IDENTIFICATION SEQ 0001

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

PRODUCT NAME: TA11 LOGIC TEST (PART 2)

DATE REVISED: 21 JUNE 76

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: JIM LACEY

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

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

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

COPYRIGHT (C) 1973,1976 BY DIGITAL EQUIPMEN CORPORATION

MAIN DEC CHANGE NOTICE
MAY BE REQUIRED FOR
PROGRAM TO OPERATE

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

MAINDEC-11-DZTAA

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=", TO ALLOW SETTING OF REGISTER BEFORE BEGINNING TEST.

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

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 PROMPTED
FOR THE VALUE WITH "SWR=XXXXXX 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 PROMPTED
FOR THE VALUE WITH "SWR=XXXXXX 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.

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS TAKES APPROXIMATELY 100 SECONDS ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 475 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

R3, R4 AND R5 ARE RESERVED FOR "DRIVE", "TACS" AND "TADB THROUGH OUT THE PROGRAM.

9. PROGRAM DESCRIPTION

THIS PROGRAM IS A SEQUENCE OF SMALL INDEPENDENT TESTS THAT CHECK THE TA11 FOR PROPER OPERATION.

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

1. TEST THE TIMING ERROR CIRCUITRY
2. TEST THE INTERRUPT CIRCUITRY
3. TEST THE SPACING FUNCTIONS
4. TEST THE FILE GAP CIRCUITRY
5. TEST THAT DATA CAN BE WRITTEN AND READ
6. TEST CRC CIRCUITRY
7. INSURE THAT DATA CAN BE WRITTEN AND READ
   WHILE THE OTHER DRIVE IS REWINDING

DZTABC.NEW TABLE OF CONTENTS


```
   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
  345          COMMON TAGS
  406          ERROR POINTER TABLE
  473              START OF TEST
  498          INITIALIZE THE COMMON TAGS
  536          TYPE PROGRAM NAME
  543          GET VALUE FOR SOFTWARE SWITCH REGISTER
  710          T1      ROUTINE TO DETERMINE TIME OF WAIT LOOPS
  728                  **********TIMING ERROR**********
  733          T2      TEST "TIMING ERROR" FOR "WRITE"
  774          T3      TEST "TIMING ERROR" FOR "READ"
  827          T4      TEST "ERROR" OUTPUT OF STATUS ROM USING A "TIMING ERROR"
  898                  **********INTERRUPTS**********
  921          T5      TEST INTERRUPT WITH READY = "1" AT LEVEL 0
  970          T6      TEST INTERRUPT WITH READY = "1" AT LEVEL 1
 1019          T7      TEST INTERRUPT WITH READY = "1" AT LEVEL 2
 1068          T10     TEST INTERRUPT WITH READY = "1" AT LEVEL 3
 1117          T11     TEST INTERRUPT WITH READY = "1" AT LEVEL 4
 1166          T12     TEST INTERRUPT WITH READY = "1" AT LEVEL 5
 1215          T13     TEST INTERRUPT WITH READY = "1" AT LEVEL 6
 1264          T14     TEST INTERRUPT WITH READY = "1" AT LEVEL 7
 1306          T15     TEST INTERRUPT WITH "TRANSFER REQUEST" = 1
 1343          T16     TEST INTERRUPT WITH "READY" = 0 AND "XFER REQ" = 0
 1382          T17     TEST INTERRUPT WITH "XFER REQ" = 1 & "TIMING ERROR" = 1
 1450                  **********SPACING FUNCTIONS*****
 1455          T20     TEST "BACK SPACE FILE GAP"
 1486          T21     TEST "SPACE FORWARD FILE GAP" FUNCTION
 1515          T22     TEST "BACK SPACE BLOCK GAP"
 1543          T23     TEST "SPACE FWD BLOCK GAP"
 1576          T24     TEST AUTOMATIC "WFG" WHEN WRITING OFF OF "CLEAR LEADER"
 1602          T25     TEST "READ" INTO FILE GAP CAUSES AN ERROR
 1634          T26     TEST "SFBG" INTO FILE GAP CAUSES AN ERROR
 1666          T27     TEST "ERROR" OUTPUT OF THE STATUS ROM WITH "FILE GAP=1"
 1735          T30     TEST BACK-SPACE-FILE-GAP INTO CLEAR LEADER
 1783          T31     TEST BACK-SPACE-BLOCK-GAP INTO CLEAR LEADER
 1828                  **********TADB**********
 1833          T32     TEST "WRITE" 377 & 0 "READ" 377 & 0
 1869          T33     TEST "WRITE & READ" A COUNT PATTERN
 1916                  *********CRC**********
 1921          T34     TEST "ERROR" WITH "CRCERR" = 1
 1990          T35     TEST "DATA OF 0 GIVES CRC OF 0"
 2038          T36     TEST "CRC" CIRCUIT USING A COUNT PATTERN
 2112          T37     TRY TO HANG "READY" ON "REWIND"
 2166          T40     TRY TO GLITCH THE "POWER SUPPLY"
 2269          T41     END OF TEST CODE
 2279          END OF PASS ROUTINE
```

DZTABC.NEW TABLE OF CONTENTS

2315 SCOPE HANDLER ROUTINE
2379 ERROR HANDLER ROUTINE
2431 ERROR TYPEOUT ROUTINE
2467 ROUTINE TO WAIT ON THE READY BIT TO SET
2496 ROUTINE TO WAIT ON TRANSFER REQUEST
2525 ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2560 ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2624 ROUTINE TO CALCULATE THE CRC
2668 ********** MANUAL ADJUSTMENT ROUTINES **********
2678 WRITE FILE GAPS FROM "BOT" TO "EOT"
2703 WRITE CONTINUOUS BLOCKS OF DATA
2735 READ CONTINUOUS BLOCKS OF DATA
2758 WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2795 READ A BLOCK OF DATA AND A FILE GAP
2823 SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2850 BACK SPACE FILE GAP
2867 SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2901 WRITE ROUTINE FOR THE MANUAL OPERATIONS
2919 READ ROUTINE FOR THE MANUAL OPERATIONS
2944 ROUTINE TO CHANGE DRIVES
2960 ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABLITY
2990 TYPE ROUTINE
3060 READ AN OCTAL NUMBER FROM THE TTY
3098 TTY INPUT ROUTINE
3237 BINARY TO OCTAL (ASCII) AND TYPE
3314 TRAP DECODER
3337 TRAP TABLE
3358 POWER DOWN AND UP ROUTINES

```
                                .TITLE  TA11 BASIC LOGIC TEST (PART 2)  MAINDEC-11-DZTAB-C
                                ;*COPYRIGHT (C) 1976
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY JAMES 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
```

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

                                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                    100000          BIT15=  100000
                    040000          BIT14=  40000
                    020000          BIT13=  20000
                    010000          BIT12=  10000
                    004000          BIT11=  4000
                    002000          BIT10=  2000
                    001000          BIT09=  1000
                    000400          BIT08=  400
                    000200          BIT07=  200
                    000100          BIT06=  100
                    000040          BIT05=  40
                    000020          BIT04=  20
                    000010          BIT03=  10
                    000004          BIT02=  4
                    000002          BIT01=  2
                    000001          BIT00=  1
                                    .EQUIV  BIT09,BIT9
                                    .EQUIV  BIT08,BIT8
                                    .EQUIV  BIT07,BIT7
                                    .EQUIV  BIT06,BIT6
                                    .EQUIV  BIT05,BIT5
                                    .EQUIV  BIT04,BIT4
                                    .EQUIV  BIT03,BIT3
                                    .EQUIV  BIT02,BIT2
                                    .EQUIV  BIT01,BIT1
                                    .EQUIV  BIT00,BIT0

                                    ;*BASIC "CPU" TRAP VECTOR ADDRESSES
```



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

```
                                ;TA11 FUNCTIONS
            000000              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

                                        .=0
            000000              ;*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
                                        .=174
            000174              DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
            000176              SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
            000200  000137  001336      JMP     @#BEGIN1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
            000204  000137  001370      JMP     @#BEGIN2                ;SELECT DRIVE(S) BEFORE STARTING TEST
            000210  000137  001376      JMP     @#BEGIN3                ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
            000214  000137  001404      JMP     @#BEGIN4                ;SETUP FOR MANUAL LOOPING
            000220  000137  013776      JMP     @#WFGSUB                ;WRITE FILE GAP FROM BOT TO EOT
            000224  000137  014062      JMP     @#WRTSUB                ;WRITE CONTINUOUS BLOCKS OF DATA
            000230  000137  014150      JMP     @#RDSUB                 ;READ CONTINUOUS BLOCKS OF DATA
            000234  000137  014230      JMP     @#WGPBLK                ;WRITE FILE GAP AND A BLOCK OF DATA
            000240  000137  014332      JMP     @#RGPBLK                ;READ BLOCK OF DATA AND INTO A FILE GAP
            000244  000137  014426      JMP     @#SFFGSB                ;SPACE FWD FILE GAP FROM BOT TO EOT
            000250  000137  014512      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
001236  017066                          EM1     ;STATUS PROBLEM
001240  017256                          DH1     ;PC     TACS
001242  017420                          DT1     ;$ERRPC $REG0
001244  000000                          0

                                ;ITEM   2
001246  017105                          EM2     ;READY FAILED TO SET
001250  017273                          DH2     ;PC     TACS    WAIT ADDRESS
001252  017426                          DT2     ;$ERRPC $REG0   SAVPC
001254  000000                          0

                                ;ITEM   3
001256  017133                          EM3     ;TRANSFER REQUEST FAILED TO SET
001260  017273                          DH2     ;PC     TACS    WAIT ADDRESS
001262  017426                          DT2     ;$ERRPC $REG0   SAVPC
001264  000000                          0

                                ;ITEM   4
001266  017174                          EM4     ;THE WRONG FLAG SET
001270  017273                          DH2     ;PC     TACS    WAIT ADDRESS
001272  017426                          DT2     ;$ERRPC $REG0   SAVPC
001274  000000                          0

                                ;ITEM   5
001276  017217                          EM5     ;DATA PROBLEM
001300  017330                          DH5     ;PC     TACS    EXPECT  RCV'D
001302  017436                          DT5     ;$ERRPC $REG0   $GDDAT  $BDDAT
001304  000000                          0

                                ;ITEM   6
001306  017234                          EM6     ;INTERRUPT PROBLEM
001310  017366                          DH6     ;PC     TACS    BR LEVEL
001312  017450                          DT6     ;$ERRPC $REG0   $GDDAT
001314  000000                          0
```



```
001316                          ITEMS2:         ;ITEMS 201-202

001316  017472                          EM201   ;TA11 FAILED TO RESPOND
001320  017544                          DH201   ;PC     TACS
001322  017460                          DT201   ;$ERRPC TACS
001324  000000                          0       ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS

001326  017521                          EM202   ;NO DRIVES AVAILABLE
001330  017561                          DH202   ;PC
001332  017466                          DT202   ;$ERRPC
001334  000000                          0       ;
```

```
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;;**************************************************************
                                        ;BEGIN1 IS FOR NORMAL START
                                        ;BEGIN2 IS FOR DRIVE SELECTION
                                        ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
                                        ;BEGIN4 IS FOR MANUAL OPERATION
                                        ;;**************************************************************
001336  005005                  BEGIN1: CLR     R5                      ;NORMAL START
001340  012737  041101  001224          MOV     #"AB,@#DRVKEY
001346  122737  000005  000041          CMPB    #5,@#41                 ;CASSETTE DDP?
001354  001015                         BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001356  022737  000260  001216          CMP     #260,@#TAVEC            ;STANDARD VECTOR?
001364  001011                         BNE     BGNCMN                  ;GO BEGIN COMMON CODE IF NO
001366  000403                         BR      BEGIN3                  ;GET DRIVES AND ADDRESSES
001370  012705  000001          BEGIN2: MOV     #1,R5                   ;ASK FOR DRIVES FLAG
001374  000405                         BR      BGNCMN                  ;BEGIN COMMON CODE
001376  012705  000002          BEGIN3: MOV     #2,R5                   ;ASK FOR DRIVES AND ADDRESSES
001402  000402                         BR      BGNCMN
001404  012705  000003          BEGIN4: MOV     #3,R5
001410                          BGNCMN:
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001410  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001414  005026                         CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001416  022706  001140                  CMP     #SWR,R6 ;;DONE?
001422  001374                         BNE     .-6             ;;LOOP BACK IF NO
001424  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001430  012737  012264  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001436  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001444  012737  012536  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001452  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001460  012737  016476  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001466  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001474  012737  016562  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001502  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001510  016767  010476  010466          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
001516  005067  177444                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001522  005067  177442                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001526  112767  000001  177361          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
001534  012767  001534  177344          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001542  012767  001542  177340          MOV     #.,$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.
001550  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001554  012737  001610  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001562  012767  177570  177350          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001570  012767  177570  177344          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001576  022777  177777  177334          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001604  001012                         BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
001606  000403                         BR      65$             ;;BRANCH IF NO TIMEOUT
```



