                        IN WHEN IT IS IMPOSSIBLE TO LOAD THE DIAGNOSTICS

                                        NOTE:   BEFORE USING THESE ROUTINES INSURE THAT R3,R4,& R5
                                                ARE SETUP PROPERLY.

                                        ** NOTE: IF USING SOFTWARE SWITCH REGISTER
                                                LOCATION SWR (=1140) MUST CONTAIN
                                                ADDRESS "SWREG" (=176).
                                                *****  PUT VALUE INTO 176    *****
                                                ** REGISTERS 3,4,&5 MUST BE SETUP **
                                                ** VIA MOVE INSTRUCTIONS **
                                        R3=       0 IF USING DRIVE A
                                                400 IF USING DRIVE B

                                        R4=     TA11 STATUS REG ADDRESS (TACS 177500)

                                        R5=     TA11 DATA BUFFER ADDRESS (TADB 177502)

                                        LOOP1 WILL LOAD THE SWITCH REGISTER INTO THE TACS.

                                        LOOP2 WILL WRITE THE CONTENTS OF THE SWITCH REGISTER
                                        ALL THE WAY TO END-OF-TAPE(EOT).

                                        LOOP3 WILL READ TO EOT. DATA WILL GO TO R0.

                                        NOTE:   LOOP2 AND LOOP3 WILL REWIND WHEN EOT IS REACHED AND
                                                THEN START OVER.
                                !

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

                                        ;LOAD SWITCH REGISTER INTO THE TACS
                                ;;**************************************************************

          000500                        .=500

000500  017714  000434          LOOP1:  MOV     @SWR,@TACS              ;LOAD TACS
000504  000775                  LOOP1   BR      LOOP1                   ;LOOP
```



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

                                        ;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
                                ;;**************************************************************
          000600                        .=600

000600  000005                  LOOP2:  RESET                           ;CLEAR ALL FLAGS
000602  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
000604  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO BOT
000610  032714  000040          1$:     BIT     #READY,@TACS            ;WAIT TILL READY COMES UP
000614  001775                          BEQ     1$
000616  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE
000622  105714                  2$:     TSTB    @TACS                   ;CHECK FOR TRANSFER REQUEST
000624  100003                          BPL     3$                      ;BR IF NOT SET
000626  017715  000306                  MOV     @SWR,@TADB              ;SEND DATA TO TA11
000632  000773                          BR      2$                      ;LOOP
000634  032714  000040          3$:     BIT     #READY,@TACS            ;DID READY SET?
000640  001357                          BNE     LOOP2                   ;START OVER IF YES
000642  000767                          BR      2$                      ;LOOP


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

                                        ;READ FROM BOT TO EOT
                                ;;**************************************************************

          000700                        .=700

000700  000005                  LOOP3:  RESET                           ;CLEAR ALL FLAGS
000702  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
000704  112714  000017                  MOVB    #REWIND!GO,@TACS        ;START A REWIND
000710  032714  000040          1$:     BIT     #READY,@TACS            ;WAIT ON REWIND TO FINISH
000714  001775                          BEQ     1$
000716  112714  000005                  MOVB    #READ!GO,@TACS          ;START A READ
000722  105714                  2$:     TSTB    @TACS                   ;CHECK TRANSFER REQ
000724  100002                          BPL     3$                      ;BR IF NOT SET
000726  011500                          MOV     @TADB,R0                ;PICKUP THE DATA
000730  000774                          BR      2$                      ;LOOP
000732  032714  000040          3$:     BIT     #READY,@TACS            ;CHECK READY
000736  001360                          BNE     LOOP3                   ;START OVER
000740  000770                          BR      2$                      ;LOOP
```

```
                                .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                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001170  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001172  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001176     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001177     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001200  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*************************************************************
001202  000000                  SAVPC:  .WORD   0                       ;STORAGE FOR THE PC
001204  000000                  SAVPS:  .WORD   0                       ;STORAGE FOR THE PS
001206  177500                  TACSL:  177500                          ;LOW BYTE ADDRESS OF TACS
001210  177501                  TACSH:  177501                          ;HIGH BYTE ADDRESS OF TACS
001212  177502                  TADBL:  177502                          ;LOW BYTE ADDRESS OF TADB
001214  177503                  TADBH:  177503                          ;HIGH BYTE ADDRESS OF TADB
001216  000260  000262          TAVEC:  260,262                         ;TA11 VECTOR ADDRESS
001222  000300                  TAPRIO: 300                             ;TA11 BR LEVEL 6
```



```
001224  000000  000000          DRVKEY: 0,0                             ;DRIVE SELECT KEY:
001230  001224                  DRVPNT: DRVKEY
001232  000000                  ASKKEY: 0
001234  000000                  CURDRV: 0                               ;CURRENT DRIVE BEING TESTED
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


001236                          $ERRTB:

                                ;NOTE:  ALL NUMBERS ARE TYPED AS 6-DIGIT OCTAL NUMBERS

                                ;ITEM   1
015712                                  EM1     ;STATUS PROBLEM
016124                                  DH1     ;PC     TACS
016272                                  DT1     ;$ERRPC $REG0
000000                                  0

                                ;ITEM   2
015731                                  EM2     ;READY FAILED TO SET
016141                                  DH2     ;PC     TACS    WAIT ADDRESS
016300                                  DT2     ;$ERRPC $REG0   SAVPC
000000                                  0

                                ;ITEM   3
015757                                  EM3     ;TRANSFER REQUEST FAILED TO SET
016141                                  DH2     ;PC     TACS    WAIT ADDRESS
016300                                  DT2     ;$ERRPC $REG0   SAVPC
000000                                  0

                                ;ITEM   4
016020                                  EM4     ;THE WRONG FLAG SET
016141                                  DH2     ;PC     TACS    WAIT ADDRESS
016300                                  DT2     ;$ERRPC $REG0   SAVPC
000000                                  0

                                ;ITEM   5
016043                                  EM5     ;COUNT PATTERN FAILED
016176                                  DH5     ;PC     TACS    EXPECT  RCV'D
016310                                  DT5     ;$ERRPC $REG0   $GDDAT  $BDDAT
000000                                  0

                                ;ITEM   6
016070                                  EM6     ;DATA PROBLEM
016234                                  DH6     ;PC     TADB
016322                                  DT6     ;$ERRPC $REG1
000000                                  0

                                ;ITEM   7
016070                                  EM6     ;DATA PROBLEM
```



```
016176                                  DH5     ;PC     TACS    EXPECT  RCV'D
016310                                  DT5     ;$ERRPC $REG0   $GDDAT  $BDDAT
000000                                  0

                                ;ITEM   10
016105                                  EM10    ;ADDRESS FAILED
016251                                  DH10    ;PC     ADDRESS
016330                                  DT10    ;$ERRPC $BDADR
000000                                  0

001336                          ITEMS2:         ;ITEMS 201-202

016350                                  EM201   ;TA11 FAILED TO RESPOND
016422                                  DH201   ;PC     TACS
016336                                  DT201   ;$ERRPC TACS
000000                                  0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS

016377                                  EM202   ;NO DRIVES AVAILABLE
016437                                  DH202   ;PC
016344                                  DT202   ;$ERRPC
000000                                  0       ;
```

```
                                        ;////////////////////////////////////////////////////////////////////
                                        ;////////////////////////////////////////////////////////////////////
                                        ;;****************************************************************

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

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

001356  005005                  BEGIN1: CLR     R5                      ;NORMAL START
001360  012737  041101  001224          MOV     #"AB,@#DRVKEY
001366  122737  000005  000041          CMPB    #5,@#41                 ;CASSETTE DDP?
001374  001015                          BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001376  022737  000260  001216          CMP     #260,@#TAVEC            ;STANDARD VECTOR?
001404  001011                          BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001406  000403                          BR      BEGIN3                  ;GET DRIVES AND ADDRESSES
001410  012705  000001          BEGIN2: MOV     #1,R5                   ;ASK FOR DRIVES FLAG
001414  000405                          BR      BGNCMN                  ;BEGIN COMMON CODE
001416  012705  000002          BEGIN3: MOV     #2,R5                   ;ASK FOR DRIVES AND ADDRESSES
001422  000402                          BR      BGNCMN
001424  012705  000003          BEGIN4: MOV     #3,R5
001430                          BGNCMN:
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001430  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001434  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001436  022706  001140                  CMP     #SWR,R6 ;;DONE?
001442  001374                          BNE     .-6             ;;LOOP BACK IF NO
001444  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001450  012737  011232  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001456  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001464  012737  011504  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001472  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001500  012737  015322  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001506  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001514  012737  015406  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001522  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001530  016767  007424  007414          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
001536  005067  177424                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001542  005067  177422                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001546  112767  000001  177341          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001554  012767  001554  177324          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001562  012767  001562  177320          MOV     #.,$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.
001570  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001574  012737  001630  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001602  012767  177570  177330          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001610  012767  177570  177324          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001616  022777  177777  177314          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001624  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001626  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
```



```
001630  012716  001636          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001634  000002                          RTI
001636  012767  000176  177274  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001644  012767  000174  177270          MOV     #DISPREG,DISPLAY
001652  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001656  005227  177777                  INC     #-1             ;;FIRST TIME?
001662  001036                          BNE     HERE            ;;BRANCH IF NO
001664  022737  011200  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
001672  001432                          BEQ     HERE            ;;BRANCH IF YES
001674  104401  001732                  TYPE    ,MSGID          ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001700  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001704  001006                          BNE     67$             ;;BRANCH IF YES
001706  026727  177226  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001714  001005                          BNE     68$             ;;BRANCH IF NO
001716  104405                          GTSWR                   ;;GET SOFT-SWR SETTINGS
001720  000403                          BR      68$
001722  112767  000001  177204  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
001730                          68$:
001730  000413                          BR      HERE            ;;GET OVER THE ASCIZ
                                ;;MSGID:        .ASCIZ  <CRLF>/MAINDEC-11-DZTAA-C/<CRLF>
001760                          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
                                        ;
                                        ;;*****************************************************************
001760  010504                BEGINX: MOV     R5,R4                   ;COPY R5
001762  005305                        DEC     R5                      ;ASK FOR DRIVES?
001764  002406                        BLT     CHKADR                  ;BR IF NO
001766  004737  012222                JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
001772  005305                        DEC     R5                      ;ASK FOR ADDRESSES?
001774  002402                        BLT     CHKADR                  ;BR IF NO
001776  004737  012332                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 $ENDAD
                                        ;
                                        ;II.    TIMEOUT DIDN'T OCCUR
                                        ;       A. CONTINUE
                                        ;
                                        ;;*****************************************************************
002002  012737  002020  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE OF TIMEOUTS
002010  005000                        CLR     R0                      ;USE AS A SWITCH
002012  005777  177170                TST     @TACSL                  ;SEE IF TA11 RESPONDS
002016  000402                        BR      2$                      ;BR IF NO TIMEOUT
002020  005200                1$:     INC     R0                      ;COME HERE ON TIMEOUT
002022  022626                        CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
002024  012737  000006  000004  2$:    MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
002032  005700                        TST     R0                      ;DID A TIMEOUT OCCUR?
002034  001412                        BEQ     3$                      ;BR IF NO
002036  104201                        ERROR   201                     ;TA11 FAILED TO RESPOND
002040  012705  000002                MOV     #2,P5                   ;DRIVES & ADDRESSES
002044  005704                        TST     R4                      ;OPERATOR INPUTS?
002046  001344                        BNE     BEGINX                  ;BR IF YES
002050  013700  000042                MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002054  001741                        BEQ     BEGINX                  ;BR IF NO MONITOR
002056  000137  011200                JMP     @#$ENDAD                ;GO TO END
002062                          3$:
```



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

                                        ;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 $ENDAD
                                        ;
                                        ;II.    BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
                                        ;       A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
                                        ;       B. CONTINUE IN PROGRAM
                                        ;
                                        ;III.   DESIRED DRIVE(S) CAN BE TESTED
                                        ;       A. CONTINUE IN PROGRAM
                                        ;
                                        ;;*****************************************************************
002062  012700  001224          CHKDRV: MOV     #DRVKEY,R0              ;PICKUP ADDRESS OF ASCII DRIVE KEY
002066  004737  013672                  JSR     PC,@#EXAM               ;GO EXAMINE FIRST DRIVE
002072  000410                        BR      1$                      ;OK TO TEST---GO CHECK NEXT
002074  116010  000001                MOVB    1(R0),(R0)              ;REPLACE 1ST WITH 2ND
002100  001412                        BEQ     2$                      ;BR IF NO 2ND DRIVE SELECTED
002102  004737  013672                JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002106  000407                        BR      2$                      ;OK TO TEST
002110  005010                        CLR     (R0)                    ;CLEAR DRIVE CODES
002112  000405                        BR      2$
002114  005200                1$:     INC     R0                      ;POINT TO 2ND
002116  004737  013672                JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002122  000401                        BR      2$                      ;OK TO TEST
002124  105010                        CLRB    (R0)                    ;CLEAR 2ND
002126  012700  001224          2$:     MOV     #DRVKEY,R0              ;RESET ADDRESS POINTERS
002132  010037  001230                MOV     R0,@#DRVPNT
002136  121060  000001                CMPB    (R0),1(R0)              ;1ST = 2ND?
002142  001002                        BNE     3$                      ;BR IF NO
002144  105060  000001                CLRB    1(R0)                   ;YES---CLEAR 2ND
002150  005710                3$:     TST     (R0)                    ;ANY DRIVES?
002152  001401                        BEQ     5$                      ;BR IF NO
002154  000412                        BR      MANUAL
002156  104202                5$:     ERROR   202                     ;NO DRIVES AVAILABLE
002160  012705  000002                MOV     #2,R5                   ;DRIVES & ADDRESS
002164  005704                        TST     R4                      ;OPERATOR INPUTS?
002166  001274                        BNE     BEGINX                  ;BR IF YES
002170  013700  000042                MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002174  001671                        BEQ     BEGINX                  ;NO MONITOR
002176  000137  011200                JMP     @#$ENDAD                ;GO TO END
002202  020427  000003          MANUAL: CMP R4,#3
002206  001002                        BNE     OK
002210  016704  175563                MOV     -1,R4
002214  010437  001232          OK:     MOV     R4,@#ASKKEY
002220  000405                        BR START
002222  104401  001732          PWRST:  TYPE    ,MSGID                  ;POWER FAIL RESTART
002226  012737  001224  001230        MOV     #DRVKEY,@#DRVPNT
```

```
002234  013777  001220  176754  START:  MOV     @#TAVEC+2,@TAVEC        ;SETUP TA11 TRAP VECTOR
002242  005077  176752                  CLR     @TAVEC+2
002246  012737  000340  177776          MOV     #340,@#PS               ;LOCKOUT ALL I/O INT
002254  013704  001206                  MOV     @#TACSL,TACS            ;SETUP TACS
002260  013705  001212                  MOV     @#TADBL,TADB            ;SETUP TADB
002264  005737  001100                  TST     @#$PASS                 ;IF FIRST PASS SETUP FOR EXTRA LONG WAIT LOOPS
002270  001024                          BNE     1$                      ;OTHERWISE USE OLD VALUES
002272  012737  077777  012116          MOV     #^CBIT15,@#MAXCNT
002300  005737  000042                  TST     @#42                    ;UNDER MONITOR CONTROL?
002304  001016                          BNE     1$                      ;SKIP TYPEOUT IF YES
002306  104401  002314                  TYPE    ,65$            ;;TYPE ASCIZ STRING
002312  000413                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/DO A MANUAL REWIND/
002342                          64$:
002342  005037  001102          1$:     CLR     @#$TSTNM                ;ZERO THE TEST NUMBER
002346  005003                          CLR     DRIVE                   ;SET DRIVE TO "A"
002350  013701  001230                  MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
002354  121127  000101                  CMPB    (R1),#'A                ;IS IT DRIVE "A"?
002360  001402                          BEQ     TDRV                    ;BR IF YES
002362  012703  000400                  MOV     #UNIT,DRIVE             ;SET DRIVE TO "B"
002366                          TDRV:
002366  104401  002374                  TYPE    ,65$            ;;TYPE ASCIZ STRING
002372  000411                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>*TESTING DRIVE *
002416                          64$:
002416  112167  176612                  MOVB    (R1)+,CURDRV            ;SETUP TO TYPE CURRENT DRIVE
002422  104401  001234                  TYPE    ,CURDRV
002426  104401  001177                  TYPE    ,$CRLF                  ;TYPE A  CR & LF
002432  105711                          TSTB    (R1)                    ;LAST DRIVE BEEN SELECTED
002434  001002                          BNE     1$                      ;BR IF NO
002436  012701  001224                  MOV     #DRVKEY,R1              ;RESET DRIVE POINTER
002442  010137  001230          1$:     MOV     R1,@#DRVPNT             ;SAVE DRIVE POINTER FOR NEXT TIME
002446  005737  001232                  TST     @#ASKKEY                ;GO START TESTING IF NO MANUAL
002452  002007                          BGE     2$                      ;       OPERATIONS REQUESTED
002454  005000                          CLR     R0
002456  000000                          HALT                            ;GIVE CONTROL TO THE OPERATOR
002460  022767  000176  176452          CMP     #SWREG,SWR              ;SOFTWARE SWITCH REG.?
002466  001001                          BNE     20$                     ;NO - LET HIM GO
002470  104405                          GTSWR                           ;GET NEW SWR VALUE
002472                          20$:
                                ;THIS CODE IS FOR ACT11 & DDP
002472  005737  000042          2$:     TST     @#42                    ;IS THERE A MONITOR?
002476  001406                          BEQ     TST1                    ;GO START TESTING IF NO
002500  010314                          MOV     DRIVE,@TACS             ;IF YES SELECT DRIVE
002502  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO BOT
002506  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY
002512  001775                          BEQ     3$                      ;FALL THRU IF READY=1
```



```
                                ;////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS WILL CHECK THE TA11
                                ;STATUS (TACS) AND DATA BUFFER (TADB) REGISTERS
                                ;TO INSURE THAT THEY RESPOND TO ADDRESSING.
                                ;////////////////////////////////////////////////////////////////////

                                ;;**************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TACS RESPONDS TO ADDRESSING
                                ;;**************************************************************
                                ;*TEST 1         TEST IF TACS EXIST
                                ;;**************************************************************
002514  000004          TST1:   SCOPE
002516  012767  000001  176442          MOV     #1,$TIMES       ;;DO 1 ITERATION
002524  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
002530  012737  002542  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
002536  005014                          CLR     @TACS                   ;TACS ARE YOU THERE?
002540  000420                          BR      2$                      ;YES -- NO TRAP
002542  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
002550  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
002552  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
002556  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
002564  010437  001122                  MOV     TACS,@#$BDADR           ;GET THE FAILING ADDRESS
002570  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TACS REGISTER
002572  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
002576  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
002602  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER

                                ;;**************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TADB RESPONDS TO ADDRESSING
                                ;;**************************************************************
                                ;*TEST 2         TEST IF TADB EXIST
                                ;;**************************************************************
002610  000004          TST2:   SCOPE
002612  012767  000001  176346          MOV     #1,$TIMES       ;;DO 1 ITERATION
002620  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
002624  012737  002636  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
002632  005015                          CLR     @TADB                   ;TADB ARE YOU THERE?
002634  000420                          BR      2$                      ;YES -- NO TRAP
002636  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
002644  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
002646  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
002652  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
002660  010537  001122                  MOV     TADB,@#$BDADR           ;GET THE FAILING ADDRESS
002664  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TADB REGISTER
002666  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
002672  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
002676  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
```

```
                                        ;;*************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TACS RESPONDS TO ADDRESSING
                                        ;;*************************************************************
                                        ;*TEST 3         TEST FOR LOW BYTE OF TACS
                                        ;;*************************************************************
002704  000004                  TST3:   SCOPE
002706  012767  000001  176252          MOV     #1,$TIMES       ;;DO 1 ITERATION
002714  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
002720  012737  002732  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
002726  105014                          CLRB    @TACS                   ;TACS ARE YOU THERE?
002730  000420                          BR      2$                      ;YES -- NO TRAP
002732  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
002740  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
002742  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
002746  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
002754  010437  001122                  MOV     TACS,@#$BDADR           ;GET THE FAILING ADDRESS
002760  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TACS REGISTER
002762  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
002766  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
002772  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER

                                        ;;*************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TADB RESPONDS TO ADDRESSING
                                        ;;*************************************************************
                                        ;*TEST 4         TEST FOR LOW BYTE OF TADB
                                        ;;*************************************************************
003000  000004                  TST4:   SCOPE
003002  012767  000001  176156          MOV     #1,$TIMES       ;;DO 1 ITERATION
003010  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
003014  012737  003026  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
003022  105015                          CLRB    @TADB                   ;TADB ARE YOU THERE?
003024  000420                          BR      2$                      ;YES -- NO TRAP
003026  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
003034  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
003036  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
003042  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
003050  010537  001122                  MOV     TADB,@#$BDADR           ;GET THE FAILING ADDRESS
003054  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TADB REGISTER
003056  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
003062  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
003066  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER

                                        ;;*************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TACSH RESPONDS TO ADDRESSING
                                        ;;*************************************************************
                                        ;*TEST 5         TEST FOR HIGH BYTE OF TACS
                                        ;;*************************************************************
003074  000004                  TST5:   SCOPE
003076  012767  000001  176062          MOV     #1,$TIMES       ;;DO 1 ITERATION
003104  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
```



```
003110  012737  003124  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
003116  105077  176066                  CLRB    @TACSH                  ;TACSH ARE YOU THERE?
003122  000421                          BR      2$                      ;YES -- NO TRAP
003124  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
003132  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
003134  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
003140  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
003146  016737  176036  001122          MOV     TACSH,@#$BDADR          ;GET THE FAILING ADDRESS
003154  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TACSH REGISTER
003156  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
003162  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
003166  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER

                                        ;;*************************************************************
                                        ;TA11 REGISTER ADDRESS TEST
                                        ;INSURE THAT TADBH RESPONDS TO ADDRESSING
                                        ;;*************************************************************
                                        ;*TEST 6         TEST FOR HIGH BYTE OF TADB
                                        ;;*************************************************************
003174  000004                  TST6:   SCOPE
003176  012767  000001  175762          MOV     #1,$TIMES       ;;DO 1 ITERATION
003204  012706  001100                  MOV     #STACK,SP               ;SETUP THE STACK POINTER
003210  012737  003224  000004          MOV     #1$,@#ERRVEC            ;SETUP FOR BUS TIMEOUT
003216  105077  175772                  CLRB    @TADBH                  ;TADBH ARE YOU THERE?
003222  000421                          BR      2$                      ;YES -- NO TRAP
003224  012737  000006  000004  1$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
003232  022626                          CMP     (SP)+,(SP)+             ;CLEAN UP THE STACK
003234  013746  000030                  MOV     @#EMTVEC,-(SP)          ;SETUP FOR THIS ERROR CALL
003240  012737  011516  000030          MOV     #ERROR1,@#EMTVEC                ;SKIP OVER THE SAVE OF TACS AND TADB
003246  016737  175742  001122          MOV     TADBH,@#$BDADR          ;GET THE FAILING ADDRESS
003254  104010                          ERROR   10                      ;ERROR -- TIME OUT OCCURRED
                                                                        ;WHEN ADDRESSING THE TADBH REGISTER
003256  012637  000030                  MOV     (SP)+,@#EMTVEC          ;RESTORE THE ERROR CALLS VECTOR
003262  000137  011124                  JMP     @#$EOP                  ;GO TO END OF PROGRAM
003266  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TRAP CATCHER
```