```
001610  012716  001616          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001614  000002                         RTI
001616  012767  000176  177314  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001624  012767  000174  177310          MOV     #DISPREG,DISPLAY
001632  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001636  005227  177777                  INC     #-1             ;;FIRST TIME?
001642  001036                         BNE     HERE            ;;BRANCH IF NO
001644  022737  012232  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
001652  001432                         BEQ     HERE            ;;BRANCH IF YES
001654  104401  001712                  TYPE    ,MSGID          ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
001660  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
001664  001006                         BNE     67$             ;;BRANCH IF YES
001666  026727  177246  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
001674  001005                         BNE     68$             ;;BRANCH IF NO
001676  104405                         GTSWR                   ;;GET SOFT-SWR SETTINGS
001700  000403                         BR      68$
001702  112767  000001  177224  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
001710                          68$:
001710  000413                         BR      HERE            ;;GET OVER THE ASCIZ
                                ;;MSGID:        .ASCIZ  <CRLF>/MAINDEC-11-DZTAB-C/<CRLF>
001740                          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
                                        ;
                                        ;;**************************************************************
001740  010504                  BEGINX: MOV     R5,R4                   ;COPY R5
001742  005305                         DEC     R5                      ;ASK FOR DRIVES?
001744  002406                         BLT     CHKADR                  ;BR IF NO
001746  004737  013254                  JSR     PC,@#ASKDRV             ;GO GET DRIVES TO BE TESTED
001752  005305                         DEC     R5                      ;ASK FOR ADDRESSES?
001754  002402                         BLT     CHKADR                  ;BR IF NO
001756  004737  013364                  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
                                ;
                                ;;*************************************************************
001762  012737  002000  000004  CHKADR: MOV     #1$,@#ERRVEC            ;IN CASE OF TIMEOUTS
001770  005000                          CLR     R0                      ;USE AS A SWITCH
001772  005777  177210                  TST     @TACSL                  ;SEE IF TA11 RESPONDS
001776  000402                          BR      2$                      ;BR IF NO TIMEOUT
002000  005200                  1$:     INC     R0                      ;COME HERE ON TIMEOUT
002002  022626                          CMP     (SP)+,(SP)+             ;CLEANUP THE STACK
002004  012737  000006  000004  2$:     MOV     #ERRVEC+2,@#ERRVEC      ;RESTORE TIMEOUT VECTOR
002012  005700                          TST     R0                      ;DID A TIMEOUT OCCUR?
002014  001414                          BEQ     3$                      ;BR IF NO
002016  104201                          ERROR   201                     ;TA11 FAILED TO RESPOND
002020  012705  000002                  MOV     #2,R5                   ;DRIVES & ADDRESSES
002024  005704                          TST     R4                      ;OPERATOR INPUTS?
002026  001344                          BNE     BEGINX                  ;BR IF YES
002030  013700  000042                  MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002034  001741                          BEQ     BEGINX                  ;BR IF NO MONITOR
002036  000137  012232                  JMP     @#$ENDAD                ;GO TO END
002042  005077  177152                  CLR     @TAVEC+2
002046                          3$:
```



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

                                ;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 $ENDAD
                                ;
                                ;II.    BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
                                ;       A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
                                ;       B. CONTINUE IN PROGRAM
                                ;
                                ;III.   DESIRED DRIVE(S) CAN BE TESTED
                                ;       A. CONTINUE IN PROGRAM
                                ;
                                ;;*************************************************************
002046  012700  001224          CHKDRV: MOV     #DRVKEY,R0              ;PICKUP ADDRESS OF ASCII DRIVE KEY
002052  004737  015046                  JSR     PC,@#EXAM               ;GO EXAMINE FIRST DRIVE
002056  000410                          BR      1$                      ;OK TO TEST---GO CHECK NEXT
002060  116010  000001                  MOVB    1(R0),(R0)              ;REPLACE 1ST WITH 2ND
002064  001412                          BEQ     2$                      ;BR IF NO 2ND DRIVE SELECTED
002066  004737  015046                  JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002072  000407                          BR      2$                      ;OK TO TEST
002074  005010                          CLR     (R0)                    ;CLEAR DRIVE CODES
002076  000405                          BR      2$
002100  005200                  1$:     INC     R0                      ;POINT TO 2ND
002102  004737  015046                  JSR     PC,@#EXAM               ;GO EXAMINE DRIVE
002106  000401                          BR      2$                      ;OK TO TEST
002110  105010                          CLRB    (R0)                    ;CLEAR 2ND
002112  012700  001224          2$:     MOV     #DRVKEY,R0              ;RESET ADDRESS POINTERS
002116  010037  001230                  MOV     R0,@#DRVPNT
002122  121060  000001                  CMPB    (R0),1(R0)              ;1ST = 2ND?
002126  001002                          BNE     3$                      ;BR IF NO
002130  105060  000001                  CLRB    1(R0)                   ;YES---CLEAR 2ND
002134  005710                  3$:     TST     (R0)                    ;ANY DRIVES?
002136  001401                          BEQ     5$                      ;BR IF NO
002140  000412                          BR      MANUAL
002142  104202                  5$:     ERROR   202                     ;NO DRIVES AVAILABLE
002144  012705  000002                  MOV     #2,R5                   ;DRIVES & ADDRESS
002150  005704                          TST     R4                      ;OPERATOR INPUTS?
002152  001272                          BNE     BEGINX                  ;BR IF YES
002154  013700  000042                  MOV     @#42,R0                 ;GET MONITOR RETURN ADDRESS
002160  001667                          BEQ     BEGINX                  ;NO MONITOR
002162  000137  012232                  JMP     @#$ENDAD                ;GO TO END
002166  020427  000003          MANUAL: CMP R4,#3
002172  001002                          BNE     OK
002174  016704  175577                  MOV     -1,R4
002200  010437  001232          OK:     MOV     R4,@#ASKKEY
002204  000405                          BR      START
002206  104401  001712          PWRST:  TYPE    ,MSGID                  ;POWER FAIL RESTART
002212  012737  001224  001230          MOV     #DRVKEY,@#DRVPNT
```

```
   655  002220  013777  001220  176770  START:  MOV     @#TAVEC+2,@TAVEC        ;SETUP TA11 TRAP VECTOR
   656  002226  005077  176766                  CLR     @TAVEC+2
   657  002232  012737  000340  177776          MOV     #340,@#PS               ;LOCKOUT ALL I/O INT
   658  002240  013704  001206                  MOV     @#TACSL,TACS            ;SETUP TACS
   659  002244  013705  001212                  MOV     @#TADBL,TADB            ;SETUP TADB
   660  002250  005737  001100                  TST     @#$PASS                 ;IF FIRST PASS SETUP FOR EXTRA LONG WAIT LOOPS
   661  002254  001003                          BNE     1$                      ;OTHERWISE USE OLD VALUES
   662  002256  012737  077777  013150          MOV     #^CBIT15,@#MAXCNT
   663  002264  005037  001102          1$:     CLR     @#STSTNM                ;ZERO THE TEST NUMBER
   664  002270  005003                          CLR     DRIVE                   ;SET DRIVE TO "A"
   665  002272  013701  001230                  MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
   666  002276  121127  000101                  CMPB    (R1),#'A                ;IS IT DRIVE "A"?
   667  002302  001402                          BEQ     TDRV                    ;BR IF YES
   668  002304  012703  000400                  MOV     #UNIT,DRIVE             ;SET DRIVE TO "B"
   669  002310                          TDRV:
   670  002310  104401  002316                  TYPE    ,65$            ;;TYPE ASCIZ STRING
   671  002314  000411                          BR      64$             ;;GET OVER THE ASCIZ
   672                                  ;;65$:  .ASCIZ  <15><12>*TESTING DRIVE *
   673  002340                          64$:
   674  002340  112167  176670                  MOVB    (R1)+,CURDRV            ;SETUP TO TYPE CURRENT DRIVE
   675  002344  104401  001234                  TYPE    ,CURDRV
   676  002350  104401  001177                  TYPE    ,$CRLF                  ;TYPE A  CR & LF
   677  002354  105711                          TSTB    (R1)                    ;LAST DRIVE BEEN SELECTED
   678  002356  001002                          BNE     1$                      ;BR IF NO
   679  002360  012701  001224                  MOV     #DRVKEY,R1              ;RESET DRIVE POINTER
   680  002364  010137  001230          1$:     MOV     R1,@#DRVPNT             ;SAVE DRIVE POINTER FOR NEXT TIME
   681  002370  005737  001232                  TST     @#ASKKEY                ;GO START TESTING IF NO MANUAL
   682  002374  002007                          BGE     2$                      ;       OPERATIONS REQUESTED
   683  002376  005000                          CLR     R0
   684  002400  000000                          HALT                            ;GIVE CONTROL TO THE OPERATOR
   685  002402  022767  000176  176530          CMP     #SWREG,SWR              ;USING S/W SWITCH REG.?
   686  002410  001001                          BNE     20$                     ;NO- GET OUT
   687  002412  104405                          GTSWR                           ;LET HIM CHANGE IT
   688  002414                          20$:                                    ;CONTINUE
   689                                  ;THIS CODE IS FOR ACT11 & DDP
   690  002414  005737  000042          2$:     TST     @#42                    ;IS THERE A MONITOR?
   691  002420  001406                          BEQ     TST1                    ;GO START TESTING IF NO
   692  002422  010314                          MOV     DRIVE,@TACS             ;IF YES SELECT DRIVE
   693  002424  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO BOT
   694  002430  032714  000040          3$:     BIT     #READY,@TACS            ;WAIT ON READY
   695  002434  001775                          BEQ     3$                      ;FALL THRU IF READY=1
```



```
   696                                  ;////////////////////////////////////////////////////////////////////
   697                                  ;////////////////////////////////////////////////////////////////////
   698                                  ;;****************************************************************
   699                                  ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX.
   700                                  ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
   701                                  ;;****************************************************************
   702                                  ;*TEST 1         ROUTINE TO DETERMINE TIME OF WAIT LOOPS
   703                                  ;;****************************************************************
   704  002436  000004                  TST1:   SCOPE
   705  002440  012767  000001  176520          MOV     #1,$TIMES       ;;DO 1 ITERATION
   706  002446  005737  001100                  TST     @#$PASS                 ;IS THIS THE FIRST PASS?
   707  002452  001021                          BNE     TST2            ;;BR IF NO
   708  002454  000005                          RESET
   709  002456  010314                          MOV     DRIVE,@TACS             ;SELECT THE DRIVE
   710  002460  112714  000017                  MOVB    #REWIND!GO,@TACS        ;START A REWIND
   711  002464  104412                          WAITREADY                       ;WAIT FOR READY
   712  002466  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
   713  002472  104412                          WAITREADY                       ;WAIT ON READY
   714  002474  163737  013124  013150          SUB     @#HGHTIM,@#MAXCNT       ;GET THE TIME IT TOOK
   715  002502  005237  013150                  INC     @#MAXCNT                ;MAKE IT BIGGER
   716  002506  006137  013150                  ROL     @#MAXCNT
   717  002512  006137  013150                  ROL     @#MAXCNT
   718                                  ;////////////////////////////////////////////////////////////////////
   719                                  ;////////////////////////////////////////////////////////////////////
   720                                  ;THE FOLLOWING TEST WILL CHECK TIMING ERRORS FOR BOTH "WRITE" AND "READ".
   721                                  ;////////////////////////////////////////////////////////////////////
   722                                  ;;****************************************************************
   723                                  ;*TEST 2         TEST "TIMING ERROR" FOR "WRITE"
   724                                  ;;****************************************************************
   725  002516  000004                  TST2:   SCOPE
   726  002520  012767  000012  176440          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
   727  002526  012767  002554  176352          MOV     #7$,$LPADR      ;;SET SCOPE LOOP ADDRESS
   728  002534  012767  002656  176426          MOV     #TST3,$ESCAPE   ;;ESCAPE TO TEST 3 ON ERROR
   729  002542  000005                          RESET
   730  002544  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
   731  002546  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
   732  002552  104412                          WAITREADY                       ;WAIT ON "READY"
   733  002554  112714  000003          7$:     MOVB    #WRITE!GO,@TACS         ;WRITE FUNCTION
   734  002560  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
   735  002562  105714                          TSTB    @TACS                   ;IS "XFER REQ" = 1
   736  002564  100401                          BMI     1$                      ;BR IF YES
   737  002566  104001                          ERROR   1                       ;NO "XFER REQ" OCCURRED
   738  002570  005015                  1$:     CLR     @TADB                   ;KNOCK DOWN "XFER REQ"
   739  002572  005000                          CLR     R0                      ;CLEAR COUNTER
   740  002574  005001                          CLR     R1
   741  002576  105714                  2$:     TSTB    @TACS                   ;WAIT FOR "XFER REQ"
   742  002600  100405                          BMI     3$
   743  002602  062700  000010                  ADD     #10,R0                  ;KEEP TRACK OF HOW LONG
   744  002606  005501                          ADC     R1                      ;IT TAKES "XFER REQ" TO SET
   745  002610  100372                          BPL     2$
   746  002612  104001                          ERROR   1                       ;"XFER REQ" FAILED TO SET
   747  002614  105714                  3$:     TSTB    @TACS                   ;SAMPLE "XFER REQ"
   748  002616  100005                          BPL     4$                      ;IF "XFER REQ" = 0 SOMETHING IS WRONG
   749  002620  162700  000001                  SUB     #1,R0                   ;DOWN COUNT THE COUNTER
   750  002624  005601                          SBC     R1                      ;SO A "TIMING ERROR WILL
   751  002626  100372                          BPL     3$                      ;OCCUR
```

```
                                        BR      5$
                                4$:     ERROR   1                       ;"XFER REQ" SHOULDN'T HAVE CLEARED
                                5$:     TSTB    @TADB                   ;CLEAR "XFER REQ"
                                        WAITREADY                       ;WAIT FOR "READY"
                                        TST     @TACS                   ;IS "ERROR" BIT SET?
                                        BMI     6$                      ;BR IF YES
                                        ERROR   1                       ;"ERROR" BIT NOT SET
                                6$:     BIT     #TIMERR,@TACS           ;IS "TIMING ERROR" = 1?
                                        BNE     TST3            ;;BR IF YES
                                        ERROR   1                       ;"TIMING ERROR" NOT SET
                                ;;**************************************************************
                                ;*TEST 3         TEST "TIMING ERROR" FOR "READ"
                                ;;**************************************************************
                                TST3:   SCOPE
                                        MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
                                        MOV     #7$,$LPADR      ;;SET SCOPE LOOP ADDRESS
                                        MOV     #TST4,$ESCAPE   ;;ESCAPE TO TEST 4 ON ERROR
                                        RESET
                                        MOV     DRIVE,@TACS             ;SELECT DRIVE
                                        MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
                                        WAITREADY                       ;WAIT FOR "READY"
                                7$:     MOVB    #WRITE!GO,@TACS         ;WRITE
                                        MOV     #6,R0                   ;WRITE THIS MANY BYTES ON TAPE
                                        WAITXFER                        ;WAIT FOR "XFER REQ"
                                8$:     MOVB    #377,@TADB              ;WRITE ONE BYTE
                                        WAITXFER                        ;WAIT FOR "XFER REQ"
                                        DEC     R0                      ;LAST BYTE OUT?
                                        BGT     8$                      ;BR IF NO
                                        BIS     #ILBS,@TACS             ;WRITE "CRC"
                                        WAITREADY                       ;WAIT FOR "READY"
                                        MOVB    #REWIND!GO,@TACS        ;BACK OVER THE BLOCK
                                        WAITREADY                       ;WAIT FOR "READY"
                                        MOVB    #READ!GO,@TACS          ;START A "READ"
                                        WAITXFER                        ;WAIT FOR "XFER REQ"
                                        TSTB    @TACS                   ;IS "XFER REQ" SET
                                        BMI     1$                      ;BR IF YES
                                        ERROR   1                       ;"XFER REQ" DIDN'T SET
                                1$:     TSTB    @TADB                   ;KNOCK DOWN "XFER REQ"
                                        CLR     R0                      ;CLEAR COUNTERS
                                        CLR     R1
                                2$:     TSTB    @TACS                   ;FIND OUT HOW LONG IT
                                        BMI     3$                      ;TAKES FOR "XFER REQ"
                                        ADD     #10.,R0                 ;TO SET
                                        ADC     R1
                                        BPL     2$
                                        ERROR   1                       ;"XFER REQ" FAILED TO SET
                                3$:     TSTB    @TACS                   ;NOW WASTE MORE THAN THE ABOVE
                                        BPL     4$                      ;AMOUNT OF TIME SO THAT
                                        SUB     #1,R0                   ;A "TE" WILL OCCUR
                                        SBC     R1
                                        BPL     3$
                                        BR      5$
                                4$:     ERROR   1                       ;"XFER REQ" CLEARED SOME HOW
                                5$:     TSTB    @TADB                   ;KNOCK DOWN "XFER REQ"
                                        BIS     #ILBS,@TACS             ;STOP THE READ
                                        WAITREADY                       ;WAIT ON "READY"
```



```
                                        TST     @TACS                   ;IS "ERROR" SET?
                                        BMI     6$                      ;BR IF YES
                                        ERROR   1                       ;"ERROR" NOT = 1
                                6$:     BIT     #TIMERR,@TACS           ;IS THERE A "TIMING ERROR"
                                        BNE     TST4            ;;GO TO NEXT TEST IF YES
                                        ERROR   1                       ;"TIMING ERROR" NOT SET
                                ;;**************************************************************
                                ;*TEST 4         TEST "ERROR" OUTPUT OF STATUS ROM USING A "TIMING ERROR"
                                ;;**************************************************************
                                TST4:   SCOPE
                                        MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
                                        MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
                                        MOV     #TST5,$ESCAPE   ;;ESCAPE TO TEST 5 ON ERROR
                                        RESET
                                        MOV     DRIVE,@TACS             ;SELECT DRIVE
                                        MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
                                        WAITREADY                       ;WAIT ON "READY" TO SET
                                        MOVB    #WFG!GO,@TACS           ;GET ON THE OXIDE
                                        WAITREADY                       ;WAIT ON "READY" TO SET
                                1$:     MOVB    #WRITE!GO,@TACS         ;START A WRITE
                                        WAITXFER                        ;WAIT FOR "TRANSFER REQ"
                                        MOVB    #377,@TADB              ;WRITE ONE BYTE
                                        CLR     R0                      ;CLEAR THE DOUBLE LENGTH COUNTER
                                        CLR     R1
                                2$:     TSTB    @TACS                   ;CHECK FOR "XFER REQ"
                                        BMI     3$                      ;BR IF SET
                                        ADD     #10.,R0                 ;COUNT UP UNTIL "XFER REQ"
                                        ADC     R1                      ;SETS OR OVERFLOW OCCURS
                                        BPL     2$
                                        ERROR   1                       ;"XFER REQ" FAILED TO SET
                                3$:     TSTB    @TACS                   ;CHECK "XFER REQ"
                                        BMI     4$                      ;BR IF SET
                                        ERROR   1                       ;"XFER REQ" SHOULDN'T HAVE CLEARED
                                4$:     SUB     #1,R0                   ;COUNT DOWN TO CAUSE A "TE"
                                        SBC     R1
                                        BPL     3$
                                        TSTB    @TADB                   ;CLEAR "XFER REQ"
                                        WAITREADY                       ;WAIT FOR READY
                                        TST     @TACS                   ;MAKE SURE "ERROR" IS SET
                                        BMI     5$
                                        ERROR   1                       ;"ERROR" DIDN'T SET
                                5$:     BIT     #TIMERR,@TACS           ;MAKE SURE "TE" IS SET
                                        BNE     6$
                                        ERROR   1                       ;"TE" FAILED TO SET
                                6$:     MOVB    #WFG,@TACS              ;CHECK "ERROR" WITH "WFG"
                                        TST     @TACS                   ;SAMPLE THE "ERROR" BIT
                                        BPL     7$                      ;BR IF "ERROR" = 0
                                        ERROR   1                       ;"ERROR" NOT = 0
                                7$:     MOVB    #WRITE,@TACS            ;CHECK "ERROR" WITH "WRITE"
                                        TST     @TACS                   ;SAMPLE THE "ERROR" BIT
                                        BMI     8$                      ;BR IF "ERROR" = 1
                                        ERROR   1                       ;"ERROR" NOT = 1
                                8$:     MOVB    #READ,@TACS             ;CHECK "ERROR" WITH "READ"
                                        TST     @TACS                   ;SAMPLE THE "ERROR" BIT
                                        BMI     9$                      ;BR IF "ERROR" = 1
                                        ERROR   1                       ;"ERROR" NOT = 1
```

```
003260  112714  000006          9$:     MOVB    #BSFG,@TACS             ;CHECK "ERROR" WITH "BSFG"
003264  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
003266  100001                          BPL     10$                     ;BR IF "ERROR" = 0
003270  104001                          ERROR   1                       ;"ERROR" NOT = 0
003272  112714  000010          10$:    MOVB    #BSBG,@TACS             ;CHECK "ERROR" WITH "BSBG"
003276  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
003300  100001                          BPL     11$                     ;BR IF "ERROR" = 0
003302  104001                          ERROR   1                       ;"ERROR" NOT = 0
003304  112714  000012          11$:    MOVB    #SFFG,@TACS             ;CHECK "ERROR" WITH "SFFG"
003310  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
003312  100001                          BPL     12$                     ;BR IF "ERROR" = 0
003314  104001                          ERROR   1                       ;"ERROR" NOT = 0
003316  112714  000014          12$:    MOVB    #SFBG,@TACS             ;CHECK "ERROR" WITH "SFBG"
003322  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
003324  100001                          BPL     13$                     ;BR IF "ERROR" = 0
003326  104001                          ERROR   1                       ;"ERROR" NOT = 0
003330  112714  000016          13$:    MOVB    #REWIND,@TACS           ;CHECK "ERROR" WITH "REWIND"
003334  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
003336  100001                          BPL     TST5            ;;BR IF "ERROR" = 0
003340  104001                          ERROR   1                       ;"ERROR" NOT = 0
                                ;////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS CHECK THE INTERRUPT CIRCUITRY FOR:
                                ;
                                ;1)     INTERRUPT OCCURS AT THE PROPER "BR" LEVEL
                                ;2)     "READY" WILL INTERRUPT
                                ;3)     "TRANSFER REQUEST" WILL INTERRUPT
                                ;4)     IMPROPER INTERRUPTS DO NOT OCCUR
                                ;5)     WITH AN ERROR PENDING AND "XFER REQ." SET THE FOLLOWING SEQUENCE OCCURS
                                ;       A)      "XFER REQ" CAUSES AN INTERRUPT
                                ;       B)      CLEARING "XFER REQ" ALLOWS "READY" TO SET
                                ;       C)      "READY" CAUSES AN INTERRUPT
                                ;
                                ;////////////////////////////////////////////////////////////////

                                ;;*************************************************************
                                        ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
                                        ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
                                        ;IT WILL THEN SET THE PRIORITY LEVEL TO 0 AND WASTE ENOUGH TIME
                                        ;FOR AN INTERRUPT TO OCCUR.
                                        ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
                                        ;AND CHECK IF AN INTERRUPT OCCURRED.
                                ;;*************************************************************
                                ;*TEST 5         TEST INTERRUPT WITH READY = "1" AT LEVEL 0
                                ;;*************************************************************
000004                  TST5:   SCOPE
003342  012767  003362  175534          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003352  012767  003520  175610          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
000005                          RESET
003362  010314                  1$:     MOV     DRIVE,@TACS
003364  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
003372  012700  000000                  MOV     #0,R0                   ;TEST AT PRIORITY LEVEL 0
003376  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
003400  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
003404  012777  003510  175604          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
003412  012777  000340  175600          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
```



```
003420  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
003426  012737  003514  000002          MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
003434  112714  000100                  MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
003440  012737  000000  177776          MOV     #0*40,@#PS              ;SET TO PRIORITY LEVEL 0
003446  006302                  2$:     ASL     R2                      ;KILL SOME TIME
003450  001376                          BNE     2$
003452  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
003460  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
003462  001005                          BNE     3$                      ;BR IF YES
003464  022737  000000  001222          CMP     #0*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
003472  002012                          BGE     6$                      ;BR IF NO
003474  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 0

003476  022737  000000  001222  3$:     CMP     #0*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
003504  002405                          BLT     6$                      ;BR IF YES
003506  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 0

003510  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
003512  000002                          RTI                             ;RETURN AFTER INTERRUPT
003514  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
003516  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0

003520  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
003522  013777  001220  175466          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
003530  005077  175464                  CLR     @TAVEC+2
003534  005037  000000                  CLR     @#0
003540  005037  000002                  CLR     @#2
                                ;;*************************************************************
                                        ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
                                        ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
                                        ;IT WILL THEN SET THE PRIORITY LEVEL TO 1 AND WASTE ENOUGH TIME
                                        ;FOR AN INTERRUPT TO OCCUR.
                                        ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
                                        ;AND CHECK IF AN INTERRUPT OCCURRED.
                                ;;*************************************************************
                                ;*TEST 6         TEST INTERRUPT WITH READY = "1" AT LEVEL 1
                                ;;*************************************************************
003544  000004                  TST6:   SCOPE
003546  012767  003564  175332          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
003554  012767  003722  175406          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
003562  000005                          RESET
003564  010314                  1$:     MOV     DRIVE,@TACS
003566  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
003574  012700  000001                  MOV     #1,R0                   ;TEST AT PRIORITY LEVEL 1
003600  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
003602  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
003606  012777  003712  175402          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
003614  012777  000340  175376          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
003622  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
003630  012737  003716  000002          MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
003636  112714  000100                  MOVB    #INT,EN,@TACS           ;SET INTERRUPT ENABLE
003642  012737  000040  177776          MOV     #1*40,@#PS              ;SET TO PRIORITY LEVEL 1
003650  006302                  2$:     ASL     R2                      ;KILL SOME TIME
003652  001376                          BNE     2$
003654  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
003662  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
```

```
 976  003664  001005                          BNE     3$                      ;BR IF YES
 977  003666  022737  000040  001222          CMP     #1*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
 978  003674  002012                          BGE     6$                      ;BR IF NO
 979  003676  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 1
 980
 981  003700  022737  000040  001222  3$:     CMP     #1*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
 982  003706  002405                          BLT     6$                      ;BR IF YES
 983  003710  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 1
 984
 985  003712  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
 986  003714  000002                          RTI                             ;RETURN AFTER INTERRUPT
 987  003716  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
 988  003720  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
 989
 990  003722  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
 991  003724  013777  001220  175264          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
 992  003732  005077  175262                  CLR     @TAVEC+2
 993  003736  005037  000000                  CLR     @#0
 994  003742  005037  000002                  CLR     @#2
 995                                  ;;*************************************************************
 996                                          ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
 997                                          ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
 998                                          ;IT WILL THEN SET THE PRIORITY LEVEL TO 2 AND WASTE ENOUGH TIME
 999                                          ;FOR AN INTERRUPT TO OCCUR.
1000                                          ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1001                                          ;AND CHECK IF AN INTERRUPT OCCURRED.
1002                                  ;;*************************************************************
1003                                  ;*TEST 7         TEST INTERRUPT WITH READY = "1" AT LEVEL 2
1004                                  ;;*************************************************************
1005  003746  000004                  TST7:   SCOPE
1006  003750  012767  003766  175130          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1007  003756  012767  004124  175204          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
1008  003764  000005                          RESET
1009  003766  010314                  1$:     MOV     DRIVE,@TACS
1010  003770  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1011  003776  012700  000002                  MOV     #2,R0                   ;TEST AT PRIORITY LEVEL 2
1012  004002  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
1013  004004  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1014  004010  012777  004114  175200          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1015  004016  012777  000340  175174          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
1016  004024  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1017  004032  012737  004120  000002          MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