```
                                        ;////////////////////////////////////////////////////////////////
                                        ;////////////////////////////////////////////////////////////////
                                        ;THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
                                        ;IN THE TACS THAT ARE NOT DRIVE DEPENDENT.
                                        ;////////////////////////////////////////////////////////////////
                                        ;;**************************************************************
                                        ;*TEST 7        TEST INITIAL CONDITION OF TACS BIT14 = 0
                                        ;;**************************************************************
003274  000004                          TST7:   SCOPE
003276  012767  003306  175602                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003304  000005                                  RESET
003306  032714  040000                  1$:     BIT     #BIT14,@TACS            ;IS BIT14 OF TACS = 0?
003312  001401                                  BEQ     TST10           ;;BR IF YES
003314  104001                                  ERROR   1                       ;ERROR -- BIT14 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 10       TEST INITIAL CONDITION OF TACS BIT11 = 0
                                        ;;**************************************************************
003316  000004                          TST10:  SCOPE
003320  012767  003330  175560                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003326  000005                                  RESET
003330  032714  004000                  1$:     BIT     #BIT11,@TACS            ;IS BIT11 OF TACS = 0?
003334  001401                                  BEQ     TST11           ;;BR IF YES
003336  104001                                  ERROR   1                       ;ERROR -- BIT11 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 11       TEST INITIAL CONDITION OF TACS BIT10 = 0
                                        ;;**************************************************************
003340  000004                          TST11:  SCOPE
003342  012767  003352  175536                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003350  000005                                  RESET
003352  032714  002000                  1$:     BIT     #BIT10,@TACS            ;IS BIT10 OF TACS = 0?
003356  001401                                  BEQ     TST12           ;;BR IF YES
003360  104001                                  ERROR   1                       ;ERROR -- BIT10 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 12       TEST INITIAL CONDITION OF TACS BIT08 = 0
                                        ;;**************************************************************
003362  000004                          TST12:  SCOPE
003364  012767  003374  175514                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003372  000005                                  RESET
003374  032714  000400                  1$:     BIT     #BIT08,@TACS            ;IS BIT08 OF TACS = 0?
003400  001401                                  BEQ     TST13           ;;BR IF YES
003402  104001                                  ERROR   1                       ;ERROR -- BIT08 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 13       TEST INITIAL CONDITION OF TACS BIT07 = 0
                                        ;;**************************************************************
003404  000004                          TST13:  SCOPE
003406  012767  003416  175472                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003414  000005                                  RESET
003416  032714  000200                  1$:     BIT     #BIT07,@TACS            ;IS BIT07 OF TACS = 0?
003422  001401                                  BEQ     TST14           ;;BR IF YES
003424  104001                                  ERROR   1                       ;ERROR -- BIT07 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 14       TEST INITIAL CONDITION OF TACS BIT06 = 0
                                        ;;**************************************************************
003426  000004                          TST14:  SCOPE
003430  012767  003440  175450                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003436  000005                                  RESET
```



```
003440  032714  000100                  1$:     BIT     #BIT06,@TACS            ;IS BIT06 OF TACS = 0?
003444  001401                                  BEQ     TST15           ;;BR IF YES
003446  104001                                  ERROR   1                       ;ERROR -- BIT06 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 15       TEST INITIAL CONDITION OF TACS BIT04 = 0
                                        ;;**************************************************************
003450  000004                          TST15:  SCOPE
003452  012767  003462  175426                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003460  000005                                  RESET
003462  032714  000020                  1$:     BIT     #BIT04,@TACS            ;IS BIT04 OF TACS = 0?
003466  001401                                  BEQ     TST16           ;;BR IF YES
003470  104001                                  ERROR   1                       ;ERROR -- BIT04 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 16       TEST INITIAL CONDITION OF TACS BIT03 = 0
                                        ;;**************************************************************
003472  000004                          TST16:  SCOPE
003474  012767  003504  175404                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003502  000005                                  RESET
003504  032714  000010                  1$:     BIT     #BIT03,@TACS            ;IS BIT03 OF TACS = 0?
003510  001401                                  BEQ     TST17           ;;BR IF YES
003512  104001                                  ERROR   1                       ;ERROR -- BIT03 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 17       TEST INITIAL CONDITION OF TACS BIT02 = 0
                                        ;;**************************************************************
003514  000004                          TST17:  SCOPE
003516  012767  003526  175362                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003524  000005                                  RESET
003526  032714  000004                  1$:     BIT     #BIT02,@TACS            ;IS BIT02 OF TACS = 0?
003532  001401                                  BEQ     TST20           ;;BR IF YES
003534  104001                                  ERROR   1                       ;ERROR -- BIT02 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 20       TEST INITIAL CONDITION OF TACS BIT01 = 0
                                        ;;**************************************************************
003536  000004                          TST20:  SCOPE
003540  012767  003550  175340                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003546  000005                                  RESET
003550  032714  000002                  1$:     BIT     #BIT01,@TACS            ;IS BIT01 OF TACS = 0?
003554  001401                                  BEQ     TST21           ;;BR IF YES
003556  104001                                  ERROR   1                       ;ERROR -- BIT01 OF TACS DIDN'T INITIALIZE PROPER
                                        ;;**************************************************************
                                        ;*TEST 21       TEST INITIAL CONDITION OF TACS BIT00 = 0
                                        ;;**************************************************************
003560  000004                          TST21:  SCOPE
003562  012767  003572  175316                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003570  000005                                  RESET
003572  032714  000001                  1$:     BIT     #BIT00,@TACS            ;IS BIT00 OF TACS = 0?
003576  001401                                  BEQ     TST22           ;;BR IF YES
003600  104001                                  ERROR   1                       ;ERROR -- BIT00 OF TACS DIDN'T INITIALIZE PROPER
```

```
                                ;////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS VALIDATES EACH READ/WRITE
                                ;BIT OF THE TACS FOR SETTING, CLEARING,
                                ;AND INITIALIZING
                                ;////////////////////////////////////////////////////////////////////
                                ;;****************************************************************
                                ;*TEST 22        TEST BIT08 OF TACS MAY BE "SET" AND "CLEARED"
                                ;;****************************************************************
003602  000004                  TST22:  SCOPE
003604  052714  000400                  BIS     #BIT08,@TACS            ;SET BIT08 OF TACS
003610  032714  000400                  BIT     #BIT08,@TACS            ;DID BIT08 SET?
003614  001001                          BNE     1$                      ;BR IF YES
003616  104001                          ERROR   1                       ;BIT08 FAILED TO SET

003620  042714  000400          1$:     BIC     #BIT08,@TACS            ;NOW CLEAR IT
003624  032714  000400                  BIT     #BIT08,@TACS            ;DID BIT08 CLEAR?
003630  001401                          BEQ     TST23           ;;BR IF YES
003632  104001                          ERROR   1                       ;ERROR -- BIT08 OF TACS FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 23        TEST BIT08 OF TACS INITIALIZE TO ZERO
                                ;;****************************************************************
003634  000004                  TST23:  SCOPE
003636  012767  000005  175322          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
003644  012714  000400          1$:     MOV     #BIT08,@TACS            ;SET BIT08 OF TACS
003650  032714  000400                  BIT     #BIT08,@TACS            ;DID BIT08 SET?
003654  001001                          BNE     2$                      ;BR IF YES
003656  104001                          ERROR   1                       ;BIT08 FAILED TO SET

003660  000005                  2$:     RESET                           ;CLEAR THE WORLD
003662  032714  000400                  BIT     #BIT08,@TACS            ;DID BIT08 CLEAR?
003666  001401                          BEQ     TST24           ;;BR IF YES
003670  104001                          ERROR   1                       ;ERROR -- RESET DIDN'T CLEAR BIT08
                                ;;****************************************************************
                                ;*TEST 24        TEST BIT06 OF TACS MAY BE "SET" AND "CLEARED"
                                ;;****************************************************************
003672  000004                  TST24:  SCOPE
003674  052714  000100                  BIS     #BIT06,@TACS            ;SET BIT06 OF TACS
003700  032714  000100                  BIT     #BIT06,@TACS            ;DID BIT06 SET?
003704  001001                          BNE     1$                      ;BR IF YES
003706  104001                          ERROR   1                       ;BIT06 FAILED TO SET

003710  042714  000100          1$:     BIC     #BIT06,@TACS            ;NOW CLEAR IT
003714  032714  000100                  BIT     #BIT06,@TACS            ;DID BIT06 CLEAR?
003720  001401                          BEQ     TST25           ;;BR IF YES
003722  104001                          ERROR   1                       ;ERROR -- BIT06 OF TACS FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 25        TEST BIT06 OF TACS INITIALIZE TO ZERO
                                ;;****************************************************************
003724  000004                  TST25:  SCOPE
003726  012767  000005  175232          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
003734  012714  000100          1$:     MOV     #BIT06,@TACS            ;SET BIT06 OF TACS
003740  032714  000100                  BIT     #BIT06,@TACS            ;DID BIT06 SET?
003744  001001                          BNE     2$                      ;BR IF YES
003746  104001                          ERROR   1                       ;BIT06 FAILED TO SET
```



```
003750  000005                  2$:     RESET                           ;CLEAR THE WORLD
003752  032714  000100                  BIT     #BIT06,@TACS            ;DID BIT06 CLEAR?
003756  001401                          BEQ     TST26           ;;BR IF YES
003760  104001                          ERROR   1                       ;ERROR -- RESET DIDN'T CLEAR BIT06
                                ;;****************************************************************
                                ;*TEST 26        TEST BIT04 OF TACS MAY BE "SET" AND "CLEARED"
                                ;;****************************************************************
003762  000004                  TST26:  SCOPE
003764  052714  000020                  BIS     #BIT04,@TACS            ;SET BIT04 OF TACS
003770  032714  000020                  BIT     #BIT04,@TACS            ;DID BIT04 SET?
003774  001001                          BNE     1$                      ;BR IF YES
003776  104001                          ERROR   1                       ;BIT04 FAILED TO SET

004000  042714  000020          1$:     BIC     #BIT04,@TACS            ;NOW CLEAR IT
004004  032714  000020                  BIT     #BIT04,@TACS            ;DID BIT04 CLEAR?
004010  001401                          BEQ     TST27           ;;BR IF YES
004012  104001                          ERROR   1                       ;ERROR -- BIT04 OF TACS FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 27        TEST BIT04 OF TACS INITIALIZE TO ZERO
                                ;;****************************************************************
004014  000004                  TST27:  SCOPE
004016  012767  000005  175142          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
004024  012714  000020          1$:     MOV     #BIT04,@TACS            ;SET BIT04 OF TACS
004030  032714  000020                  BIT     #BIT04,@TACS            ;DID BIT04 SET?
004034  001001                          BNE     2$                      ;BR IF YES
004036  104001                          ERROR   1                       ;BIT04 FAILED TO SET

004040  000005                  2$:     RESET                           ;CLEAR THE WORLD
004042  032714  000020                  BIT     #BIT04,@TACS            ;DID BIT04 CLEAR?
004046  001401                          BEQ     TST30           ;;BR IF YES
004050  104001                          ERROR   1                       ;ERROR -- RESET DIDN'T CLEAR BIT04
                                ;;****************************************************************
                                ;*TEST 30        TEST BIT03 OF TACS MAY BE "SET" AND "CLEARED"
                                ;;****************************************************************
004052  000004                  TST30:  SCOPE
004054  052714  000010                  BIS     #BIT03,@TACS            ;SET BIT03 OF TACS
004060  032714  000010                  BIT     #BIT03,@TACS            ;DID BIT03 SET?
004064  001001                          BNE     1$                      ;BR IF YES
004066  104001                          ERROR   1                       ;BIT03 FAILED TO SET

004070  042714  000010          1$:     BIC     #BIT03,@TACS            ;NOW CLEAR IT
004074  032714  000010                  BIT     #BIT03,@TACS            ;DID BIT03 CLEAR?
004100  001401                          BEQ     TST31           ;;BR IF YES
004102  104001                          ERROR   1                       ;ERROR -- BIT03 OF TACS FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 31        TEST BIT03 OF TACS INITIALIZE TO ZERO
                                ;;****************************************************************
004104  000004                  TST31:  SCOPE
004106  012767  000005  175052          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
004114  012714  000010          1$:     MOV     #BIT03,@TACS            ;SET BIT03 OF TACS
004120  032714  000010                  BIT     #BIT03,@TACS            ;DID BIT03 SET?
004124  001001                          BNE     2$                      ;BR IF YES
004126  104001                          ERROR   1                       ;BIT03 FAILED TO SET

004130  000005                  2$:     RESET                           ;CLEAR THE WORLD
004132  032714  000010                  BIT     #BIT03,@TACS            ;DID BIT03 CLEAR?
```

```
1072  004136  001401                        BEQ     TST32           ;;BR IF YES
1073  004140  104001                        ERROR   1               ;ERROR -- RESET DIDN'T CLEAR BIT03
1074                                  ;;*****************************************************************
1075                                  ;*TEST 32        TEST BIT02 OF TACS MAY BE "SET" AND "CLEARED"
1076                                  ;;*****************************************************************
1077  004142  000004                  TST32:  SCOPE
1078  004144  052714  000004                 BIS     #BIT02,@TACS            ;SET BIT02 OF TACS
1079  004150  032714  000004                 BIT     #BIT02,@TACS            ;DID BIT02 SET?
1080  004154  001001                        BNE     1$                      ;BR IF YES
1081  004156  104001                        ERROR   1                       ;BIT02 FAILED TO SET
1082
1083  004160  042714  000004          1$:   BIC     #BIT02,@TACS            ;NOW CLEAR IT
1084  004164  032714  000004                 BIT     #BIT02,@TACS            ;DID BIT02 CLEAR?
1085  004170  001401                        BEQ     TST33           ;;BR IF YES
1086  004172  104001                        ERROR   1                       ;ERROR -- BIT02 OF TACS FAILED TO CLEAR
1087                                  ;;*****************************************************************
1088                                  ;*TEST 33        TEST BIT02 OF TACS INITIALIZE TO ZERO
1089                                  ;;*****************************************************************
1090  004174  000004                  TST33:  SCOPE
1091  004176  012767  000005  174762         MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1092  004204  012714  000004          1$:   MOV     #BIT02,@TACS            ;SET BIT02 OF TACS
1093  004210  032714  000004                 BIT     #BIT02,@TACS            ;DID BIT02 SET?
1094  004214  001001                        BNE     2$                      ;BR IF YES
1095  004216  104001                        ERROR   1                       ;BIT02 FAILED TO SET
1096
1097  004220  000005                  2$:   RESET                           ;CLEAR THE WORLD
1098  004222  032714  000004                 BIT     #BIT02,@TACS            ;DID BIT02 CLEAR?
1099  004226  001401                        BEQ     TST34           ;;BR IF YES
1100  004230  104001                        ERROR   1                       ;ERROR -- RESET DIDN'T CLEAR BIT02
1101                                  ;;*****************************************************************
1102                                  ;*TEST 34        TEST BIT01 OF TACS MAY BE "SET" AND "CLEARED"
1103                                  ;;*****************************************************************
1104  004232  000004                  TST34:  SCOPE
1105  004234  052714  000002                 BIS     #BIT01,@TACS            ;SET BIT01 OF TACS
1106  004240  032714  000002                 BIT     #BIT01,@TACS            ;DID BIT01 SET?
1107  004244  001001                        BNE     1$                      ;BR IF YES
1108  004246  104001                        ERROR   1                       ;BIT01 FAILED TO SET
1109
1110  004250  042714  000002          1$:   BIC     #BIT01,@TACS            ;NOW CLEAR IT
1111  004254  032714  000002                 BIT     #BIT01,@TACS            ;DID BIT01 CLEAR?
1112  004260  001401                        BEQ     TST35           ;;BR IF YES
1113  004262  104001                        ERROR   1                       ;ERROR -- BIT01 OF TACS FAILED TO CLEAR
1114                                  ;;*****************************************************************
1115                                  ;*TEST 35        TEST BIT01 OF TACS INITIALIZE TO ZERO
1116                                  ;;*****************************************************************
1117  004264  000004                  TST35:  SCOPE
1118  004266  012767  000005  174672         MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1119  004274  012714  000002          1$:   MOV     #BIT01,@TACS            ;SET BIT01 OF TACS
1120  004300  032714  000002                 BIT     #BIT01,@TACS            ;DID BIT01 SET?
1121  004304  001001                        BNE     2$                      ;BR IF YES
1122  004306  104001                        ERROR   1                       ;BIT01 FAILED TO SET
1123
1124  004310  000005                  2$:   RESET                           ;CLEAR THE WORLD
1125  004312  032714  000002                 BIT     #BIT01,@TACS            ;DID BIT01 CLEAR?
1126  004316  001401                        BEQ     TST36           ;;BR IF YES
1127  004320  104001                        ERROR   1                       ;ERROR -- RESET DIDN'T CLEAR BIT01
```



```
1128                                  ;;*****************************************************************
1129                                  ;*TEST 36        COUNT PATTERN TO TACS<04:01>
1130                                  ;;*****************************************************************
1131  004322  000004                  TST36:  SCOPE
1132  004324  012767  004370  174636         MOV     #TST37,$ESCAPE  ;;ESCAPE TO TEST 37 ON ERROR
1133  004332  005014                        CLR     @TACS                   ;SETUP TACS FOR COUNTING
1134  004334  005000                        CLR     R0                      ;SETUP R0 FOR COUNTING
1135  004336  012702  000002                MOV     #2,R2                   ;SET THE INCREMENT
1136  004342  011401                  1$:   MOV     @TACS,R1                ;GET TACS
1137  004344  042701  177741                BIC     #^C36,R1                ;CLEAR THE JUNK
1138  004350  020100                        CMP     R1,R0                   ;COUNT OK?
1139  004352  001401                        BEQ     2$                      ;BR IF YES
1140  004354  104005                        ERROR   5                       ;ERROR--BAD COUNT
1141  004356  060200                  2$:   ADD     R2,R0                   ;UPDATE R0
1142  004360  060214                        ADD     R2,@TACS                ;COUNT TACS
1143  004362  032700  000036                BIT     #36,R0                  ;READY?
1144  004366  001365                        BNE     1$                      ;NO--GO DO TEST COUNT PATTERN
1145
```

```
1146                                    ;////////////////////////////////////////////////////////////////////
1147                                    ;////////////////////////////////////////////////////////////////////
1148                                    ;THE FOLLOWING TESTS WILL CHECK THE INITIAL STATE OF ALL BITS
1149                                    ;IN THE TACS THAT ARE DRIVE AND FUNCTION DEPENDENT.
1150                                    ;NOTE: FOR ALL OF THESE TEST TO PASS
1151                                    ;THE DRIVE UNDERGOING TEST MUST BE
1152                                    ;ON CLEAR LEADER AND WRITE ENABLED.
1153                                    ;////////////////////////////////////////////////////////////////////
1154                                    ;;****************************************************************
1155                                    ;*TEST 37       TEST INITIAL CONDITION OF TACS BIT05 = 1
1156                                    ;;****************************************************************
1157  004370  000004                   TST37:  SCOPE
1158  004372  012767  004404  174506           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1159  004400  000005                           RESET
1160  004402  010314                           MOV     DRIVE,@TACS             ;SELECT DRIVE
1161  004404  032714  000040           1$:     BIT     #BIT05,@TACS            ;IS BIT05 OF TACS = 1?
1162  004410  001001                           BNE     TST40           ;;BR IF YES
1163  004412  104001                           ERROR   1                       ;ERROR -- BIT05 OF TACS DIDN'T INITIALIZE PROPER
1164                                    ;;****************************************************************
1165                                    ;*TEST 40       TEST INITIAL CONDITION OF TACS BIT09 = 0
1166                                    ;;****************************************************************
1167  004414  000004                   TST40:  SCOPE
1168  004416  012767  004430  174462           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1169  004424  000005                           RESET
1170  004426  010314                           MOV     DRIVE,@TACS             ;SELECT DRIVE
1171  004430  032714  001000           1$:     BIT     #BIT09,@TACS            ;IS BIT09 OF TACS = 0?
1172  004434  001401                           BEQ     TST41           ;;BR IF YES
1173  004436  104001                           ERROR   1                       ;ERROR -- BIT09 OF TACS DIDN'T INITIALIZE PROPER
1174                                    ;;****************************************************************
1175                                    ;*TEST 41       TEST INITIAL CONDITION OF TACS BIT12 = 0
1176                                    ;;****************************************************************
1177  004440  000004                   TST41:  SCOPE
1178  004442  012767  004460  174436           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1179  004450  000005                           RESET
1180  004452  010314                           MOV     DRIVE,@TACS             ;SELECT DRIVE
1181  004454  052714  000002                   BIS     #WRITE,@TACS            ;LOAD WRITE FUNCTION
1182  004460  032714  010000           1$:     BIT     #BIT12,@TACS            ;IS BIT12 OF TACS = 0?
1183  004464  001401                           BEQ     TST42           ;;BR IF YES
1184  004466  104001                           ERROR   1                       ;ERROR -- BIT12 OF TACS DIDN'T INITIALIZE PROPER
1185                                    ;;****************************************************************
1186                                    ;*TEST 42       TEST INITIAL CONDITION OF TACS BIT13 = 1
1187                                    ;;****************************************************************
1188  004470  000004                   TST42:  SCOPE
1189  004472  012767  004504  174406           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1190  004500  000005                           RESET
1191  004502  010314                           MOV     DRIVE,@TACS             ;SELECT DRIVE
1192  004504  032714  020000           1$:     BIT     #BIT13,@TACS            ;IS BIT13 OF TACS = 1?
1193  004510  001001                           BNE     TST43           ;;BR IF YES
1194  004512  104001                           ERROR   1                       ;ERROR -- BIT13 OF TACS DIDN'T INITIALIZE PROPER
1195                                    ;;****************************************************************
1196                                    ;*TEST 43       TEST INITIAL CONDITION OF TACS BIT15 = 1
1197                                    ;;****************************************************************
1198  004514  000004                   TST43:  SCOPE
1199  004516  012767  004530  174362           MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1200  004524  000005                           RESET
1201  004526  010314                           MOV     DRIVE,@TACS             ;SELECT DRIVE
```



```
1202  004530  032714  100000           1$:     BIT     #BIT15,@TACS            ;IS BIT15 OF TACS = 1?
1203  004534  001001                           BNE     TST44           ;;BR IF YES
1204  004536  104001                           ERROR   1                       ;ERROR -- BIT15 OF TACS DIDN'T INITIALIZE PROPER
```