1018  004040  112714  000100                  MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1019  004044  012737  000100  177776          MOV     #2*40,@#PS              ;SET TO PRIORITY LEVEL 2
1020  004052  006302                  2$:     ASL     R2                      ;KILL SOME TIME
1021  004054  001376                          BNE     2$
1022  004056  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
1023  004064  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
1024  004066  001005                          BNE     3$                      ;BR IF YES
1025  004070  022737  000100  001222          CMP     #2*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1026  004076  002012                          BGE     6$                      ;BR IF NO
1027  004100  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 2
1028
1029  004102  022737  000100  001222  3$:     CMP     #2*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1030  004110  002405                          BLT     6$                      ;BR IF YES
1031  004112  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 2
```



```
1032
1033  004114  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
1034  004116  000002                          RTI                             ;RETURN AFTER INTERRUPT
1035  004120  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1036  004122  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1037
1038  004124  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1039  004126  013777  001220  175062          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1040  004134  005077  175060                  CLR     @TAVEC+2
1041  004140  005037  000000                  CLR     @#0
1042  004144  005037  000002                  CLR     @#2
1043                                  ;;*************************************************************
1044                                          ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1045                                          ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1046                                          ;IT WILL THEN SET THE PRIORITY LEVEL TO 3 AND WASTE ENOUGH TIME
1047                                          ;FOR AN INTERRUPT TO OCCUR.
1048                                          ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1049                                          ;AND CHECK IF AN INTERRUPT OCCURRED.
1050                                  ;;*************************************************************
1051                                  ;*TEST 10        TEST INTERRUPT WITH READY = "1" AT LEVEL 3
1052                                  ;;*************************************************************
1053  004150  000004                  TST10:  SCOPE
1054  004152  012767  004170  174726          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1055  004160  012767  004326  175002          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
1056  004166  000005                          RESET
1057  004170  010314                  1$:     MOV     DRIVE,@TACS
1058  004172  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1059  004200  012700  000003                  MOV     #3,R0                   ;TEST AT PRIORITY LEVEL 3
1060  004204  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
1061  004206  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1062  004212  012777  004316  174776          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1063  004220  012777  000340  174772          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
1064  004226  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1065  004234  012737  004322  000002          MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
1066  004242  112714  000100                  MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1067  004246  012737  000140  177776          MOV     #3*40,@#PS              ;SET TO PRIORITY LEVEL 3
1068  004254  006302                  2$:     ASL     R2                      ;KILL SOME TIME
1069  004256  001376                          BNE     2$
1070  004260  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
1071  004266  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
1072  004270  001005                          BNE     3$                      ;BR IF YES
1073  004272  022737  000140  001222          CMP     #3*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1074  004300  002012                          BGE     6$                      ;BR IF NO
1075  004302  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 3
1076
1077  004304  022737  000140  001222  3$:     CMP     #3*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1078  004312  002405                          BLT     6$                      ;BR IF YES
1079  004314  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 3
1080
1081  004316  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
1082  004320  000002                          RTI                             ;RETURN AFTER INTERRUPT
1083  004322  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1084  004324  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1085
1086  004326  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1087  004330  013777  001220  174660          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
```

```
1088  004336  005077  174656                  CLR     @TAVEC+2
1089  004342  005037  000000                  CLR     @#0
1090  004346  005037  000002                  CLR     @#2
1091                                  ;;*************************************************************
1092                                          ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1093                                          ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1094                                          ;IT WILL THEN SET THE PRIORITY LEVEL TO 4 AND WASTE ENOUGH TIME
1095                                          ;FOR AN INTERRUPT TO OCCUR.
1096                                          ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1097                                          ;AND CHECK IF AN INTERRUPT OCCURRED.
1098                                  ;;*************************************************************
1099                                  ;*TEST 11        TEST INTERRUPT WITH READY = "1" AT LEVEL 4
1100                                  ;;*************************************************************
1101  004352  000004                  TST11:  SCOPE
1102  004354  012767  004372  174524          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1103  004362  012767  004530  174600          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
1104  004370  000005                          RESET
1105  004372  010314                  1$:     MOV     DRIVE,@TACS
1106  004374  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1107  004402  012700  000004                  MOV     #4,R0                   ;TEST AT PRIORITY LEVEL 4
1108  004406  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
1109  004410  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1110  004414  012777  004520  174574          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1111  004422  012777  000340  174570          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
1112  004430  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1113  004436  012737  004524  000002          MOV     #5$,@#2                 ;    IT GOES TO LOCATION 0
1114  004444  112714  000100                  MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1115  004450  012737  000200  177776          MOV     #4*40,@#PS              ;SET TO PRIORITY LEVEL 4
1116  004456  006302                  2$:     ASL     R2                      ;KILL SOME TIME
1117  004460  001376                          BNE     2$
1118  004462  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
1119  004470  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
1120  004472  001005                          BNE     3$                      ;BR IF YES
1121  004474  022737  000200  001222          CMP     #4*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1122  004502  002012                          BGE     6$                      ;BR IF NO
1123  004504  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 4
1124
1125  004506  022737  000200  001222  3$:     CMP     #4*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1126  004514  002405                          BLT     6$                      ;BR IF YES
1127  004516  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 4
1128
1129  004520  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
1130  004522  000002                          RTI                             ;RETURN AFTER INTERRUPT
1131  004524  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1132  004526  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1133
1134  004530  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1135  004532  013777  001220  174456          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1136  004540  005077  174454                  CLR     @TAVEC+2
1137  004544  005037  000000                  CLR     @#0
1138  004550  005037  000002                  CLR     @#2
1139                                  ;;*************************************************************
1140                                          ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1141                                          ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1142                                          ;IT WILL THEN SET THE PRIORITY LEVEL TO 5 AND WASTE ENOUGH TIME
1143                                          ;FOR AN INTERRUPT TO OCCUR.
```



```
1144                                          ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1145                                          ;AND CHECK IF AN INTERRUPT OCCURRED.
1146                                  ;;*************************************************************
1147                                  ;*TEST 12        TEST INTERRUPT WITH READY = "1" AT LEVEL 5
1148                                  ;;*************************************************************
1149  004554  000004                  TST12:  SCOPE
1150  004556  012767  004574  174322          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1151  004564  012767  004732  174376          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
1152  004572  000005                          RESET
1153  004574  010314                  1$:     MOV     DRIVE,@TACS
1154  004576  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1155  004604  012700  000005                  MOV     #5,R0                   ;TEST AT PRIORITY LEVEL 5
1156  004610  005001                          CLR     R1                      ;CLEAR INTERRUPT KEY
1157  004612  012702  000001                  MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1158  004616  012777  004722  174372          MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1159  004624  012777  000340  174366          MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
1160  004632  012737  000137  000000          MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1161  004640  012737  004726  000002          MOV     #5$,@#2                 ;    IT GOES TO LOCATION 0
1162  004646  112714  000100                  MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1163  004652  012737  000240  177776          MOV     #5*40,@#PS              ;SET TO PRIORITY LEVEL 5
1164  004660  006302                  2$:     ASL     R2                      ;KILL SOME TIME
1165  004662  001376                          BNE     2$
1166  004664  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
1167  004672  005701                          TST     R1                      ;DID AN INTERRUPT OCCUR?
1168  004674  001005                          BNE     3$                      ;BR IF YES
1169  004676  022737  000240  001222          CMP     #5*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1170  004704  002012                          BGE     6$                      ;BR IF NO
1171  004706  104006                          ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 5
1172
1173  004710  022737  000240  001222  3$:     CMP     #5*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1174  004716  002405                          BLT     6$                      ;BR IF YES
1175  004720  104006                          ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 5
1176
1177  004722  005201                  4$:     INC     R1                      ;SET INTERRUPT KEY
1178  004724  000002                          RTI                             ;RETURN AFTER INTERRUPT
1179  004726  022626                  5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1180  004730  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1181
1182  004732  005014                  6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1183  004734  013777  001220  174254          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1184  004742  005077  174252                  CLR     @TAVEC+2
1185  004746  005037  000000                  CLR     @#0
1186  004752  005037  000002                  CLR     @#2
1187                                  ;;*************************************************************
1188                                          ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1189                                          ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1190                                          ;IT WILL THEN SET THE PRIORITY LEVEL TO 6 AND WASTE ENOUGH TIME
1191                                          ;FOR AN INTERRUPT TO OCCUR.
1192                                          ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1193                                          ;AND CHECK IF AN INTERRUPT OCCURRED.
1194                                  ;;*************************************************************
1195                                  ;*TEST 13        TEST INTERRUPT WITH READY = "1" AT LEVEL 6
1196                                  ;;*************************************************************
1197  004756  000004                  TST13:  SCOPE
1198  004760  012767  004776  174120          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1199  004766  012767  005134  174174          MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
```

```
1200  004774  000005                        RESET
1201  004776  010314                1$:     MOV     DRIVE,@TACS
1202  005000  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1203  005006  012700  000006                MOV     #6,R0                   ;TEST AT PRIORITY LEVEL 6
1204  005012  005001                        CLR     R1                      ;CLEAR INTERRUPT KEY
1205  005014  012702  000001                MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1206  005020  012777  005124  174170        MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1207  005026  012777  000340  174164        MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
1208  005034  012737  000137  000000        MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1209  005042  012737  005130  000002        MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
1210  005050  112714  000100                MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1211  005054  012737  000300  177776        MOV     #6*40,@#PS              ;SET TO PRIORITY LEVEL 6
1212  005062  006302                2$:     ASL     R2                      ;KILL SOME TIME
1213  005064  001376                        BNE     2$
1214  005066  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT THE WORLD
1215  005074  005701                        TST     R1                      ;DID AN INTERRUPT OCCUR?
1216  005076  001005                        BNE     3$                      ;BR IF YES
1217  005100  022737  000300  001222        CMP     #6*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1218  005106  002012                        BGE     6$                      ;BR IF NO
1219  005110  104006                        ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 6
1220
1221  005112  022737  000300  001222  3$:     CMP     #6*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1222  005120  002405                        BLT     6$                      ;BR IF YES
1223  005122  104006                        ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 6
1224
1225  005124  005201                4$:     INC     R1                      ;SET INTERRUPT KEY
1226  005126  000002                        RTI                             ;RETURN AFTER INTERRUPT
1227  005130  022626                5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1228  005132  104006                        ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1229
1230  005134  005014                6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1231  005136  013777  001220  174052        MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1232  005144  005077  174050                CLR     @TAVEC+2
1233  005150  005037  000000                CLR     @#0
1234  005154  005037  000002                CLR     @#2
1235                                ;;**************************************************************
1236                                ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1237                                ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1238                                ;IT WILL THEN SET THE PRIORITY LEVEL TO 7 AND WASTE ENOUGH TIME
1239                                ;FOR AN INTERRUPT TO OCCUR.
1240                                ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1241                                ;AND CHECK IF AN INTERRUPT OCCURRED.
1242                                ;;**************************************************************
1243                                ;*TEST 14        TEST INTERRUPT WITH READY = "1" AT LEVEL 7
1244                                ;;**************************************************************
1245  005160  000004                TST14:  SCOPE
1246  005162  012767  005200  173716        MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1247  005170  012767  005336  173772        MOV     #6$,$ESCAPE     ;;ESCAPE TO 6$ ON ERROR
1248  005176  000005                        RESET
1249  005200  010314                1$:     MOV     DRIVE,@TACS
1250  005202  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1251  005210  012700  000007                MOV     #7,R0                   ;TEST AT PRIORITY LEVEL 7
1252  005214  005001                        CLR     R1                      ;CLEAR INTERRUPT KEY
1253  005216  012702  000001                MOV     #1,R2                   ;SETUP TO WASTE A LITTLE TIME
1254  005222  012777  005326  173766        MOV     #4$,@TAVEC              ;INTERRUPT RETURN
1255  005230  012777  000340  173762        MOV     #340,@TAVEC+2           ;LOCK OUT THE WORLD IF INTERRUPT
```



```
1256  005236  012737  000137  000000        MOV     #137,@#0                ;SETUP TO CATCH INTERRUPT IF
1257  005244  012737  005332  000002        MOV     #5$,@#2                 ;   IT GOES TO LOCATION 0
1258  005252  112714  000100                MOVB    #INT.EN,@TACS           ;SET INTERRUPT ENABLE
1259  005256  012737  000340  177776        MOV     #7*40,@#PS              ;SET TO PRIORITY LEVEL 7
1260  005264  006302                2$:     ASL     R2                      ;KILL SOME TIME
1261  005266  001376                        BNE     2$
1262  005270  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT THE WORLD
1263  005276  005701                        TST     R1                      ;DID AN INTERRUPT OCCUR?
1264  005300  001005                        BNE     3$                      ;BR IF YES
1265  005302  022737  000340  001222        CMP     #7*40,@#TAPRIO          ;WAS AN INTERRUPT EXPECTED?
1266  005310  002012                        BGE     6$                      ;BR IF NO
1267  005312  104006                        ERROR   6                       ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 7
1268
1269  005314  022737  000340  001222  3$:     CMP     #7*40,@#TAPRIO          ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1270  005322  002405                        BLT     6$                      ;BR IF YES
1271  005324  104006                        ERROR   6                       ;INTERRUPT OCCURRED AT PRIORITY LEVEL 7
1272
1273  005326  005201                4$:     INC     R1                      ;SET INTERRUPT KEY
1274  005330  000002                        RTI                             ;RETURN AFTER INTERRUPT
1275  005332  022626                5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1276  005334  104006                        ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1277
1278  005336  005014                6$:     CLR     @TACS                   ;CLEAR INTERRUPT ENABLE
1279  005340  013777  001220  173650        MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1280  005346  005077  173646                CLR     @TAVEC+2
1281  005352  005037  000000                CLR     @#0
1282  005356  005037  000002                CLR     @#2
1283                                ;;**************************************************************
1284                                ;*TEST 15        TEST INTERRUPT WITH "TRANSFER REQUEST" = 1
1285                                ;;**************************************************************
1286  005362  000004                TST15:  SCOPE
1287  005364  012767  000010  173574        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
1288  005372  012767  005420  173506        MOV     #4$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1289  005400  012767  005530  173562        MOV     #3$,$ESCAPE     ;;ESCAPE TO 3$ ON ERROR
1290  005406  000005                        RESET
1291  005410  010314                        MOV     DRIVE,@TACS             ;SELECT DRIVE
1292  005412  112714  000017                MOVB    #REWIND+GO,@TACS        ;POSITION THE TAPE
1293  005416  104412                        WAITREADY                       ;GO WAIT FOR "READY" TO SET
1294  005420  010314                4$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
1295  005422  112714  000001                MOVB    #WFG+GO,@TACS           ;START A "WRITE FILE GAP"
1296  005426  104412                        WAITREADY                       ;GO WAIT FOR "READY" TO SET
1297  005430  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1298  005436  012701  000001                MOV     #1,R1                   ;SETUP TO WASTE SOME TIME
1299  005442  012777  005526  173546        MOV     #2$,@TAVEC              ;SETUP INTERRUPT RETURN
1300  005450  012777  000340  173542        MOV     #340,@TAVEC+2           ;PRIORITY "7" ON INTERRUPT
1301  005456  012737  000137  000000        MOV     #137,@#0                ;CATCH INTERRUPT IF IT GOES
1302  005464  012737  005522  000002        MOV     #5$,@#2                 ;   TO LOCATION 0
1303  005472  112714  000103                MOVB    #WRITE!INT.EN!GO,@TACS  ;START A "WRITE" WITH "INT. EN."
1304  005476  104413                        WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
1305  005500  005037  177776                CLR     @#PS                    ;LEVEL 0
1306  005504  005000                        CLR     R0
1307  005506  006301                1$:     ASL     R1                      ;KILL A LITTLE TIME
1308  005510  001376                        BNE     1$
1309  005512  012737  000340  177776        MOV     #340,@#PS               ;LOCK OUT ALL INTERRUPTS
1310  005520  104006                        ERROR   6                       ;INTERRUPT FOR XFER REQ. FAILED
1311  005522  022626                5$:     CMP     (SP)+,(SP)+             ;POP THE STACK
```

```
1312  005524  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1313  005526  022626                  2$:     CMP     (SP)+,(SP)+             ;POP STACK
1314  005530  000005                  3$:     RESET                           ;CLEAR ALL
1315  005532  013777  001220  173456          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1316  005540  005077  173454                  CLR     @TAVEC+2
1317  005544  005037  000000                  CLR     @#0
1318  005550  005037  000002                  CLR     @#2
1319                                  ;;*************************************************************
1320                                  ;*TEST 16        TEST INTERRUPT WITH "READY" = 0 AND "XFER REQ" = 0
1321                                  ;;*************************************************************
1322  005554  000004                  TST16:  SCOPE
1323  005556  012767  000001  173402          MOV     #1,$TIMES       ;;DO 1 ITERATION
1324  005564  012767  005724  173376          MOV     #5$,$ESCAPE     ;;ESCAPE TO 5$ ON ERROR
1325  005572  012767  005724  000064          MOV     #5$,3$
1326  005600  012701  000001                  MOV     #1,R1
1327  005604  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT THE WORLD
1328  005612  000005                          RESET                           ;CLEAR THE WORLD
1329  005614  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1330  005616  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1331  005620  012777  005714  173370          MOV     #4$,@TAVEC              ;RETURN IF INTERRUPT OCCURS
1332  005626  012777  000340  173364          MOV     #340,@TAVEC+2           ;LEVEL "7" IF INTERRUPT
1333  005634  012737  000137  000000          MOV     #137,@#0                ;CATCH INTERRUPT IF IT GOES TO
1334  005642  012737  005720  000002          MOV     #7$,@#2                 ;  LOCATION 0
1335  005650  112714  000117                  MOVB    #REWIND!INT.EN!GO,@TACS
1336  005654  032714  000240          1$:     BIT     #TR.REQ!READY,@TACS     ;GIVE READY TIME TO CLEAR
1337  005660  001404                          BEQ     2$
1338  005662  005327                          DEC     (PC)+
1339  005664  000000                  3$:     0
1340  005666  001372                          BNE     1$
1341  005670  104001                          ERROR   1                       ;READY OR XFER REQ = "1"
1342  005672  005037  177776          2$:     CLR     @#PS                    ;ALLOW INTERRUPT
1343  005676  005000                          CLR     R0                      ;AT LEVEL 0
1344  005700  006301                  6$:     ASL     R1                      ;GIVE INTERRUPT TIME TO COME IN
1345  005702  001376                          BNE     6$
1346  005704  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT INTERRUPT
1347  005712  000404                          BR      5$                      ;EXIT WITH NO ERRORS
1348  005714  022626                  4$:     CMP     (SP)+,(SP)+             ;POP STACK
1349  005716  104006                          ERROR   6                       ;INTERRUPT OCCURRED
1350  005720  022626                  7$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1351  005722  104006                          ERROR   6                       ;INTERRUPTTED TO LOCATION 0
1352  005724  000005                  5$:     RESET                           ;CLEAR THE WORLD
1353  005726  013777  001220  173262          MOV     @#TAVEC+2,@TAVEC        ;SET TRAP CATCHER
1354  005734  005077  173260                  CLR     @TAVEC+2
1355  005740  005037  000000                  CLR     @#0
1356  005744  005037  000002                  CLR     @#2
1357                                  ;;*************************************************************
1358                                  ;*TEST 17        TEST INTERRUPT WITH "XFER REQ" = 1 & "TIMING ERROR" = 1
1359                                  ;;*************************************************************
1360  005750  000004                  TST17:  SCOPE
1361  005752  012767  000012  173206          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1362  005760  012767  006014  173120          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1363  005766  012767  006222  173174          MOV     #11$,$ESCAPE    ;;ESCAPE TO 11$ ON ERROR
1364  005774  000005                          RESET                           ;CLEAR ALL
1365  005776  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1366  006000  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO "BOT"
1367  006004  104412                          WAITREADY                       ;WAIT ON READY
```



```
1368  006006  112714  000001                  MOVB    #WFG!GO,@TACS           ;GET ON OXIDE
1369  006012  104412                          WAITREADY
1370  006014  010314                  1$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
1371  006016  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE
1372  006022  104413                          WAITXFER                        ;WAIT ON "TRANSFER REQUEST"
1373  006024  112715  000377                  MOVB    #377,@TADB              ;WRITE A BYTE
1374  006030  005000                          CLR     R0                      ;ZERO THE COUNTER
1375  006032  005001                          CLR     R1
1376  006034  105714                  2$:     TSTB    @TACS                   ;CHECK "XFER REQ"
1377  006036  100405                          BMI     3$                      ;BR IF SET
1378  006040  062700  000010                  ADD     #10,R0                  ;FIND OUT HOW LONG IT TAKES
1379  006044  005501                          ADC     R1                      ;FOR "XFER REQ" TO SET
1380  006046  100372                          BPL     2$
1381  006050  104001                          ERROR   1                       ;"XFER REQ" DIDN'T SET
1382  006052  105714                  3$:     TSTB    @TACS                   ;CHECK "XFER REQ" IF IT IS = 0
1383  006054  100005                          BPL     4$                      ;SOMETHING IS WRONG
1384  006056  162700  000001                  SUB     #1,R0                   ;DOWN COUNT THE COUNTER
1385  006062  005601                          SBC     R1                      ;SO A "TIMING ERROR" WILL OCCUR
1386  006064  100372                          BPL     3$
1387  006066  000401                          BR      5$
1388
1389  006070  104001                  4$:     ERROR   1                       ;"XFER REQ" CLEARED
1390
1391  006072  012777  006160  173116  5$:     MOV     #8$,@TAVEC              ;SETUP  INTERRRUPT RETURN
1392  006100  012777  000340  173112          MOV     #340,@TAVEC+2           ;PRIORITY "7" ON INTERRUPT
1393  006106  012737  000137  000000          MOV     #137,@#0                ;CATCH INTERRUPT IF IT GOES
1394  006114  012737  006154  000002          MOV     #7$,@#2                 ;       TO LOCATION 0
1395  006122  012701  000001                  MOV     #1,R1                   ;SETUP TO WASTE SOME TIME
1396  006126  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
1397  006132  005000                          CLR     R0                      ;AT LEVEL "0"
1398  006134  052714  000100                  BIS     #INT.EN,@TACS           ;TURN ON "INTERRUPT ENABLE"
1399  006140  006301                  6$:     ASL     R1                      ;GIVE INTERRUPT TIME TO OCCUR
1400  006142  001376                          BNE     6$
1401  006144  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT INTERRUPTS
1402  006152  104006                          ERROR   6                       ;FAILED TO INTERRUPT WITH "XFER REQ" = 1
1403
1404  006154  022626                  7$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1405  006156  104006                          ERROR   6                       ;INTERRUPT WENT TO LOCATION 0
1406
1407  006160  022626                  8$:     CMP     (SP)+,(SP)+             ;POP THE STACK
1408  006162  105715                          TSTB    @TADB                   ;KNOCK DOWN "XFER REQ"
1409  006164  104412                          WAITREADY                       ;WAIT ON "READY" CAUSED BY "TIMING ERROR"
1410  006166  012777  006220  173022          MOV     #10$,@TAVEC             ;SETUP INTERRUPT RETURN
1411  006174  012701  000001                  MOV     #1,R1                   ;SETUP TO WASTE TIME
1412  006200  005037  177776                  CLR     @#PS                    ;ALLOW INTERRUPTS
1413  006204  006301                  9$:     ASL     R1                      ;GIVE INTERRUPTS TIME TO OCCUR
1414  006206  001376                          BNE     9$
1415  006210  012737  000340  177776          MOV     #340,@#PS               ;LOCK OUT INTERRUPTS
1416  006216  104006                          ERROR   6                       ;"INTERRUPT FAILED TO OCCUR FOR READY
1417
1418  006220  022626                  10$:    CMP     (SP)+,(SP)+             ;POP THE STACK
1419  006222  000005                  11$:    RESET                           ;CLEAR ALL
1420  006224  012777  001220  172764          MOV     #TAVEC+2,@TAVEC         ;RESTORE THE TRAP CATCHER
1421  006232  005077  172762                  CLR     @TAVEC+2
1422  006236  005037  000000                  CLR     @#0
1423  006242  005037  000002                  CLR     @#2
```

```
                                        ;////////////////////////////////////////////////////////////////
                                        ;////////////////////////////////////////////////////////////////
                                        ;THE FOLLOWING TEST ARE USED TO CHECK THE BASIC OPERATION OF THE SPACING FUNCTIONS
                                        ;////////////////////////////////////////////////////////////////
                                        ;;************************************************************
                                        ;*TEST 20        TEST "BACK SPACE FILE GAP"
                                        ;;************************************************************
006246  000004                  TST20:  SCOPE
006250  012767  000012  172710          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
006256  012767  006274  172622          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006264  012767  006370  172676          MOV     #TST21,$ESCAPE  ;;ESCAPE TO TEST 21 ON ERROR
006272  000005                          RESET
006274  010314                  3$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
006276  112714  000017                  MOVB    #REWIND!GO,@TACS        ;POSITION TAPE ON CLEAR LEADER
006302  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006304  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP OFF OF CLEAR LEADER
006310  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006312  112714  000003                  MOVB    #WRITE!GO,@TACS         ;PUT A DATA BLOCK ON THE TAPE
006316  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006320  112715  000377                  MOVB    #377,@TADB              ;KNOCK DOWN XFER REQ
006324  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006326  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
006332  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006334  112714  000007                  MOVB    #BSFG!GO,@TACS          ;BACK UP OVER THE DATA
006340  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006342  032714  004000                  BIT     #FGAP,@TACS             ;CHECK FOR FILE GAP
006346  001001                          BNE     1$                      ;BR IF "FILE GAP" =1
006350  104001                          ERROR   1                       ;ERROR=NO FILE GAP INDICATION
006352  005714                  1$:     TST     @TACS                   ;IS "ERROR" =1?
006354  100001                          BPL     2$                      ;BR IF NO
006356  104001                          ERROR   1                       ;"ERROR" =1
006360  032714  020000          2$:     BIT     #LEADER,@TACS           ;IS "CLEAR LEADER" =1?
006364  001401                          BEQ     TST21           ;;BR IF NO
006366  104001                          ERROR   1                       ;"WFG" OR "BSFG" FAILED
                                        ;;************************************************************
                                        ;*TEST 21        TEST "SPACE FORWARD FILE GAP" FUNCTION
                                        ;;************************************************************
006370  000004                  TST21:  SCOPE
006372  012767  000005  172566          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006400  012767  006464  172500          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006406  012767  006510  172554          MOV     #TST22,$ESCAPE  ;;ESCAPE TO TEST 22 ON ERROR
006414  000005                          RESET
006416  010314                          MOV     DRIVE,@TACS             ;SELECT A DRIVE
006420  112714  000017                  MOVB    #REWIND!GO,@TACS        ;POSITION THE TAPE
006424  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006426  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE FILE GAP OFF OF CLEAR LEADER
006432  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006434  112714  000003                  MOVB    #WRITE!GO,@TACS         ;PUT SOME DATA ON THE TAPE
006440  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006442  112715  000377                  MOVB    #377,@TADB              ;KNOCK DOWN XFER REQ
006446  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006450  052714  000020                  BIS     #ILBS,@TACS             ;WRITE CRC AND SHUT DOWN
006454  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006456  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE FILE GAP AFTER THE RECORD