```
                                ;////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS CHECKS EACH READ ONLY
                                ;BIT OF THE TACS FOR READ ONLY CAPABILITY
                                ;////////////////////////////////////////////////////////////////
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT15.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 44        TEST TACS BIT15 IS READ ONLY
                                ;;************************************************************
004540  000004                  TST44:  SCOPE
004542  012767  004552  174336          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
004550  000005                          RESET                   ;INITIALIZE ALL BITS
004552  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
004554  010001                          MOV     R0,R1           ;PUT STATUS IN R1
004556  005101                          COM     R1              ;COMPLEMENT R1
004560  042700  077777                  BIC     #^C<BIT15>,R0   ;GET RID OF UNDESIRED BITS
004564  042701  077777                  BIC     #^C<BIT15>,R1
004570  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT15
004572  011401                          MOV     @TACS,R1        ;READ IT BACK
004574  042701  077777                  BIC     #^C<BIT15>,R1   ;ONLY LOOK AT BIT15
004600  020001                          CMP     R0,R1           ;DID IT CHANGE?
004602  001401                          BEQ     TST45           ;;BR IF NO
004604  104001                          ERROR   1               ;ERROR -- BIT15 CHANGED
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT14.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 45        TEST TACS BIT14 IS READ ONLY
                                ;;************************************************************
004606  000004                  TST45:  SCOPE
004610  012767  004620  174270          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
004616  000005                          RESET                   ;INITIALIZE ALL BITS
004620  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
004622  010001                          MOV     R0,R1           ;PUT STATUS IN R1
004624  005101                          COM     R1              ;COMPLEMENT R1
004626  042700  137777                  BIC     #^C<BIT14>,R0   ;GET RID OF UNDESIRED BITS
004632  042701  137777                  BIC     #^C<BIT14>,R1
004636  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT14
004640  011401                          MOV     @TACS,R1        ;READ IT BACK
004642  042701  137777                  BIC     #^C<BIT14>,R1   ;ONLY LOOK AT BIT14
004646  020001                          CMP     R0,R1           ;DID IT CHANGE?
004650  001401                          BEQ     TST46           ;;BR IF NO
004652  104001                          ERROR   1               ;ERROR -- BIT14 CHANGED
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT13.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 46        TEST TACS BIT13 IS READ ONLY
                                ;;************************************************************
004654  000004                  TST46:  SCOPE
004656  012767  004666  174222          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
004664  000005                          RESET                   ;INITIALIZE ALL BITS
004666  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
004670  010001                          MOV     R0,R1           ;PUT STATUS IN R1
```



```
004672  005101                          COM     R1              ;COMPLEMENT R1
004674  042700  157777                  BIC     #^C<BIT13>,R0   ;GET RID OF UNDESIRED BITS
004700  042701  157777                  BIC     #^C<BIT13>,R1
004704  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT13
004706  011401                          MOV     @TACS,R1        ;READ IT BACK
004710  042701  157777                  BIC     #^C<BIT13>,R1   ;ONLY LOOK AT BIT13
004714  020001                          CMP     R0,R1           ;DID IT CHANGE?
004716  001401                          BEQ     TST47           ;;BR IF NO
004720  104001                          ERROR   1               ;ERROR -- BIT13 CHANGED
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT12.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 47        TEST TACS BIT12 IS READ ONLY
                                ;;************************************************************
004722  000004                  TST47:  SCOPE
004724  012767  004734  174154          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
004732  000005                          RESET                   ;INITIALIZE ALL BITS
004734  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
004736  010001                          MOV     R0,R1           ;PUT STATUS IN R1
004740  005101                          COM     R1              ;COMPLEMENT R1
004742  042700  167777                  BIC     #^C<BIT12>,R0   ;GET RID OF UNDESIRED BITS
004746  042701  167777                  BIC     #^C<BIT12>,R1
004752  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT12
004754  011401                          MOV     @TACS,R1        ;READ IT BACK
004756  042701  167777                  BIC     #^C<BIT12>,R1   ;ONLY LOOK AT BIT12
004762  020001                          CMP     R0,R1           ;DID IT CHANGE?
004764  001401                          BEQ     TST50           ;;BR IF NO
004766  104001                          ERROR   1               ;ERROR -- BIT12 CHANGED
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT11.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 50        TEST TACS BIT11 IS READ ONLY
                                ;;************************************************************
004770  000004                  TST50:  SCOPE
004772  012767  005002  174106          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
005000  000005                          RESET                   ;INITIALIZE ALL BITS
005002  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
005004  010001                          MOV     R0,R1           ;PUT STATUS IN R1
005006  005101                          COM     R1              ;COMPLEMENT R1
005010  042700  173777                  BIC     #^C<BIT11>,R0   ;GET RID OF UNDESIRED BITS
005014  042701  173777                  BIC     #^C<BIT11>,R1
005020  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT11
005022  011401                          MOV     @TACS,R1        ;READ IT BACK
005024  042701  173777                  BIC     #^C<BIT11>,R1   ;ONLY LOOK AT BIT11
005030  020001                          CMP     R0,R1           ;DID IT CHANGE?
005032  001401                          BEQ     TST51           ;;BR IF NO
005034  104001                          ERROR   1               ;ERROR -- BIT11 CHANGED
                                ;;************************************************************
                                        ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT10.
                                        ;THEN CHECK IT TO SEE IF IT CHANGED.
                                ;;************************************************************
                                ;*TEST 51        TEST TACS BIT10 IS READ ONLY
                                ;;************************************************************
005036  000004                  TST51:  SCOPE
```

```
 1317  005040  012767  005050  174040          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 1318  005046  000005                          RESET                   ;INITIALIZE ALL BITS
 1319  005050  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
 1320  005052  010001                          MOV     R0,R1           ;PUT STATUS IN R1
 1321  005054  005101                          COM     R1              ;COMPLEMENT R1
 1322  005056  042700  175777                  BIC     #^C<BIT10>,R0   ;GET RID OF UNDESIRED BITS
 1323  005062  042701  175777                  BIC     #^C<BIT10>,R1
 1324  005066  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT10
 1325  005070  011401                          MOV     @TACS,R1        ;READ IT BACK
 1326  005072  042701  175777                  BIC     #^C<BIT10>,R1   ;ONLY LOOK AT BIT10
 1327  005076  020001                          CMP     R0,R1           ;DID IT CHANGE?
 1328  005100  001401                          BEQ     TST52           ;;BR IF NO
 1329  005102  104001                          ERROR   1               ;ERROR -- BIT10 CHANGED
 1330                                  ;;**************************************************************
 1331                                  ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT09.
 1332                                  ;THEN CHECK IT TO SEE IF IT CHANGED.
 1333                                  ;;**************************************************************
 1334                                  ;*TEST 52        TEST TACS BIT09 IS READ ONLY
 1335                                  ;;**************************************************************
 1336  005104  000004                  TST52:  SCOPE
 1337  005106  012767  005116  173772          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 1338  005114  000005                          RESET                   ;INITIALIZE ALL BITS
 1339  005116  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
 1340  005120  010001                          MOV     R0,R1           ;PUT STATUS IN R1
 1341  005122  005101                          COM     R1              ;COMPLEMENT R1
 1342  005124  042700  176777                  BIC     #^C<BIT09>,R0   ;GET RID OF UNDESIRED BITS
 1343  005130  042701  176777                  BIC     #^C<BIT09>,R1
 1344  005134  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT09
 1345  005136  011401                          MOV     @TACS,R1        ;READ IT BACK
 1346  005140  042701  176777                  BIC     #^C<BIT09>,R1   ;ONLY LOOK AT BIT09
 1347  005144  020001                          CMP     R0,R1           ;DID IT CHANGE?
 1348  005146  001401                          BEQ     TST53           ;;BR IF NO
 1349  005150  104001                          ERROR   1               ;ERROR -- BIT09 CHANGED
 1350                                  ;;**************************************************************
 1351                                  ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT07.
 1352                                  ;THEN CHECK IT TO SEE IF IT CHANGED.
 1353                                  ;;**************************************************************
 1354                                  ;*TEST 53        TEST TACS BIT07 IS READ ONLY
 1355                                  ;;**************************************************************
 1356  005152  000004                  TST53:  SCOPE
 1357  005154  012767  005164  173724          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 1358  005162  000005                          RESET                   ;INITIALIZE ALL BITS
 1359  005164  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
 1360  005166  010001                          MOV     R0,R1           ;PUT STATUS IN R1
 1361  005170  005101                          COM     R1              ;COMPLEMENT R1
 1362  005172  042700  177577                  BIC     #^C<BIT07>,R0   ;GET RID OF UNDESIRED BITS
 1363  005176  042701  177577                  BIC     #^C<BIT07>,R1
 1364  005202  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT07
 1365  005204  011401                          MOV     @TACS,R1        ;READ IT BACK
 1366  005206  042701  177577                  BIC     #^C<BIT07>,R1   ;ONLY LOOK AT BIT07
 1367  005212  020001                          CMP     R0,R1           ;DID IT CHANGE?
 1368  005214  001401                          BEQ     TST54           ;;BR IF NO
 1369  005216  104001                          ERROR   1               ;ERROR -- BIT07 CHANGED
 1370                                  ;;**************************************************************
 1371                                  ;THIS TEST WILL TRY TO CHANGE THE STATE OF BIT05.
 1372                                  ;THEN CHECK IT TO SEE IF IT CHANGED.
```



```
 1373                                  ;;**************************************************************
 1374                                  ;*TEST 54        TEST TACS BIT05 IS READ ONLY
 1375                                  ;;**************************************************************
 1376  005220  000004                  TST54:  SCOPE
 1377  005222  012767  005232  173656          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 1378  005230  000005                          RESET                   ;INITIALIZE ALL BITS
 1379  005232  011400                  1$:     MOV     @TACS,R0        ;GET TACS FOR COMPARING
 1380  005234  010001                          MOV     R0,R1           ;PUT STATUS IN R1
 1381  005236  005101                          COM     R1              ;COMPLEMENT R1
 1382  005240  042700  177737                  BIC     #^C<BIT05>,R0   ;GET RID OF UNDESIRED BITS
 1383  005244  042701  177737                  BIC     #^C<BIT05>,R1
 1384  005250  010114                          MOV     R1,@TACS        ;TRY TO CHANGE BIT05
 1385  005252  011401                          MOV     @TACS,R1        ;READ IT BACK
 1386  005254  042701  177737                  BIC     #^C<BIT05>,R1   ;ONLY LOOK AT BIT05
 1387  005260  020001                          CMP     R0,R1           ;DID IT CHANGE?
 1388  005262  001401                          BEQ     TST55           ;;BR IF NO
 1389  005264  104001                          ERROR   1               ;ERROR -- BIT05 CHANGED
 1390
```

```
                                ;////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS WILL CHECK FOR DUAL AND/OR FAULTY SELECTION
                                ;OF THE TA11 ADDRESSES
                                ;////////////////////////////////////////////////////////////////
                                ;;************************************************************
                                ;*TEST 55        TEST HIGH BYTE SELECTION OF TACS
                                ;;************************************************************
005266  000004                  TST55:  SCOPE
005270  012767  005300  173610          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
005276  000005                          RESET
005300  012714  000004          1$:     MOV     #BIT02,@TACS            ;SET TACS BIT02
005304  105077  173700                  CLRB    @TACSH                  ;CLEAR THE HIGH BYTE
005310  132714  000004                  BITB    #BIT02,@TACS            ;IS BIT02 STILL SET?
005314  001001                          BNE     TST56           ;;BR IF YES
005316  104001                          ERROR   1                       ;CLRB TO HIGH BYTE OF TACS CLEARED LOW BYTE
                                ;;************************************************************
                                ;*TEST 56        TEST LOW BYTE SELECTION OF TACS
                                ;;************************************************************
005320  000004                  TST56:  SCOPE
005322  012767  005332  173556          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
005330  000005                          RESET
005332  012714  000400          1$:     MOV     #BIT08,@TACS            ;SET BIT08 OF TACS
005336  105014                          CLRB    @TACS                   ;CLEAR THE LOW BYTE OF TACS
005340  132777  000001  173642          BITB    #BIT00,@TACSH           ;IS BIT08 OF TACS STILL SET?
005346  001001                          BNE     TST57           ;;BR IF YES
005350  104001                          ERROR   1                       ;CLRB TO LOW BYTE OF TACS CLEARED THE HIGH BYTE
                                ;;************************************************************
                                ;*TEST 57        TEST "OUT LOW" SIGNAL
                                ;;************************************************************
005352  000004                  TST57:  SCOPE
005354  012767  005412  173606          MOV     #TST60,$ESCAPE  ;;ESCAPE TO TEST 60 ON ERROR
005362  005014                          CLR     @TACS                   ;ZERO THE TA11 STATUS
005364  032714  000536                  BIT     #UNIT!INT.EN!ILBS!FUNCTION,@TACS
005370  001401                          BEQ     1$
005372  104001                          ERROR   1                       ;TACS DIDN'T CLEAR PROPERLY
005374  112777  000136  173606  1$:     MOVB    #INT.EN!ILBS!FUNCTION,@TACSH    ;LOAD HIGH BYTE OF TACS
005402  032714  000136                  BIT     #INT.EN!ILBS!FUNCTION,@TACS     ;CHECK LOW BYTE
005406  001401                          BEQ     TST60           ;;BR IF LOW BYTE IS OK
005410  104001                          ERROR   1                       ;LOW BYTE IMPROPERLY SELECTED
                                ;;************************************************************
                                ;*TEST 60        TEST "OUT HIGH" SIGNAL
                                ;;************************************************************
005412  000004                  TST60:  SCOPE
005414  012767  005450  173546          MOV     #TST61,$ESCAPE  ;;ESCAPE TO TEST 61 ON ERROR
005422  005014                          CLR     @TACS                   ;ZERO THE TA11 STATUS
005424  032714  000536                  BIT     #UNIT!INT.EN!ILBS!FUNCTION,@TACS
005430  001401                          BEQ     1$
005432  104001                          ERROR   1                       ;TACS DIDN'T CLEAR PROPERLY
005434  112714  000400          1$:     MOVB    #BIT08,@TACS            ;LOAD LOW BYTE OF TACS
005440  032714  000400                  BIT     #BIT08,@TACS            ;DID BIT08 GET SET?
005444  001401                          BEQ     TST61           ;;BR IF NO
005446  104001                          ERROR   1                       ;HIGH BYTE IMPROPERLY SELECTED
                                ;;************************************************************
                                ;*TEST 61        TEST "SELECT 00" ISN'T STUCK
                                ;;************************************************************
```



```
005450  000004                  TST61:  SCOPE
005452  012767  005506  173510          MOV     #TST62,$ESCAPE  ;;ESCAPE TO TEST 62 ON ERROR
005460  005014                          CLR     @TACS                   ;CLEAR TA11 STATUS
005462  032714  000536                  BIT     #UNIT!INT.EN!ILBS!FUNCTION,@TACS
005466  001401                          BEQ     1$
005470  104001                          ERROR   1                       ;TACS DIDN'T CLEAR PROPERLY
005472  012715  177776          1$:     MOV     #^C1,@TADB              ;LOAD THE DATA BUFFER
005476  032714  000536                  BIT     #UNIT!INT.EN!ILBS!FUNCTION,@TACS
005502  001401                          BEQ     TST62           ;;BR IF TACS DIDN'T CHANGE
005504  104001                          ERROR   1                       ;TACS CHANGED WHEN LOADING TADB


                                ;;************************************************************
                                ;THE FOLLOWING TEST INSURES THAT THE STATE OF THE "ERROR" INDICATOR IS
                                ;PROPER FOR ALL FUNCTIONS WHEN THE TAPE IS AT "CLEAR LEADER"
                                ;;************************************************************
                                ;*TEST 62        TEST "ERROR" FOR "CLEAR LEADER"
                                ;;************************************************************
005506  000004                  TST62:  SCOPE
005510  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
005512  032714  020000                  BIT     #LEADER,@TACS           ;CHECK FOR CLEAR LEADER
005516  001001                          BNE     1$                      ;BR IF ON CLEAR LEADER
005520  104001                          ERROR   1                       ;NOT ON CLEAR LEADER
005522  112714  000000          1$:     MOVB    #WFG,@TACS              ;CHECK "ERROR" WITH "WFG"
005526  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005530  100401                          BMI     2$                      ;BR IF "ERROR" = 1
005532  104001                          ERROR   1                       ;"ERROR" NOT = 1
005534  112714  000002          2$:     MOVB    #WRITE,@TACS            ;CHECK "ERROR" WITH "WRITE"
005540  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005542  100401                          BMI     3$                      ;BR IF "ERROR" = 1
005544  104001                          ERROR   1                       ;"ERROR" NOT = 1
005546  112714  000004          3$:     MOVB    #READ,@TACS             ;CHECK "ERROR" WITH "READ"
005552  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005554  100401                          BMI     4$                      ;BR IF "ERROR" = 1
005556  104001                          ERROR   1                       ;"ERROR" NOT = 1
005560  112714  000006          4$:     MOVB    #BSFG,@TACS             ;CHECK "ERROR" WITH "BSFG"
005564  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005566  100401                          BMI     5$                      ;BR IF "ERROR" = 1
005570  104001                          ERROR   1                       ;"ERROR" NOT = 1
005572  112714  000010          5$:     MOVB    #BSBG,@TACS             ;CHECK "ERROR" WITH "BSBG"
005576  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005600  100401                          BMI     6$                      ;BR IF "ERROR" = 1
005602  104001                          ERROR   1                       ;"ERROR" NOT = 1
005604  112714  000012          6$:     MOVB    #SFFG,@TACS             ;CHECK "ERROR" WITH "SFFG"
005610  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005612  100401                          BMI     7$                      ;BR IF "ERROR" = 1
005614  104001                          ERROR   1                       ;"ERROR" NOT = 1
005616  112714  000014          7$:     MOVB    #SFBG,@TACS             ;CHECK "ERROR" WITH "SFBG"
005622  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005624  100401                          BMI     8$                      ;BR IF "ERROR" = 1
005626  104001                          ERROR   1                       ;"ERROR" NOT = 1
005630  112714  000016          8$:     MOVB    #REWIND,@TACS           ;CHECK "ERROR" WITH "REWIND"
005634  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
005636  100001                          BPL     TST63           ;;BR IF "ERROR" = 0
005640  104001                          ERROR   1                       ;"ERROR" NOT = 0
```

```
                                ;////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS WILL CHECK BASIC OPERATIONS OF THE
                                ;"REWIND" AND "WRITE FILE GAP" FUNCTIONS.
                                ;////////////////////////////////////////////////////////////////////
                                ;;*****************************************************************
                                ;*TEST 63        TEST THAT "READY" SETS WHEN "REWIND" IS COMPLETED
                                ;;*****************************************************************
005642  000004                  TST63:  SCOPE
005644  012767  000005  173314          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
005652  012767  005670  173226          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
005660  000005                          RESET
005662  012767  005742  173300          MOV     #TST64,$ESCAPE  ;;ESCAPE TO TEST 64 ON ERROR
005670  010314                  3$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
005672  005001                          CLR     R1                      ;WASTE SOME TIME
005674  012700  000144                  MOV     #100.,R0
005700  112714  000017                  MOVB    #REWIND!GO,@TACS                ;START A REWIND
005704  032714  000040          1$:     BIT     #READY,@TACS            ;SAMPLE READY
005710  001010                          BNE     2$                      ;GET OUT IF READY IS SET
005712  005201                          INC     R1
005714  001373                          BNE     1$
005716  005300                          DEC     R0
005720  001371                          BNE     1$
005722  032714  000040                  BIT     #READY,@TACS            ;IS "READY" BIT SET?
005726  001001                          BNE     2$                      ;BR IF YES
005730  104001                          ERROR   1                       ;ERROR--"READY" BIT DIDN'T SET
005732  032714  020000          2$:     BIT     #LEADER,@TACS           ;CHECK FOR CLEAR LEADER
005736  001001                          BNE     TST64           ;;BR IF CLEAR LEADER NE 0
005740  104001                          ERROR   1                       ;NOT ON CLEAR LEADER
                                                                        ;AFTER DOING A REWIND
                                ;;*****************************************************************
                                ;*TEST 64        TEST THAT "READY" SETS WHEN "WRITE FILE GAP" IS COMPLETED
                                ;;*****************************************************************
005742  000004                  TST64:  SCOPE
005744  012767  000005  173214          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
005752  012767  005770  173126          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
005760  000005                          RESET
005762  012767  006042  173200          MOV     #TST65,$ESCAPE  ;;ESCAPE TO TEST 65 ON ERROR
005770  010314                  3$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
005772  005001                          CLR     R1                      ;WASTE SOME TIME
005774  012700  000144                  MOV     #100.,R0
006000  112714  000001                  MOVB    #WFG!GO,@TACS           ;START A WFG
006004  032714  000040          1$:     BIT     #READY,@TACS            ;SAMPLE READY
006010  001010                          BNE     2$                      ;GET OUT IF READY IS SET
006012  005201                          INC     R1
006014  001373                          BNE     1$
006016  005300                          DEC     R0
006020  001371                          BNE     1$
006022  032714  000040                  BIT     #READY,@TACS            ;IS "READY" BIT SET?
006026  001001                          BNE     2$                      ;BR IF YES
006030  104001                          ERROR   1                       ;ERROR--"READY" BIT DIDN'T SET
006032  032714  020000          2$:     BIT     #LEADER,@TACS           ;CHECK FOR CLEAR LEADER
006036  001401                          BEQ     TST65           ;;BR IF CLEAR LEADER EQ 0
006040  104001                          ERROR   1                       ;ON CLEAR LEADER
                                                                        ;AFTER DOING A WRITE FILE GAP
                                ;;*****************************************************************
```



```
                                ;*TEST 65        TEST "READY" CLEARS WHEN STARTING A "REWIND" FUNCTION
                                ;;*****************************************************************
006042  000004                  TST65:  SCOPE
006044  012767  000005  173114          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006052  000005                          RESET
006054  012701  000001                  MOV     #1,R1                   ;SET TIMER
006060  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006062  104412                          WAITREADY                       ;WAIT FOR "READY"
006064  112714  000017                  MOVB    #REWIND!GO,@TACS                ;START A "REWIND"
006070  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
006074  001403                          BEQ     TST66           ;;BR IF "READY" = 0
006076  006301                          ASL     R1                      ;IS TIME UP?
006100  001373                          BNE     1$                      ;BR IF NO
006102  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
                                ;;*****************************************************************
                                ;*TEST 66        TEST "READY" CLEARS WHEN STARTING A "WRITE FILE GAP" FUNCTION
                                ;;*****************************************************************
006104  000004                  TST66:  SCOPE
006106  012767  000005  173052          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006114  000005                          RESET
006116  012701  000001                  MOV     #1,R1                   ;SET TIMER
006122  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006124  104412                          WAITREADY                       ;WAIT FOR "READY"
006126  112714  000001                  MOVB    #WFG!GO,@TACS           ;START A "WRITE FILE GAP"
006132  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
006136  001403                          BEQ     TST67           ;;BR IF "READY" = 0
006140  006301                          ASL     R1                      ;IS TIME UP?
006142  001373                          BNE     1$                      ;BR IF NO
006144  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
                                ;;*****************************************************************
                                ;*TEST 67        TEST THAT "RESET" CANNOT ABORT A "REWIND"
                                ;;*****************************************************************
006146  000004                  TST67:  SCOPE
006150  012767  000001  173010          MOV     #1,$TIMES       ;;DO 1 ITERATION
006156  012767  006234  173004          MOV     #5$,$ESCAPE     ;;ESCAPE TO 5$ ON ERROR
006164  000005                          RESET
006166  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006170  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006172  112714  000017                  MOVB    #REWIND!GO,@TACS        ;LOAD REWIND AND GO
006176  005067  000010                  CLR     3$
006202  032714  000040          2$:     BIT     #READY,@TACS            ;WAIT FOR READY TO CLEAR
006206  001404                          BEQ     4$
006210  105227                          INCB    (PC)+
006212  000000                  3$:     0
006214  001372                          BNE     2$
006216  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
006220  000005                  4$:     RESET                           ;TRY TO STOP REWIND
006222  050314                          BIS     DRIVE,@TACS             ;SELECT DRIVE
006224  032714  000040                  BIT     #READY,@TACS            ;IS READY SET?
006230  001401                          BEQ     5$                      ;BR IF NO
006232  104001                          ERROR   1                       ;"READY" WAS SET
006234  104412                  5$:     WAITREADY                       ;WAIT ON READY TO SET
                                ;;*****************************************************************
                                ;THIS TEST WILL START A "WRITE FILE GAP" AND WAIT FOR "READY" TO CLEAR
                                ;AS SOON AS "READY" CLEARS A "RESET" IS PERFORMED (WHICH SHOULD
                                ;CLEAR THE FUNCTION AND SET READY) THEN "READY" IS EXAMINED
```

```
                                        ;TO INSURE THAT IT IS SET
                                        ;;************************************************************
                                        ;*TEST 70       TEST "RESET" WILL ABORT A "WRITE FILE GAP"
                                        ;;************************************************************
006236  000004                  TST70:  SCOPE
006240  012767  000005  172720          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006246  012767  006264  172632          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006254  012767  006332  172706          MOV     #TST71,$ESCAPE  ;;ESCAPE TO TEST 71 ON ERROR
006262  000005                          RESET
006264  010314                  1$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
006266  104412                          WAITREADY                       ;WAIT ON READY
006270  112714  000001                  MOVB    #WFG!GO,@TACS           ;START A WRITE FILE GAP FUNCTION
006274  005067  000010                  CLR     3$
006300  032714  000040          2$:     BIT     #READY,@TACS            ;WAIT FOR READY TO CLEAR
006304  001404                          BEQ     4$                      ;BR IF "READY" = 0
006306  105227                          INCB    (PC)+                   ;WASTE SOME TIME
006310  000000                  3$:     0
006312  001372                          BNE     2$
006314  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
006316  000005                  4$:     RESET                           ;KILL WRITE FILE GAP
006320  010314                          MOV     DRIVE,@TACS
006322  032714  000040                  BIT     #READY,@TACS
006326  001001                          BNE     TST71           ;;BR IF "RESET" SET "READY"
006330  104001                          ERROR   1                       ;"READY" DIDN'T SET
                                        ;;************************************************************
                                        ;*TEST 71       TEST "WFG" AND "REWIND" FOR NO ERRORS
                                        ;;************************************************************
006332  000004                  TST71:  SCOPE
006334  012767  000005  172624          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006342  012767  006362  172536          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006350  012767  006424  172612          MOV     #TST72,$ESCAPE  ;;ESCAPE TO TEST 72 ON ERROR
006356  000005                          RESET
006360  010314                          MOV     DRIVE,@TACS
006362  112714  000017          1$:     MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
006366  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006370  005714                          TST     @TACS                   ;IS "ERROR" SET?
006372  100001                          BPL     2$                      ;BR IF NO
006374  104001                          ERROR   1                       ;"ERROR" = 1 AFTER "REWIND"
006376  032714  020000          2$:     BIT     #LEADER,@TACS           ;IS "CLEAR LEADER" = 1?
006402  001001                          BNE     3$                      ;BR IF YES
006404  104001                          ERROR   1                       ;NOT ON "CLEAR LEADER" AFTER "REWIND"
006406  112714  000001          3$:     MOVB    #WFG!GO,@TACS           ;GET OFF OF "CLEAR LEADER"
006412  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006414  032714  120000                  BIT     #ERROR!LEADER,@TACS     ;SHOULD BE OFF OF "CLEAR LEADER"
                                                                        ;WITH NO "ERROR"
006420  001401                          BEQ     .+4                     ;BR IF "WFG" MOVED TAPE WITH NO "ERROR"
006422  104001                          ERROR   1                       ;"WFG" FAILED

                                        ;;************************************************************
                                        ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX,
                                        ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
                                        ;;************************************************************
                                        ;*TEST 72       ROUTINE TO DETERMINE TIME OF WAIT LOOPS
                                        ;;************************************************************
006424  000004                  TST72:  SCOPE
006426  012767  000001  172532          MOV     #1,$TIMES       ;;DO 1 ITERATION
```



```
006434  005737  001100                  TST     @#$PASS                 ;IS THIS THE FIRST PASS?
006440  001021                          BNE     TST73           ;;BR IF NO
006442  000005                          RESET
006444  010314                          MOV     DRIVE,@TACS             ;SELECT THE DRIVE
006446  112714  000017                  MOVB    #REWIND!GO,@TACS        ;START A REWIND
006452  104412                          WAITREADY                       ;WAIT FOR READY
006454  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
006460  104412                          WAITREADY                       ;WAIT ON READY
006462  163737  012072  012116          SUB     @#HGHTIM,@#MAXCNT       ;GET THE TIME IT TOOK
006470  005237  012116                  INC     @#MAXCNT                ;MAKE IT BIGGER
006474  006137  012116                  ROL     @#MAXCNT
006500  006137  012116                  ROL     @#MAXCNT
```

```
                                ;////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS ARE USED TO CHECK VARIOUS OPERATIONS WHEN AT "CLEAR LEADER"
                                ;////////////////////////////////////////////////////////////////////
                                ;;****************************************************************
                                ;*TEST 73        TEST THAT "CLEAR LEADER" GOES FALSE ON "REWIND"
                                ;;****************************************************************
006504  000004                  TST73:  SCOPE
006506  012767  000005  172452          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006514  012767  006600  172446          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
006522  000005                          RESET
006524  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006526  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO CLEAR LEADER
006532  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006534  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
006540  001001                          BNE     2$                      ;BR IF YES
006542  104001                          ERROR   1                       ;ERROR--NOT AT CLEAR LEADER
006544  005214                  2$:     INC     @TACS                   ;START ANOTHER REWIND
006546  005067  000010                  CLR     4$
006552  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON "READY" TO CLEAR
006556  001404                          BEQ     5$
006560  105227                          INCB    (PC)+
006562  000000                  4$:     0
006564  001372                          BNE     3$
006566  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
006570  032714  020000          5$:     BIT     #LEADER,@TACS           ;DID "CLEAR LEADER" GO FALSE?
006574  001401                          BEQ     6$                      ;BR IF YES
006576  104001                          ERROR   1                       ;CLEAR LEADER DIDN'T CLEAR
006600  005067  172364          6$:     CLR     $ESCAPE
006604  104412                          WAITREADY
                                ;;****************************************************************
                                ;*TEST 74        TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FILE GAP FUNCTION"
                                ;;****************************************************************
006606  000004                  TST74:  SCOPE
006610  012767  000005  172350          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006616  012767  006646  172262          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006624  012767  006722  172336          MOV     #TST75,$ESCAPE  ;;ESCAPE TO TEST 75 ON ERROR
006632  000005                          RESET
006634  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006636  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
006640  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
006644  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006646  012700  000001          1$:     MOV     #1,R0
006652  010314                          MOV     DRIVE,@TACS
006654  112714  000001                  MOVB    #WFG!GO,@TACS           ;START A WRITE FILE GAP FUNCTION
006660  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
006664  001402                          BEQ     4$
006666  006300                          ASL     R0
006670  100373                          BPL     3$
006672  000005                  4$:     RESET
006674  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006676  112714  000001                  MOVB    #WFG!GO,@TACS           ;START A WRITE FILE GAP FUNCTION
006702  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006704  005714                          TST     @TACS                   ;DID ERROR BIT SET?
006706  100401                          BMI     2$                      ;BR IF YES
006710  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
```



```
006712  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
006716  001001                          BNE     TST75           ;;BR IF SET
006720  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
                                ;;****************************************************************
                                ;*TEST 75        TEST "READY" CLEARS WHEN STARTING A "WRITE" FUNCTION
                                ;;****************************************************************
006722  000004                  TST75:  SCOPE
006724  012767  000005  172234          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006732  000005                          RESET
006734  012701  000001                  MOV     #1,R1                   ;SET TIMER
006740  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
006742  104412                          WAITREADY                       ;WAIT FOR "READY"
006744  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A "WRITE"
006750  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
006754  001403                          BEQ     TST76           ;;BR IF "READY" = 0
006756  006301                          ASL     R1                      ;IS TIME UP?
006760  001373                          BNE     1$                      ;BR IF NO
006762  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 76        TEST "READY" CLEARS WHEN STARTING A "READ" FUNCTION
                                ;;****************************************************************
006764  000004                  TST76:  SCOPE
006766  012767  000005  172172          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006774  000005                          RESET
006776  012701  000001                  MOV     #1,R1                   ;SET TIMER
007002  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
007004  104412                          WAITREADY                       ;WAIT FOR "READY"
007006  112714  000005                  MOVB    #READ!GO,@TACS          ;START A "READ"
007012  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
007016  001403                          BEQ     TST77           ;;BR IF "READY" = 0
007020  006301                          ASL     R1                      ;IS TIME UP?
007022  001373                          BNE     1$                      ;BR IF NO
007024  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 77        TEST "READY" CLEARS WHEN STARTING A "BACK SPACE FILE GAP" FUNCTION
                                ;;****************************************************************
007026  000004                  TST77:  SCOPE
007030  012767  000005  172130          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
007036  000005                          RESET
007040  012701  000001                  MOV     #1,R1                   ;SET TIMER
007044  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
007046  104412                          WAITREADY                       ;WAIT FOR "READY"
007050  112714  000007                  MOVB    #BSFG!GO,@TACS          ;START A "BACK SPACE FILE GAP"
007054  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
007060  001403                          BEQ     TST100          ;;BR IF "READY" = 0
007062  006301                          ASL     R1                      ;IS TIME UP?
007064  001373                          BNE     1$                      ;BR IF NO
007066  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
                                ;;****************************************************************
                                ;*TEST 100       TEST "READY" CLEARS WHEN STARTING A "BACK SPACE BLOCK GAP" FUNCTION
                                ;;****************************************************************
007070  000004                  TST100: SCOPE
007072  012767  000005  172066          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
007100  000005                          RESET
007102  012701  000001                  MOV     #1,R1                   ;SET TIMER
007106  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
```

```
1794  007110  104412                          WAITREADY                       ;WAIT FOR "READY"
1795  007112  112714  000011                  MOVB    #BSBG!GO,@TACS          ;START A "BACK SPACE BLOCK GAP"
1796  007116  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
1797  007122  001403                          BEQ     TST101          ;;BR IF "READY" = 0
1798  007124  006301                          ASL     R1                      ;IS TIME UP?
1799  007126  001373                          BNE     1$                      ;BR IF NO
1800  007130  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
1801                                  ;;*****************************************************************
1802                                  ;*TEST 101       TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD FILE GAP" FUNCTION
1803                                  ;;*****************************************************************
1804  007132  000004                  TST101: SCOPE
1805  007134  012767  000005  172024          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1806  007142  000005                          RESET
1807  007144  012701  000001                  MOV     #1,R1                   ;SET TIMER
1808  007150  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1809  007152  104412                          WAITREADY                       ;WAIT FOR "READY"
1810  007154  112714  000013                  MOVB    #SFFG!GO,@TACS          ;START A "SPACE FORWARD FILE GAP"
1811  007160  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
1812  007164  001403                          BEQ     TST102          ;;BR IF "READY" = 0
1813  007166  006301                          ASL     R1                      ;IS TIME UP?
1814  007170  001373                          BNE     1$                      ;BR IF NO
1815  007172  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
1816                                  ;;*****************************************************************
1817                                  ;*TEST 102       TEST "READY" CLEARS WHEN STARTING A "SPACE FORWARD BLOCK GAP" FUNCTION
1818                                  ;;*****************************************************************
1819  007174  000004                  TST102: SCOPE
1820  007176  012767  000005  171762          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1821  007204  000005                          RESET
1822  007206  012701  000001                  MOV     #1,R1                   ;SET TIMER
1823  007212  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1824  007214  104412                          WAITREADY                       ;WAIT FOR "READY"
1825  007216  112714  000015                  MOVB    #SFBG!GO,@TACS          ;START A "SPACE FORWARD BLOCK GAP"
1826  007222  032714  000040          1$:     BIT     #READY,@TACS            ;CHECK "READY" BIT
1827  007226  001403                          BEQ     TST103          ;;BR IF "READY" = 0
1828  007230  006301                          ASL     R1                      ;IS TIME UP?
1829  007232  001373                          BNE     1$                      ;BR IF NO
1830  007234  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
1831                                  ;;*****************************************************************
1832                                  ;*TEST 103       TEST FOR "CLEAR LEADER ERROR" FOR "WRITE FUNCTION"
1833                                  ;;*****************************************************************
1834  007236  000004                  TST103: SCOPE
1835  007240  012767  000005  171720          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1836  007246  012767  007276  171632          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1837  007254  012767  007370  171706          MOV     #TST104,$ESCAPE ;;ESCAPE TO TEST 104 ON ERROR
1838  007262  000005                          RESET
1839  007264  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1840  007266  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1841  007270  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
1842  007274  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1843  007276  012700  000001          1$:     MOV     #1,R0
1844  007302  010314                          MOV     DRIVE,@TACS
1845  007304  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE FUNCTION
1846  007310  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1847  007314  001402                          BEQ     4$
1848  007316  006300                          ASL     R0
1849  007320  100373                          BPL     3$
```



```
1850  007322  000005                  4$:     RESET
1851  007324  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1852  007326  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE FUNCTION
1853  007332  012700  000001                  MOV     #1,R0                   ;WAIT A WHILE FOR XFER REQ.
1854  007336  105714                  5$:     TSTB    @TACS                   ;XFER REQ = 1?
1855  007340  100402                          BMI     6$                      ;BR IF YES
1856  007342  005200                          INC     R0                      ;WAITED LONG ENOUGH?
1857  007344  001374                          BNE     5$                      ;BR IF NO
1858  007346  105715                  6$:     TSTB    @TADB                   ;CLEAR TRANSFER REQ.
1859  007350  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1860  007352  005714                          TST     @TACS                   ;DID ERROR BIT SET?
1861  007354  100401                          BMI     2$                      ;BR IF YES
1862  007356  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
1863  007360  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
1864  007364  001001                          BNE     TST104          ;;BR IF SET
1865  007366  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
1866                                  ;;*****************************************************************
1867                                  ;*TEST 104       TEST FOR "CLEAR LEADER ERROR" FOR "READ FUNCTION"
1868                                  ;;*****************************************************************
1869  007370  000004                  TST104: SCOPE
1870  007372  012767  000005  171566          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1871  007400  012767  007430  171500          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1872  007406  012767  007504  171554          MOV     #TST105,$ESCAPE ;;ESCAPE TO TEST 105 ON ERROR
1873  007414  000005                          RESET
1874  007416  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1875  007420  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1876  007422  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
1877  007426  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1878  007430  012700  000001          1$:     MOV     #1,R0
1879  007434  010314                          MOV     DRIVE,@TACS
1880  007436  112714  000005                  MOVB    #READ!GO,@TACS          ;START A READ FUNCTION
1881  007442  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1882  007446  001402                          BEQ     4$
1883  007450  006300                          ASL     R0
1884  007452  100373                          BPL     3$
1885  007454  000005                  4$:     RESET
1886  007456  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1887  007460  112714  000005                  MOVB    #READ!GO,@TACS          ;START A READ FUNCTION
1888  007464  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1889  007466  005714                          TST     @TACS                   ;DID ERROR BIT SET?
1890  007470  100401                          BMI     2$                      ;BR IF YES
1891  007472  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
1892  007474  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
1893  007500  001001                          BNE     TST105          ;;BR IF SET
1894  007502  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
1895                                  ;;*****************************************************************
1896                                  ;*TEST 105       TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE FILE GAP FUNCTION"
1897                                  ;;*****************************************************************
1898  007504  000004                  TST105: SCOPE
1899  007506  012767  000005  171452          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1900  007514  012767  007544  171364          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1901  007522  012767  007620  171440          MOV     #TST106,$ESCAPE ;;ESCAPE TO TEST 106 ON ERROR
1902  007530  000005                          RESET
1903  007532  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1904  007534  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1905  007536  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
```

```
1906  007542  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1907  007544  012700  000001          1$:     MOV     #1,R0
1908  007550  010314                          MOV     DRIVE,@TACS
1909  007552  112714  000007                  MOVB    #BSFG!GO,@TACS          ;START A BACK SPACE FILE GAP FUNCTION
1910  007556  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1911  007562  001402                          BEQ     4$
1912  007564  006300                          ASL     R0
1913  007566  100373                          BPL     3$
1914  007570  000005                  4$:     RESET
1915  007572  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1916  007574  112714  000007                  MOVB    #BSFG!GO,@TACS          ;START A BACK SPACE FILE GAP FUNCTION
1917  007600  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1918  007602  005714                          TST     @TACS                   ;DID ERROR BIT SET?
1919  007604  100401                          BMI     2$                      ;BR IF YES
1920  007606  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
1921  007610  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
1922  007614  001001                          BNE     TST106          ;;BR IF SET
1923  007616  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
1924                                  ;;*************************************************************
1925                                  ;*TEST 106       TEST FOR "CLEAR LEADER ERROR" FOR "BACK SPACE BLOCK GAP FUNCTION"
1926                                  ;;*************************************************************
1927  007620  000004                  TST106: SCOPE
1928  007622  012767  000005  171336          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1929  007630  012767  007660  171250          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1930  007636  012767  007734  171324          MOV     #TST107,$ESCAPE ;;ESCAPE TO TEST 107 ON ERROR
1931  007644  000005                          RESET
1932  007646  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1933  007650  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1934  007652  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
1935  007656  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1936  007660  012700  000001          1$:     MOV     #1,R0
1937  007664  010314                          MOV     DRIVE,@TACS
1938  007666  112714  000011                  MOVB    #BSBG!GO,@TACS          ;START A BACK SPACE BLOCK GAP FUNCTION
1939  007672  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1940  007676  001402                          BEQ     4$
1941  007700  006300                          ASL     R0
1942  007702  100373                          BPL     3$
1943  007704  000005                  4$:     RESET
1944  007706  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1945  007710  112714  000011                  MOVB    #BSBG!GO,@TACS          ;START A BACK SPACE BLOCK GAP FUNCTION
1946  007714  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1947  007716  005714                          TST     @TACS                   ;DID ERROR BIT SET?
1948  007720  100401                          BMI     2$                      ;BR IF YES
1949  007722  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
1950  007724  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
1951  007730  001001                          BNE     TST107          ;;BR IF SET
1952  007732  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
1953                                  ;;*************************************************************
1954                                  ;*TEST 107       TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD FILE GAP FUNCTION"
1955                                  ;;*************************************************************
1956  007734  000004                  TST107: SCOPE
1957  007736  012767  000005  171222          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1958  007744  012767  007774  171134          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1959  007752  012767  010050  171210          MOV     #TST110,$ESCAPE ;;ESCAPE TO TEST 110 ON ERROR
1960  007760  000005                          RESET
1961  007762  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
```



```
1962  007764  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1963  007766  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
1964  007772  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1965  007774  012700  000001          1$:     MOV     #1,R0
1966  010000  010314                          MOV     DRIVE,@TACS
1967  010002  112714  000013                  MOVB    #SFFG!GO,@TACS          ;START A SPACE FWD FILE GAP FUNCTION
1968  010006  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1969  010012  001402                          BEQ     4$
1970  010014  006300                          ASL     R0
1971  010016  100373                          BPL     3$
1972  010020  000005                  4$:     RESET
1973  010022  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1974  010024  112714  000013                  MOVB    #SFFG!GO,@TACS          ;START A SPACE FWD FILE GAP FUNCTION
1975  010030  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1976  010032  005714                          TST     @TACS                   ;DID ERROR BIT SET?
1977  010034  100401                          BMI     2$                      ;BR IF YES
1978  010036  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
1979  010040  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
1980  010044  001001                          BNE     TST110          ;;BR IF SET
1981  010046  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
1982                                  ;;*************************************************************
1983                                  ;*TEST 110       TEST FOR "CLEAR LEADER ERROR" FOR "SPACE FWD BLOCK GAP FUNCTION"
1984                                  ;;*************************************************************
1985  010050  000004                  TST110: SCOPE
1986  010052  012767  000005  171106          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1987  010060  012767  010110  171020          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1988  010066  012767  010164  171074          MOV     #TST111,$ESCAPE ;;ESCAPE TO TEST 111 ON ERROR
1989  010074  000005                          RESET
1990  010076  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1991  010100  104412                          WAITREADY                       ;MAKE SURE TA11 IS READY
1992  010102  112714  000017                  MOVB    #REWIND+GO,@TACS        ;LOAD REWIND AND GO
1993  010106  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1994  010110  012700  000001          1$:     MOV     #1,R0
1995  010114  010314                          MOV     DRIVE,@TACS
1996  010116  112714  000015                  MOVB    #SFBG!GO,@TACS          ;START A SPACE FWD BLOCK GAP FUNCTION
1997  010122  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY TO CLEAR
1998  010126  001402                          BEQ     4$
1999  010130  006300                          ASL     R0
2000  010132  100373                          BPL     3$
2001  010134  000005                  4$:     RESET
2002  010136  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
2003  010140  112714  000015                  MOVB    #SFBG!GO,@TACS          ;START A SPACE FWD BLOCK GAP FUNCTION
2004  010144  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
2005  010146  005714                          TST     @TACS                   ;DID ERROR BIT SET?
2006  010150  100401                          BMI     2$                      ;BR IF YES
2007  010152  104001                          ERROR   1                       ;"ERROR" BIT (BIT15) FAILED TO SET
2008  010154  032714  020000          2$:     BIT     #LEADER,@TACS           ;HOW ABOUT CLEAR LEADER BIT
2009  010160  001001                          BNE     TST111          ;;BR IF SET
2010  010162  104001                          ERROR   1                       ;"CLEAR LEADER" NOT SET
2011                                  ;;*************************************************************
2012                                  ;*TEST 111       TEST "GO" BIT IS WRITE ONLY
2013                                  ;;*************************************************************
2014  010164  000004                  TST111: SCOPE
2015  010166  012767  000005  170772          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
2016  010174  012767  010206  170704          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
2017  010202  000005                          RESET
```

```
2018  010204  010314                        MOV     DRIVE,@TACS             ;LOAD THE DRIVE
2019  010206  052714  000001          1$:   BIS     #1,@TACS                ;SET "GO" BIT
2020  010212  032714  000001                BIT     #1,@TACS                ;TRY TO READ IT
2021  010216  001401                        BEQ     2$                      ;NO ERROR IF "GO" BIT=0
2022  010220  104001                        ERROR   1                       ;GO BIT WASN'T 0
2023  010222  104412                  2$:   WAITREADY                       ;WAIT ON "READY"
```