006462  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006464  112714  000017          1$:     MOVB    #REWIND!GO,@TACS        ;REPOSITION THE TAPE
```



```
                                        WAITREADY                       ;GO WAIT FOR "READY" TO SET
006470  104412
006472  112714  000013                  MOVB    #SFFG!GO,@TACS          ;SPACE OVER THE RECORD
006476  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006500  032714  004000                  BIT     #FGAP,@TACS             ;SETTING IN A FILE GAP?
006504  001001                          BNE     TST22           ;;BR IF YES
006506  104001                          ERROR   1                       ;NO FILE GAP INDICATION
                                        ;;************************************************************
                                        ;*TEST 22        TEST "BACK SPACE BLOCK GAP"
                                        ;;************************************************************
006510  000004                  TST22:  SCOPE
006512  012767  000005  172446          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006520  012767  006536  172360          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006526  012767  006622  172434          MOV     #TST23,$ESCAPE  ;;ESCAPE TO TEST 23 ON ERROR
006534  000005                          RESET
006536  010314                  1$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
006540  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO CLEAR LEADER
006544  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006546  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
006552  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006554  112714  000003                  MOVB    #WRITE!GO,@TACS         ;WRITE A BLOCK OF DATA
006560  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006562  112715  000377                  MOVB    #377,@TADB              ;WRITE ONE BYTE
006566  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006570  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
006574  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006576  112714  000011                  MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA
006602  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006604  005714                          TST     @TACS                   ;CHECK FOR ERRORS
006606  100001                          BPL     2$                      ;BR IF NONE
006610  104001                          ERROR   1                       ;ERROR OCCURRED
006612  032714  020000          2$:     BIT     #LEADER,@TACS           ;IS CLEAR LEADER=1?
006616  001401                          BEQ     TST23           ;;BR IF NO
006620  104001                          ERROR   1                       ;ON CLEAR LEADER
                                        ;;************************************************************
                                        ;*TEST 23        TEST "SPACE FWD BLOCK GAP"
                                        ;;************************************************************
006622  000004                  TST23:  SCOPE
006624  012767  000005  172334          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
006632  012767  006716  172246          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
006640  012767  006742  172322          MOV     #TST24,$ESCAPE  ;;ESCAPE TO TEST 24 ON ERROR
006646  000005                          RESET
006650  010314                          MOV     DRIVE,@TACS
006652  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO CLEAR LEADER
006656  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006660  112714  000001                  MOVB    #WFG!GO,@TACS           ;PUT FILE GAP ON TAPE
006664  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006666  112714  000003                  MOVB    #WRITE!GO,@TACS         ;WRITE ONE BYTE BLOCK
006672  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006674  112715  000377                  MOVB    #377,@TADB
006700  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
006702  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
006706  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006710  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP AFTER THE DATA
006714  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
006716  112714  000017          1$:     MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
006722  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
```

```
1536  006724  112714  000015                  MOVB    #SFBG!GO,@TACS          ;SEE IF SFBG WILL SPACE OVER THE BLOCK OF DATA
1537  006730  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1538  006732  032714  124000                  BIT     #ERROR!LEADER!FGAP,@TACS        ;"ERROR" SHOULD BE CLEAR
1539  006736  001401                          BEQ     TST24           ;;SO SHOULD "CLEAR LEADER" AND "FILE GAP"
1540  006740  104001                          ERROR   1                       ;TAPE IS IN THE WRONG PLACE AFTER A SFBG
1541                                  ;////////////////////////////////////////////////////////////////////
1542                                  ;////////////////////////////////////////////////////////////////////
1543                                  ;THE FOLLOWING TESTS ARE USED TO CHECK THE "FILE GAP" CIRCUITS
1544                                  ;////////////////////////////////////////////////////////////////////
1545                                  ;;************************************************************
1546                                  ;*TEST 24        TEST AUTOMATIC "WFG" WHEN WRITING OFF OF "CLEAR LEADER"
1547                                  ;;************************************************************
1548  006742  000004                  TST24:  SCOPE
1549  006744  012767  000012  172214          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1550  006752  012767  006770  172126          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1551  006760  012767  007050  172202          MOV     #TST25,$ESCAPE  ;;ESCAPE TO TEST 25 ON ERROR
1552  006766  000005                          RESET
1553  006770  010314                  1$:     MOV     DRIVE,@TACS             ;SELECT DRIVE
1554  006772  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO CLEAR LEADER
1555  006776  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1556  007000  112714  000003                  MOVB    #WRITE!GO,@TACS         ;WRITE "AUTO, FILE GAP"
1557  007004  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
1558  007006  012715  000377                  MOV     #377,@TADB              ;WRITE "DATA"
1559  007012  104413                          WAITXFER                        ;GO WAIT ON "TRANSFER REQUEST" TO SET
1560  007014  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
1561  007020  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1562  007022  112714  000007                  MOVB    #BSFG!GO,@TACS          ;BACK OVER THE DATA BLOCK
1563  007026  104412                          WAITREADY                       ;GO WAIT FOR "READY" TO SET
1564  007030  032714  004000                  BIT     #FGAP,@TACS             ;IN FILE GAP?
1565  007034  001001                          BNE     2$                      ;BR IF YES
1566  007036  104001                          ERROR   1                       ;NO FILE GAP INDICATION
1567  007040  032714  120000          2$:     BIT     #ERROR!LEADER,@TACS     ;ERROR SHOULD BE CLEAR
1568  007044  001401                          BEQ     TST25           ;;SO SHOULD "CLEAR LEADER"
1569  007046  104001                          ERROR   1                       ;ERROR OR CLEAR LEADER ON A (1)
1570                                  ;;************************************************************
1571                                  ;*TEST 25        TEST "READ" INTO FILE GAP CAUSES AN ERROR
1572                                  ;;************************************************************
1573  007050  000004                  TST25:  SCOPE
1574  007052  012767  000012  172106          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1575  007060  012767  007106  172020          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1576  007066  012767  007176  172074          MOV     #TST26,$ESCAPE  ;;ESCAPE TO TEST 26 ON ERROR
1577  007074  000005                          RESET
1578  007076  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1579  007100  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
1580  007104  104412                          WAITREADY                       ;WAIT ON "READY"
1581  007106  112714  000003          1$:     MOVB    #WRITE!GO,@TACS         ;START A WRITE
1582  007112  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1583  007114  112715  000377                  MOVB    #377,@TADB              ;WRITE ONE BYTE
1584  007120  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1585  007122  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
1586  007126  104412                          WAITREADY                       ;WAIT ON "READY"
1587  007130  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A "FILE GAP"
1588  007134  104412                          WAITREADY                       ;WAIT ON "READY"
1589  007136  112714  000011                  MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA
1590  007142  104412                          WAITREADY                       ;WAIT ON "READY"
1591  007144  112714  000015                  MOVB    #SFBG!GO,@TACS          ;SPACE OVER THE DATA BLOCK
```



```
1592  007150  104412                          WAITREADY                       ;WAIT ON "READY"
1593  007152  112714  000005                  MOVB    #READ!GO,@TACS          ;START A "READ"
1594  007156  104412                          WAITREADY                       ;WAIT ON "READY"
1595  007160  005714                          TST     @TACS                   ;DID AN "ERROR" OCCUR?
1596  007162  100401                          BMI     2$                      ;BR IF YES
1597  007164  104001                          ERROR   1                       ;ERROR FAILED TO SET
1598  007166  032714  004000          2$:     BIT     #FGAP,@TACS             ;IS FILE GAP=1?
1599  007172  001001                          BNE     TST26           ;;BR IF YES
1600  007174  104001                          ERROR   1                       ;FILE GAP NOT=1
1601                                  ;;************************************************************
1602                                  ;*TEST 26        TEST "SFBG" INTO FILE GAP CAUSES AN ERROR
1603                                  ;;************************************************************
1604  007176  000004                  TST26:  SCOPE
1605  007200  012767  000012  171760          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1606  007206  012767  007234  171672          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1607  007214  012767  007324  171746          MOV     #TST27,$ESCAPE  ;;ESCAPE TO TEST 27 ON ERROR
1608  007222  000005                          RESET
1609  007224  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1610  007226  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
1611  007232  104412                          WAITREADY                       ;WAIT ON "READY"
1612  007234  112714  000003          1$:     MOVB    #WRITE!GO,@TACS         ;START A WRITE
1613  007240  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1614  007242  112715  000377                  MOVB    #377,@TADB              ;WRITE ONE BYTE
1615  007246  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1616  007250  052714  000020                  BIS     #ILBS,@TACS             ;WRITE"CRC"
1617  007254  104412                          WAITREADY                       ;WAIT ON "READY"
1618  007256  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A "FILE GAP"
1619  007262  104412                          WAITREADY                       ;WAIT ON "READY"
1620  007264  112714  000011                  MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA
1621  007270  104412                          WAITREADY                       ;WAIT ON "READY"
1622  007272  112714  000015                  MOVB    #SFBG!GO,@TACS          ;SPACE OVER THE DATA BLOCK
1623  007276  104412                          WAITREADY                       ;WAIT ON "READY"
1624  007300  112714  000015                  MOVB    #SFBG!GO,@TACS          ;START A "SFBG"
1625  007304  104412                          WAITREADY                       ;WAIT ON "READY"
1626  007306  005714                          TST     @TACS                   ;DID AN "ERROR" OCCUR?
1627  007310  100401                          BMI     2$                      ;BR IF YES
1628  007312  104001                          ERROR   1                       ;ERROR FAILED TO SET
1629  007314  032714  004000          2$:     BIT     #FGAP,@TACS             ;IS FILE GAP=1?
1630  007320  001001                          BNE     TST27           ;;BR IF YES
1631  007322  104001                          ERROR   1                       ;FILE GAP NOT=1
1632                                  ;;************************************************************
1633                                  ;*TEST 27        TEST "ERROR" OUTPUT OF THE STATUS ROM WITH "FILE GAP=1"
1634                                  ;;************************************************************
1635  007324  000004                  TST27:  SCOPE
1636  007326  012767  000012  171632          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1637  007334  012767  007430  171544          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1638  007342  012767  007574  171620          MOV     #TST30,$ESCAPE  ;;ESCAPE TO TEST 30 ON ERROR
1639  007350  000005                          RESET
1640  007352  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1641  007354  104412                          WAITREADY                       ;MAKE SURE READY IS SET
1642  007356  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
1643  007362  104412                          WAITREADY                       ;WAIT ON "READY"
1644  007364  112714  000001                  MOVB    #WFG!GO,@TACS           ;GET OFF OF CLEAR LEADER
1645  007370  104412                          WAITREADY                       ;WAIT FOR READY
1646  007372  112714  000003                  MOVB    #WRITE!GO,@TACS         ;START A WRITE
1647  007376  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
```

```
1648  007400  112715  000377                  MOVB    #377,@TADB              ;WRITE ONE BYTE
1649  007404  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1650  007406  112715  000377                  MOVB    #377,@TADB              ;WRITE
1651  007412  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1652  007414  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
1653  007420  104412                          WAITREADY                       ;WAIT ON "READY"
1654  007422  112714  000001                  MOVB    #WFG!GO,@TACS           ;WRITE A "FILE GAP"
1655  007426  104412                          WAITREADY                       ;WAIT ON "READY"
1656  007430  112714  000017          1$:     MOVB    #REWIND!GO,@TACS        ;BACK OVER THE DATA
1657  007434  104412                          WAITREADY                       ;WAIT ON "READY"
1658  007436  112714  000013                  MOVB    #SFFG!GO,@TACS          ;DO A SPACE FWD FILE GAP
1659  007442  104412                          WAITREADY
1660  007444  032714  004000                  BIT     #FGAP,@TACS             ;IS FILE GAP=1?
1661  007450  001001                          BNE     2$                      ;BR IF YES
1662  007452  104001                          ERROR   1                       ;FILE GAP NOT=1
1663                                  ;NOW CHECK "ERROR" BIT FOR ALL FUNCTIONS
1664  007454  112714  000000          2$:     MOVB    #WFG,@TACS              ;CHECK "ERROR" WITH "WFG"
1665  007460  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1666  007462  100001                          BPL     3$                      ;BR IF "ERROR" = 0
1667  007464  104001                          ERROR   1                       ;"ERROR" NOT = 0
1668  007466  112714  000002          3$:     MOVB    #WRITE,@TACS            ;CHECK "ERROR" WITH "WRITE"
1669  007472  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1670  007474  100001                          BPL     4$                      ;BR IF "ERROR" = 0
1671  007476  104001                          ERROR   1                       ;"ERROR" NOT = 0
1672  007500  112714  000004          4$:     MOVB    #READ,@TACS             ;CHECK "ERROR" WITH "READ"
1673  007504  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1674  007506  100401                          BMI     5$                      ;BR IF "ERROR" = 1
1675  007510  104001                          ERROR   1                       ;"ERROR" NOT = 1
1676  007512  112714  000006          5$:     MOVB    #BSFG,@TACS             ;CHECK "ERROR" WITH "BSFG"
1677  007516  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1678  007520  100001                          BPL     6$                      ;BR IF "ERROR" = 0
1679  007522  104001                          ERROR   1                       ;"ERROR" NOT = 0
1680  007524  112714  000010          6$:     MOVB    #BSBG,@TACS             ;CHECK "ERROR" WITH "BSBG"
1681  007530  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1682  007532  100001                          BPL     7$                      ;BR IF "ERROR" = 0
1683  007534  104001                          ERROR   1                       ;"ERROR" NOT = 0
1684  007536  112714  000012          7$:     MOVB    #SFFG,@TACS             ;CHECK "ERROR" WITH "SFFG"
1685  007542  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1686  007544  100001                          BPL     8$                      ;BR IF "ERROR" = 0
1687  007546  104001                          ERROR   1                       ;"ERROR" NOT = 0
1688  007550  112714  000014          8$:     MOVB    #SFBG,@TACS             ;CHECK "ERROR" WITH "SFBG"
1689  007554  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1690  007556  100401                          BMI     9$                      ;BR IF "ERROR" = 1
1691  007560  104001                          ERROR   1                       ;"ERROR" NOT = 1
1692  007562  112714  000016          9$:     MOVB    #REWIND,@TACS           ;CHECK "ERROR" WITH "REWIND"
1693  007566  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1694  007570  100001                          BPL     TST30           ;;BR IF "ERROR" = 0
1695  007572  104001                          ERROR   1                       ;"ERROR" NOT = 0
```



```
1696                                    ;////////////////////////////////////////////////////////////////////
1697                                    ;////////////////////////////////////////////////////////////////////
1698                                    ;THE FOLLOWING TESTS INSURE THAT BACKING INTO "CLEAR LEADER" CAUSES AN ERROR
1699                                    ;////////////////////////////////////////////////////////////////////
1700                                    ;;****************************************************************
1701                                    ;*TEST 30        TEST BACK-SPACE-FILE-GAP INTO CLEAR LEADER
1702                                    ;;****************************************************************
1703  007574  000004                    TST30:  SCOPE
1704  007576  012767  000012  171362            MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1705  007604  012767  007700  171274            MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1706  007612  012767  010000  171350            MOV     #TST31,$ESCAPE  ;;ESCAPE TO TEST 31 ON ERROR
1707  007620  000005                            RESET
1708  007622  010314                            MOV     DRIVE,@TACS             ;SELECT DRIVE
1709  007624  104412                            WAITREADY                       ;MAKE SURE READY IS SET
1710  007626  112714  000017                    MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
1711  007632  104412                            WAITREADY                       ;WAIT ON "READY"
1712  007634  112714  000001                    MOVB    #WFG!GO,@TACS           ;GET OFF OF CLEAR LEADER
1713  007640  104412                            WAITREADY                       ;WAIT FOR READY
1714  007642  112714  000003                    MOVB    #WRITE!GO,@TACS         ;START A WRITE
1715  007646  104413                            WAITXFER                        ;WAIT ON "XFER REQ"
1716  007650  112715  000377                    MOVB    #377,@TADB              ;WRITE ONE BYTE
1717  007654  104413                            WAITXFER                        ;WAIT ON "XFER REQ"
1718  007656  112715  000377                    MOVB    #377,@TADB              ;WRITE
1719  007662  104413                            WAITXFER                        ;WAIT ON "XFER REQ"
1720  007664  052714  000020                    BIS     #ILBS,@TACS             ;WRITE "CRC"
1721  007670  104412                            WAITREADY                       ;WAIT ON "READY"
1722  007672  112714  000001                    MOVB    #WFG!GO,@TACS           ;WRITE A "FILE GAP"
1723  007676  104412                            WAITREADY                       ;WAIT ON "READY"
1724  007700  112714  000017          1$:       MOVB    #REWIND!GO,@TACS        ;BACK OVER THE DATA
1725  007704  104412                            WAITREADY                       ;WAIT ON "READY"
1726  007706  112714  000013                    MOVB    #SFFG!GO,@TACS          ;DO A SPACE FWD FILE GAP
1727  007712  104412                            WAITREADY
1728  007714  032714  004000                    BIT     #FGAP,@TACS             ;IS FILE GAP=1?
1729  007720  001001                            BNE     2$                      ;BR IF YES
1730  007722  104001                            ERROR   1                       ;FILE GAP NOT=1
1731  007724  112714  000007          2$:       MOVB    #BSFG!GO,@TACS          ;GO TO FIRST FILE GAP ON TAPE
1732  007730  104412                            WAITREADY                       ;GO WAIT FOR "READY" TO SET
1733  007732  032714  004000                    BIT     #FGAP,@TACS             ;IN A FILE GAP?
1734  007736  001001                            BNE     3$                      ;BR IF YES
1735  007740  104001                            ERROR   1                       ;DIDN'T STOP IN A GAP
1736  007742  032714  100000          3$:       BIT     #ERROR,@TACS            ;DID WE GET AN ERROR?
1737  007746  001401                            BEQ     4$                      ;BR IF NO
1738  007750  104001                            ERROR   1                       ;AN ERROR OCCURRED
1739  007752  112714  000007          4$:       MOVB    #BSFG!GO,@TACS          ;BACK ONTO THE CLEAR LEADER
1740  007756  104412                            WAITREADY                       ;GO WAIT FOR "READY" TO SET
1741  007760  032714  100000                    BIT     #ERROR,@TACS            ;CHECK FOR AN ERROR
1742  007764  001001                            BNE     5$                      ;BR IF "ERROR" BIT IS SET
1743  007766  104001                            ERROR   1                       ;"ERROR" BIT FAILED TO SET
1744  007770  032714  020000          5$:       BIT     #LEADER,@TACS           ;CHECK FOR CLEAR LEADER
1745  007774  001001                            BNE     TST31           ;;BR IF "CLEAR LEADER" = 1
1746  007776  104001                            ERROR   1                       ;"ERROR" WASN'T DUE TO "CLEAR LEADER"
```

```
                                ;;*************************************************************
                                ;*TEST 31        TEST BACK-SPACE-BLOCK-GAP INTO CLEAR LEADER
                                ;;*************************************************************
010000  000004                  TST31:  SCOPE
010002  012767  000012  171156          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
010010  012767  010104  171070          MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
010016  012767  010172  171144          MOV     #TST32,$ESCAPE  ;;ESCAPE TO TEST 32 ON ERROR
010024  000005                        RESET
010026  010314                        MOV     DRIVE,@TACS             ;SELECT DRIVE
010030  104412                        WAITREADY                       ;MAKE SURE READY IS SET
010032  112714  000017                MOVB    #REWIND!GO,@TACS        ;GO TO BEGINNING OF TAPE
010036  104412                        WAITREADY                       ;WAIT ON "READY"
010040  112714  000001                MOVB    #WFG!GO,@TACS           ;GET OFF OF CLEAR LEADER
010044  104412                        WAITREADY                       ;WAIT FOR READY
010046  112714  000003                MOVB    #WRITE!GO,@TACS         ;START A WRITE
010052  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010054  112715  000377                MOVB    #377,@TADB              ;WRITE ONE BYTE
010060  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010062  112715  000377                MOVB    #377,@TADB              ;WRITE
010066  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010070  052714  000020                BIS     #ILBS,@TACS             ;WRITE "CRC"
010074  104412                        WAITREADY                       ;WAIT ON "READY"
010076  112714  000001                MOVB    #WFG!GO,@TACS           ;WRITE A "FILE GAP"
010102  104412                        WAITREADY                       ;WAIT ON "READY"
010104  112714  000017          1$:   MOVB    #REWIND!GO,@TACS        ;BACK OVER THE DATA
010110  104412                        WAITREADY                       ;WAIT ON "READY"
010112  112714  000013                MOVB    #SFFG!GO,@TACS          ;DO A SPACE FWD FILE GAP
010116  104412                        WAITREADY
010120  032714  004000                BIT     #FGAP,@TACS             ;IS FILE GAP=1?
010124  001001                        BNE     2$                      ;BR IF YES
010126  104001                        ERROR   1                       ;FILE GAP NOT=1
010130  112714  000011          2$:   MOVB    #BSBG!GO,@TACS          ;GO TO FIRST FILE GAP ON TAPE
010134  104412                        WAITREADY                       ;GO WAIT FOR "READY" TO SET
010136  032714  124000                BIT     #ERROR!FGAP!LEADER,@TACS        ;DID WE GET ANY ERROR?
010142  001401                        BEQ     3$                      ;BR IF NO
010144  104001                        ERROR   1                       ;AN ERROR OCCURRED
010146  112714  000011          3$:   MOVB    #BSBG!GO,@TACS          ;BSBG WHILE IN A FILE GAP
010152  104412                        WAITREADY                       ;GO WAIT FOR "READY" TO SET
010154  005714                        TST     @TACS                   ;"ERROR" BIT SHOULD BE SET
010156  100401                        BMI     4$                      ;BR IF IT IS
010160  104001                        ERROR   1                       ;"ERROR" BIT WASN'T SET
010162  032714  020000          4$:   BIT     #LEADER,@TACS           ;"CLEAR LEADER" SHOULD BE SET
010166  001001                        BNE     TST32           ;;BR IF "CLEAR LEADER" = 1
010170  104001                        ERROR   1                       ;"CLEAR LEADER" NOT SET
                                ;////////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING TESTS INSURE THAT DATA CAN BE WRITING ONTO AND READ FROM THE TAPE
                                ;////////////////////////////////////////////////////////////////////
                                ;;*************************************************************
                                ;*TEST 32        TEST "WRITE" 377 & 0 "READ" 377 & 0
                                ;;*************************************************************
010172  000004                  TST32:  SCOPE
010174  012767  000012  170764          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
010202  012767  010230  170676          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
010210  012767  010330  170752          MOV     #TST33,$ESCAPE  ;;ESCAPE TO TEST 33 ON ERROR
010216  000005                        RESET
```



```
010220  010314                        MOV     DRIVE,@TACS             ;SELECT DRIVE
010222  112714  000017                MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
010226  104412                        WAITREADY                       ;WAIT ON "READY"
010230  112714  000003          3$:   MOVB    #WRITE!GO,@TACS         ;WRITE DATA
010234  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010236  112715  000377                MOVB    #377,@TADB              ;1ST BYTE
010242  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010244  112715  000000                MOVB    #0,@TADB                ;2ND BYTE
010250  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010252  052714  000020                BIS     #ILBS,@TACS             ;WRITE "CRC"
010256  104412                        WAITREADY                       ;WAIT ON "READY"
010260  112714  000011                MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA
010264  104412                        WAITREADY                       ;WAIT ON "READY"
010266  112714  000005                MOVB    #READ!GO,@TACS          ;START A "READ"
010272  104413                        WAITXFER                        ;WAIT ON "XFER REQ"
010274  011501                        MOV     @TADB,R1                ;PUT THIS BYTE IN R1
010276  012700  000377                MOV     #377,R0                 ;PUT WHAT IT SHOULD BE IN R0
010302  020100                        CMP     R1,R0                   ;DID DATA READ GOOD?
010304  001401                        BEQ     1$                      ;BR IF YES
010306  104005                        ERROR   5                       ;DATA WASN'T = 377
010310  104413                1$:     WAITXFER                        ;WAIT ON "XFER"
010312  005000                        CLR     R0
010314  011501                        MOV     @TADB,R1                ;READ
010316  001401                        BEQ     2$                      ;BR IF DATA = 000
010320  104005                        ERROR   5                       ;DATA WASN'T = 000
010322  052714  000020          2$:   BIS     #ILBS,@TACS             ;SHUT DOWN
010326  104412                        WAITREADY                       ;WAIT ON "READY"
                                ;;*************************************************************
                                ;*TEST 33        TEST "WRITE & READ" A COUNT PATTERN
                                ;;*************************************************************
010330  000004                  TST33:  SCOPE
010332  012767  000012  170626          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
010340  012767  010442  170540          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
010346  012767  010516  170614          MOV     #TST34,$ESCAPE  ;;ESCAPE TO TEST 34 ON ERROR
010354  000005                        RESET                           ;CLEAR ALL
010356  010314                        MOV     DRIVE,@TACS             ;SELECT DRIVE
010360  112714  000017                MOVB    #REWIND!GO,@TACS        ;GO TO "BOT"
010364  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
010366  112714  000001                MOVB    #WFG!GO,@TACS           ;GET ON OXIDE
010372  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
010374  012700  000377                MOV     #377,R0                 ;FIRST DATA PATTERN AND COUNTER
010400  112714  000003                MOVB    #WRITE!GO,@TACS         ;START A WRITE
010404  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
010406  010015                1$:     MOV     R0,@TADB                ;WRITE A BYTE ON TAPE
010410  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
010412  011501                        MOV     @TADB,R1                ;GET BACK THE LAST BYTE WRITTEN
010414  020001                        CMP     R0,R1                   ;       AND CHECK IT
010416  001401                        BEQ     2$                      ;BR IF IT LOOKS GOOD
010420  104005                        ERROR   5                       ;THE DATA IN TADB WAS BAD
010422  005300                2$:     DEC     R0                      ;NEXT PATTERN
010424  002370                        BGE     1$                      ;BR IF MORE TO DO
010426  052714  000020                BIS     #ILBS,@TACS             ;WRITE THE "CRC"
010432  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
010434  005714                        TST     @TACS                   ;ANY ERRORS OCCUR?
010436  100001                        BPL     3$                      ;BR IF NO
010440  104001                        ERROR   1                       ;ERROR OCCURRED DURING WRITE
```

```
1859  010442  112714  000011          3$:     MOVB    #BSBG!GO,@TACS          ;GO TO BEGINNING OF BLOCK
1860  010446  104412                          WAITREADY                       ;WAIT ON "READY" TO SET
1861  010450  112714  000005                  MOVB    #READ!GO,@TACS          ;START A "READ"
1862  010454  012700  000377                  MOV     #377,R0                 ;FIRST DATA PATTERN AND COUNTER
1863  010460  104413                  4$:     WAITXFER                        ;WAIT ON "XFER REQ" TO SET
1864  010462  011501                          MOV     @TADB,R1                ;READ A BYTE FROM TAPE
1865  010464  020001                          CMP     R0,R1                   ;IS IT VALID?
1866  010466  001401                          BEQ     5$                      ;BR IF YES
1867  010470  104005                          ERROR   5                       ;BAD DATA READ FROM TAPE
1868  010472  005300                  5$:     DEC     R0                      ;NEXT PATTERN
1869  010474  002371                          BGE     4$                      ;BR IF MORE TO READ
1870  010476  104413                          WAITXFER                        ;WAIT ON "XFER REQ" TO SET
1871  010500  052714  000020                  BIS     #ILBS,@TACS             ;SHUT DOWN THE "READ" OPERATION
1872  010504  104412                          WAITREADY                       ;WAIT ON "READY" TO SET
1873  010506  032714  140000                  BIT     #ERROR!CRCERR,@TACS     ;ERROR AND CRCERR SHOULD BE = 0
1874  010512  001401                          BEQ     TST34           ;;BR IF THEY ARE
1875  010514  104001                          ERROR   1                       ;(ERROR ! CRCERR) = 1
1876                                  ;////////////////////////////////////////////////////////////////////////
1877                                  ;////////////////////////////////////////////////////////////////////////
1878                                  ;THE FOLLOWING TESTS ARE USED TO INSURE THAT THE "CRC" CIRCUITRY FUNCTIONS PROPERLY
1879                                  ;////////////////////////////////////////////////////////////////////////
1880                                  ;;**************************************************************
1881                                  ;*TEST 34        TEST "ERROR" WITH "CRCERR" = 1
1882                                  ;;**************************************************************
1883  010516  000004                  TST34:  SCOPE
1884  010520  012767  000012  170440          MOV     #10.,$TIMES     ;;DO 10. ITERATIONS
1885  010526  012767  010562  170352          MOV     #20$,$LPADR     ;;SET SCOPE LOOP ADDRESS
1886  010534  012767  010776  170426          MOV     #TST35,$ESCAPE  ;;ESCAPE TO TEST 35 ON ERROR
1887  010542  000005                          RESET
1888  010544  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1889  010546  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO CLEAR LEADER
1890  010552  104412                          WAITREADY                       ;WAIT ON "READY"
1891  010554  112714  000001                  MOVB    #WFG!GO,@TACS           ;GET OFF OF CLEAR LEADER
1892  010560  104412                          WAITREADY                       ;WAIT FOR READY
1893  010562  112714  000003          20$:    MOVB    #WRITE!GO,@TACS         ;START A WRITE
1894  010566  012700  000006                  MOV     #6,R0                   ;SETUP FOR 6 BYTES
1895  010572  104413                  1$:     WAITXFER                        ;WAIT ON "XFER REQ"
1896  010574  005300                          DEC     R0                      ;DOWN COUNT
1897  010576  002403                          BLT     2$                      ;DONE?
1898  010600  112715  000377                  MOVB    #377,@TADB              ;NO--WRITE ON TAPE
1899  010604  000772                          BR      1$                      ;LOOP
1900  010606  052714  000020          2$:     BIS     #ILBS,@TACS             ;WRITE "CRC"
1901  010612  104412                          WAITREADY                       ;WAIT ON "READY"
1902  010614  112714  000011                  MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA BLOCK
1903  010620  104412                          WAITREADY                       ;WAIT ON "READY"
1904  010622  112714  000005                  MOVB    #READ!GO,@TACS          ;START A "READ"
1905  010626  012700  000003                  MOV     #3,R0                   ;DO 3 BYTES
1906  010632  104413                  3$:     WAITXFER                        ;WAIT FOR "XFER REQ"
1907  010634  005300                          DEC     R0                      ;COUNT # OF BYTES
1908  010636  002402                          BLT     4$
1909  010640  105715                          TSTB    @TADB                   ;CLEAR "XFER REQ"