```
2024                                    ;////////////////////////////////////////////////////////////////////
2025                                    ;////////////////////////////////////////////////////////////////////
2026                                    ;THE FOLLOWING TESTS CHECKS FOR PROPER OPERATION OF "TRANSFER REQUEST"
2027                                    ;AND "ILBS" FOR BOTH WRITING AND READING
2028                                    ;////////////////////////////////////////////////////////////////////
2029
2030                                    ;;************************************************************
2031                                    ;*TEST 112      TEST TRANSFER REQUEST & ILBS FOR "WRITE"
2032                                    ;;************************************************************
2033  010224  000004                    TST112: SCOPE
2034  010226  012767  000012  170732            MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
2035  010234  012767  010276  170644            MOV     #8$,$LPADR      ;;SET SCOPE LOOP ADDRESS
2036  010242  012767  010374  170720            MOV     #TST113,$ESCAPE ;;ESCAPE TO TEST 113 ON ERROR
2037  010250  000005                            RESET
2038  010252  010314                            MOV     DRIVE,@TACS             ;SELECT DRIVE
2039  010254  104412                            WAITREADY                       ;INSURE DRIVE IS READY
2040  010256  112714  000017                    MOVB    #REWIND+GO,@TACS        ;GO TO BOT
2041  010262  104412                            WAITREADY                       ;GO WAIT FOR "READY" TO SET
2042  010264  112714  000001                    MOVB    #WFG+GO,@TACS           ;GET OFF OF CLEAR LEADER
2043  010270  104412                            WAITREADY                       ;GO WAIT FOR "READY" TO SET
2044  010272  000005                            RESET
2045  010274  010314                            MOV     DRIVE,@TACS
2046  010276  112714  000002          8$:       MOVB    #WRITE,@TACS            ;LOAD "WRITE" FUNCTION
2047  010302  012700  000001                    MOV     #1,R0
2048  010306  006300                    1$:     ASL     R0                      ;KILL A LITTLE TIME
2049  010310  001376                            BNE     1$
2050  010312  105714                            TSTB    @TACS
2051  010314  100001                            BPL     2$                      ;INSURE XFER REQ=0
2052  010316  104001                            ERROR   1                       ;XFER REQ. SET BEFORE "GO"
2053  010320  005214                    2$:     INC     @TACS                   ;SET "GO"
2054  010322  012700  000001                    MOV     #1,R0
2055  010326  006300                    3$:     ASL     R0                      ;KILL A LITTLE TIME
2056  010330  001376                            BNE     3$
2057  010332  105714                            TSTB    @TACS
2058  010334  100401                            BMI     4$                      ;CHECK THAT XFER EQ=1
2059  010336  104001                            ERROR   1                       ;XFER REQ FAILED TO SET
2060  010340  105015                    4$:     CLRB    @TADB                   ;KNOCK DOWN XFER REQ.
2061  010342  105714                            TSTB    @TACS
2062  010344  100001                            BPL     5$                      ;CHECK THAT "XFER REQ" IS CLEAR
2063  010346  104001                            ERROR   1                       ;XFER REQ FAILED TO CLEAR
2064  010350  104413                    5$:     WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
2065  010352  052714  000020                    BIS     #ILBS,@TACS             ;SET ILBS--WRITE CRC
2066  010356  105714                            TSTB    @TACS                   ;DID "XFER REQ" CLEAR?
2067  010360  100001                            BPL     6$                      ;BR IF YES
2068  010362  104001                            ERROR   1                       ;"ILBS" DIDN'T CLEAR "XFER REQ"
2069  010364  104412                    6$:     WAITREADY                       ;GO WAIT FOR "READY" TO SET
2070  010366  005714                            TST     @TACS                   ;ANY ERROR?
2071  010370  100001                            BPL     TST113          ;;BR IF NO
2072  010372  104001                            ERROR   1                       ;AN ERROR OCCURRED
2073                                    ;;************************************************************
2074                                    ;*TEST 113      TEST "TRANSFER REQ" AND "ILBS" FOR "READ"
2075                                    ;;************************************************************
2076  010374  000004                    TST113: SCOPE
2077  010376  012767  000012  170562            MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
2078  010404  012767  010472  170474            MOV     #4$,$LPADR      ;;SET SCOPE LOOP ADDRESS
2079  010412  012767  010534  170550            MOV     #TST114,$ESCAPE ;;ESCAPE TO TEST 114 ON ERROR
```

```
 2080  010420  000005                          RESET
 2081  010422  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
 2082  010424  104412                          WAITREADY                       ;CHECK READY
 2083  010426  052714  000017                  BIS     #REWIND+GO,@TACS
 2084  010432  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
 2085  010434  112714  000001                  MOVB    #WFG!GO,@TACS           ;GET OFF OF "CLEAR LEADER"
 2086  010440  104412                          WAITREADY                       ;GO WAIT ON "READY"
 2087  010442  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE
 2088  010446  104413                          WAITXFER
 2089  010450  112715  000377                  MOVB    #377,@TADB              ;WRITE ON TAPE
 2090  010454  104413                          WAITXFER
 2091  010456  112715  000377                  MOVB    #377,@TADB              ;WRITE ON TAPE
 2092  010462  104413                          WAITXFER
 2093  010464  052714  000020                  BIS     #ILBS,@TACS
 2094  010470  104412                          WAITREADY
 2095  010472  112714  000017          4$:     MOVB    #REWIND!GO,@TACS
 2096  010476  104412                          WAITREADY
 2097  010500  112714  000005                  MOVB    #READ!GO,@TACS          ;START A READ
 2098  010504  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
 2099  010506  005715                          TST     @TADB                   ;CLEAR XFER REQ
 2100  010510  105714                          TSTB    @TACS                   ;DID XFER REQ CLEAR?
 2101  010512  100001                          BPL     2$                      ;BR IF YES
 2102  010514  104001                          ERROR   1                       ;XFER REQ FAILED TO CLEAR
 2103  010516  104413                  2$:     WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
 2104  010520  052714  000020                  BIS     #ILBS,@TACS             ;START ILBS--CLEARS XFER REQ
 2105  010524  105714                          TSTB    @TACS                   ;DID XFER REQ CLEAR?
 2106  010526  100001                          BPL     3$                      ;BR IF YES
 2107  010530  104001                          ERROR   1                       ;XFER REQ FAILED TO CLEAR
 2108  010532  104412                  3$:     WAITREADY                       ;GO WAIT FOR "READY" TO SET
 2109                                  ;;**************************************************************
 2110                                  ;*TEST 114       TEST THAT TADB (READ BUFFER) INITIALIZES TO 0'S
 2111                                  ;;**************************************************************
 2112  010534  000004                  TST114: SCOPE
 2113  010536  012767  010546  170342          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 2114  010544  000005                          RESET
 2115  010546  011500                  1$:     MOV     @TADB,R0                ;PUT TADB IN R0
 2116  010550  001401                          BEQ     TST115          ;;CHECK FOR ALL ZEROS
 2117  010552  104006                          ERROR   6                       ;TADB NOT ALL ZEROS
 2118                                  ;;**************************************************************
 2119                                  ;*TEST 115       TEST TADB (WRITE BUFFER) FOR WRITE ONLY
 2120                                  ;;**************************************************************
 2121  010554  000004                  TST115: SCOPE
 2122  010556  012715  177777                  MOV     #-1,@TADB               ;LOAD TADB WITH 1'S
 2123  010562  011500                          MOV     @TADB,R0                ;PUT TADB IN R0
 2124  010564  001401                          BEQ     TST116          ;;BR IF IT DIDN'T CHANGE
 2125  010566  104006                          ERROR   6                       ;ERROR
```



```
 2126                                  ;;**************************************************************
 2127                                  ;*TEST 116       FLOAT A "1" THROUGH TADB
 2128                                  ;;**************************************************************
 2129  010570  000004                  TST116: SCOPE
 2130  010572  012767  000012  170366          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
 2131  010600  012767  010634  170300          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 2132  010606  012767  010666  170354          MOV     #4$,$ESCAPE     ;;ESCAPE TO 4$ ON ERROR
 2133  010614  000005                          RESET
 2134  010616  010314                          MOV     DRIVE,@TACS
 2135  010620  112714  000017                  MOVB    #REWIND!GO,@TACS        ;POSITION THE TAPE
 2136  010624  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
 2137  010626  112714  000001                  MOVB    #WFG!GO,@TACS           ;GET OFF OF "CLEAR LEADER"
 2138  010632  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
 2139  010634  012700  000001          1$:     MOV     #1,R0                   ;INITIALIZE THE PATTERN
 2140  010640  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE FUNCTION
 2141  010644  104413                          WAITXFER                        ;GO WAIT ON"TRANSFER REQUEST" TO SET
 2142  010646  110015                  2$:     MOVB    R0,@TADB                ;LOAD TADB WRITE BUFFER
 2143  010650  104413                          WAITXFER                        ;WAIT ON "TRANSFER REQUEST"
 2144  010652  011501                          MOV     @TADB,R1                ;READ THE DATA BUFFER
 2145  010654  120001                          CMPB    R0,R1                   ;CHECK THE DATA
 2146  010656  001401                          BEQ     3$                      ;BR IF DATA IS GOOD
 2147  010660  104007                          ERROR   7                       ;ERROR---TADB NOT EQUAL R0
 2148  010662  106300                  3$:     ASLB    R0                      ;NEXT PATTERN
 2149  010664  103370                          BCC     2$                      ;BR IF MORE TO DO
 2150  010666  152714  000020          4$:     BISB    #ILBS,@TACS             ;WRITE CRC
 2151  010672  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
```

```
                                        ;;*****************************************************************
                                        ;*TEST 117      FLOAT A  "0" THROUGH TADB
                                        ;;*****************************************************************
010674  000004                          TST117: SCOPE
010676  012767  000012  170262                  MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
010704  012767  010740  170174                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
010712  012767  010772  170250                  MOV     #4$,$ESCAPE     ;;ESCAPE TO 4$ ON ERROR
010720  000005                                  RESET
010722  010314                                  MOV     DRIVE,@TACS
010724  112714  000017                          MOVB    #REWIND!GO,@TACS        ;POSITION THE TAPE
010730  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
010732  112714  000001                          MOVB    #WFG!GO,@TACS           ;GET OFF OF "CLEAR LEADER"
010736  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
010740  012700  177776          1$:             MOV     #-2,R0                  ;INITIALIZE THE PATTERN
010744  112714  000003                          MOVB    #WRITE!GO,@TACS         ;START A WRITE FUNCTION
010750  104413                                  WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
010752  110015                  2$:             MOVB    R0,@TADB                ;LOAD INTO WRITE BUFFER
010754  104413                                  WAITXFER                        ;WAIT ON "TRANSFER REQUEST"
010756  011501                                  MOV     @TADB,R1                ;READ THE DATA BUFFER
010760  120001                                  CMPB    R0,R1                   ;CHECK THE DATA
010762  001401                                  BEQ     3$                      ;BR IF DATA IS GOOD
010764  104007                                  ERROR   7                       ;ERROR---TADB NOT EQUAL R0
010766  106300                  3$:             ASLB    R0                      ;NEXT PATTERN
010770  103770                                  BCS     2$                      ;BR IF MORE TO DO
010772  152714  000020          4$:             BISB    #ILBS,@TACS             ;WRITE CRC
010776  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
```



```
                                        ;;*****************************************************************
                                        ;*TEST 120      COUNT PATTERN TO TADB
                                        ;;*****************************************************************
011000  000004                          TST120: SCOPE
011002  012767  000012  170156                  MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
011010  012767  011044  170070                  MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
011016  012767  011074  170144                  MOV     #4$,$ESCAPE     ;;ESCAPE TO 4$ ON ERROR
011024  000005                                  RESET
011026  010314                                  MOV     DRIVE,@TACS
011030  112714  000017                          MOVB    #REWIND!GO,@TACS        ;POSITION THE TAPE
011034  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
011036  112714  000001                          MOVB    #WFG!GO,@TACS           ;GET OFF OF "CLEAR LEADER"
011042  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
011044  005000                  1$:             CLR     R0                      ;INITIALIZE THE COUNT PATTERN
011046  112714  000003                          MOVB    #WRITE!GO,@TACS         ;START A WRITE FUNCTION
011052  104413                                  WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
011054  110015                  2$:             MOVB    R0,@TADB                ;LOAD TADB WRITE BUFFER
011056  104413                                  WAITXFER                        ;WAIT ON "TRANSFER REQUEST"
011060  011501                                  MOV     @TADB,R1                ;READ THE DATA BUFFER
011062  120001                                  CMPB    R0,R1                   ;CHECK THE DATA
011064  001401                                  BEQ     3$                      ;BR IF DATA IS GOOD
011066  104005                                  ERROR   5                       ;ERROR---TADB NOT EQUAL R0
011070  105200                  3$:             INCB    R0                      ;UPDATE THE PATTERN
011072  001370                                  BNE     2$                      ;LOOP IF NON-ZERO
011074  152714  000020          4$:             BISB    #ILBS,@TACS             ;WRITE CRC
011100  104412                                  WAITREADY                       ;GO WAIT FOR "READY" TO SET
```

```
                                        ;;************************************************************
                                        ;*TEST 121       END OF TEST CODE
                                        ;;************************************************************
011102  000004                          TST121: SCOPE
011104  012767  000001  170054                  MOV     #1,$TIMES       ;;DO 1 ITERATION
011112  000005                                  RESET
011114  010314                                  MOV     DRIVE,@TACS             ;SELECT DRIVE
011116  112714  000017                          MOVB    #REWIND!GO,@TACS
011122  104412                                  WAITREADY
                                        .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.

011124                                  $EOP:
011124  000004                                  SCOPE
011126  005067  167750                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
011132  005067  170030                          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
011136  005267  167736                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
011142  042767  100000  167730                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
011150  005327                                  DEC     (PC)+           ;;LOOP?
011152  000001                          $EOPCT: .WORD   1
011154  003015                                  BGT     $DOAGN          ;;YES
011156  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
011160  000001                          $ENDCT: .WORD   1
011162  011152                                  $EOPCT
011164  104401  011217                          TYPE    ,$ENDMG         ;;TYPE "END PASS"
011170  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
011174  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
011176  000005                                  RESET                   ;;CLEAR THE WORLD
011200  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
011202  000240                                  NOP                     ;;SAVE ROOM
011204  000240                                  NOP                     ;;FOR
011206  000240                                  NOP                     ;;ACT11
011210                                  $DOAGN:
011210  000137                                  JMP     @(PC)+          ;;RETURN
011212  002234                          $RTNAD: .WORD   START
011214     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
011217     015  042412  042116          $ENDMG: .ASCIZ  <15><12>/END PASS/
011224  050040  051501  000123
```



```
                                        .SBTTL  SCOPE HANDLER ROUTINE

                                        ;;************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<7:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT

011232                                  $SCOPE:
011232  104406                                  CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
011234  032777  040000  167676          1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
011242  001111                                  BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
011244  000416                          $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
011246  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
011252  012737  011272  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
011260  005737  177060                          TST     @#177060        ;;TIME OUT ON XOR?
011264  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
011270  000463                                  BR      $SVLAD          ;;GO TO THE NEXT TEST
011272  022626                          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
011274  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
011300  000423                                  BR      7$              ;;LOOP ON THE PRESENT TEST
011302                                  6$:;#####END OF CODE FOR THE XOR TESTER#####
011302  032777  000400  167630                  BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
011310  001404                                  BEQ     2$              ;;BR IF NO
011312  127767  167622  167562                  CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
011320  001462                                  BEQ     $OVER           ;;BR IF YES
011322  105767  167555          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
011326  001421                                  BEQ     3$              ;;BR IF NO
011330  126767  167561  167545                  CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
011336  101015                                  BHI     3$              ;;BR IF NO
011340  032777  001000  167572                  BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
011346  001404                                  BEQ     4$              ;;BR IF NO
011350  016767  167534  167530          7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
011356  000443                                  BR      $OVER
011360  105067  167517          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
011364  005067  167576                          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
011370  000415                                  BR      1$              ;;ESCAPE TO THE NEXT TEST
011372  032777  004000  167540          3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
011400  001011                                  BNE     1$              ;;BR IF YES
011402  005767  167472                          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
011406  001406                                  BEQ     1$              ;;      INHIBIT ITERATIONS
011410  005267  167470                          INC     $ICNT           ;;INCREMENT ITERATION COUNT
011414  026767  167546  167462                  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
011422  002021                                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
011424  012767  000001  167452          1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
011432  016767  000044  167526                  MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
011440  105267  167436          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
011444  011667  167436                          MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
```

```
 2305  011450  011667  167434                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 2306  011454  005067  167510                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 2307  011460  112767  000001  167427          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 2308  011466  016777  167410  167446  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 2309  011474  016716  167406                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 2310  011500  000002                          RTI                     ;;FIXES PS
 2311  011502  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
```



```
 2312                                  .SBTTL  ERROR HANDLER ROUTINE
 2313
 2314                                  ;;**************************************************************
 2315                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 2316                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 2317                                  ;*AND GO TO TYPERR ON ERROR
 2318                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 2319                                  ;*SW15=1        HALT ON ERROR
 2320                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
 2321                                  ;*SW10=1        BELL ON ERROR
 2322                                  ;*SW09=1        LOOP ON ERROR
 2323                                  ;*CALL
 2324                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 2325
 2326  011504                          $ERROR:
 2327  011504  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 2328  011506  011437  001162                  MOV     @TACS,@#$REG0           ;SAVE THE STATUS REG.
 2329  011512  011537  001164                  MOV     @TADB,@#$REG1           ;SAVE THE DATA BUFFER
 2330  011516  010037  001124          ERROR1: MOV     R0,@#$GDDAT             ;R0 WILL CONTAIN THE GOOD DATA
 2331  011522  010137  001126                  MOV     R1,@#$BDDAT             ;R1 WILL CONTAIN THE BAD DATA
 2332  011526  105267  167351          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 2333  011532  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 2334  011534  016777  167342  167400          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 2335  011542  032777  002000  167370          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 2336  011550  001402                          BEQ     1$              ;;NO - SKIP
 2337  011552  104401  001172                  TYPE    ,$BELL          ;;RING BELL
 2338  011556  005267  167330          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 2339  011562  011667  167330                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 2340  011566  162767  000002  167322          SUB     #2,$ERRPC
 2341  011574  117767  167316  167312          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 2342  011602  032777  020000  167330          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 2343  011610  001004                          BNE     20$             ;;SKIP TYPEOUTS
 2344  011612  004767  000060                  JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
 2345  011616  104401  001177                  TYPE    ,$CRLF
 2346  011622                          20$:
 2347  011622  005777  167312          2$:     TST     @SWR            ;;HALT ON ERROR
 2348  011626  100002                          BPL     3$              ;;SKIP IF CONTINUE
 2349  011630  000000                          HALT                    ;;HALT ON ERROR!
 2350  011632  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
 2351  011634  032777  001000  167276  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 2352  011642  001402                          BEQ     4$              ;;BR IF NO
 2353  011644  016716  167240                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 2354  011650  005767  167314          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 2355  011654  001402                          BEQ     5$              ;;BR IF NONE
 2356  011656  016716  167306                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 2357  011662                          5$:
 2358  011662  022737  011200  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
 2359  011670  001001                          BNE     6$              ;;BRANCH IF NO
 2360  011672  000000                          HALT                    ;;YES
 2361  011674                          6$:
 2362  011674  000002                          RTI                     ;;RETURN
 2363
```

```
                                ;;*****************************************************************
                                ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES

011676  104401  001177          TYPERR: TYPE    ,$CRLF                  ;TYPE A CARRIAGE RETURN & LINE FEED
011702  010046                          MOV     R0,-(SP)                ;SAVE R0
011704  113700  001114                  MOVB    @#$ITEMB,R0             ;PICKUP THE ITEM INDEX
011710  005300                          DEC     R0                      ;ADJUST THE INDEX
011712  006300                          ASL     R0                      ;SO IT WILL WORK FOR
011714  006300                          ASL     R0                      ;THE ERROR TABLE
011716  006300                          ASL     R0
011720  062700  001236                  ADD     #$ERRTB,R0              ;FORM THE TABLE POINTER
011724  012067  000002                  MOV     (R0)+,1$                ;PICKUP "ERROR MESSAGE" POINTER
011730  104401                          TYPE                            ;TYPE "ERROR MESSAGE"
011732  000000                  1$:     0                               ;"ERROR MESSAGE POINTER" GOES HERE
011734  104401  001177                  TYPE    ,$CRLF
011740  012067  000004                  MOV     (R0)+,2$                ;PICKUP "DATA HEADER" POINTER
011744  001404                          BEQ     3$                      ;IF "0" DON'T TYPE
011746  104401                          TYPE                            ;TYPE "DATA HEADER
011750  000000                  2$:     0                               ;"DATA HEADER" POINTER GOES HERE
011752  104401  001177                  TYPE    ,$CRLF
011756  012000                  3$:     MOV     (R0)+,R0                ;PICKUP "DATA POINTER"
011760  001004                          BNE     5$                      ;IF THERE IS DATA TO TYPE GO DO IT
011762  012600                  4$:     MOV     (SP)+,R0                ;RESTORE R0
011764  104401  001177                  TYPE    ,$CRLF                  ;TYPE A CARRAGE RETURN&LINE FEED
011770  000207                          RTS     PC                      ;RETURN
011772                          5$:
011772  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                                                                ;;TYPE DATA
011774  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
011776  005710                          TST     (R0)                    ;TERMINATOR?
012000  001770                          BEQ     4$                      ;BR IF YES
012002  104401  012010                  TYPE    ,6$                     ;TYPE 2 SPACES
012006  000771                          BR      5$                      ;LOOP
012010  020040    000           6$:     .ASCIZ  /  /                    ;ASCII STRING OF 2 SPACES
        012014                          .EVEN
```



```
                                ;;*****************************************************************
                                ;ROUTINE TO WAIT ON THE READY BIT

012014                          WAIT.ON.READY:
012014  005067  000044                  CLR     WAIT2                   ;SETUP MAX. TIME TO WAIT ON "READY"
012020  016767  000072  000044          MOV     MAXCNT,HGHTIM
012026  012637  001202                  MOV     (SP)+,@#SAVPC           ;GET THE PC OF THE WAITREADY INSTRUCTION
012032  162737  000002  001202          SUB     #2,@#SAVPC
012040  012637  001204                  MOV     (SP)+,@#SAVPS           ;SAVE THE PS
012044  032714  000040          WAIT1:  BIT     #READY,@TACS            ;READY=1?
012050  001013                          BNE     WAIT3                   ;GO ON IF YES
012052  105714                          TSTB    @TACS                   ;CHECK TRANSFER REQUEST
012054  100002                          BPL     WAIT4
012056  104004                          ERROR   4                       ;"TRANSFER REQUEST" SET WHILE WAITING ON "READY"
012060  000407                          BR      WAIT3
012062  005227                  WAIT4:  INC     (PC)+                   ;COUNT FAST COUNTER
012064  000000                  WAIT2:  0
012066  001366                          BNE     WAIT1                   ;GO CHECK "READY" AGAIN
012070  005327                          DEC     (PC)+                   ;COUNT LOOP COUNTER
012072  000000                  HGHTIM: 0
012074  003363                          BGT     WAIT1                   ;GO LOOP AGAIN
012076  104002                          ERROR   2                       ;"READY" FAILED TO SET
012100  013746  001204          WAIT3:  MOV     @#SAVPS,-(SP)           ;GET THE STATUS BACK
012104  013746  001202                  MOV     @#SAVPC,-(SP)           ;GET THE PC
012110  062716  000002                  ADD     #2,(SP)
012114  000002                          RTI
012116  000000                  MAXCNT: 0

                                ;;*****************************************************************
                                ;ROUTINE TO WAIT ON TRANSFER REQUEST

012120                          WAIT.FOR.XFER.REQ:
012120  005067  000044                  CLR     2$                      ;SETUP WASTE TIME LOOP
012124  013767  012116  000044          MOV     @#MAXCNT,3$
012132  012637  001202                  MOV     (SP)+,@#SAVPC           ;GET THE PC OF THE WAITXFER INSTRUCTION
012136  162737  000002  001202          SUB     #2,@#SAVPC
012144  012637  001204                  MOV     (SP)+,@#SAVPS           ;SAVE THE PS
012150  105714                  1$:     TSTB    @TACS                   ;CHECK XFER REQ
012152  100414                          BMI     4$                      ;EXIT IF SET
012154  032714  000040                  BIT     #READY,@TACS            ;LOOK AT READY
012160  001402                          BEQ     5$                      ;BR IF "READY" ISN'T SET
012162  104004                          ERROR   4                       ;"READY" SET WHILE WAITING FOR "XFER REQ"
012164  000407                          BR      4$
012166  005227                  5$:     INC     (PC)+                   ;COUNT
012170  000000                  2$:     0
012172  001366                          BNE     1$                      ;BR IF MORE TO DO
012174  005327                          DEC     (PC)+
012176  000000                  3$:     0
012200  003363                          BGT     1$
012202  104003                          ERROR   3                       ;"TRANSFER REQUEST" FAILED TO SET
012204  013746  001204          4$:     MOV     @#SAVPS,-(SP)           ;GET THE STATUS BACK
012210  013746  001202                  MOV     @#SAVPC,-(SP)           ;GET THE PC
012214  062716  000002                  ADD     #2,(SP)
012220  000002                          RTI                             ;GO BACK
```

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

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

012222  104401  015570          ASKDRV: TYPE    ,MSGDRV                 ;<CRLF>"DRIVE(S)? "
012226  005067  166772                  CLR     DRVKEY
012232  104410                          RDLIN                           ;GO GET A DRIVE
012234  012600                          MOV     (SP)+,R0                ;SETUP TO CHECK FOR VALID DRIVE(S)
012236  105710                          TSTB    @R0                     ;WAS A DRIVE SELECTED?
012240  001425                          BEQ     NOTLGL                  ;BR IF NO
012242  012701  001224                  MOV     #DRVKEY,R1
012246  122710  000101          LOOP:   CMPB    #'A,@R0                 ;WAS DRIVE "A" SELECTED?
012252  001002                          BNE     NOTA                    ;BR IF NO
012254  112021                          MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "A"
012256  000411                          BR      NEXT
012260  122710  000102          NOTA:   CMPB    #'B,@R0                 ;WAS DRIVE "B" SELECTED?
012264  001002                          BNE     NOTB                    ;BR IF NO
012266  112021                          MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "B"
012270  000404                          BR      NEXT
012272  122710  000054          NOTB:   CMPB    #54,@R0                 ;WAS A COMMA TYPED?
012276  001006                          BNE     NOTLGL                  ;BR IF NO
012300  105720                          TSTB    (R0)+                   ;DUMP THE COMMA
012302  105710          NEXT:   TSTB    @R0                     ;TERMINATOR?
012304  001406                          BEQ     EXIT                    ;BR IF YES
012306  022701  001226                  CMP     #DRVKEY+2,R1            ;TWO DRIVES SELECTED?
012312  101355                          BHI     LOOP                    ;BR IF NO
012314  104401  001176          NOTLGL: TYPE    ,$QUES                  ;ILLEGAL INPUT DETECTED
012320  000740                          BR      ASKDRV                  ;GO TRY AGAIN
012322  005767  166676          EXIT:   TST     DRVKEY                  ;ANY DRIVE SELECTED?
012326  001772                          BEQ     NOTLGL                  ;BR IF NO
012330  000207                          RTS     PC
```



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

012332  010046                  ASKADR: MOV     R0,-(SP)                ;SAVE R0
012334  104401  015603          1$:     TYPE    ,MSGASK                 ;"TACS?"
012340  104411                          RDOCT                           ;GET VALUE
012342  012600                          MOV     (SP)+,R0                ;PICK UP THE OCTAL NUMBER
012344  001423                          BEQ     3$                      ;IF "0" USE OLD VALUES
012346  020027  160000                  CMP     R0,#160000              ;MAKE SURE IT IS A BUS ADDRESS
012352  103770                          BLO     1$
012354  010037  001206                  MOV     R0,@#TACSL              ;SAVE TOE TACS
012360  062700  000002                  ADD     #2,R0                   ;STEP TO TADB ADDRESS
012364  010037  001212                  MOV     R0,@#TADBL              ;AND SAVE IT
012370  013737  001206  001210          MOV     @#TACSL,@#TACSH         ;SETUP TACS UPPER
012376  005237  001210                  INC     @#TACSH                 ;BYTE POINTER

012402  013737  001212  001214          MOV     @#TADBL,@#TADBH         ;SETUP TADB UPPER
012410  005237  001214                  INC             @#TADBH                 ;BYTE POINTER
012414  104401  015613          3$:     TYPE    ,MSGVEC                 ;"VECTOR?"
012420  104411                          RDOCT
012422  012600                          MOV     (SP)+,R0
012424  001411                          BEQ     5$
012426  020027  001000                  CMP     R0,#1000                ;MAKE SUPE ADDRESS IS IN VECTOR AREA
012432  103370                          BHIS    3$
012434  010037  001216                  MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
012440  062700  000002                  ADD     #2,R0
012444  010037  001220                  MOV     R0,@#TAVEC+2
012450  104401  015623          5$:     TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
012454  104411                          RDOCT
012456  012600                          MOV     (SP)+,R0
012460  001413                          BEQ     6$                      ;IF "0" USE OLD VALUE
012462  020027  000007                  CMP     R0,#7                   ;MAKE SURE ITS VALID
012466  101370                          BHI     5$
012470  000300                          SWAB    R0                      ;PUT INTO HIGH BYTE
012472  006200                          ASR     R0                      ;AND SHIFT
012474  006200                          ASR     R0                      ;INTO PROPER
012476  006200                          ASR     R0                      ;POSITION
012500  042700  177437                  BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
012504  010037  001222                  MOV     R0,@#TAPRIO             ;STORE IT AWAY
012510  104401  015635          6$:     TYPE    ,MTACS                  ;TACS="
012514  016746  166466                  MOV     TACSL,-(SP)     ;;SAVE TACSL FOR TYPEOUT
012520  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012522  104401  015643                  TYPE    ,MTADB                  ;"TADB="
012526  016746  166460                  MOV     TADBL,-(SP)     ;;SAVE TADBL FOR TYPEOUT
012532  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012534  104401  015652                  TYPE    ,MTAVEC                 ;"VECTOR="
012540  016746  166452                  MOV     TAVEC,-(SP)     ;;SAVE TAVEC FOR TYPEOUT
012544  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012546  104401  015663                  TYPE    ,MTAPRI                 ;"PRIORITY="
012552  016746  166444                  MOV     TAPRIO,-(SP)    ;;SAVE TAPRIO FOR TYPEOUT
012556  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012560  104401  015676                  TYPE    ,MSGOK                  ;"OK?"
012564  104407                          RDCHR                           ;GO READ ONE CHARACTER
012566  012600                          MOV     (SP)+,R0                ;GET IT
```

```
2545  012570  022700  000015          CMP     #15,R0                  ;IS IT "CR"?
2546  012574  001406                  BEQ     7$                      ;BRANCH IF YES
2547  012576  022700  000131          CMP     #'Y,R0                  ;IS IT "Y"?
2548  012602  001403                  BEQ     7$                      ;IT WAS
2549  012604  104401  001176          TYPE    ,$QUES                  ;TYPE "?"
2550  012610  000651                  BR      1$                      ;AND LET HIM CORRECT THEM
2551  012612  104401  015704  7$:     TYPE    ,MYES                   ;TYPE OUT "YES"
2552  012616  012600                  MOV     (SP)+,R0                ;RESTORE R0
2553  012620  000207                  RTS     PC                      ;AND RETURN
```



```
2554                                ;////////////////////////////////////////////////////////////////////////
2555                                ;////////////////////////////////////////////////////////////////////////
2556                                ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60
2557                                ;NOTE:  ### BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ###
2558                                ;////////////////////////////////////////////////////////////////////////
2559
2560
2561
2562                                ;;*************************************************************
2563                                ;       WRITE FILE GAPS FROM "BOT" TO "EOT"
2564                                        ;START AT 220
2565                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
2566                                        ;THE "WRITE DELAY MONO".
2567                                ;;*************************************************************
2568  012622  012706  001100  WFGSUB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2569  012626  013704  001206          MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2570  012632  013705  001212          MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2571  012636  000005                  RESET                           ;RESET THE WORLD
2572  012640  012737  012622  001110  MOV     #WFGSUB,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2573  012646  004737  013626          JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2574  012652  010314                  MOV     DRIVE,@TACS             ;SELECT DRIVE
2575  012654  112714  000017          MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2576  012660  032714  000040  100$:   BIT     #READY,@TACS            ;WAIT ON READY
2577  012664  001775                  BEQ     100$
2578  012666  112714  000001  1$:     MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
2579  012672  104412                  WAITREADY                       ;WAIT ON READY
2580  012674  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2581  012700  001772                  BEQ     1$                      ;BR IF NO
2582  012702  000000                  HALT                            ;STOP IF YES
2583  012704  000746                  BR      WFGSUB                  ;LOOP ON CONT.
2584
2585
2586                                ;;*************************************************************
2587                                ;       WRITE CONTINUOUS BLOCKS OF DATA
2588                                        ;START AT 224
2589                                        ;THE PROGRAM WILL HALT THREE(3) TIMES
2590                                        ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2591                                        ;****  IF USING SOFTWARE SWITCH REGISTER
2592                                        ;       PRESS CONTINUE   AND
2593                                        ;      PROGRAM WILL TYPE "SWR=XXXXXX NEW="
2594                                        ;       AND ALLOW A NEW VALUE TO BE INPUT
2595                                        ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2596                                        ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2597                                        ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2598                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
2599                                ;;*************************************************************
2600  012706  004737  013370  WRTSUB: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
2601  012712  012706  001100  WLOOP:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2602  012716  013704  001206          MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2603  012722  013705  001212          MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2604  012726  000005                  RESET                           ;RESET THE WORLD
2605  012730  012737  012712  001110  MOV     #WLOOP,@#$LPERR         ;SETUP THE LOOP ON ERROR ADDRESS
2606  012736  004737  013626          JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2607  012742  010314                  MOV     DRIVE,@TACS             ;SELECT DRIVE
2608  012744  112714  000017          MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2609  012750  032714  000040  100$:   BIT     #READY,@TACS            ;WAIT ON READY
```

```
2610  012754  001775                    BEQ     100$
2611  012756  004737  013504    1$:     JSR     PC,@#WRTBLK             ;WRITE A BLOCK
2612  012762  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2613  012766  001773                    BEQ     1$                      ;BR IF NO
2614  012770  000000                    HALT                            ;STOP IF "EOT"
2615  012772  000747                    BR      WLOOP                   ;LOOP IF CONT.
2616
2617
2618                              ;;*************************************************************
2619                              ;       READ CONTINUOUS BLOCKS OF DATA
2620                              ;START AT 230
2621                              ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
2622                              ;AND THE "THRESHOLD POT".
2623                              ;;*************************************************************
2624  012774  012706  001100    RDSUB:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2625  013000  013704  001206            MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2626  013004  013705  001212            MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2627  013010  000005                    RESET                           ;RESET THE WORLD
2628  013012  012737  012774  001110    MOV     #RDSUB,@#$LPERR         ;SETUP THE LOOP ON ERROR ADDRESS
2629  013020  004737  013626            JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2630  013024  010314                    MOV     DRIVE,@TACS             ;SELECT DRIVE
2631  013026  112714  000017            MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2632  013032  032714  000040    100$:   BIT     #READY,@TACS            ;WAIT ON READY
2633  013036  001775                    BEQ     100$
2634  013040  004737  013546    1$:     JSR     PC,@#RDBLK              ;READ A BLOCK
2635  013044  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2636  013050  001351                    BNE     RDSUB                   ;BR IF YES---LOOP
2637  013052  000772                    BR      1$
2638
2639
2640                              ;;*************************************************************
2641                              ;       WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2642                              ;START AT 234
2643                              ;THE PROGRAM WILL HALT THREE(3) TIMES
2644                              ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2645                              ;****  IF USING SOFTWARE SWITCH REGISTER
2646                              ;       PRESS CONTINUE   AND
2647                              ;      PROGRAM WILL TYPE "SWR=XXXXXX NEW="
2648                              ;       AND ALLOW A NEW VALUE TO BE INPUT
2649                              ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2650                              ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2651                              ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2652                              ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
2653                              ;AND THE "GAP TIME MONO".
2654                              ;;*************************************************************
2655  013054  004737  013370    WGPBLK: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
2656  013060  012706  001100    WGBLOP: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2657  013064  013704  001206            MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2658  013070  013705  001212            MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2659  013074  000005                    RESET                           ;RESET THE WORLD
2660  013076  012737  013060  001110    MOV     #WGBLOP,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2661  013104  004737  013626            JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2662  013110  010314                    MOV     DRIVE,@TACS             ;SELECT DRIVE
2663  013112  112714  000017            MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2664  013116  032714  000040    100$:   BIT     #READY,@TACS            ;WAIT ON READY
2665  013122  001775                    BEQ     100$
```



```
2666  013124  112714  000001    1$:     MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
2667  013130  104412                    WAITREADY                       ;WAIT ON READY
2668  013132  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2669  013136  001005                    BNE     2$                      ;BR IF YES
2670  013140  004737  013504            JSR     PC,@#WRTBLK             ;WRITE A BLOCK
2671  013144  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2672  013150  001765                    BEQ     1$                      ;BR IF NO
2673  013152  000000    2$:             HALT                            ;STOP AT "EOT"
2674  013154  000741                    BR      WGBLOP                  ;START OVER ON CONT.
2675
2676
2677                              ;;*************************************************************
2678                              ;       READ A BLOCK OF DATA AND A FILE GAP
2679                              ;START AT 240
2680                              ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
2681                              ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
2682                              ;AND THE "TAPE BLANK MONO".
2683                              ;;*************************************************************
2684  013156  012706  001100    RGPBLK: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2685  013162  013704  001206            MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2686  013166  013705  001212            MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2687  013172  000005                    RESET                           ;RESET THE WORLD
2688  013174  012737  013156  001110    MOV     #RGPBLK,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2689  013202  004737  013626            JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2690  013206  010314                    MOV     DRIVE,@TACS             ;SELECT DRIVE
2691  013210  112714  000017            MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2692  013214  032714  000040    100$:   BIT     #READY,@TACS            ;WAIT ON READY
2693  013220  001775                    BEQ     100$
2694  013222  004737  013546    1$:     JSR     PC,@#RDBLK              ;READ A BLOCK OF DATA
2695  013226  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2696  013232  001351                    BNE     RGPBLK                  ;BR IF YES
2697  013234  112714  000015            MOVB    #SFBG!GO,@TACS          ;GET INTO A FILE GAP
2698  013240  104412                    WAITREADY
2699  013242  032714  020000            BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2700  013246  001343                    BNE     RGPBLK                  ;BR IF YES
2701  013250  000764                    BR      1$                      ;LOOP
2702
2703
2704                              ;;*************************************************************
2705                              ;       SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2706                              ;START AT 244
2707                              ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
2708                              ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
2709                              ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
2710                              ;(SIGNAL MONO CAN BE CHECKED).
2711                              ;;*************************************************************
2712  013252  012706  001100    SFFGSB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2713  013256  013704  001206            MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2714  013262  013705  001212            MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2715  013266  000005                    RESET                           ;RESET THE WORLD
2716  013270  012737  013252  001110    MOV     #SFFGSB,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2717  013276  004737  013626            JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2718  013302  010314                    MOV     DRIVE,@TACS             ;SELECT DRIVE
2719  013304  112714  000017            MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2720  013310  032714  000040    100$:   BIT     #READY,@TACS            ;WAIT ON READY
2721  013314  001775                    BEQ     100$
```

```
2722  013316  112714  000013          1$:     MOVB    #SFFG!GO,@TACS          ;SPACE INTO A FILE GAP
2723  013322  104412                          WAITREADY                       ;WAIT ON READY
2724  013324  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2725  013330  001772                          BEQ     1$                      ;BR IF NO
2726  013332  000000                          HALT                            ;STOP AT "EOT"
2727  013334  000746                          BR      SFFGSB                  ;LOOP ON CONT.
2728
2729
2730                                  ;;**************************************************************
2731                                  ;       BACK SPACE FILE GAP
2732                                  ;START AT 250
2733                                  ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
2734                                  ;;**************************************************************
2735  013336  000005          BSFGSB: RESET                                   ;RESET THE WORLD
2736  013340  012737  013336  001110          MOV     #BSFGSB,@#$LPERR                ;LOOP ON ERROR ADDRESS
2737  013346  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
2738  013350  112714  000007          1$:     MOVB    #BSFG!GO,@TACS          ;BACK SPACE A FILE GAP
2739  013354  104412                          WAITREADY                       ;WAIT ON READY
2740  013356  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2741  013362  001772                          BEQ     1$                      ;BR IF NO
2742  013364  000000                          HALT                            ;STOP AT BOT
2743  013366  000763                          BR      BSFGSB                  ;START OVER ON CONT.
2744
2745
2746                                  ;;**************************************************************
2747                                  ;       SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2748                                  ;;**************************************************************
2749  013370  005000          SETBUF: CLR     R0
2750  013372  000000                          HALT                            ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
2751  013374  022767  000176  165536          CMP     #SWREG,SWR              ;SOFTWARE SWITCH REG.?
2752  013402  001001                          BNE     20$                     ;NO-LET HIM BE
2753  013404  104405                          GTSWR                           ;GET NEW VALUE
2754  013406                          20$:                                    ;CONTINUE
2755  013406  157700  165526                  BISB    @SWR,R0                 ;PICKUP THE BYTE COUNT
2756  013412  001006                          BNE     2$                      ;BR IF NON-ZERO
2757  013414  105777  165521                  TSTB    @SWR+1                  ;CHECK IF GREATER THAN 377
2758  013420  001402                          BEQ     1$                      ;BR IF NO
2759  013422  012700  000376                  MOV     #376,R0                 ;SET FOR MAX ALLOWED
2760  013426  005200                  1$:     INC     R0                      ;MAKE IT 377 OR 1
2761  013430  010037  013500          2$:     MOV     R0,@#BLKLIM             ;SETUP THE BLOCK LIMIT
2762  013434  005037  013502                  CLR     @#PATTRN
2763  013440  000000                          HALT                            ;OPERATOR PUTS PATTERN IN SWR<7:0>
2764  013442  022767  000176  165470          CMP     #SWREG,SWR              ;SOFTWARE SWITCH REG.?
2765  013450  001001                          BNE     21$                     ;NO-LET HIM BE
2766  013452  104405                          GTSWR                           ;GET NEW VALUE
2767  013454                          21$:                                    ;CONTINUE
2768  013454  117737  165460  013502          MOVB    @SWP,@#PATTRN           ;PICK UP THE PATTERN
2769  013462  000000                          HALT                            ;SET OPERATIONAL SWITCHES
2770  013464  022767  000176  165446          CMP     #SWREG,SWR              ;SOFTWARE SWITCH REG.?
2771  013472  001001                          BNE     22$                     ;NO-LET HIM BE
2772  013474  104405                          GTSWR                           ;GET NEW VALUE
2773  013476                          22$:                                    ;CONTINUE
2774  013476  000207                          RTS     PC                      ;RETURN
2775  013500  000000                  BLKLIM: 0                               ;READ AND WRITE BLOCK SIZE
2776  013502  000000                  PATTRN: 0                               ;PATTERN TO GO ON THE TAPE
2777
```