1910  010642  000773                          BR      3$
1911  010644  052714  000020          4$:     BIS     #ILBS,@TACS             ;DO "ILBS"
1912  010650  104412                          WAITREADY                       ;WAIT ON "READY"
1913  010652  005714                          TST     @TACS                   ;CHECK FOR "ERROR"
1914  010654  100401                          BMI     5$                      ;BR IF "ERROR"
```



```
1915  010656  104001                          ERROR   1                       ;"ERROR" BIT NOT SET
1916  010660  032714  040000          5$:     BIT     #CRCERR,@TACS           ;CHECK FOR "CRC" ERROR
1917  010664  001001                          BNE     6$                      ;BR IF "CRC" ERROR
1918  010666  104001                          ERROR   1                       ;NO "CRC" ERROR
1919                                  ;THE FOLLOWING CODE IS USED TO CHECK THE ROM
1920  010670  112714  000000          6$:     MOVB    #WFG,@TACS              ;CHECK "ERROR" WITH "WFG"
1921  010674  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1922  010676  100001                          BPL     7$                      ;BR IF "ERROR" = 0
1923  010700  104001                          ERROR   1                       ;"ERROR" NOT = 0
1924  010702  112714  000002          7$:     MOVB    #WRITE,@TACS            ;CHECK "ERROR" WITH "WRITE"
1925  010706  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1926  010710  100401                          BMI     8$                      ;BR IF "ERROR" = 1
1927  010712  104001                          ERROR   1                       ;"ERROR" NOT = 1
1928  010714  112714  000006          8$:     MOVB    #BSFG,@TACS             ;CHECK "ERROR" WITH "BSFG"
1929  010720  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1930  010722  100001                          BPL     9$                      ;BR IF "ERROR" = 0
1931  010724  104001                          ERROR   1                       ;"ERROR" NOT = 0
1932  010726  112714  000010          9$:     MOVB    #BSBG,@TACS             ;CHECK "ERROR" WITH "BSBG"
1933  010732  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1934  010734  100001                          BPL     10$                     ;BR IF "ERROR" = 0
1935  010736  104001                          ERROR   1                       ;"ERROR" NOT = 0
1936  010740  112714  000012          10$:    MOVB    #SFFG,@TACS             ;CHECK "ERROR" WITH "SFFG"
1937  010744  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1938  010746  100001                          BPL     11$                     ;BR IF "ERROR" = 0
1939  010750  104001                          ERROR   1                       ;"ERROR" NOT = 0
1940  010752  112714  000014          11$:    MOVB    #SFBG,@TACS             ;CHECK "ERROR" WITH "SFBG"
1941  010756  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1942  010760  100001                          BPL     12$                     ;BR IF "ERROR" = 0
1943  010762  104001                          ERROR   1                       ;"ERROR" NOT = 0
1944  010764  112714  000016          12$:    MOVB    #REWIND,@TACS           ;CHECK "ERROR" WITH "REWIND"
1945  010770  005714                          TST     @TACS                   ;SAMPLE THE "ERROR" BIT
1946  010772  100001                          BPL     TST35           ;;BR IF "ERROR" = 0
1947  010774  104001                          ERROR   1                       ;"ERROR" NOT = 0
1948                                  ;;**************************************************************
1949                                  ;*TEST 35        TEST "DATA OF 0 GIVES CRC OF 0"
1950                                  ;;**************************************************************
1951  010776  000004                  TST35:  SCOPE
1952  011000  012767  000005  170160          MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1953  011006  012767  011034  170072          MOV     #6$,$LPADR      ;;SET SCOPE LOOP ADDRESS
1954  011014  012767  011160  170146          MOV     #TST36,$ESCAPE  ;;ESCAPE TO TEST 36 ON ERROR
1955  011022  000005                          RESET
1956  011024  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
1957  011026  112714  000017                  MOVB    #REWIND!GO,@TACS        ;GO TO "CLEAR LEADER"
1958  011032  104412                          WAITREADY                       ;WAIT ON "READY"
1959  011034  112714  000003          6$:     MOVB    #WRITE!GO,@TACS         ;WRITE THE DATA
1960  011040  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1961  011042  105015                          CLRB    @TADB                   ;1ST BYTE = 0
1962  011044  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1963  011046  105015                          CLRB    @TADB                   ;2ND BYTE = 0
1964  011050  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
1965  011052  052714  000020                  BIS     #ILBS,@TACS             ;WRITE "CRC"
1966  011056  104412                          WAITREADY                       ;WAIT ON "READY"
1967  011060  112714  000011                  MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA
1968  011064  104412                          WAITREADY                       ;WAIT ON "READY"
1969  011066  112714  000005                  MOVB    #READ!GO,@TACS          ;START A "READ"
1970  011072  104413                          WAITXFER                        ;WAIT ON "XFER REQ" FIRST
```

```
1971  011074  005000                        CLR     R0                      ;SET R0 TO WHAT THE DATA SHOULD BE
1972  011076  011501                        MOV     @TADB,R1                ;READ DATA BYTE
1973  011100  001401                        BEQ     1$                      ;BR IF DATA = 0
1974  011102  104005                        ERROR   5                       ;1ST BYTE NOT = 0
1975  011104  104413              1$:       WAITXFER                        ;WAIT ON "XFER REQ"
1976  011106  011501                        MOV     @TADB,R1                ;READ SECOND DATA BYTE
1977  011110  001401                        BEQ     2$                      ;BR IF 2ND BYTE = 0
1978  011112  104005                        ERROR   5                       ;DATA WASN'T = 0
1979  011114  104413              2$:       WAITXFER                        ;WAIT ON "XFER REQ"
1980  011116  011501                        MOV     @TADB,R1                ;READ FIRST CRC BYTE
1981  011120  001401                        BEQ     3$                      ;BR IF 1ST CRC BYTE = 0
1982  011122  104005                        ERROR   5                       ;1ST BYTE OF CRC BAD
1983  011124  104413              3$:       WAITXFER                        ;WAIT ON "XFER REQ"
1984  011126  011501                        MOV     @TADB,R1                ;READ SECOND CRC BYTE
1985  011130  001401                        BEQ     4$                      ;BR IF 2ND CRC BYTE = 0
1986  011132  104005                        ERROR   5                       ;2ND BYTE OF CRC BAD
1987  011134  052714  000020      4$:       BIS     #ILBS,@TACS
1988  011140  104412                        WAITREADY                       ;WAIT FOR "READY"
1989  011142  005714                        TST     @TACS                   ;"ERROR" SHOULD BE = 1
1990  011144  100401                        BMI     5$                      ;BR IF "ERROR" IS  = 1
1991  011146  104001                        ERROR   1
1992  011150  032714  040000      5$:       BIT     #CRCERR,@TACS           ;IS  THE ERROR  A "CRC" ERROR?
1993  011154  001001                        BNE     TST36           ;;GO TO NEXT TEST IF YES
1994  011156  104001                        ERROR   1                       ;THE ERROR WASN'T A "CRC" ERROR
1995                                ;;**************************************************************
1996                                ;*TEST 36        TEST "CRC" CIRCUIT USING A COUNT PATTERN
1997                                ;;**************************************************************
1998  011160  000004              TST36:  SCOPE
1999  011162  012767  000012  167776        MOV     #10.,sTIMES     ;;DO 10. ITERATIONS
2000  011170  012767  011272  167710        MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
2001  011176  012767  011412  167764        MOV     #TST37,$ESCAPE  ;;ESCAPE TO TEST 37 ON ERROR
2002  011204  000005                        RESET                           ;CLEAR ALL
2003  011206  010314                        MOV     DRIVE,@TACS             ;SELECT DRIVE
2004  011210  112714  000017                MOVB    #REWIND!GO,@TACS        ;GO TO "BOT"
2005  011214  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
2006  011216  112714  000001                MOVB    #WFG!GO,@TACS           ;GET ON OXIDE
2007  011222  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
2008  011224  012700  000377                MOV     #377,R0                 ;FIRST DATA PATTERN AND COUNTER
2009  011230  112714  000003                MOVB    #WRITE!GO,@TACS         ;START A WRITE
2010  011234  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
2011  011236  010015              1$:       MOV     R0,@TADB                ;WRITE A BYTE ON TAPE
2012  011240  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
2013  011242  011501                        MOV     @TADB,R1                ;GET BACK THE LAST BYTE WRITTEN
2014  011244  020001                        CMP     R0,R1                   ;       AND CHECK IT
2015  011246  001401                        BEQ     2$                      ;BR IF IT LOOKS GOOD
2016  011250  104005                        ERROR   5                       ;THE DATA IN TADB WAS BAD
2017  011252  005300              2$:       DEC     R0                      ;NEXT PATTERN
2018  011254  002370                        BGE     1$                      ;BR IF MORE TO DO
2019  011256  052714  000020              BIS     #ILBS,@TACS             ;WRITE THE "CRC"
2020  011262  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
2021  011264  005714                        TST     @TACS                   ;ANY ERRORS OCCUR?
2022  011266  100001                        BPL     3$                      ;BR IF NO
2023  011270  104001                        ERROR   1                       ;ERROR OCCURRED DURING WRITE
2024  011272  112714  000011      3$:       MOVB    #BSBG!GO,@TACS          ;BACK OVER THE DATA BLOCK
2025  011276  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
2026  011300  112714  000005                MOVB    #READ!GO,@TACS          ;START A "READ"
```



```
2027  011304  012700  000377                MOV     #377,R0                 ;FIRST DATA PATTERN AND COUNTER
2028  011310  005037  013774                CLR     @#CRC.WD                ;INITIALIZES THE "CRC WORD"
2029  011314  004737  013654      4$:       JSR     PC,@#DO.CRC             ;COMBINE THIS BYTE [R0] WITH THE "CRC WORD"
2030  011320  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
2031  011322  011501                        MOV     @TADB,R1                ;READ A BYTE FROM TAPE
2032  011324  020001                        CMP     R0,R1                   ;IS IT VALID?
2033  011326  001401                        BEQ     5$                      ;BR IF YES
2034  011330  104005                        ERROR   5                       ;BAD DATA READ FROM TAPE
2035  011332  005300              5$:       DEC     R0                      ;NEXT PATTERN
2036  011334  002367                        BGE     4$                      ;BR IF MORE TO READ
2037  011336  005000                        CLR     R0                      ;GET LOW BYTE OF "CRC WORD"
2038  011340  153700  013774                BISB    @#CRC.WD,R0
2039  011344  104413                        WAITXFER                        ;WAIT ON "XFER REQ" TO SET
2040  011346  052714  000020                BIS     #ILBS,@TACS             ;SHUT DOWN THE "READ" OPERATION
2041  011352  011501                        MOV     @TADB,R1                ;PICK FIRST BYTE OF TU60 "CRC"
2042  011354  020001                        CMP     R0,R1                   ;IS IT GOOD?
2043  011356  001401                        BEQ     6$                      ;BR IF YES
2044  011360  104005                        ERROR   5                       ;FIRST BYTE OF TU60 "CRC" IS BAD
2045  011362  005000                        CLR     R0                      ;GET HIGH BYTE OF "CRC WORD"
2046  011364  153700  013775                BISB    @#CRC.WD+1,R0
2047  011370  104412                        WAITREADY                       ;WAIT ON "READY" TO SET
2048  011372  011501                        MOV     @TADB,R1                ;READ THE 2ND BYTE OF TU60 "CRC"
2049  011374  020001                        CMP     R0,R1                   ;IS IT GOOD?
2050  011376  001401                        BEQ     7$                      ;BR IF YES
2051  011400  104005                        ERROR   5                       ;HIGH BYTE OF "CRC" IS BAD
2052  011402  032714  140000      7$:       BIT     #ERROR!CRCERR,@TACS     ;ERROR AND CRCERR SHOULD BE = 0
2053  011406  001401                        BEQ     TST37           ;;BR IF THEY ARE
2054  011410  104001                        ERROR   1                       ;(ERROR ! CRCERR) = 1
2055                                ;;**************************************************************
2056                                ;THIS TEST REQUIRES BOTH DRIVES .
2057                                ;FOR THE FOLLOWING DESCRIPTION DRIVE "1" IS THE DRIVE UNDER TEST
2058                                ;AND DRIVE "2" IS THE OTHER DRIVE
2059                                ;
2060                                ;TEST DESCRIPTION
2061                                ;1)     REWIND DRIVE 1
2062                                ;2)     WAIT FOR READY TO SET
2063                                ;3)     START DRIVE 2 REWINDING
2064                                ;4)     WAIT FOR READY TO CLEAR
2065                                ;5)     SELECT DRIVE 1  AND CHECK THAT READY IS STILL SET
2066                                ;6)     SELECT DRIVE 2 AND CHECK THAT READY IS STILL CLEAR
2067                                ;
2068                                ;;**************************************************************
2069                                ;*TEST 37        TRY TO HANG "READY" ON "REWIND"
2070                                ;;**************************************************************
2071  011412  000004              TST37:  SCOPE
2072  011414  012767  000005  167544        MOV     #5,sTIMES       ;;DO 5 ITERATIONS
2073  011422  012767  011436  167456        MOV     #1$,$LPADR      ;;SET SCOPE LOOP ADDRESS
2074  011430  012767  011552  167532        MOV     #TST40,$ESCAPE  ;;ESCAPE TO TEST 40 ON ERROR
2075  011436  105737  001225      1$:       TSTB    @#DRVKEY+1              ;TESTING TWO DRIVES?
2076  011442  001443                        BEQ     TST40           ;;BR IF NO
2077  011444  012701  000001                MOV     #1,R1                   ;SET TIMER
2078  011450  010302                        MOV     DRIVE,R2                ;SET R2 TO THE OTHER DRIVE
2079  011452  062702  000400                ADD     #UNIT,R2
2080  011456  042702  177377                BIC     #^CUNIT,R2
2081  011462  000005                        RESET                           ;CLEAR ALL
2082  011464  010314                        MOV     DRIVE,@TACS             ;SELECT 1ST DRIVE
```

```
                                        WAITREADY                       ;WAIT ON "READY"
                                        MOV     R2,@TACS                ;SELECT 2ND DRIVE
                                        WAITREADY                       ;WAIT ON "READY"
                                        MOVB    #REWIND!GO,@TACS        ;SEND 2ND DRIVE TO BOT
                                2$:     ASL     R1                      ;GIVE "READY" TIME TO CLEAR
                                        BNE     3$
                                        ERROR   1                       ;"READY" FAILED TO CLEAR
                                                                        ;NOTE:  THIS ERROR OCCURRED ON THE
                                                                        ;2ND DRIVE, I.E. IF TESTING DRIVE A
                                                                        ;THEN DRIVE B FAILED
                                3$:     BIT     #READY,@TACS            ;WAIT FOR "READY" TO CLEAR
                                        BNE     2$
                                        MOV     DRIVE,@TACS             ;SELECT 1ST DRIVE
                                        MOVB    #REWIND,@TACS           ;LOAD A "REWIND"
                                        BIT     #READY,@TACS            ;CHECK "READY"
                                        BNE     4$                      ;BR IF STILL SET
                                        ERROR   1                       ;"READY" WAS CLEAR
                                4$:     MOV     R2,@TACS                ;SELECT 2ND DRIVE
                                        MOVB    #REWIND,@TACS           ;LOAD A REWIND
                                        BIT     #READY,@TACS            ;CHECK "READY"
                                        BEQ     5$                      ;BR IF STILL CLEAR
                                        ERROR   1                       ;"READY" WAS SET
                                5$:     WAITREADY                       ;WAIT ON "READY"
                                ;;*****************************************************************
                                ;THIS TEST REQUIRES BOTH DRIVES .
                                ;FOR THE FOLLOWING DESCRIPTION DRIVE "1" IS THE DRIVE UNDER TEST
                                ;AND DRIVE "2" IS THE OTHER DRIVE
                                ;
                                ;TEST DESCRIPTION
                                ;1)     REWIND DRIVE 1
                                ;2)     WFG ON DRIVE 1
                                ;3)     START A REWIND ON DRIVE 2
                                ;4)     WHILE DRIVE 2 IS REWINDING WRITE DATA ON DRIVE 1
                                ;5)     CHECK FOR ERRORS
                                ;6)     START A REWIND ON DRIVE 2
                                ;7)     WHILE DRIVE 2 IS REWINDING READ DATA FROM DRIVE 1
                                ;8)     CHECK FOR ERRORS
                                ;
                                ;;*****************************************************************
                                ;*TEST 40        TRY TO GLITCH THE "POWER SUPPLY"
                                ;;*****************************************************************
011552  000004                TST40:  SCOPE
011554  012767  000005  167404        MOV     #5,STIMES       ;;DO 5 ITERATIONS
011562  012767  011576  167316        MOV     #1$,SLPADR      ;;SET SCOPE LOOP ADDRESS
011570  012767  012134  167372        MOV     #TST41,SESCAPE  ;;ESCAPE TO TEST 41 ON ERROR
011576  105737  001225        1$:     TSTB    @#DRVKEY+1              ;TESTING TWO DRIVES?
011602  001554                        BEQ     TST41           ;;BR IF NO
011604  012701  000001                MOV     #1,R1                   ;SET TIMER
011610  010302                        MOV     DRIVE,R2                ;SET R2 TO THE OTHER DRIVE
011612  062702  000400                ADD     #UNIT,R2
011616  042702  177377                BIC     #^CUNIT,R2
011622  000005                        RESET                           ;CLEAR ALL
011624  010314                        MOV     DRIVE,@TACS             ;SELECT 1ST DRIVE
011626  112714  000017                MOVB    #REWIND!GO,@TACS        ;REWIND TO BOT
011632  104412                        WAITREADY                       ;WAIT ON "READY"
011634  112714  000001                MOVB    #WFG!GO,@TACS           ;GET ON OXIDE
```



```
                                        WAITREADY                       ;WAIT ON "READY"
                                        MOV     R2,@TACS                ;SELECT 2ND DRIVE
                                        WAITREADY                       ;WAIT ON "READY"
                                        MOVB    #REWIND!GO,@TACS        ;START A "REWIND"
                                2$:     ASL     R1                      ;GIVE "READY" TIME TO CLEAR
                                        BNE     3$
                                        ERROR   1                       ;"READY" FAILED TO CLEAR
                                                                        ;NOTE:  THIS ERROR OCCURRED ON THE
                                                                        ;2ND DRIVE, I.E. IF TESTING DRIVE A
                                                                        ;THEN DRIVE B FAILED
                                3$:     BIT     #READY,@TACS            ;WAIT FOR "READY" TO CLEAR
                                        BNE     2$
                                        MOV     DRIVE,@TACS             ;SELECT 1ST DRIVE
                                        MOVB    #WRITE!GO,@TACS         ;WRITE WHILE THE OTHER DRIVE IS "REWINDING"
                                        WAITXFER                        ;WAIT ON "XFER REQ"
                                        MOVB    #377,@TADB              ;WRITE 1ST BYTE
                                        WAITXFER                        ;WAIT ON "XFER REQ"
                                        CLRB    @TADB                   ;WRITE 2ND BYTE
                                        WAITXFER                        ;WAIT ON "XFER REQ"
                                        BIS     #ILBS,@TACS             ;WRITE "CRC" & SHUT DOWN
                                        WAITREADY                       ;WAIT ON "READY"
                                        TST     @TACS                   ;ANY ERROR?
                                        BPL     4$                      ;BR IF NO
                                        ERROR   1                       ;ERROR OCCURRED DURING "WRITE"
                                4$:     MOVB    #BSBG!GO,@TACS          ;POSITION TAPE FOR "READ"
                                        WAITREADY                       ;WAIT ON "READY"
                                        TST     @TACS                   ;ANY ERROR?
                                        BPL     5$                      ;BR IF NO
                                        ERROR   1                       ;ERROR OCCURRED DURING "BSBG"
                                5$:     MOVB    #READ!GO,@TACS          ;START A "READ"
                                        WAITXFER                        ;WAIT ON "XFER REQ"
                                        MOV     #377,R0                 ;FIRST DATA PATTERN
                                        MOV     @TADB,R1                ;PICKUP FIST BYTE
                                        CMP     R0,R1                   ;IS IT GOOD?
                                        BEQ     6$                      ;BR IF NO
                                        ERROR   5                       ;1ST DATA BYTE IS BAD
                                6$:     WAITXFER                        ;WAIT ON "XFER REQ"
                                        CLR     R0                      ;2ND DATA PATTERN
                                        MOV     @TADB,R1                ;PICKUP 2ND BYTE
                                        BEQ     7$                      ;BR IF IT IS GOOD
                                        ERROR   5                       ;2ND BYTE IS BAD
                                7$:     WAITXFER                        ;WAIT ON "XFER REQ"
                                        BIS     #ILBS,@TACS             ;SHUT DOWN
                                        WAITREADY                       ;WAIT ON "READY"
                                        TST     @TACS                   ;ERROR?
                                        BPL     8$                      ;BR IF NO
                                        ERROR   1                       ;ERROR OCCURRED
                                8$:     MOVB    #BSBG!GO,@TACS          ;POSITION FOR NEXT "READ"
                                        WAITREADY                       ;WAIT ON "READY"
                                        TST     @TACS                   ;ERROR?
                                        BPL     9$                      ;BR IF NO
                                        ERROR   1                       ;ERROR OCCURRED DURING "BSBG
                                9$:     MOV     R2,@TACS                ;SELECT 2ND DRIVE
                                        WAITREADY                       ;WAIT ON "READY"
                                        MOVB    #REWIND!GO,@TACS        ;START A "REWIND"
                                        MOV     #1,R1                   ;SET THE TIMER
```

```
2195  012042  005201                  10$:    INC     R1                      ;GIVE "READY" TIME TO CLEAR
2196  012044  001001                          BNE     11$
2197  012046  104001                          ERROR   1                       ;"READY" FAILED TO CLEAR
2198                                                                          ;NOTE:  THIS ERROR OCCURRED ON THE
2199                                                                          ;2ND DRIVE, I.E. IF TESTING DRIVE A
2200                                                                          ;THEN DRIVE B FAILED
2201  012050  032714  000040          11$:    BIT     #READY,@TACS            ;WAIT FOR "READY" TO CLEAR
2202  012054  001372                          BNE     10$
2203  012056  010314                          MOV     DRIVE,@TACS             ;SELECT 1ST DRIVE
2204  012060  104412                          WAITREADY                       ;WAIT ON "READY"
2205  012062  112714  000005                  MOVB    #READ!GO,@TACS          ;"READ" WHILE OTHER DRIVE IS "REWINDING"
2206  012066  104413                          WAITXFER                        ;WAIT ON "XFER REQ"
2207  012070  012700  000377                  MOV     #377,R0                 ;1ST DATA PATTERN
2208  012074  011501                          MOV     @TADB,R1                ;READ 1ST BYTE
2209  012076  020001                          CMP     R0,R1                   ;IS IT GOOD?
2210  012100  001401                          BEQ     12$                     ;BR IF NO
2211  012102  104005                          ERROR   5                       ;1ST BYTE FAILED
2212  012104  104413                  12$:    WAITXFER                        ;WAIT ON "XFER REQ"
2213  012106  005000                          CLR     R0                      ;2ND DATA PATTERN
2214  012110  011501                          MOV     @TADB,R1                ;READ 2ND BYTE
2215  012112  001401                          BEQ     13$                     ;BR IF IT IS GOOD
2216  012114  104005                          ERROR   5                       ;2ND BYTE FAILED
2217  012116  104413                  13$:    WAITXFER                        ;WAIT ON "XFER REQ"
2218  012120  052714  000020                  BIS     #ILBS,@TACS             ;SHUT DOWN
2219  012124  104412                          WAITREADY                       ;WAIT ON "READY"
2220  012126  005714                          TST     @TACS                   ;ERROR?
2221  012130  100001                          BPL     TST41           ;;BR IF NO
2222  012132  104001                          ERROR   1                       ;ERROR OCCURRED
2223                                  ;;**************************************************************
2224                                  ;*TEST 41        END OF TEST CODE
2225                                  ;;**************************************************************
2226  012134  000004                  TST41:  SCOPE
2227  012136  012767  000001  167022          MOV     #1,$TIMES       ;;DO 1 ITERATION
2228  012144  000005                          RESET
2229  012146  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
2230  012150  112714  000017                  MOVB    #REWIND!GO,@TACS
2231  012154  104412                          WAITREADY
2232                                  .SBTTL  END OF PASS ROUTINE
2233
2234                                  ;;**************************************************************
2235                                  ;*INCREMENT THE PASS NUMBER ($PASS)
2236                                  ;*TYPE "END PASS"
2237                                  ;*IF THERES A MONITOR GO TO IT
2238                                  ;*IF THERE ISN'T JUMP TO START
2239                                  ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
2240                                  ;*SENDMG CAN BE CHANGED TO 7.
2241
2242  012156                          $EOP:
2243  012156  000004                          SCOPE
2244  012160  005067  166716                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
2245  012164  005067  166776                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
2246  012170  005267  166704                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
2247  012174  042767  100000  166676          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
2248  012202  005327                          DEC     (PC)+           ;;LOOP?
2249  012204  000001                  $EOPCT: .WORD   1
2250  012206  003015                          BGT     $DOAGN          ;;YES
```



```
2251  012210  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
2252  012212  000001                  $ENDCT: .WORD   1
2253  012214  012204                          $EOPCT
2254  012216  104401  012251                  TYPE    ,$ENDMG         ;;TYPE "END PASS"
2255  012222  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
2256  012226  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
2257  012230  000005                          RESET                   ;;CLEAR THE WORLD
2258  012232  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
2259  012234  000240                          NOP                     ;;SAVE ROOM
2260  012236  000240                          NOP                     ;;FOR
2261  012240  000240                          NOP                     ;;ACT11
2262  012242                          $DOAGN:
2263  012242  000137                          JMP     @(PC)+          ;;RETURN
2264  012244  002220                  $RTNAD: .WORD   START
2265  012246     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
2266  012251     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS/
2267  012256  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

012264                          $SCOPE:
012264  104406                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
012266  032777  040000  166644  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
012274  001111                          BNE     $OVER           ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
012276  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
012300  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
012304  012737  012324  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
012312  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
012316  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
012322  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
012324  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
012326  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
012332  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
012334                          6$:;#####END OF CODE FOR THE XOR TESTER#####
012334  032777  000400  166576          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
012342  001404                          BEQ     2$              ;;BR IF NO
012344  127767  166570  166530          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
012352  001462                          BEQ     $OVER           ;;BR IF YES
012354  105767  166523          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
012360  001421                          BEQ     3$              ;;BR IF NO
012362  126767  166527  166513          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
012370  101015                          BHI     3$              ;;BR IF NO
012372  032777  001000  166540          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
012400  001404                          BEQ     4$              ;;BR IF NO
012402  016767  166502  166476  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
012410  000443                          BR      $OVER
012412  105067  166465          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
012416  005067  166544                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
012422  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
012424  032777  004000  166506  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
012432  001011                          BNE     1$              ;;BR IF YES
012434  005767  166440                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
012440  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
012442  005267  166436                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
012446  026767  166514  166430          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
012454  002021                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
012456  012767  000001  166420  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
012464  016767  000044  166474          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
012472  105267  166404          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
012476  011667  166404                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
```



```
012502  011667  166402                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
012506  005067  166456                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
012512  112767  000001  166375          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
012520  016777  166356  166414  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
012526  016716  166354                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
012532  000002                          RTI                     ;;FIXES PS
012534  003720                  $MXCNT: 2000.                   ;;MAX. NUMBER OF ITERATIONS
```

```
                                .SBTTL  ERROR HANDLER ROUTINE

                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                ;*AND GO TO TYPERR ON ERROR
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW15=1         HALT ON ERROR
                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                ;*SW10=1         BELL ON ERROR
                                ;*SW09=1         LOOP ON ERROR
                                ;*CALL
                                ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

012536                          $ERROR:
012536  104406                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
012540  011437  001162                   MOV     @TACS,@#$REG0           ;SAVE THE STATUS REG.
012544  011537  001164                   MOV     @TADB,@#$REG1           ;SAVE THE DATA BUFFER
012550  010037  001124                   MOV     R0,@#$GDDAT             ;R0 WILL CONTAIN THE GOOD DATA
012554  010137  001126                   MOV     R1,@#$BDDAT             ;R1 WILL CONTAIN THE BAD DATA
012560  105267  166317           7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
012564  001775                   BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
012566  016777  166310  166346         MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
012574  032777  002000  166336         BIT     #BIT10,@SWR     ;;BELL ON ERROR?
012602  001402                   BEQ     1$              ;;NO - SKIP
012604  104401  001172                   TYPE    ,$BELL          ;;RING BELL
012610  005267  166276           1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
012614  011667  166276                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
012620  162767  000002  166270         SUB     #2,$ERRPC
012626  117767  166264  166260         MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
012634  032777  020000  166276         BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
012642  001004                   BNE     20$             ;;SKIP TYPEOUTS
012644  004767  000060                   JSR     PC,TYPERR       ;;GO TO USER ERROR ROUTINE
012650  104401  001177                   TYPE    ,$CRLF
012654                          20$:
012654  005777  166260           2$:     TST     @SWR            ;;HALT ON ERROR
012660  100002                   BPL     3$              ;;SKIP IF CONTINUE
012662  000000                   HALT                    ;;HALT ON ERROR!
012664  104406                   CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
012666  032777  001000  166244   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
012674  001402                   BEQ     4$              ;;BR IF NO
012676  016716  166206                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
012702  005767  166262           4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
012706  001402                   BEQ     5$              ;;BR IF NONE
012710  016716  166254                   MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
012714                          5$:
012714  022737  012232  000042         CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
012722  001001                   BNE     6$              ;;BRANCH IF NO
012724  000000                   HALT                    ;;YES
012726                          6$:
012726  000002                   RTI                     ;;RETURN
```