```
2778
2779                                  ;;**************************************************************
2780                                  ;       WRITE ROUTINE FOR THE MANUAL OPERATIONS
2781                                  ;;**************************************************************
2782  013504  013701  013500  WRTBLK: MOV     @#BLKLIM,R1             ;PICKUP THE BLOCK SIZE
2783  013510  112714  000003          MOVB    #WRITE!GO,@TACS         ;START A WRITE
2784  013514  104413          1$:     WAITXFER                        ;WAIT ON TRANSFER REQUEST
2785  013516  032714  000040          BIT     #READY,@TACS            ;DID READY SET?
2786  013522  001010                  BNE     3$                      ;BR IF YES
2787  013524  005301                  DEC     R1                      ;COUNT THIS REQUEST
2788  013526  002403                  BLT     2$                      ;BR IF TIME FOR ILBS
2789  013530  113715  013502          MOVB    @#PATTRN,@TADB          ;PUT DATA ON TAPE
2790  013534  000767                  BR      1$                      ;LOOP
2791  013536  052714  000020  2$:     BIS     #ILBS,@TACS             ;WRITE CRC AND SHUT DOWN
2792  013542  104412                  WAITREADY                       ;WAIT ON THE READY FLAG
2793  013544  000207          3$:     RTS     PC
2794
2795
2796                                  ;;**************************************************************
2797                                  ;       READ ROUTINE FOR THE MANUAL OPERATIONS
2798                                  ;;**************************************************************
2799  013546  013702  013500  RDBLK:  MOV     @#BLKLIM,R2             ;PICKUP THE BLOCK SIZE
2800  013552  013700  013502          MOV     @#PATTRN,R0             ;USE THIS DATA PATTERN TO COMPARE TO
2801  013556  112714  000005          MOVB    #READ!GO,@TACS          ;START A READ
2802  013562  104413          1$:     WAITXFER                        ;WAIT ON TRANSFER REQUEST
2803  013564  032714  000040          BIT     #READY,@TACS            ;IS READY SET?
2804  013570  001012                  BNE     3$                      ;BR IF YES
2805  013572  005302                  DEC     R2                      ;COUNT THIS REQUEST
2806  013574  002405                  BLT     2$                      ;BR IF TIME FOR ILBS
2807  013576  011501                  MOV     @TADB,R1                ;READ THE DATA BUFFER
2808  013600  120001                  CMPB    R0,R1                   ;CHECK THE DATA
2809  013602  001767                  BEQ     1$                      ;BR IF OK
2810  013604  104007                  ERROR   7                       ;BAD DATA
2811  013606  000406                  BR      4$                      ;GET OUT
2812  013610  052714  000020  2$:     BIS     #ILBS,@TACS             ;READ ILBS
2813  013614  104412                  WAITREADY                       ;WAIT ON READY
2814  013616  005714          3$:     TST     @TACS                   ;CHECK FOR ERROR
2815  013620  100001                  BPL     4$                      ;BR IF NONE
2816  013622  104001                  ERROR   1                       ;ERROR OCCURRED
2817  013624  000207          4$:     RTS     PC                      ;RETURN
2818
2819
2820                                  ;;**************************************************************
2821                                  ;       ROUTINE TO CHANGE DRIVES
2822  013626  105777  165306  NXTDRV: TSTB    @SWR                    ;IS SW07 ON A (1)?
2823  013632  100416                  BMI     3$                      ;BR IF YES
2824  013634  005003                  CLR     DRIVE                   ;SET DRIVE TO "A"
2825  013636  013701  001230          MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
2826  013642  122127  000101          CMPB    (R1)+,#'A               ;IS IT DRIVE "A"?
2827  013646  001402                  BEQ     1$                      ;BR IF YES
2828  013650  012703  000400          MOV     #UNIT,DRIVE             ;SET DRIVE TO "B"
2829  013654  105711          1$:     TSTB    (R1)                    ;LAST DRIVE BEEN SELECTED
2830  013656  001002                  BNE     2$                      ;BR IF NO
2831  013660  012701  001224          MOV     #DRVKEY,R1              ;RESET DRIVE POINTER
2832  013664  010137  001230  2$:     MOV     R1,@#DRVPNT             ;SAVE DRIVE POINTER FOR NEXT TIME
2833  013670  000207          3$:     RTS     PC                      ;GO BACK
```

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

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

013672  013701  001206          EXAM:   MOV     @#TACSL,R1              ;PICKUP THE "CONTROL & STATUS" REG. ADR.
013676  005011                          CLR     (R1)                    ;DRIVE="A", FUNCTION="WFG"
013700  122710  000101                  CMPB    #'A,(R0)                ;EXAMINE DRIVE "A"?
013704  001402                          BEQ     1$                      ;BR IF YES
013706  052711  000400                  BIS     #UNIT,(R1)              ;SELECT DRIVE "B"
013712  032711  000040          1$:     BIT     #READY,(R1)             ;WAIT ON READY
013716  001775                          BEQ     1$
013720  005711                          TST     (R1)                    ;ANY ERROR?
013722  100024                          BPL     4$                      ;BR IF NO
013724  032711  001000                  BIT     #OFFLINE,(R1)           ;ERROR DUE TO "OFF LINE"?
013730  001017                          BNE     3$                      ;BR IF YES
013732  032711  010000                  BIT     #WRTLOCK,(R1)           ;ERROR DUE TO "WRITE LOCK"?
013736  001411                          BEQ     2$                      ;BR IF NO
013740  122777  000201  165172          CMPB    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
013746  001412                          BEQ     4$                      ;BR IF YES
013750  122777  000203  165162          CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
013756  001406                          BEQ     4$                      ;BR IF YES
013760  000403                          BR      3$                      ;TAKE THE ERROR EXIT
013762  032711  020000          2$:     BIT     #LEADER,(R1)            ;ERROR DUE TO "CLEAR LEADER"?
013766  001002                          BNE     4$                      ;BR IF YES
013770  062716  000002          3$:     ADD     #2,(SP)                 ;TAKE ERROR RETURN
013774  000207                  4$:     RTS     PC                      ;RETURN
```



```
                                        .SBTTL  TYPE ROUTINE

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

013776  105767  165155          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
014002  100002                          BPL     1$              ;;BR IF YES
014004  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
014006  000407                          BR      3$              ;;LEAVE
014010  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
014012  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
014016  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
014020  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
014022  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
014024  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
014026  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
014032  000002                          RTI                     ;;RETURN
014034  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
014040  001430                          BEQ     8$
014042  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
014046  001006                          BNE     5$
014050  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
014052  104401                          TYPE                    ;;TYPE A CR AND LF
014054  001177                          $CRLF
014056  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
014062  000755                          BR      2$              ;;GET NEXT CHARACTER
014064  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
014070  126726  165062          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
014074  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
014076  016746  165052                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
014102  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
014106  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
014110  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
014114  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
014120  000770                          BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

014122  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
014126  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
014132  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
014140  001372                          BNE     9$              ;;TAB STOP
014142  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
```

```
 2922  014144  000724                          BR      2$              ;;GET NEXT CHARACTER
 2923  014146  105777  164776          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 2924  014152  100375                          BPL     $TYPEC
 2925  014154  116677  000002  164770        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 2926  014162  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 2927  014170  001003                          BNE     1$              ;;BRANCH IF NO
 2928  014172  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 2929  014176  000406                          BR      $TYPEX          ;;EXIT
 2930  014200  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 2931  014206  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 2932  014210  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 2933  014212  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 2934  014214  000207          $TYPEX: RTS     PC
 2935
 2936                          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 2937
 2938                          ;;**************************************************************
 2939                          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 2940                          ;*CHANGE IT TO BINARY.
 2941                          ;*CALL:
 2942                          ;*      RDOCT                   ;;READ AN OCTAL NUMBER
 2943                          ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
 2944                          ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 2945
 2946  014216  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 2947  014220  016666  000004  000002        MOV     4(SP),2(SP)     ;;INPUT NUMBER
 2948  014226  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 2949  014230  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 2950  014232  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 2951  014234  104410                  1$:   RDLIN                   ;;READ AN ASCIZ LINE
 2952  014236  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 2953  014240  005001                        CLR     R1              ;;CLEAR DATA WORD
 2954  014242  005002                        CLR     R2
 2955  014244  112046                  2$:   MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 2956  014246  001412                        BEQ     3$              ;;IF ZERO GET OUT
 2957  014250  006301                        ASL     R1              ;;*2
 2958  014252  006102                        ROL     R2
 2959  014254  006301                        ASL     R1              ;;*4
 2960  014256  006102                        ROL     R2
 2961  014260  006301                        ASL     R1              ;;*8
 2962  014262  006102                        ROL     R2
 2963  014264  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 2964  014270  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 2965  014272  000764                        BR      2$              ;;LOOP
 2966  014274  005726                  3$:   TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 2967  014276  010166  000012                MOV     R1,12(SP)       ;;SAVE THE RESULT
 2968  014302  010267  000010                MOV     R2,$HIOCT
 2969  014306  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
 2970  014310  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
 2971  014312  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 2972  014314  000002                        RTI                     ;;RETURN
 2973  014316  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
 2974                          .SBTTL  TTY INPUT ROUTINE
 2975
 2976                          ;;**************************************************************
 2977                          .ENABL  LSB
```



```
 2978
 2979                          ;;**************************************************************
 2980                          ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 2981                          ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 2982                          ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 2983                          ;*WHEN OPERATING IN TTY FLAG MODE.
 2984  014320  022767  000176  164612  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 2985  014326  001074                          BNE     15$             ;;BRANCH IF NO
 2986  014330  105777  164610                  TSTB    @$TKS           ;;CHAR THERE?
 2987  014334  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
 2988  014336  117746  164604                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
 2989  014342  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
 2990  014346  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 2991  014352  001062                          BNE     15$             ;;NO, RETURN TO USER
 2992  014354  126727  164554  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
 2993  014362  001456                          BEQ     15$             ;;BRANCH IF YES
 2994
 2995  014364  104401  015045                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
 2996  014370  104401  015052          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
 2997  014374  016746  163576                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
 2998  014400  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2999  014402  104401  015063                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
 3000  014406  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
 3001  014410  005046                          CLR     -(SP)           ;;THE NEW SWR
 3002  014412  105777  164526          7$:     TSTB    @$TKS           ;;CHAR THERE?
 3003  014416  100375                          BPL     7$              ;;IF NOT TRY AGAIN
 3004
 3005  014420  117746  164522                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
 3006  014424  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 3007
 3008
 3009
 3010  014430  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
 3011  014434  001005                          BNE     10$             ;;BRANCH IF NOT
 3012  014436  104401  015040                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
 3013  014442  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
 3014  014446  000757                          BR      19$             ;;LET'S TRY IT AGAIN
 3015
 3016
 3017  014450  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
 3018  014454  001022                          BNE     16$             ;;BRANCH IF NO
 3019  014456  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
 3020  014462  001403                          BEQ     11$             ;;BRANCH IF YES
 3021  014464  016677  000002  164446          MOV     2(SP),@SWR      ;;SAVE NEW SWR
 3022  014472  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
 3023  014476  104401  001177          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
 3024  014502  126727  164427  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
 3025  014510  001003                          BNE     15$             ;;BRANCH IF NOT
 3026  014512  012777  000100  164424          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
 3027  014520  000002          15$:    RTI                     ;;RETURN
 3028  014522  004767  177420          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
 3029  014526  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
 3030  014532  002420                          BLT     18$             ;;BRANCH IF YES
 3031  014534  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
 3032  014540  003015                          BGT     18$             ;;BRANCH IF YES
 3033  014542  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
```

```
3034  014546  005766  000002          TST     2(SP)           ;;IS THIS THE FIRST CHAR
3035  014552  001403                  BEQ     17$             ;;BRANCH IF YES
3036  014554  006316                  ASL     (SP)            ;;NO, SHIFT PRESENT
3037  014556  006316                  ASL     (SP)            ;;   CHAR OVER TO MAKE
3038  014560  006316                  ASL     (SP)            ;;   ROOM FOR NEW ONE.
3039  014562  005266  000002  17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
3040  014566  056616  177776          BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
3041  014572  000707                  BR      7$              ;;GET THE NEXT ONE
3042  014574  104401  001176  18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3043  014600  000720                  BR      20$             ;;SIMULATE CONTROL-U
3044                          .DSABL  LSB
3045
3046
3047                          ;;*************************************************************
3048                          ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3049                          ;*CALL:
3050                          ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
3051                          ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3052                          ;*                              ;;WITH PARITY BIT STRIPPED OFF
3053                          ;
3054
3055  014602  011646  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
3056  014604  016666  000004  000002  MOV     4(SP),2(SP)     ;;SAVE THE PS
3057  014612  105777  164326  1$:     TSTB    @$TKS           ;;WAIT FOR
3058  014616  100375                  BPL     1$              ;;A CHARACTER
3059  014620  117766  164322  000004  MOVB    @$TKB,4(SP)     ;;READ THE TTY
3060  014626  042766  177600  000004  BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3061  014634  026627  000004  000023  CMP     4(SP),#23       ;;IS IT A CONTROL-S?
3062  014642  001013                  BNE     3$              ;;BRANCH IF NO
3063  014644  105777  164274  2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
3064  014650  100375                  BPL     2$              ;;LOOP UNTIL ITS THERE
3065  014652  117746  164270          MOVB    @$TKB,-(SP)     ;;GET CHARACTER
3066  014656  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3067  014662  022627  000021          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3068  014666  001366                  BNE     2$              ;;IF NOT DISCARD IT
3069  014670  000750                  BR      1$              ;;YES, RESUME
3070  014672  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
3071  014700  002407                  BLT     4$              ;;BRANCH IF YES
3072  014702  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
3073  014710  003003                  BGT     4$              ;;BRANCH IF YES
3074  014712  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
3075  014720  000002          4$:     RTI                     ;;GO BACK TO USER
3076                          ;;*************************************************************
3077                          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3078                          ;*CALL:
3079                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3080                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3081                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3082
3083  014722  010346  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3084  014724  012703  015030  1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
3085  014730  022703  015040  2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
3086  014734  101405                  BLOS    4$              ;;BR IF YES
3087  014736  104407                  RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
3088  014740  112613                  MOVB    (SP)+,(R3)      ;;GET CHARACTER
3089  014742  122713  000177  10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
```



```
3090  014746  001003                  BNE     3$              ;;SKIP IF NOT
3091  014750  104401  001176  4$:     TYPE    ,$QUES          ;;TYPE A '?'
3092  014754  000763                  BR      1$              ;;CLEAR THE BUFFER AND LOOP
3093  014756  111367  000044  3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
3094  014762  104401  015026          TYPE    ,9$
3095  014766  122723  000015          CMPB    #15,(R3)+       ;;CHECK FOR RETURN
3096  014772  001356                  BNE     2$              ;;LOOP IF NOT RETURN
3097  014774  105063  177777          CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3098  015000  104401  001200          TYPE    ,$LF            ;;TYPE A LINE FEED
3099  015004  012603                  MOV     (SP)+,R3        ;;RESTORE R3
3100  015006  011646                  MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3101  015010  016666  000004  000002  MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3102  015016  012766  015030  000004  MOV     #$TTYIN,4(SP)
3103  015024  000002                  RTI                     ;;RETURN
3104  015026     000          9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
3105  015027     000                  .BYTE   0               ;;TERMINATOR
3106  015030  000010          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
3107  015040  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
3108  015045     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
3109  015052  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
3110  015060  020075     000
3111  015063     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
3112  015070  036440  000040
3113                          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3114
3115                          ;;*************************************************************
3116                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3117                          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3118                          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3119                          ;*CALL:
3120                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3121                          ;*      TYPOS                   ;;CALL FOR TYPEOUT
3122                          ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3123                          ;*      .BYTE   M               ;;M=1 OR 0
3124                          ;*                                      ;;1=TYPE LEADING ZEROS
3125                          ;*                                      ;;0=SUPPRESS LEADING ZEROS
3126                          ;*
3127                          ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3128                          ;*$TYPOS OR $TYPOC
3129                          ;*CALL:
3130                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3131                          ;*      TYPON                   ;;CALL FOR TYPEOUT
3132                          ;*
3133                          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3134                          ;*CALL:
3135                          ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3136                          ;*      TYPOC                   ;;CALL FOR TYPEOUT
3137
3138  015074  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
3139  015100  116667  000001  000211  MOVB    1(SP),$0FILL    ;;LOAD ZERO FILL SWITCH
3140  015106  112667  000207          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3141  015112  062716  000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
3142  015116  000406                  BR      $TYPON
3143  015120  112767  000001  000171  $TYPOC: MOVB    #1,$0FILL       ;;SET THE ZERO FILL SWITCH
3144  015126  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
3145  015134  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
```

```
3146  015142  010346                          MOV     R3,-(SP)        ;;SAVE R3
3147  015144  010446                          MOV     R4,-(SP)        ;;SAVE R4
3148  015146  010546                          MOV     R5,-(SP)        ;;SAVE R5
3149  015150  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
3150  015154  005404                          NEG     R4
3151  015156  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3152  015162  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3153  015166  116704  000125                  MOVB    $0FILL,R4       ;;GET THE ZERO FILL SWITCH
3154  015172  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3155  015176  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
3156  015200  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
3157  015202  000404                          BR      3$              ;;GO DO MSB
3158  015204  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
3159  015206  006105                          ROL     R5
3160  015210  006105                          ROL     R5
3161  015212  010503                          MOV     R5,R3
3162  015214  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3163  015216  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
3164  015222  100016                          BPL     7$              ;;BR IF NO
3165  015224  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
3166  015230  001002                          BNE     4$              ;;TEST FOR 0
3167  015232  005704                          TST     R4              ;;SUPPRESS THIS 0?
3168  015234  001403                          BEQ     5$              ;;BR IF YES
3169  015236  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3170  015240  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3171  015244  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3172  015250  110367  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
3173  015254  104401  015314                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
3174  015260  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
3175  015264  003347                          BGT     2$              ;;BR IF MORE TO DO
3176  015266  002402                          BLT     6$              ;;BR IF DONE
3177  015270  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3178  015272  000744                          BR      2$              ;;GO DO THE LAST DIGIT
3179  015274  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
3180  015276  012604                          MOV     (SP)+,R4        ;;RESTORE R4
3181  015300  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3182  015302  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3183  015310  012616                          MOV     (SP)+,(SP)
3184  015312  000002                          RTI                     ;;RETURN
3185  015314     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3186  015315     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3187  015316     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3188  015317     000          $0FILL: .BYTE   0               ;;ZERO FILL SWITCH
3189  015320  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```



```
3190                                    .SBTTL  TRAP DECODER
3191
3192                                    ;;*****************************************************************
3193                                    ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3194                                    ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3195                                    ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3196                                    ;*GO TO THAT ROUTINE.
3197
3198  015322  010046                    $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
3199  015324  016600  000002                    MOV     2(SP),R0        ;;GET TRAP ADDRESS
3200  015330  005740                            TST     -(R0)           ;;BACKUP BY 2
3201  015332  111000                            MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
3202  015334  006300                            ASL     R0              ;;POSITION FOR INDEXING
3203  015336  016000  015356                    MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
3204  015342  000200                            RTS     R0              ;;GO TO ROUTINE
3205
3206
3207                                    ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3208
3209  015344  011646                    $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
3210  015346  016666  000004  000002            MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3211  015354  000002                            RTI                     ;;RESTORE THE PSW
3212
3213                                    .SBTTL  TRAP TABLE
3214
3215                                    ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3216                                    ;*BY THE "TRAP" INSTRUCTION.
3217
3218                                    ;       ROUTINE
3219                                    ;       -------
3220  015356  015344                    $TRPAD: .WORD   $TRAP2
3221  015360  013776                            $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
3222  015362  015120                            $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3223  015364  015074                            $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3224  015366  015134                            $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3225
3226  015370  014370                            $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
3227
3228  015372  014320                            $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
3229  015374  014602                            $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
3230  015376  014722                            $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3231  015400  014216                            $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
3232  015402  012014                            WAIT.ON.READY   ;;CALL=WAITREADY        TRAP+12(104412) WAIT ON THE READY BIT TO
3233  015404  012120                            WAIT.FOR.XFER   ;;CALL=WAITXFER TRAP+13(104413) WAIT ON XFER REQ.
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES

                                        ;;**************************************************************
                                        ;POWER DOWN ROUTINE
015406  012737  015552  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
015414  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
015422  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015424  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015426  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015430  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
015432  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
015434  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
015436  017746  163476                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
015442  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
015446  012737  015460  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
015454  000000                          HALT
015456  000776                          BR      .-2             ;;HANG UP

                                        ;;**************************************************************
                                        ;POWER UP ROUTINE
015460  012737  015552  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
015466  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
015472  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
015476  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
015502  001375                          BNE     1$              ;;OF  WORD
015504  012677  163430                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
015510  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
015512  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
015514  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
015516  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
015520  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015522  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015524  012737  015406  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
015532  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
015540  104401                          TYPE                    ;REPORT THE POWER FAILURE
015542  015560                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
015544  012716                          MOV     (PC)+,(SP)      ;;RESTART AT PWRST
015546  002222                  $PWRAD: .WORD   PWRST           ;;RESTART ADDRESS
015550  000002                          RTI
015552  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
015554  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
015556  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
015560  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
015566  000122
                                        .EVEN
```



```
015570  042200  044522  042526  MSGDRV: .ASCIZ  <CRLF>/DRIVE(S)?/
015576  051450  037451     000
015603     015  052012  041501  MSGASK: .ASCIZ  <15><12>/TACS?/
015610  037523     000
015613     126  041505  047524  MSGVEC: .ASCIZ  /VECTOR?/
015620  037522     000
015623     120  044522  051117  MSGPRI: .ASCIZ  /PRIORITY?/
015630  052111  037531     000
015635     124  041501  036523  MTACS:  .ASCIZ  /TACS=/
015642     000
015643     040  040524  041104  MTADB:  .ASCIZ  / TADB=/
015650  000075
015652  053040  041505  047524  MTAVEC: .ASCIZ  / VECTOR=/
015660  036522     000
015663     040  051120  047511  MTAPRI: .ASCIZ  / PRIORITY=/
015670  044522  054524  000075
015676  005015  045517  000077  MSGOK:  .ASCIZ  <15><12>/OK?/
015704  042531  006523  000012  MYES:   .ASCIZ  /YES/<15><12>
015712  052123  052101  051525  EM1:    .ASCIZ  /STATUS PROBLEM/
015720  050040  047522  046102
015726  046505     000
015731     042  042522  042101  EM2:    .ASCIZ  /"READY" FAILED TO SET/
015736  021131  043040  044501
015744  042514  020104  047524
015752  051440  052105     000
015757     042  051124  047101  EM3:    .ASCIZ  /"TRANSFER REQUEST" FAILED TO SET/
015764  043123  051105  051040
015772  050505  042525  052123
016000  020042  040506  046111
016006  042105  052040  020117
016014  042523  000124
016020  044124  020105  051127  EM4:    .ASCIZ  /THE WRONG FLAG SET/
016026  047117  020107  046106
016034  043501  051440  052105
016042     000
016043     103  052517  052116  EM5:    .ASCIZ  /COUNT PATTERN FAILED/
016050  050040  052101  042524
016056  047122  043040  044501
016064  042514  000104
016070  040504  040524  050040  EM6:    .ASCIZ  /DATA PROBLEM/
016076  047522  046102  046505
016104     000
016105     101  042104  042522  EM10:   .ASCIZ  /ADDRESS FAILED/
016112  051523  043040  044501
016120  042514  000104
016124  041520  020040  020040  DH1:    .ASCIZ  /PC      TACS/
016132  020040  040524  051503
016140     000
016141     120  020103  020040  DH2:    .ASCIZ  /PC      TACS    WAIT ADDRESS/
016146  020040  052040  041501
016154  020123  020040  053440
016162  044501  020124  042101
016170  051104  051505  000123
016176  041520  020040  020040  DH5:    .ASCIZ  /PC      TACS    EXPECT  RCV'D/
016204  020040  040524  051503
016212  020040  020040  054105
```

```
  3335  016220  042520  052103  020040
  3336  016226  041522  023526  000104
  3337  016234  041520  020040  020040  DH6:    .ASCIZ  /PC      TADB/
  3338  016242  020040  040524  041104
  3339  016250     000
  3340  016251     120  020103  020040  DH10:   .ASCIZ  /PC      ADDRESS/
  3341  016256  020040  040440  042104
  3342  016264  042522  051523     000
  3343          016272                  .EVEN
  3344  016272  001116  001162  000000  DT1:    .WORD   $ERRPC,$REG0,0
  3345  016300  001116  001162  001202  DT2:    .WORD   $ERRPC,$REG0,SAVPC,0
  3346  016306  000000
  3347  016310  001116  001162  001124  DT5:    .WORD   $ERRPC,$REG0,$GDDAT,$BDDAT,0
  3348  016316  001126  000000
  3349  016322  001116  001164  000000  DT6:    .WORD   $ERRPC,$REG1,0
  3350  016330  001116  001122  000000  DT10:   .WORD   $ERRPC,$BDADR,0
  3351  016336  001116  001206  000000  DT201:  .WORD   $ERRPC,TACSL,0
  3352
  3353  016344  001116  000000          DT202:  .WORD   $ERRPC,0
  3354
  3355  016350  040524  030461  043040  EM201:  .ASCIZ  "TA11 FAILED TO RESPOND"
  3356  016356  044501  042514  020104
  3357  016364  047524  051040  051505
  3358  016372  047520  042116     000
  3359  016377     116  020117  051104  EM202:  .ASCIZ  "NO DRIVE AVAILABLE"
  3360  016404  053111  020105  053101
  3361  016412  044501  040514  046102
  3362  016420  000105
  3363  016422  041520  020040  020040  DH201:  .ASCIZ  /PC      TACS/
  3364  016430  020040  040524  051503
  3365  016436     000
  3366  016437     120  000103          DH202:  .ASCIZ  /PC/
  3367          000001                  .END
```



```
ASKADR  012332            572    2494#
ASKDRV  012222            569    2461#   2484
ASKKEY  001232            394#    659*    695
BEGINX  001760            566#    602     604     652     654
BEGIN1  001356            231     485#
BEGIN2  001410            232     492#
BEGIN3  001416            233     491     494#
BEGIN4  001424            234     496#
BGNCMN  001430            488     490     493     495     497#
BIT0  = 000001            166#
BIT00 = 000001            156#    166     208     956    1415    2857    2859
BIT01 = 000002            155#    165     207     947    1105    1106    1110    1111    1119    1120    1125    2859
BIT02 = 000004            154#    164     206     938    1078    1079    1083    1084    1092    1093    1098    1402    1404
BIT03 = 000010            153#    163     205     929    1051    1052    1056    1057    1065    1066    1071
BIT04 = 000020            152#    162     204     920    1024    1025    1029    1030    1038    1039    1044
BIT05 = 000040            151#    161     203    1161    1382    1383    1386
BIT06 = 000100            150#    160     202     911     997     998    1002    1003    1011    1012    1017
BIT07 = 000200            149#    159     201     902    1362    1363    1366    2857    2859
BIT08 = 000400            148#    158     200     893     970     971     975     976     984     985     990    1413    1440
                         1441    2279
BIT09 = 001000            147#    157     199    1171    1342    1343    1346    2287    2351
BIT1  = 000002            165#
BIT10 = 002000            146#    198     884    1322    1323    1326    2335
BIT11 = 004000            145#    197     875    1302    1303    1306    2294
BIT12 = 010000            144#    196    1182    1282    1283    1286
BIT13 = 020000            143#    195    1192    1262    1263    1266    2342
BIT14 = 040000            142#    194     866    1242    1243    1246    2265
BIT15 = 100000            141#    193     670    1202    1222    1223    1226
BIT2  = 000004            164#
BIT3  = 000010            163#
BIT4  = 000020            162#
BIT5  = 000040            161#
BIT6  = 000100            160#
BIT7  = 000200            159#
BIT8  = 000400            158#
BIT9  = 001000            157#
BLKLIM  013500           2761*   2775#   2782    2799
BPTVEC= 000014            173#
BSBG  = 000010            186#    1486    1795    1938    1945
BSFG  = 000006            185#    1482    1780    1909    1916    2738
BSFGSB  013336            241     2735#   2736    2743
CHKADR  002002            568     571     590#
CHKDRV  002062            628#
CKSWR = 104406           2264     2327    2350    3228#
CR    = 000015             81#    2926    2936
CRCERR= 040000            194#
CRLF  = 000200             82#     554    2897    2936    3279
CURDRV  001234            395#    688*    689
DDISP = 177570             88#     364     525
DH1     016124            416    3324#
DH10    016251            458    3340#
DH2     016141            422     428     434    3327#
DH201   016422            465    3363#
DH202   016437            470    3366#
DH5     016176            440     452    3332#
DH6     016234            446    3337#
```

```
DISPLA  001142          364#    525*    533*    2308*   2334*
DISPRE  000174          228#    533
DRVKEY  001224          392#    393     486*    628     641     662     693     2462*   2467    2481    2485    2831
DRVPNT  001230          393#    642*    662*    679     694*    2825    2832#
DSWR  = 177570           87#    363     524
DT1     016272          417     3344#
DT10    016330          459     3350#
DT2     016300          423     429     435     3345#
DT201   016336          466     3351#
DT202   016344          471     3353#
DT5     016310          441     453     3347#
DT6     016322          447     3349#
EMTVEC= 000030          176#    508*    509*    731     732*    736*    754     755*    759*    777     778*    782*    800
                        801*    805*    823     824*    828*    846     847*    851*
EM1     015712          415     3297#
EM10    016105          457     3321#
EM2     015731          421     3300#
EM201   016350          464     3355#
EM202   016377          469     3359#
EM3     015757          427     3304#
EM4     016020          433     3310#
EM5     016043          439     3314#
EM6     016070          445     451     3318#
ERROR = 100000          193#    1657
ERROR1  011516          732     755     778     801     824     847     2330#
ERRVEC= 000004          169#    522     523*    534*    590#    596*    726*    729*    738*    749*    752*    761*    772*
                        775*    784*    795*    798*    807*    818*    821*    830*    841*    844*    853*    2270    2271*
                       2273*   2276*
EXAM    013672          629     633     638     2844#
EXIT    012322         2480    2485#
FGAP  = 004000          197#
FUNCTI= 000016          209#    1424    1427    1428    1437    1450    1454
FUNC0 = 000002          207#    209
FUNC1 = 000004          206#    209
FUNC2 = 000010          205#    209
GNS   = ****** U        227     553     675     686     3221    3222    3223    3224    3226    3228    3229    3230    3231
                       3232    3233
GO    = 000001          208#    303     306     325     328     707     1518    1543    1566    1581    1596    1625    1647
                       1655    1674    1676    1694    1722    1726    1733    1750    1765    1780    1795    1810    1825
                       1841    1845    1852    1876    1880    1887    1905    1909    1916    1934    1938    1945    1963
                       1967    1974    1992    1996    2003    2040    2042    2083    2085    2087    2095    2097    2135
                       2137    2140    2161    2163    2166    2187    2189    2192    2211    2575    2578    2608    2631
                       2663    2666    2691    2697    2719    2722    2738    2783    2801
GTSWR = 104405          548     701     2753    2766    2772    3226#
HERE    001760          539     541     552     554#
HGHTIM  012072         1678    2404*   2418#
HT    = 000011           79#    2895    2936
ILBS  = 000020          204#    1424    1427    1428    1437    1450    1454    2065    2093    2104    2150    2176    2202
                       2791    2812
INT.EN= 000100          202#    1424    1427    1428    1437    1450    1454
IOTVEC= 000020          174#    506*    507*
ITEMS2  001336          462#
LEADER= 020000          195#    1467    1528    1553    1652    1657    1696    1707    1738    1863    1892    1921    1950
                       1979    2008    2580    2612    2635    2668    2671    2695    2699    2724    2740    2862
LF    = 000012           80#    2930    2936
LOOP    012246         2468#    2482
```



```
LOOP1   000500          291#    292
LOOP2   000600          301#    312
LOOP3   000700          323#    334
MANUAL  002202          648     656#
MAXCNT  012116          670*    1678*   1679*   1680*   1681*   2404    2425#   2432
MSGASK  015603         2495     3281#
MSGDRV  015570         2461     3279#
MSGID   001732          542     554#    661
MSGOK   015676         2542     3295#
MSGPRI  015623         2518     3285#
MSGVEC  015613         2509     3283#
MTACS   015635         2530     3287#
MTADB   015643         2533     3289#
MTAPRI  015663         2539     3293#
MTAVEC  015652         2536     3291#
MYES    015704         2551     3296#
NEXT    012302         2471     2475    2479#
NOTA    012260         2469     2472#
NOTB    012272         2473     2476#
NOTLGL  012314         2466     2477    2483#   2486
NXTDRV  013626         2573     2606    2629    2661    2689    2717    2822#
OFFLIN= 001000          199#    2853
OK      002214          657     659#
PATTRN  013502         2762*    2768*   2776#   2789    2800
PIRQ  = 177772           86#
PIRQVE= 000240          180#
PR0   = 000000          103#
PR1   = 000040          104#
PR2   = 000100          105#
PR3   = 000140          106#
PR4   = 000200          107#
PR5   = 000240          108#
PR6   = 000300          109#
PR7   = 000340          110#
PS    = 177776           83#     84      665*
PSW   = 177776           84#
PWRST   002222          661#    3271
PWRVEC= 000024          175#    512*    513*    3238*   3239*   3248*   3254*   3266*   3267*
RDBLK   013546         2634     2694    2799#
RDCHR = 104407         2543     3087    3229#
RDLIN = 104410         2463     2951    3230#
RDOC1 = 104411         2496     2510    2519    3231#
RDSUB   012774          237     2624#   2628    2636
READ  = 000004          184#    328     1478    1765    1880    1887    2097    2801
READY = 000040          203#    304     311     326     333     708     1519    1525    1544    1550    1567    1582    1598
                       1606    1627    1635    1701    1727    1751    1766    1781    1796    1811    1826    1846    1881
                       1910    1939    1968    1997    2408    2438    2576    2609    2632    2664    2692    2720    2785
                       2803    2849
RESVEC= 000010          170#
REWIND= 000016          189#    303     325     707     1498    1518    1566    1596    1647    1674    1694    1722    1841
                       1876    1905    1934    1963    1992    2040    2083    2095    2135    2161    2187    2211    2575
                       2608    2631    2663    2691    2719
RGPBLK  013156          239     2684#   2688    2696    2700
SAVPC   001202          384#    2405*   2406*   2422    2433*   2434*   2450    3345
SAVPS   001204          385#    2407*   2421    2435*   2449
SETBUF  013370         2600     2655    2749#
```

```
SFBG  = 000014          188#  1494    1825    1996    2003    2697
SFFG  = 000012          187#  1490    1810    1967    1974    2722
SFFGSB  013252          240   2712#   2716    2727
STACK = 001100           74#   504     725     748     771     794     817     840    2568    2601    2624    2656    2684
                        2712
START   002234          660    663#   2245
STKLMT= 177774           85#
SWR     001140          291    309     363#    502     524#    526     532#    546     699    2265    2279    2281    2287
                        2294   2335    2342    2347    2351    2751    2755    2757    2764    2768    2770    2822    2857
                        2859   2984    3021#   3246    3259#
SWREG   000176          229#   532     546     699    2751    2764    2770    2984    2997
SW0   = 000001          138#
SW00  = 000001          128#   138
SW01  = 000002          127#   137
SW02  = 000004          126#   136
SW03  = 000010          125#   135
SW04  = 000020          124#   134
SW05  = 000040          123#   133
SW06  = 000100          122#   132
SW07  = 000200          121#   131
SW08  = 000400          120#   130
SW09  = 001000          119#   129
SW1   = 000002          137#
SW10  = 002000          118#
SW11  = 004000          117#
SW12  = 010000          116#
SW13  = 020000          115#
SW14  = 040000          114#
SW15  = 100000          113#
SW2   = 000004          136#
SW3   = 000010          135#
SW4   = 000020          134#
SW5   = 000040          133#
SW6   = 000100          132#
SW7   = 000200          131#
SW8   = 000400          130#
SW9   = 001000          129#
TACSH   001210          387#   819#    825    1403#   1415    1427#   2504#   2505#
TACSL   001206          386#   592     666    2501#   2504    2531    2569    2602    2625    2657    2685    2713    2844
                        3351
TADBH   001214          389#   842#    848    2507#   2508#
TADBL   001212          388#   667    2503#   2507    2534    2570    2603    2626    2658    2686    2714
TAPRIO  001222          391#  2529#   2540
TAVEC   001216          390#   489     663#    664#   2515#   2517#   2537
TBITVE= 000014          171#
TDRV    002366          681    683#
TIMERR= 002000          198#
TKVEC = 000060          178#
TPVEC = 000064          179#
TRAPVE= 000034          177#   510#    511#
TRTVEC= 000014          172#
TP.REQ= 000200          201#
TST1    002514          705    723#
TST10   003316          867    872#
TST100  007070         1782   1789#
TST101  007132         1797   1804#
```



```
TST102  007174          1812   1819#
TST103  007236          1827   1834#
TST104  007370          1837   1864    1869#
TST105  007504          1872   1893    1898#
TST106  007620          1901   1922    1927#
TST107  007734          1930   1951    1956#
TST11   003340           876    881#
TST110  010050          1959   1980    1985#
TST111  010164          1988   2009    2014#
TST112  010224          2033#
TST113  010374          2036   2071    2076#
TST114  010534          2079   2112#
TST115  010554          2116   2121#
TST116  010570          2124   2129#
TST117  010674          2155#
TST12   003362           885    890#
TST120  011000          2181#
TST121  011102          2207#
TST13   003404           894    899#
TST14   003426           903    908#
TST15   003450           912    917#
TST16   003472           921    926#
TST17   003514           930    935#
TST2    002610           746#
TST20   003536           939    944#
TST21   003560           948    953#
TST22   003602           957    969#
TST23   003634           977    982#
TST24   003672           991    996#
TST25   003724          1004   1009#
TST26   003762          1018   1023#
TST27   004014          1031   1036#
TST3    002704           769#
TST30   004052          1045   1050#
TST31   004104          1058   1063#
TST32   004142          1072   1077#
TST33   004174          1085   1090#
TST34   004232          1099   1104#
TST35   004264          1112   1117#
TST36   004322          1126   1131#
TST37   004370          1132   1157#
TST4    003000           792#
TST40   004414          1162   1167#
TST41   004440          1172   1177#
TST42   004470          1183   1188#
TST43   004514          1193   1198#
TST44   004540          1203   1216#
TST45   004606          1228   1236#
TST46   004654          1248   1256#
TST47   004722          1268   1276#
TST5    003074           815#
TST50   004770          1288   1296#
TST51   005036          1308   1316#
TST52   005104          1328   1336#
TST53   005152          1348   1356#
TST54   005220          1368   1376#
```

```
TST55   005266          1388    1399#
TST56   005320          1405    1410#
TST57   005352          1416    1421#
TST6    003174           838#
TST60   005412          1422    1429    1434#
TST61   005450          1435    1442    1447#
TST62   005506          1448    1455    1465#
TST63   005642          1500    1510#
TST64   005742          1514    1529    1535#
TST65   006042          1539    1554    1560#
TST66   006104          1568    1575#
TST67   006146          1583    1590#
TST7    003274           863#
TST70   006236          1618#
TST71   006332          1621    1636    1641#
TST72   006424          1644    1668#
TST73   006504          1671    1689#
TST74   006606          1715#
TST75   006722          1718    1739    1744#
TST76   006764          1752    1759#
TST77   007026          1767    1774#
TYPE  = 104401           542     661     673     684     689     690    2235    2337    2345    2367    2376    2378    2381
                        2383    2387    2395    2461    2483    2495    2509    2518    2530    2533    2536    2539    2542
                        2549    2551    2900    2995    2996    2999    3012    3023    3042    3091    3094    3098    3173
                        3221#   3268
TYPERR  011676          2344    2367#
TYPOC = 104402          2392    2532    2535    2538    2541    2998    3222#
TYPON = 104404          3224#
TYPOS = 104403          3223#
UNIT  = 000400           200#    682    1424    1437    1450    1454    2828    2848
WAITRE= 104412          1565    1580    1595    1609    1624    1648    1656    1675    1677    1695    1711    1721    1723
                        1734    1749    1764    1779    1794    1809    1824    1840    1842    1859    1875    1877    1888
                        1904    1906    1917    1933    1935    1946    1962    1964    1975    1991    1993    2004    2023
                        2039    2041    2043    2069    2082    2084    2086    2094    2096    2108    2136    2138    2151
                        2162    2164    2177    2188    2190    2203    2212    2579    2667    2698    2723    2739    2792
                        2813    3232#
WAITXF= 104413          2064    2088    2090    2092    2098    2103    2141    2143    2167    2169    2193    2195    2784
                        2802    3233#
WAIT.F  012120          2430#   3233
WAIT.O  012014          2402#   3232
WAIT1   012044          2408#   2416    2419
WAIT2   012064          2403*   2415#
WAIT3   012100          2409    2413    2421#
WAIT4   012062          2411    2414#
WFG   = 000000           182#   1470    1543    1581    1625    1655    1676    1726    1733    2042    2085    2137    2163
                        2189    2578    2666
WFGSUB  012622           235    2568#   2572    2583
WGBLOP  013060          2656#   2660    2674
WGPBLK  013054           238    2655#
WLOOP   012712          2601#   2605    2615
WRITE = 000002           183#    306    1181    1474    1750    1845    1852    2046    2087    2140    2166    2192    2783
WRTBLK  013504          2611    2670    2782#
WRTLOC= 010000           196#   2855
WRTSUB  012706           236    2600#
$AUTOB  001134           360#    550*   2992    3113
$BDADR  001122           355#    733*    756*    779*    802*    825*    848*   3350
```



```
$BDDAT  001126           357#   2331*   3347
$BELL   001172           379#   2337    2363
$CHARC  014212          2902*   2912*   2919    2928*   2933#
$CKSWR  014320          2984#   3228
$CMTAG  001100           343#    499     500     508     514     515     516
$CM1  = 000002           375#    376#    377#
$CM2  = 000004           375#    376#    377#
$CM3  = 000002           373#    375
$CNTLG  015045          2995    3108#
$CNTLU  015040          3012    3107#
$CRLF   001177           381#    690    2345    2363    2367    2378    2383    2387    2901    2936    3023    3107
$DOAGN  011210          2231    2237    2243#
$ENDAD  011200           540     605     655    2239#   2358
$ENDCT  011160           514    2233#
$ENDMG  011217          2235    2247#
$ENULL  011214          2246#
$EOP    011124           737     760     783     806     829     852    2223#
$EOPCT  011152           514*   2230#   2234
$ERFLG  001103           346#   2254    2283    2285    2291*   2312    2332*   2363
$ERMAX  001115           352#    517*   2285    2307*   2312
$ERROR  011504           508    2326#
$ERRPC  001116           353#   2339*   2340*   2341    2363    3344    3345    3347    3349    3350    3351    3353
$ERRTB  001236           410#   2374
$ERTTL  001112           350#   2338*   2363
$ESCAP  001170           378#    516*   1132*   1422*   1435*   1448*   1514*   1539*   1592*   1621*   1644*   1691*   1710*
                        1718*   1837*   1872*   1901*   1930*   1959*   1988*   2036*   2079*   2132*   2158*   2184*   2306*
                        2354    2356    2363
$FILLC  001156           371#   2905    2936
$FILLS  001155           370#   2936
$GDADR  001120           354#
$GDDAT  001124           356#   2330*   3347
$GET42  011170          2236#
$GTSWR  014370          2996#   3226
$HD   = 000000            11
$HIOCT  014316          2968*   2973#
$ICNT   001104           347#   2298*   2299    2301*   2311
$ILLUP  015552          3238    3254    3273#
$INTAG  001135           361#   3024    3113
$ITEMB  001114           351#   2341*   2363    2369
$LF     001200           382#   2363    2936    3098    3107
$LPADR  001106           348#    518*    864*    873*    882*    891*    900*    909*    918*    927*    936*    945*    954*
                        1158*   1168*   1178*   1189*   1199*   1217*   1237*   1257*   1277*   1297*   1317*   1337*   1357*
                        1377*   1400*   1411*   1512*   1537*   1620*   1643*   1717*   1836*   1871*   1900*   1929*   1958*
                        1987*   2016*   2035*   2078*   2113*   2131*   2157*   2183*   2289*   2304*   2309    2311
$LPERR  001110           349#    519*   2289    2305*   2311    2353    2572*   2605*   2628*   2660*   2688*   2716*   2736*
$MAIL = ****** U         536     546    2304    2347    2889
$MNEW   015063          2999    3111#
$MSWR   015052          2996    3109#
$MXCNT  011502          2302    2311#
$NULL   001154           369#   2907    2936
$NWTST= 000001           720#    743#    766#    789#    812#    835#    860#    869#    878#    887#    896#    905#    914#
                         923#    932#    941#    950#    966#    979#    993#   1006#   1020#   1033#   1047#   1060#   1074#
                        1087#   1101#   1114#   1128#   1154#   1164#   1174#   1185#   1195#   1213#   1233#   1253#   1273#
                        1293#   1313#   1333#   1353#   1373#   1396#   1407#   1418#   1431#   1444#   1462#   1507#   1532#
                        1557#   1572#   1587#   1615#   1638#   1665#   1686#   1712#   1741#   1756#   1771#   1786#   1801#
                        1816#   1831#   1866#   1895#   1924#   1953#   1982#   2011#   2030#   2073#   2109#   2118#   2126#
```

DZTAAC.NEW CROSS REFERENCE TABLE -- USER SYMBOLS

```
                    2152#   2178#   2204#
$OCNT   015316      3145*   3174*   3187#
$OMODE  015320      3140*   3144*   3149    3152*   3163*   3189#
$OVER   011466      2266    2282    2290    2300    2308#
$PASS   001100       344#    668    1670    2227*   2228*   2246    2296    2312
$POWER  015560      3269    3276#
$PWRAD  015546      3271#
$PWRDN  015406       512    3238#   3266
$PWRMG  015542      3269#
$PWRUP  015460      3248    3254#
$QUES   001176       380#   2363    2483    2549    2936    3042    3091    3107
$RDCHR  014602      3055#   3229
$RDDEC= ****** U    3232
$RDLIN  014722      3083#   3230
$RDOCT  014216      2946#   3231
$RDSZ = 000010      3076#
$REGAD  001160       373#
$REG0   001162       375#   2328*   3344    3345    3347
$REG1   001164       376#   2329*   3349
$RTNAD  011212      2245#
$R2A  = ****** U    3232
$SAVRE= ****** U    3232
$SAVR6  015556      3247*   3255    3256*   3257*   3275#
$SCOPE  011232       506    2263#
$SETUP= 000137       485#    505     506     508     510     512     514     515     516     518     540     543    2225
                    2264    2327    2350    2358    2979    3113
$STUP = 177777       485#
$SVLAD  011440      2274    2303#
$SWR  = 167400         1#     11      62      63      64      65      66      67      68     377     378     379     515
                     516     518     519     724     747     770     793     816     839     864     873     882     891
                     900     909     918     927     936     945     954     970     983     997    1010    1024    1037
                    1051    1064    1078    1091    1105    1118    1132    1158    1168    1178    1189    1199    1217
                    1237    1257    1277    1297    1317    1337    1357    1377    1400    1411    1422    1435    1448
                    1466    1511    1536    1561    1576    1591    1619    1642    1669    1690    1716    1745    1760
                    1775    1790    1805    1820    1835    1870    1899    1928    1957    1986    2015    2034    2077
                    2113    2122    2130    2156    2182    2208    2218    2226    2238    2244    2246    2255    2256
                    2257    2258    2259    2265    2277    2279    2280    2283    2284    2285    2292    2293    2294
                    2305    2308    2311    2318    2319    2320    2321    2322    2335    2342    2347    2351    2363
                    3272
$SWRMK= 000000       68      69    2259    2260    2281
$TIMES  001156       377#    515*    724*    747*    770*    793*    816*    839*    983*   1010*   1037*   1064*   1091*
                    1118*   1511*   1536*   1561*   1576*   1591*   1619*   1642*   1669*   1690*   1716*   1745*   1760*
                    1775*   1790*   1805*   1820*   1835*   1870*   1899*   1928*   1957*   1986*   2015*   2034*   2077*
                    2130*   2156*   2182*   2208*   2226*   2292*   2299    2302*   2311
$TKB    001146       366#   2977    2988    3005    3059    3065
$TKS    001144       365#   2977    2986    3002    3026*   3057    3063
$TN   = 000122         1#     11     720     724#    743     747#    766     770#    789     793#    812     816#    835
                     839#    860     864#    867     869     873#    876     878     882#    885     887     891#    894
                     896     900#    903     905     909#    912     914     918#    921     923     927#    930     932
                     936#    939     941     945#    948     950     954#    957     966     970#    977     979     983#
                     991     993     997#   1004    1006    1010#   1018    1020    1024#   1031    1033    1037#   1045
                    1047    1051#   1058    1060    1064#   1072    1074    1078#   1085    1087    1091#   1099    1101
                    1105#   1112    1114    1118#   1126    1128    1132#   1154    1158#   1162    1164    1168#   1172
                    1174    1178#   1183    1185    1189#   1193    1195    1199#   1203    1213    1217#   1228    1233
                    1237#   1248    1253    1257#   1268    1273    1277#   1288    1293    1297#   1308    1313    1317#
                    1328    1333    1337#   1348    1353    1357#   1368    1373    1377#   1388    1396    1400#   1405
```


DZTAAC.NEW CROSS REFERENCE TABLE -- USER SYMBOLS

```
                    1407    1411#   1416    1418    1422#   1429    1431    1435#   1442    1444    1448#   1455    1462
                    1466#   1500    1507    1511#   1514    1529    1532    1536#   1539    1554    1557    1561#   1568
                    1572    1576#   1583    1587    1591#   1615    1619#   1621    1636    1638    1642#   1644    1665
                    1669#   1671    1686    1690#   1712    1716#   1718    1739    1741    1745#   1752    1756    1760#
                    1767    1771    1775#   1782    1786    1790#   1797    1801    1805#   1812    1816    1820#   1827
                    1831    1835#   1837    1864    1866    1870#   1872    1893    1895    1899#   1901    1922    1924
                    1928#   1930    1951    1953    1957#   1959    1980    1982    1986#   1988    2009    2011    2015#
                    2030    2034#   2036    2071    2073    2077#   2079    2109    2113#   2116    2118    2122#   2124
                    2126    2130#   2152    2156#   2178    2182#   2204    2208#
$TPB    001152       368#   2925*   2936
$TPFLG  001157       372#   2883    2936
$TPS    001150       367#   2923    2936
$TRAP   015322       510    3198#
$TRAP2  015344      3209#   3220
$TRP  = 000014      3213#   3222#   3223#   3224#   3225#   3226    3227#   3228    3229#   3230#   3231#   3232#   3233#
                    3234#
$TRPAD  015356      3203    3220#
$TSTNM  001102       345#    677#   2225*   2254    2281    2303*   2308    2312    2334    2363
$TTYIN  015030      3084    3085    3102    3106#
$TYPBN= ****** U    3225
$TYPDS= ****** U    3225
$TYPE   013776      2883#   3213    3221
$TYPEC  014146      2904    2911    2918    2923#   2924    3028
$TYPEX  014214      2929    2931    2934#
$TYPOC  015120      3143#   3222
$TYPON  015134      3142    3145#   3224
$TYPOS  015074      3138#   3223
$XTSTR  011244      2268#
$$GET4= 000000      2238#
$0FILL  015317      3139*   3143*   3153    3188#
$40CAT= ****** U    2265    2344
.     = 016442       223#    227#    289#    299#    321#    342#    383     503     518     519     554#    676#    687#
                    1659    2246    2249    2311    2312    2363    2398#   2936    2977    3106#   3107    3113    3250
                    3274    3343#
```

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

*TAAC,TAAC/SOL/CRF:SYM=DZTAAC.NEW
RUN-TIME: 58 48 3 SECONDS
RUN-TIME RATIO: 283/111=2.5
CORE USED: 26K (51 PAGES)