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

012730  104401  001177           TYPERR: TYPE    ,$CRLF                  ;TYPE A CARRIAGE RETURN & LINE FEED
012734  010046                   MOV     R0,-(SP)                ;SAVE R0
012736  113700  001114                   MOVB    @#$ITEMB,R0             ;PICKUP THE ITEM INDEX
012742  005300                   DEC     R0                      ;ADJUST THE INDEX
012744  006300                   ASL     R0                      ;SO IT WILL WORK FOR
012746  006300                   ASL     R0                      ;THE ERROR TABLE
012750  006300                   ASL     R0
012752  062700  001236                   ADD     #$ERRTB,R0              ;FORM THE TABLE POINTER
012756  012067  000002                   MOV     (R0)+,1$                ;PICKUP "ERROR MESSAGE" POINTER
012762  104401                   TYPE                            ;TYPE "ERROR MESSAGE"
012764  000000           1$:     0                               ;"ERROR MESSAGE POINTER" GOES HERE
012766  104401  001177                   TYPE    ,$CRLF
012772  012067  000004                   MOV     (R0)+,2$                ;PICKUP "DATA HEADER" POINTER
012776  001404                   BEQ     3$                      ;IF "0" DON'T TYPE
013000  104401                   TYPE                            ;TYPE "DATA HEADER
013002  000000           2$:     0                               ;"DATA HEADER" POINTER GOES HERE
013004  104401  001177                   TYPE    ,$CRLF
013010  012000           3$:     MOV     (R0)+,R0                ;PICKUP "DATA POINTER"
013012  001004                   BNE     5$                      ;IF THERE IS DATA TO TYPE GO DO IT
013014  012600           4$:     MOV     (SP)+,R0                ;RESTORE R0
013016  104401  001177                   TYPE    ,$CRLF                  ;TYPE A CARRAGE RETURN&LINE FEED
013022  000207                   RTS     PC                      ;RETURN
013024                          5$:
013024  013046                   MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                                                        ;;TYPE DATA
013026  104402                   TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
013030  005710                   TST     (R0)                    ;TERMINATOR?
013032  001770                   BEQ     4$                      ;BR IF YES
013034  104401  013042           TYPE    ,6$                     ;TYPE 2 SPACES
013040  000771                   BR      5$                      ;LOOP
013042  020040      000  6$:     .ASCIZ  /  /                    ;ASCII STRING OF 2 SPACES
013046                           .EVEN
```

```
2418                                    ;;*****************************************************************
2419                                    ;ROUTINE TO WAIT ON THE READY BIT
2420
2421  013046                            WAIT.ON.READY:
2422  013046  005067  000044                   CLR     WAIT2                   ;SETUP MAX. TIME TO WAIT ON "READY"
2423  013052  016767  000072  000044           MOV     MAXCNT,HGHTIM
2424  013060  012637  001202                   MOV     (SP)+,@#SAVPC           ;GET THE PC OF THE WAITREADY INSTRUCTION
2425  013064  162737  000002  001202           SUB     #2,@#SAVPC
2426  013072  012637  001204                   MOV     (SP)+,@#SAVPS           ;SAVE THE PS
2427  013076  032714  000040           WAIT1:  BIT     #READY,@TACS            ;READY=1?
2428  013102  001013                           BNE     WAIT3                   ;GO ON IF YES
2429  013104  105714                           TSTB    @TACS                   ;CHECK TRANSFER REQUEST
2430  013106  100002                           BPL     WAIT4
2431  013110  104004                           ERROR   4                       ;"TRANSFER REQUEST" SET WHILE WAITING ON "READY"
2432  013112  000407                           BR      WAIT3
2433  013114  005227           WAIT4:  INC     (PC)+                   ;COUNT FAST COUNTER
2434  013116  000000           WAIT2:  0
2435  013120  001366                           BNE     WAIT1                   ;GO CHECK "READY" AGAIN
2436  013122  005327                           DEC     (PC)+                   ;COUNT LOOP COUNTER
2437  013124  000000           HGHTIM: 0
2438  013126  003363                           BGT     WAIT1                   ;GO LOOP AGAIN
2439  013130  104002                           ERROR   2                       ;"READY" FAILED TO SET
2440  013132  013746  001204           WAIT3:  MOV     @#SAVPS,-(SP)           ;GET THE STATUS BACK
2441  013136  013746  001202                   MOV     @#SAVPC,-(SP)           ;GET THE PC
2442  013142  062716  000002                   ADD     #2,(SP)
2443  013146  000002                           RTI
2444  013150  000000           MAXCNT: 0
2445
2446                                    ;;*****************************************************************
2447                                    ;ROUTINE TO WAIT ON TRANSFER REQUEST
2448
2449  013152                            WAIT.FOR.XFER.REQ:
2450  013152  005067  000044                   CLR     2$                      ;SETUP WASTE TIME LOOP
2451  013156  013767  013150  000044           MOV     @#MAXCNT,3$
2452  013164  012637  001202                   MOV     (SP)+,@#SAVPC           ;GET THE PC OF THE WAITXFER INSTRUCTION
2453  013170  162737  000002  001202           SUB     #2,@#SAVPC
2454  013176  012637  001204                   MOV     (SP)+,@#SAVPS           ;SAVE THE PS
2455  013202  105714           1$:     TSTB    @TACS                   ;CHECK XFER REQ
2456  013204  100414                           BMI     4$                      ;EXIT IF SET
2457  013206  032714  000040                   BIT     #READY,@TACS            ;LOOK AT READY
2458  013212  001402                           BEQ     5$                      ;BR IF "READY" ISN'T SET
2459  013214  104004                           ERROR   4                       ;"READY" SET WHILE WAITING FOR "XFER REQ"
2460  013216  000407                           BR      4$
2461  013220  005227           5$:     INC     (PC)+                   ;COUNT
2462  013222  000000           2$:     0
2463  013224  001366                           BNE     1$                      ;BR IF MORE TO DO
2464  013226  005327                           DEC     (PC)+
2465  013230  000000           3$:     0
2466  013232  003363                           BGT     1$
2467  013234  104003                           ERROR   3                       ;"TRANSFER REQUEST" FAILED TO SET
2468  013236  013746  001204           4$:     MOV     @#SAVPS,-(SP)           ;GET THE STATUS BACK
2469  013242  013746  001202                   MOV     @#SAVPC,-(SP)           ;GET THE PC
2470  013246  062716  000002                   ADD     #2,(SP)
2471  013252  000002                           RTI                             ;GO BACK
```



```
2472                                    ;;*****************************************************************
2473
2474                                    .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2475
2476                                    ;CALL
2477                                    ;       JSR     PC,@#ASKDRV
2478                                    ;       RETURN                  ;NOTE: R0 AND R1 ARE DESTROYED
2479
2480  013254  104401  016744           ASKDRV: TYPE    ,MSGDRV                 ;<CRLF>"DRIVE(S)? "
2481  013260  005067  165740                   CLR     DRVKEY
2482  013264  104410                           RDLIN                           ;GO GET A DRIVE
2483  013266  012600                           MOV     (SP)+,R0                ;SETUP TO CHECK FOR VALID DRIVE(S)
2484  013270  105710                           TSTB    @R0                     ;WAS A DRIVE SELECTED?
2485  013272  001425                           BEQ     NOTLGL                  ;BR IF NO
2486  013274  012701  001224                   MOV     #DRVKEY,R1
2487  013300  122710  000101           LOOP:   CMPB    #'A,@R0                 ;WAS DRIVE "A" SELECTED?
2488  013304  001002                           BNE     NOTA                    ;BR IF NO
2489  013306  112021                           MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "A"
2490  013310  000411                           BR      NEXT
2491  013312  122710  000102           NOTA:   CMPB    #'B,@R0                 ;WAS DRIVE "B" SELECTED?
2492  013316  001002                           BNE     NOTB                    ;BR IF NO
2493  013320  112021                           MOVB    (R0)+,(R1)+             ;SET KEY FOR DRIVE "B"
2494  013322  000404                           BR      NEXT
2495  013324  122710  000054           NOTB:   CMPB    #54,@R0                 ;WAS A COMMA TYPED?
2496  013330  001006                           BNE     NOTLGL                  ;BR IF NO
2497  013332  105720                           TSTB    (R0)+                   ;DUMP THE COMMA
2498  013334  105710           NEXT:   TSTB    @R0                     ;TERMINATOR?
2499  013336  001406                           BEQ     EXIT                    ;BR IF YES
2500  013340  022701  001226                   CMP     #DRVKEY+2,R1            ;TWO DRIVES SELECTED?
2501  013344  101355                           BHI     LOOP                    ;BR IF NO
2502  013346  104401  001176           NOTLGL: TYPE    ,$QUES                  ;ILLEGAL INPUT DETECTED
2503  013352  000740                           BR      ASKDRV                  ;GO TRY AGAIN
2504  013354  005767  165644           EXIT:   TST     DRVKEY                  ;ANY DRIVE SELECTED?
2505  013360  001772                           BEQ     NOTLGL                  ;BR IF NO
2506  013362  000207                           RTS     PC
2507
2508                                    ;;*****************************************************************
2509                                    ;CALL
2510                                    ;JSR    PC,@#ASKADR
2511
2512  013364  010046           ASKADR: MOV     R0,-(SP)                ;SAVE R0
2513  013366  104401  016760           1$:     TYPE    ,MSGASK                 ;"TACS?"
2514  013372  104411                           RDOCT                           ;GET VALUE
2515  013374  012600                           MOV     (SP)+,R0                ;PICK UP THE OCTAL NUMBER
2516  013376  001423                           BEQ     3$                      ;IF "0" USE OLD VALUES
2517  013400  020027  160000                   CMP     R0,#160000              ;MAKE SURE IT IS A BUS ADDRESS
2518  013404  103770                           BLO     1$
2519  013406  010037  001206                   MOV     R0,@#TACSL              ;SAVE TOE TACS
2520  013412  062700  000002                   ADD     #2,R0                   ;STEP TO TADB ADDRESS
2521  013416  010037  001212                   MOV     R0,@#TADBL              ;AND SAVE IT
2522  013422  013737  001206  001210           MOV     @#TACSL,@#TACSH ;SETUP TACS UPPER
2523  013430  005237  001210                   INC     @#TACSH                 ;BYTE POINTER
2524
2525  013434  013737  001212  001214           MOV     @#TADBL,@#TADBH         ;SETUP TADB UPPER
2526  013442  005237  001214                   INC     @#TADBH                 ;BYTE POINTER
2527  013446  104401  016770           3$:     TYPE    ,MSGVEC                 ;"VECTOR?"
```

```
2528  013452  104411                        RDOCT
2529  013454  012600                        MOV     (SP)+,R0
2530  013456  001411                        BEQ     5$
2531  013460  020027  001000                CMP     R0,#1000                ;MAKE SURE ADDRESS IS IN VECTOR AREA
2532  013464  103370                        BHIS    3$
2533  013466  010037  001216                MOV     R0,@#TAVEC              ;SAVE AS VECTOR ADDRESS
2534  013472  062700  000002                ADD     #2,R0
2535  013476  010037  001220                MOV     R0,@#TAVEC+2
2536  013502  104401  017000        5$:     TYPE    ,MSGPRI                 ;ASK FOR PRIORITY
2537  013506  104411                        RDOCT
2538  013510  012600                        MOV     (SP)+,R0
2539  013512  001413                        BEQ     6$                      ;IF "0" USE OLD VALUE
2540  013514  020027  000007                CMP     R0,#7                   ;MAKE SURE ITS VALID
2541  013520  101370                        BHI     5$
2542  013522  000300                        SWAB    R0                      ;PUT INTO HIGH BYTE
2543  013524  006200                        ASR     R0                      ;AND SHIFT
2544  013526  006200                        ASR     R0                      ;INTO PROPER
2545  013530  006200                        ASR     R0                      ;POSITION
2546  013532  042700  177437                BIC     #^C<340>,R0             ;SAVE ONLY PRIORITY BITS
2547  013536  010037  001222                MOV     R0,@#TAPRIO             ;STORE IT AWAY
2548  013542  104401  017012        6$:     TYPE    ,MTACS                  ;TACS="
2549  013546  016746  165434                MOV     TACSL,-(SP)     ;;SAVE TACSL FOR TYPEOUT
2550  013552  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2551  013554  104401  017020                TYPE    ,MTADB                  ;"TADB="
2552  013560  016746  165426                MOV     TADBL,-(SP)     ;;SAVE TADBL FOR TYPEOUT
2553  013564  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2554  013566  104401  017027                TYPE    ,MTAVEC                 ;"VECTOR="
2555  013572  016746  165420                MOV     TAVEC,-(SP)     ;;SAVE TAVEC FOR TYPEOUT
2556  013576  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2557  013600  104401  017040                TYPE    ,MTAPRI                 ;"PRIORITY="
2558  013604  016746  165412                MOV     TAPRIO,-(SP)    ;;SAVE TAPRIO FOR TYPEOUT
2559  013610  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2560  013612  104401  017053                TYPE    ,MSGOK                  ;"OK?"
2561  013616  104407                        RDCHR                   ;GO READ ONE CHARACTER
2562  013620  012600                        MOV     (SP)+,R0                ;GET IT
2563  013622  022700  000015                CMP     #15,R0                  ;IS IT "CR"?
2564  013626  001406                        BEQ     7$                      ;BRANCH IF YES
2565  013630  022700  000131                CMP     #"Y,R0                  ;IS IT "Y"?
2566  013634  001403                        BEQ     7$                      ;IT WAS
2567  013636  104401  001176                TYPE    ,$QUES                  ;TYPE "?"
2568  013642  000651                        BR      1$                      ;AND LET HIM CORRECT THEM
2569  013644  104401  017061        7$:     TYPE    ,MYES                   ;TYPE OUT "YES"
2570  013650  012600                        MOV     (SP)+,R0                ;RESTORE R0
2571  013652  000207                        RTS     PC                      ;AND RETURN
```



```
2572                                    ;;*****************************************************************
2573
2574                                    ;THIS ROUTINE WILL CALCULATE THE CRC
2575                                    ;CALL:
2576                                    ;       JSR     PC,DO.CRC               ;R0=1 BYTE OF DATA
2577
2578  013654                            DO.CRC:
2579  013654  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2580  013656  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2581  013660  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2582  013662  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2583  013664  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2584  013666  012767  000010  000054            MOV     #8.,3$                  ;MAKE EIGHT ITERATIONS
2585  013674  016703  000074                    MOV     CRC.WD,R3               ;PICKUP THE CRC WORD
2586  013700  005001                    1$:     CLR     R1
2587  013702  010302                            MOV     R3,R2                   ;GET THE PARTIAL CRC WORD
2588  013704  042702  177776                    BIC     #^CBIT00,R2             ;STRIP AWAY EVERYTHING BUT BIT00
2589  013710  006000                            ROR     R0                      ;PULL OFF THIS BIT
2590  013712  006101                            ROL     R1                      ;AND SETUP TO XOR IT
2591  013714  010104                            MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R2"
2592  013716  040204                            BIC     R2,R4
2593  013720  040102                            BIC     R1,R2
2594  013722  050402                            BIS     R4,R2                   ;RESULTS TO "R2"
2595  013724  006002                            ROR     R2
2596  013726  103006                            BCC     2$
2597  013730  012701  040002                    MOV     #BIT14!BIT01,R1
2598  013734  010104                            MOV     R1,R4                   ;FORM THE XOR OF "R1" AND "R3"
2599  013736  040304                            BIC     R3,R4
2600  013740  040103                            BIC     R1,R3
2601  013742  050403                            BIS     R4,R3                   ;RESULTS TO "R3"
2602  013744  006003                    2$:     ROR     R3
2603  013746  005327                            DEC     (PC)+
2604  013750  000000                    3$:     0
2605  013752  003352                            BGT     1$                      ;BR IF MORE BITS TO DO
2606  013754  010367  000014                    MOV     R3,CRC.WD
2607  013760  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
2608  013762  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
2609  013764  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
2610  013766  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
2611  013770  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
2612  013772  000207                            RTS     PC
2613  013774  000000                    CRC.WD: 0                               ;CRC WORD
2614                                    ;;*****************************************************************
```

```
                                ;////////////////////////////////////////////////////////////////
                                ;////////////////////////////////////////////////////////////////
                                ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TU60
                                ;NOTE:  ### BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ###
                                ;////////////////////////////////////////////////////////////////
                                ;;**************************************************************
                                ;       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".
                                ;;**************************************************************
013776  012706  001100          WFGSUB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
014002  013704  001206                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
014006  013705  001212                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
014012  000005                          RESET                           ;RESET THE WORLD
014014  012737  013776  001110          MOV     #WFGSUB,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
014022  004737  015002                  JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
014026  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
014030  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
014034  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
014040  001775                          BEQ     100$
014042  112714  000001          1$:     MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
014046  104412                          WAITREADY                       ;WAIT ON READY
014050  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
014054  001772                          BEQ     1$                      ;BR IF NO
014056  000000                          HALT                            ;STOP IF YES
014060  000746                          BR      WFGSUB                  ;LOOP ON CONT.


                                ;;**************************************************************
                                ;       WRITE CONTINUOUS BLOCKS OF DATA
                                        ;START AT 224
                                        ;THE PROGRAM WILL HALT THREE(3) TIMES
                                        ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
                                        ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
                                        ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
                                        ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
                                        ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
                                        ;       EACH HALT OPERATOR WILL BE PROMPTED
                                        ;       FOR THE VALUE WITH "SWR=XXXXXX NEW="
                                ;;**************************************************************
014062  004737  014544          WRTSUB: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
014066  012706  001100          WLOOP:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
014072  013704  001206                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
014076  013705  001212                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
014102  000005                          RESET                           ;RESET THE WORLD
014104  012737  014066  001110          MOV     #WLOOP,@#$LPERR         ;SETUP THE LOOP ON ERROR ADDRESS
014112  004737  015002                  JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
014116  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
014120  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
014124  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
014130  001775                          BEQ     100$
```



```
014132  004737  014660          1$:     JSR     PC,@#WRTBLK             ;WRITE A BLOCK
014136  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
014142  001773                          BEQ     1$                      ;BR IF NO
014144  000000                          HALT                            ;STOP IF "EOT"
014146  000747                          BR      WLOOP                   ;LOOP IF CONT.


                                ;;**************************************************************
                                ;       READ CONTINUOUS BLOCKS OF DATA
                                        ;START AT 230
                                        ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
                                        ;AND THE "THRESHOLD POT".
                                ;;**************************************************************
014150  012706  001100          RDSUB:  MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
014154  013704  001206                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
014160  013705  001212                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
014164  000005                          RESET                           ;RESET THE WORLD
014166  012737  014150  001110          MOV     #RDSUB,@#$LPERR         ;SETUP THE LOOP ON ERROR ADDRESS
014174  004737  015002                  JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
014200  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
014202  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
014206  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
014212  001775                          BEQ     100$
014214  004737  014722          1$:     JSR     PC,@#RDBLK              ;READ A BLOCK
014220  032714  020000                  BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
014224  001351                          BNE     RDSUB                   ;BR IF YES---LOOP
014226  000772                          BR      1$


                                ;;**************************************************************
                                ;       WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
                                        ;START AT 234
                                        ;THE PROGRAM WILL HALT THREE(3) TIMES
                                        ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
                                        ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
                                        ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
                                        ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
                                        ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
                                        ;       EACH HALT OPERATOR WILL BE PROMPTED
                                        ;       FOR THE VALUE WITH "SWR=XXXXXX NEW="
                                        ;AND THE "GAP TIME MONO".
                                        ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
                                ;;**************************************************************
014230  004737  014544          WGPBLK: JSR     PC,@#SETBUF             ;GET BLOCK SIZE AND PATTERN
014234  012706  001100          WGBLOP: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
014240  013704  001206                  MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
014244  013705  001212                  MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
014250  000005                          RESET                           ;RESET THE WORLD
014252  012737  014234  001110          MOV     #WGBLOP,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
014260  004737  015002                  JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
014264  010314                          MOV     DRIVE,@TACS             ;SELECT DRIVE
014266  112714  000017                  MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
014272  032714  000040          100$:   BIT     #READY,@TACS            ;WAIT ON READY
014276  001775                          BEQ     100$
014300  112714  000001          1$:     MOVB    #WFG!GO,@TACS           ;WRITE A FILE GAP
014304  104412                          WAITREADY                       ;WAIT ON READY
```

```
2727  014306  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2728  014312  001005                  BNE     2$                      ;BR IF YES
2729  014314  004737  014660          JSR     PC,@#WRTBLK             ;WRITE A BLOCK
2730  014320  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2731  014324  001765                  BEQ     1$                      ;BR IF NO
2732  014326  000000          2$:     HALT                            ;STOP AT "EOT"
2733  014330  000741                  BR      WGBLOP                  ;START OVER ON CONT.
2734
2735
2736                                  ;;**************************************************************
2737                                  ;       READ A BLOCK OF DATA AND A FILE GAP
2738                                  ;       ;START AT 240
2739                                  ;       ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
2740                                  ;       ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
2741                                  ;       ;AND THE "TAPE BLANK MONO".
2742                                  ;;**************************************************************
2743  014332  012706  001100  RGPBLK: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2744  014336  013704  001206          MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2745  014342  013705  001212          MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2746  014346  000005                  RESET                           ;RESET THE WORLD
2747  014350  012737  014332  001110  MOV     #RGPBLK,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2748  014356  004737  015002          JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2749  014362  010314                  MOV     DRIVE,@TACS             ;SELECT DRIVE
2750  014364  112714  000017          MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2751  014370  032714  000040  100$:   BIT     #READY,@TACS            ;WAIT ON READY
2752  014374  001775                  BEQ     100$
2753  014376  004737  014722  1$:     JSR     PC,@#RDBLK              ;READ A BLOCK OF DATA
2754  014402  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2755  014406  001351                  BNE     RGPBLK                  ;BR IF YES
2756  014410  112714  000015          MOVB    #SFBG!GO,@TACS          ;GET INTO A FILE GAP
2757  014414  104412                  WAITREADY
2758  014416  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2759  014422  001343                  BNE     RGPBLK                  ;BR IF YES
2760  014424  000764                  BR      1$                      ;LOOP
2761
2762
2763                                  ;;**************************************************************
2764                                  ;       SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2765                                  ;       ;START AT 244
2766                                  ;       ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
2767                                  ;       ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
2768                                  ;       ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
2769                                  ;       ;(SIGNAL MONO CAN BE CHECKED).
2770                                  ;;**************************************************************
2771  014426  012706  001100  SFFGSB: MOV     #STACK,SP               ;KEEP THE STACK OUT OF THE WAY
2772  014432  013704  001206          MOV     @#TACSL,TACS            ;SETUP THE TA11 STATUS AND
2773  014436  013705  001212          MOV     @#TADBL,TADB            ;DATA BUFFER REGISTERS
2774  014442  000005                  RESET                           ;RESET THE WORLD
2775  014444  012737  014426  001110  MOV     #SFFGSB,@#$LPERR                ;SETUP THE LOOP ON ERROR ADDRESS
2776  014452  004737  015002          JSR     PC,@#NXTDRV             ;GO SETUP FOR NEXT DRIVE
2777  014456  010314                  MOV     DRIVE,@TACS             ;SELECT DRIVE
2778  014460  112714  000017          MOVB    #REWIND!GO,@TACS        ;SEND TAPE TO "BOT"
2779  014464  032714  000040  100$:   BIT     #READY,@TACS            ;WAIT ON READY
2780  014470  001775                  BEQ     100$
2781  014472  112714  000013  1$:     MOVB    #SFFG!GO,@TACS          ;SPACE INTO A FILE GAP
2782  014476  104412                  WAITREADY                       ;WAIT ON READY
```



```
2783  014500  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2784  014504  001772                  BEQ     1$                      ;BR IF NO
2785  014506  000000                  HALT                            ;STOP AT "EOT"
2786  014510  000746                  BR      SFFGSB                  ;LOOP ON CONT.
2787
2788
2789                                  ;;**************************************************************
2790                                  ;       BACK SPACE FILE GAP
2791                                  ;       ;START AT 250
2792                                  ;       ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
2793                                  ;;**************************************************************
2794  014512  000005          BSFGSB: RESET                           ;RESET THE WORLD
2795  014514  012737  014512  001110  MOV     #BSFGSB,@#$LPERR                ;LOOP ON ERROR ADDRESS
2796  014522  010314                  MOV     DRIVE,@TACS             ;SELECT DRIVE
2797  014524  112714  000007  1$:     MOVB    #BSFG!GO,@TACS          ;BACK SPACE A FILE GAP
2798  014530  104412                  WAITREADY                       ;WAIT ON READY
2799  014532  032714  020000          BIT     #LEADER,@TACS           ;AT "CLEAR LEADER"?
2800  014536  001772                  BEQ     1$                      ;BR IF NO
2801  014540  000000                  HALT                            ;STOP AT BOT
2802  014542  000763                  BR      BSFGSB                  ;START OVER ON CONT.
2803
2804
2805                                  ;;**************************************************************
2806                                  ;       SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2807                                  ;;**************************************************************
2808  014544  005000          SETBUF: CLR     R0
2809  014546  000000                  HALT                            ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
2810  014550  022767  000176  164362  CMP     #SWREG,SWR              ;USING S/W SWITCH REG.?
2811  014556  001001                  BNE     20$                     ;NO- GET OUT
2812  014560  104405                  GTSWR                           ;LET HIM CHANGE IT
2813  014562                  20$:                                    ;CONTINUE
2814  014562  157700  164352          BISB    @SWR,R0         ;PICKUP THE BYTE COUNT
2815  014566  001006                  BNE     2$                      ;BR IF NON-ZERO
2816  014570  105777  164345          TSTB    @SWR+1                  ;CHECK IF GREATER THAN 377
2817  014574  001402                  BEQ     1$                      ;BR IF NO
2818  014576  012700  000376          MOV     #376,R0                 ;SET FOR MAX ALLOWED
2819  014602  005200          1$:     INC     R0                      ;MAKE IT 377 OR 1
2820  014604  010037  014654  2$:     MOV     R0,@#BLKLIM             ;SETUP THE BLOCK LIMIT
2821  014610  005037  014656          CLR     @#PATTRN
2822  014614  000000                  HALT                            ;OPERATOR PUTS PATTERN IN SWR<7:0>
2823  014616  022767  000176  164314  CMP     #SWREG,SWR              ;USING S/W SWITCH REG.?
2824  014624  001001                  BNE     21$                     ;NO- GET OUT
2825  014626  104405                  GTSWR                           ;LET HIM CHANGE IT
2826  014630                  21$:                                    ;CONTINUE
2827  014630  117737  164304  014656  MOVB    @SWR,@#PATTRN           ;PICK UP THE PATTERN
2828  014636  000000                  HALT                            ;SET OPERATIONAL SWITCHES
2829  014640  022767  000176  164272  CMP     #SWREG,SWR              ;USING S/W SWITCH REG.?
2830  014646  001001                  BNE     22$                     ;NO- GET OUT
2831  014650  104405                  GTSWR                           ;LET HIM CHANGE IT
2832  014652                  22$:                                    ;CONTINUE
2833  014652  000207                  RTS     PC                      ;RETURN
2834  014654  000000          BLKLIM: 0                               ;READ AND WRITE BLOCK SIZE
2835  014656  000000          PATTRN: 0                               ;PATTERN TO GO ON THE TAPE
2836
2837
2838                                  ;;**************************************************************
```

```
                                        ;       WRITE ROUTINE FOR THE MANUAL OPERATIONS
                                        ;;*************************************************************
014660  013701  014654                  WRTBLK: MOV     @#BLKLIM,R1             ;PICKUP THE BLOCK SIZE
014664  112714  000003                          MOVB    #WRITE!GO,@TACS         ;START A WRITE
014670  104413                          1$:     WAITXFER                        ;WAIT ON TRANSFER REQUEST
014672  032714  000040                          BIT     #READY,@TACS            ;DID READY SET?
014676  001010                                  BNE     3$                      ;BR IF YES
014700  005301                                  DEC     R1                      ;COUNT THIS REQUEST
014702  002403                                  BLT     2$                      ;BR IF TIME FOR ILBS
014704  113715  014656                          MOVB    @#PATTRN,@TADB          ;PUT DATA ON TAPE
014710  000767                                  BR      1$                      ;LOOP
014712  052714  000020                  2$:     BIS     #ILBS,@TACS             ;WRITE CRC AND SHUT DOWN
014716  104412                                  WAITREADY                       ;WAIT ON THE READY FLAG
014720  000207                          3$:     RTS     PC
                                        ;;*************************************************************
                                        ;       READ ROUTINE FOR THE MANUAL OPERATIONS
                                        ;;*************************************************************
014722  013702  014654                  RDBLK:  MOV     @#BLKLIM,R2             ;PICKUP THE BLOCK SIZE
014726  013700  014656                          MOV     @#PATTRN,R0             ;USE THIS DATA PATTERN TO COMPARE TO
014732  112714  000005                          MOVB    #READ!GO,@TACS          ;START A READ
014736  104413                          1$:     WAITXFER                        ;WAIT ON TRANSFER REQUEST
014740  032714  000040                          BIT     #READY,@TACS            ;IS READY SET?
014744  001012                                  BNE     3$                      ;BR IF YES
014746  005302                                  DEC     R2                      ;COUNT THIS REQUEST
014750  002405                                  BLT     2$                      ;BR IF TIME FOR ILBS
014752  011501                                  MOV     @TADB,R1                ;READ THE DATA BUFFER
014754  120001                                  CMPB    R0,R1                   ;CHECK THE DATA
014756  001767                                  BEQ     1$                      ;BR IF OK
014760  104005                                  ERROR   5                       ;BAD DATA
014762  000406                                  BR      4$                      ;GET OUT
014764  052714  000020                  2$:     BIS     #ILBS,@TACS             ;READ ILBS
014770  104412                                  WAITREADY                       ;WAIT ON READY
014772  005714                          3$:     TST     @TACS                   ;CHECK FOR ERROR
014774  100001                                  BPL     4$                      ;BR IF NONE
014776  104001                                  ERROR   1                       ;ERROR OCCURRED
015000  000207                          4$:     RTS     PC                      ;RETURN
                                        ;;*************************************************************
                                        ;       ROUTINE TO CHANGE DRIVES
015002  105777  164132                  NXTDRV: TSTB    @SWR                    ;IS SW07 ON A (1)?
015006  100416                                  BMI     3$                      ;BR IF YES
015010  005003                                  CLR     DRIVE                   ;SET DRIVE TO "A"
015012  013701  001230                          MOV     @#DRVPNT,R1             ;GET DRIVE POINTER
015016  122127  000101                          CMPB    (R1)+,#'A               ;IS IT DRIVE "A"?
015022  001402                                  BEQ     1$                      ;BR IF YES
015024  012703  000400                          MOV     #UNIT,DRIVE             ;SET DRIVE TO "B"
015030  105711                          1$:     TSTB    (R1)                    ;LAST DRIVE BEEN SELECTED
015032  001002                                  BNE     2$                      ;BR IF NO
015034  012701  001224                          MOV     #DRVKEY,R1              ;RESET DRIVE POINTER
015040  010137  001230                  2$:     MOV     R1,@#DRVPNT             ;SAVE DRIVE POINTER FOR NEXT TIME
015044  000207                          3$:     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

015046  013701  001206                  EXAM:   MOV     @#TACSL,R1              ;PICKUP THE "CONTROL & STATUS" REG. ADR.
015052  005011                                  CLR     (R1)                    ;DRIVE="A", FUNCTION="WFG"
015054  122710  000101                          CMPB    #'A,(R0)                ;EXAMINE DRIVE "A"?
015060  001402                                  BEQ     1$                      ;BR IF YES
015062  052711  000400                          BIS     #UNIT,(R1)              ;SELECT DRIVE "B"
015066  032711  000040                  1$:     BIT     #READY,(R1)             ;WAIT ON READY
015072  001775                                  BEQ     1$
015074  005711                                  TST     (R1)                    ;ANY ERROR?
015076  100024                                  BPL     4$                      ;BR IF NO
015100  032711  001000                          BIT     #OFFLINE,(R1)           ;ERROR DUE TO "OFF LINE"?
015104  001017                                  BNE     3$                      ;BR IF YES
015106  032711  010000                          BIT     #WRTLOCK,(R1)           ;ERROR DUE TO "WRITE LOCK"?
015112  001411                                  BEQ     2$                      ;BR IF NO
015114  122777  000201  164016                  CMPB    #BIT07!BIT00,@SWR       ;"READONLY" SELECTED? (RD1PAS)
015122  001412                                  BEQ     4$                      ;BR IF YES
015124  122777  000203  164006                  CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
015132  001406                                  BEQ     4$                      ;BR IF YES
015134  000403                                  BR      3$                      ;TAKE THE ERROR EXIT
015136  032711  020000                  2$:     BIT     #LEADER,(R1)            ;ERROR DUE TO "CLEAR LEADER"?
015142  001002                                  BNE     4$                      ;BR IF YES
015144  062716  000002                  3$:     ADD     #2,(SP)                 ;TAKE ERROR RETURN
015150  000207                          4$:     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
                                        ;*
015152  105767  164001          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
015156  100002                          BPL     1$              ;;BR IF YES
015160  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
015162  000407                          BR      3$              ;;LEAVE
015164  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
015166  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
015172  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
015174  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
015176  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
015200  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
015202  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
015206  000002                          RTI                     ;;RETURN
015210  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
015214  001430                          BEQ     8$
015216  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
015222  001006                          BNE     5$
015224  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
015226  104401                          TYPE                    ;;TYPE A CR AND LF
015230  001177                          $CRLF
015232  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
015236  000755                          BR      2$              ;;GET NEXT CHARACTER
015240  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
015244  126726  163706          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
015250  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
015252  016746  163676                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
015256  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
015262  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
015264  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
015270  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
015274  000770                          BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

015276  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
015302  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
015306  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
015314  001372                          BNE     9$              ;;TAB STOP
015316  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
```



```
015320  000724                          BR      2$              ;;GET NEXT CHARACTER
015322  105777  163622          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
015326  100375                          BPL     $TYPEC
015330  116677  000002  163614          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
015336  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
015344  001003                          BNE     1$              ;;BRANCH IF NO
015346  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
015352  000406                          BR      $TYPEX          ;;EXIT
015354  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
015362  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
015364  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
015366  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
015370  000207                  $TYPEX: RTS     PC

                                        .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                        ;;**************************************************************
                                        ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY.
                                        ;*CALL:
                                        ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                        ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                        ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT

015372  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
015374  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
015402  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
015404  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
015406  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
015410  104410                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
015412  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
015414  005001                          CLR     R1              ;;CLEAR DATA WORD
015416  005002                          CLR     R2
015420  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
015422  001412                          BEQ     3$              ;;IF ZERO GET OUT
015424  006301                          ASL     R1              ;;*2
015426  006102                          ROL     R2
015430  006301                          ASL     R1              ;;*4
015432  006102                          ROL     R2
015434  006301                          ASL     R1              ;;*8
015436  006102                          ROL     R2
015440  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
015444  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
015446  000764                          BR      2$              ;;LOOP
015450  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
015452  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
015456  010267  000010                  MOV     R2,$HIOCT
015462  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
015464  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
015466  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
015470  000002                          RTI                     ;;RETURN
015472  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                                        .SBTTL  TTY INPUT ROUTINE

                                        ;;**************************************************************
                                        .ENABL  LSB
```

```
3037
3038                                    ;;**************************************************************
3039                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3040                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3041                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3042                                    ;*WHEN OPERATING IN TTY FLAG MODE.
3043  015474  022767  000176  163436  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
3044  015502  001074                          BNE     15$             ;;BRANCH IF NO
3045  015504  105777  163434                  TSTB    @$TKS           ;;CHAR THERE?
3046  015510  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
3047  015512  117746  163430                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
3048  015516  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
3049  015522  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
3050  015526  001062                          BNE     15$             ;;NO, RETURN TO USER
3051  015530  126727  163400  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
3052  015536  001456                          BEQ     15$             ;;BRANCH IF YES
3053
3054  015540  104401  016221                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
3055  015544  104401  016226          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
3056  015550  016746  162422                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
3057  015554  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3058  015556  104401  016237                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
3059  015562  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
3060  015564  005046                          CLR     -(SP)           ;;THE NEW SWR
3061  015566  105777  163352          7$:     TSTB    @$TKS           ;;CHAR THERE?
3062  015572  100375                          BPL     7$              ;;IF NOT TRY AGAIN
3063
3064  015574  117746  163346                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
3065  015600  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3066
3067
3068
3069  015604  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
3070  015610  001005                          BNE     10$             ;;BRANCH IF NOT
3071  015612  104401  016214                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
3072  015616  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
3073  015622  000757                          BR      19$             ;;LET'S TRY IT AGAIN
3074
3075
3076  015624  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
3077  015630  001022                          BNE     16$             ;;BRANCH IF NO
3078  015632  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
3079  015636  001403                          BEQ     11$             ;;BRANCH IF YES
3080  015640  016677  000002  163272          MOV     2(SP),@SWR      ;;SAVE NEW SWR
3081  015646  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
3082  015652  104401  001177          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
3083  015656  126727  163253  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
3084  015664  001003                          BNE     15$             ;;BRANCH IF NOT
3085  015666  012777  000100  163250          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
3086  015674  000002          15$:    RTI                             ;;RETURN
3087  015676  004767  177420          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
3088  015702  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
3089  015706  002420                          BLT     18$             ;;BRANCH IF YES
3090  015710  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
3091  015714  003015                          BGT     18$             ;;BRANCH IF YES
3092  015716  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
```



```
3093  015722  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
3094  015726  001403                          BEQ     17$             ;;BRANCH IF YES
3095  015730  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
3096  015732  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
3097  015734  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
3098  015736  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
3099  015742  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
3100  015746  000707                          BR      7$              ;;GET THE NEXT ONE
3101  015750  104401  001176          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3102  015754  000720                          BR      20$             ;;SIMULATE CONTROL-U
3103                                  .DSABL  LSB
3104
3105
3106                                    ;;**************************************************************
3107                                    ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3108                                    ;*CALL:
3109                                    ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
3110                                    ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3111                                    ;*                              ;;WITH PARITY BIT STRIPPED OFF
3112                                    ;
3113
3114  015756  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
3115  015760  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
3116  015766  105777  163152          1$:     TSTB    @$TKS           ;;WAIT FOR
3117  015772  100375                          BPL     1$              ;;A CHARACTER
3118  015774  117766  163146  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
3119  016002  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3120  016010  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
3121  016016  001013                          BNE     3$              ;;BRANCH IF NO
3122  016020  105777  163120          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
3123  016024  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
3124  016026  117746  163114                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
3125  016032  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3126  016036  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3127  016042  001366                          BNE     2$              ;;IF NOT DISCARD IT
3128  016044  000750                          BR      1$              ;;YES, RESUME
3129  016046  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
3130  016054  002407                          BLT     4$              ;;BRANCH IF YES
3131  016056  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
3132  016064  003003                          BGT     4$              ;;BRANCH IF YES
3133  016066  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
3134  016074  000002                  4$:     RTI                     ;;GO BACK TO USER
3135                                    ;;**************************************************************
3136                                    ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3137                                    ;*CALL:
3138                                    ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3139                                    ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3140                                    ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3141
3142  016076  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3143  016100  012703  016204          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
3144  016104  022703  016214          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
3145  016110  101405                          BLOS    4$              ;;BR IF YES
3146  016112  104407                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
3147  016114  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
3148  016116  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
```

```
3149  016122  001003                        BNE     3$              ;;SKIP IF NOT
3150  016124  104401  001176          4$:   TYPE    ,$QUES          ;;TYPE A '?'
3151  016130  000763                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
3152  016132  111367  000044          3$:   MOVB    (R3),9$         ;;ECHO THE CHARACTER
3153  016136  104401  016202                TYPE    ,9$
3154  016142  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
3155  016146  001356                        BNE     2$              ;;LOOP IF NOT RETURN
3156  016150  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3157  016154  104401  001200                TYPE    ,$LF            ;;TYPE A LINE FEED
3158  016160  012603                        MOV     (SP)+,R3        ;;RESTORE R3
3159  016162  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3160  016164  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3161  016172  012766  016204  000004        MOV     #$TTYIN,4(SP)
3162  016200  000002                        RTI                     ;;RETURN
3163  016202     000                 9$:    .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
3164  016203     000                        .BYTE   0               ;;TERMINATOR
3165  016204  000010                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
3166  016214  052536  005015  000   $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
3167  016221     136  006507  000012 $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
3168  016226  005015  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
3169  016234  020075     000
3170  016237     040  047040  053505 $MNEW:  .ASCIZ  /  NEW = /
3171  016244  036440  000040
3172                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3173
3174                                  ;;**************************************************************
3175                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3176                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3177                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3178                                  ;*CALL:
3179                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3180                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
3181                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3182                                  ;*      .BYTE   M               ;;M=1 OR 0
3183                                  ;*                                      ;;1=TYPE LEADING ZEROS
3184                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
3185                                  ;*
3186                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3187                                  ;*$TYPOS OR $TYPOC
3188                                  ;*CALL:
3189                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3190                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
3191                                  ;*
3192                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3193                                  ;*CALL:
3194                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3195                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
3196
3197  016250  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
3198  016254  116667  000001  000211        MOVB    1(SP),$0FILL    ;;LOAD ZERO FILL SWITCH
3199  016262  112667  000207                MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3200  016266  062716  000002                ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
3201  016272  000406                        BR      $TYPON
3202  016274  112767  000001  000171 $TYPOC: MOVB    #1,$0FILL       ;;SET THE ZERO FILL SWITCH
3203  016302  112767  000006  000165        MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
3204  016310  112767  000005  000154 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
```



```
3205  016316  010346                        MOV     R3,-(SP)        ;;SAVE R3
3206  016320  010446                        MOV     R4,-(SP)        ;;SAVE R4
3207  016322  010546                        MOV     R5,-(SP)        ;;SAVE R5
3208  016324  116704  000145                MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
3209  016330  005404                        NEG     R4
3210  016332  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3211  016336  110467  000132                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3212  016342  116704  000125                MOVB    $0FILL,R4       ;;GET THE ZERO FILL SWITCH
3213  016346  016605  000012                MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3214  016352  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
3215  016354  006105                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
3216  016356  000404                        BR      3$              ;;GO DO MSB
3217  016360  006105                2$:     ROL     R5              ;;FORM THIS DIGIT
3218  016362  006105                        ROL     R5
3219  016364  006105                        ROL     R5
3220  016366  010503                        MOV     R5,R3
3221  016370  006103                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3222  016372  105367  000076                DECB    $OMODE          ;;TYPE THIS DIGIT?
3223  016376  100016                        BPL     7$              ;;BR IF NO
3224  016400  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
3225  016404  001002                        BNE     4$              ;;TEST FOR 0
3226  016406  005704                        TST     R4              ;;SUPPRESS THIS 0?
3227  016410  001403                        BEQ     5$              ;;BR IF YES
3228  016412  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3229  016414  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3230  016420  052703  000040          5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3231  016424  110367  000040                MOVB    R3,8$           ;;SAVE FOR TYPING
3232  016430  104401  016470                TYPE    ,8$             ;;GO TYPE THIS DIGIT
3233  016434  105367  000032          7$:   DECB    $OCNT           ;;COUNT BY 1
3234  016440  003347                        BGT     2$              ;;BR IF MORE TO DO
3235  016442  002402                        BLT     6$              ;;BR IF DONE
3236  016444  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3237  016446  000744                        BR      2$              ;;GO DO THE LAST DIGIT
3238  016450  012605                6$:     MOV     (SP)+,R5        ;;RESTORE R5
3239  016452  012604                        MOV     (SP)+,R4        ;;RESTORE R4
3240  016454  012603                        MOV     (SP)+,R3        ;;RESTORE R3
3241  016456  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3242  016464  012616                        MOV     (SP)+,(SP)
3243  016466  000002                        RTI                     ;;RETURN
3244  016470     000                 8$:    .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3245  016471     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3246  016472     000                $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3247  016473     000                $0FILL: .BYTE   0               ;;ZERO FILL SWITCH
3248  016474  000000                $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .SBTTL  TRAP DECODER

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

016476  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
016500  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
016504  005740                          TST     -(R0)           ;;BACKUP BY 2
016506  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
016510  006300                          ASL     R0              ;;POSITION FOR INDEXING
016512  016000  016532                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
016516  000200                          RTS     R0              ;;GO TO ROUTINE


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

016520  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
016522  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
016530  000002                          RTI                     ;;RESTORE THE PSW


                                .SBTTL  TRAP TABLE

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

                                ;       ROUTINE
                                ;       -------
016532  016520                  $TRPAD: .WORD   $TRAP2
016534  015152                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
016536  016274                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
016540  016250                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
016542  016310                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)

016544  015544                          $GTSWR  ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING

016546  015474                          $CKSWR  ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
016550  015756                          $RDCHR  ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
016552  016076                          $RDLIN  ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
016554  015372                          $RDOCT  ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
016556  013046                          WAIT.ON.READY   ;;CALL=WAITREADY        TRAP+12(104412) WAIT ON THE READY BIT TO
016560  013152                          WAIT.FOR.XFER   ;;CALL=WAITXFER TRAP+13(104413) WAIT ON XFER REQ.
```



```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*************************************************************
                                ;POWER DOWN ROUTINE
016562  012737  016726  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
016570  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
016576  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016600  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016602  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016604  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
016606  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
016610  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
016612  017746  162322                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
016616  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
016622  012737  016634  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
016630  000000                          HALT
016632  000776                          BR      .-2             ;;HANG UP

                                ;;*************************************************************
                                ;POWER UP ROUTINE
016634  012737  016726  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
016642  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
016646  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
016652  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
016656  001375                          BNE     1$              ;;OF  WORD
016660  012677  162254                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
016664  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
016666  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
016670  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
016672  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016674  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016676  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
016700  012737  016562  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
016706  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
016714  104401                          TYPE                    ;REPORT THE POWER FAILURE
016716  016734          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
016720  012716                          MOV     (PC)+,(SP)      ;;RESTART AT PWRST
016722  002206          $PWRAD: .WORD   PWRST                   ;;RESTART ADDRESS
016724  000002                          RTI
016726  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
016730  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
016732  000000          $SAVR6: 0                               ;;PUT THE SP HERE
016734  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
016742  000122
                                .EVEN
```

```
 3338  016744  042200  044522  042526  MSGDRV: .ASCIZ  <CRLF>"DRIVE(S)? "
 3339  016752  051450  037451  000040
 3340  016760  005015  040524  051503  MSGASK: .ASCIZ  <15><12>/TACS?/
 3341  016766  000077
 3342  016770  042526  052103  051117  MSGVEC: .ASCIZ  /VECTOR?/
 3343  016776  000077
 3344  017000  051120  047511  044522  MSGPRI: .ASCIZ  /PRIORITY?/
 3345  017006  054524  000077
 3346  017012  040524  051503  000075  MTACS:  .ASCIZ  /TACS=/
 3347  017020  052040  042101  036502  MTADB:  .ASCIZ  / TADB=/
 3348  017026     000
 3349  017027     040  042526  052103  MTAVEC: .ASCIZ  / VECTOR=/
 3350  017034  051117  000075
 3351  017040  050040  044522  051117  MTAPRI: .ASCIZ  / PRIORITY=/
 3352  017046  052111  036531     000
 3353  017053     015  047412  037513  MSGOK:  .ASCIZ  <15><12>/OK?/
 3354  017060     000
 3355  017061     131  051505  000200  MYES:   .ASCIZ  /YES/<CRLF>
 3356  017066  052123  052101  051525  EM1:    .ASCIZ  /STATUS PROBLEM/
 3357  017074  050040  047522  046102
 3358  017102  046505     000
 3359  017105     042  042522  042101  EM2:    .ASCIZ  /"READY" FAILED TO SET/
 3360  017112  021131  043040  044501
 3361  017120  042514  020104  047524
 3362  017126  051440  052105     000
 3363  017133     042  051124  047101  EM3:    .ASCIZ  /"TRANSFER REQUEST" FAILED TO SET/
 3364  017140  043123  051105  051040
 3365  017146  050505  042525  052123
 3366  017154  020042  040506  046111
 3367  017162  042105  052040  020117
 3368  017170  042523  000124
 3369  017174  044124  020105  051127  EM4:    .ASCIZ  /THE WRONG FLAG SET/
 3370  017202  047117  020107  046106
 3371  017210  043501  051440  052105
 3372  017216     000
 3373  017217     104  052101  020101  EM5:    .ASCIZ  /DATA PROBLEM/
 3374  017224  051120  041117  042514
 3375  017232  000115
 3376  017234  047111  042524  051122  EM6:    .ASCIZ  /INTERRUPT PROBLEM/
 3377  017242  050125  020124  051120
 3378  017250  041117  042514  000115
 3379  017256  041520  020040  020040  DH1:    .ASCIZ  /PC      TACS/
 3380  017264  020040  040524  051503
 3381  017272     000
 3382  017273     120  020103  020040  DH2:    .ASCIZ  /PC      TACS    WAIT ADDRESS/
 3383  017300  020040  052040  041501
 3384  017306  020123  020040  053440
 3385  017314  044501  020124  042101
 3386  017322  051104  051505  000123
 3387  017330  041520  020040  020040  DH5:    .ASCIZ  /PC      TACS    EXPECT  RCV'D/
 3388  017336  020040  040524  051503
 3389  017344  020040  020040  054105
 3390  017352  042520  052103  020040
 3391  017360  041522  023526  000104
 3392  017366  041520  020040  020040  DH6:    .ASCIZ  /PC      TACS    BR LEVEL/
 3393  017374  020040  040524  051503
```



```
 3394  017402  020040  020040  051102
 3395  017410  046040  053105  046105
 3396  017416     000
 3397          017420                          .EVEN
 3398  017420  001116  001162  000000  DT1:    .WORD   $ERRPC,$REG0,0
 3399  017426  001116  001162  001202  DT2:    .WORD   $ERRPC,$REG0,SAVPC,0
 3400  017434  000000
 3401  017436  001116  001162  001124  DT5:    .WORD   $ERRPC,$REG0,$GDDAT,$BDDAT,0
 3402  017444  001126  000000
 3403  017450  001116  001162  001124  DT6:    .WORD   $ERRPC,$REG0,$GDDAT,0
 3404  017456  000000
 3405  017460  001116  001206  000000  DT201:  .WORD   $ERRPC,TACSL,0
 3406
 3407  017466  001116  000000          DT202:  .WORD   $ERRPC,0
 3408
 3409  017472  040524  030461  043040  EM201:  .ASCIZ  "TA11 FAILED TO RESPOND"
 3410  017500  044501  042514  020104
 3411  017506  047524  051040  051505
 3412  017514  047520  042116     000
 3413  017521     116  020117  051104  EM202:  .ASCIZ  "NO DRIVE AVAILABLE"
 3414  017526  053111  020105  053101
 3415  017534  044501  040514  046102
 3416  017542  000105
 3417  017544  041520  020040  020040  DH201:  .ASCIZ  /PC      TACS/
 3418  017552  020040  040524  051503
 3419  017560     000
 3420  017561     120  000103          DH202:  .ASCIZ  /PC/
 3421          000001                          .END
```

```
ASKADR  013364           563    2512#
ASKDRV  013254           560    2480#   2503
ASKKEY  001232           396#    651*    681
BEGINX  001740           557#    593     595      644      646
BEGIN1  001336           231     476#
BEGIN2  001370           232     483#
BEGIN3  001376           233     482     485#
BEGIN4  001404           234     487#
BGNCMN  001410           479     481     484      486      488#
BIT0  = 000001           166#
BIT00 = 000001           156#    166     208     2588     2916     2918
BIT01 = 000002           155#    165     207     2597     2918
BIT02 = 000004           154#    164     206
BIT03 = 000010           153#    163     205
BIT04 = 000020           152#    162     204
BIT05 = 000040           151#    161     203
BIT06 = 000100           150#    160     202
BIT07 = 000200           149#    159     201     2916     2918
BIT08 = 000400           148#    158     200     2298
BIT09 = 001000           147#    157     199     2306     2370
BIT1  = 000002           165#
BIT10 = 002000           146#    198    2354
BIT11 = 004000           145#    197    2313
BIT12 = 010000           144#    196
BIT13 = 020000           143#    195    2361
BIT14 = 040000           142#    194    2284     2597
BIT15 = 100000           141#    193     662
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  014654          2820*   2834#   2841    2858
BPTVEC= 000014           173#
BSBG  = 000010           186#    868    1505    1589    1620    1680    1778    1783    1814    1859    1902    1932    1967
                        2024    2163    2186
BSFG  = 000006           185#    864    1447    1562    1676    1731    1739    1928    2797
BSFGSB  014512           241    2794#   2795    2802
CHKADR  001762           559     562     581#
CHKDRV  002046           620#
CKSWR = 104406          2283    2346    2369    3287#
CR    = 000015            81#   2985    2995
CRCERR= 040000           194#   1873    1916    1992    2052
CRC.WD  013774          2028*   2038    2046    2585    2606*   2613#
CRLF  = 000200            82#    545    2956    2995    3338    3355
CURDRV  001234           397#    674*    675
DDISP = 177570            88#    365     516
DH1     017256           418    3379#
DH2     017273           424     430     436    3382#
DH201   017544           456    3417#
DH202   017561           461    3420#
DH5     017330           442    3387#
DH6     017366           448    3392#
```



```
DISPLA  001142           365#    516*    524*   2327*   2353*
DISPRE  000174           228#    524
DO.CRC  013654          2029    2578#
DRVKEY  001224           394#    395     477*    620     633     654     679    2075    2128    2481*   2486    2500    2504
                        2890
DRVPNT  001230           395#    634*    654*    665     680*   2884    2891*
DSWR  = 177570            87#    364     515
DT1     017420           419    3398#
DT2     017426           425     431     437    3399#
DT201   017460           457    3405#
DT202   017466           462    3407#
DT5     017436           443    3401#
DT6     017450           449    3403#
EMTVEC= 000030           176#    499*    500*
EM1     017066           417    3356#
EM2     017105           423    3359#
EM201   017472           455    3409#
EM202   017521           460    3413#
EM3     017133           429    3363#
EM4     017174           435    3369#
EM5     017217           441    3373#
EM6     017234           447    3376#
ERROR = 100000           193#   1538    1567    1736    1741    1780    1873    2052
ERRVEC= 000004           169#    513     514*    525*    581*    587*   2289    2290*   2292*   2295*
EXAM    015046           621     625     630    2903#
EXIT    013354          2499    2504#
FGAP  = 004000           197#   1449    1483    1538    1564    1598    1629    1660    1728    1733    1775    1780
FUNCTI= 000016           209#
FUNC0 = 000002           207#    209
FUNC1 = 000004           206#    209
FUNC2 = 000010           205#    209
GNS   = ****** U         227     544     672    3280    3281    3282    3283    3285    3287    3288    3289    3290    3291
                        3292
GO    = 000001           208#    303     306     325     328     693     710     712     731     733     771     773     782
                         784     823     825     827    1292    1295    1303    1335    1366    1368    1371    1437    1439
                        1441    1447    1467    1469    1471    1477    1479    1481    1495    1497    1499    1505    1522
                        1524    1526    1532    1534    1536    1554    1556    1562    1579    1581    1587    1589    1591
                        1593    1610    1612    1618    1620    1622    1624    1642    1644    1646    1654    1656    1658
                        1710    1712    1714    1722    1724    1726    1731    1739    1757    1759    1761    1769    1771
                        1773    1778    1783    1804    1806    1814    1816    1839    1841    1844    1859    1861    1889
                        1891    1893    1902    1904    1957    1959    1967    1969    2004    2006    2009    2024    2026
                        2086    2136    2138    2142    2152    2163    2168    2186    2193    2205    2230    2636    2639
                        2668    2691    2722    2725    2750    2756    2778    2781    2797    2842    2860
GTSWR = 104405           539     687    2812    2825    2831   3285#
HERE    001740           530     532     543     545#
HGHTIM  013124           714    2423*   2437#
HT    = 000011            79#   2954    2995
ILBS  = 000020           204#    780     806    1445    1475    1503    1530    1560    1585    1616    1652    1720    1767
                        1812    1828    1854    1871    1900    1911    1965    1987    2019    2040    2158    2181    2218
                        2850    2871
INT.EN= 000100           202#    922     970    1018    1066    1114    1162    1210    1258    1303    1335    1398
IOTVEC= 000020           174#    497*    498*
ITEMS2  001316           453#
LEADER= 020000           195#   1455    1510    1538    1567    1744    1780    1788    2641    2672    2695    2727    2730
                        2754    2758    2783    2799    2921
LF    = 000012            80#   2989    2995
```

```
LOOP    013300          2487#   2501
LOOP1   000500           291#    292
LOOP2   000600           301#    312
LOOP3   000700           323#    334
MANUAL  002166           640     648#
MAXCNT  013150           662*    714*    715*    716*    717*    2423    2444#   2451
MSGASK  016760          2513    3340#
MSGDRV  016744          2480    3338#
MSGID   001712           533     545#    653
MSGOK   017053          2560    3353#
MSGPRI  017000          2536    3344#
MSGVEC  016770          2527    3342#
MTACS   017012          2548    3346#
MTADB   017020          2551    3347#
MTAPRI  017040          2557    3351#
MTAVEC  017027          2554    3349#
MYES    017061          2569    3355#
NEXT    013334          2490    2494    2498#
NOTA    013312          2488    2491#
NOTB    013324          2492    2495#
NOTLGL  013346          2485    2496    2502#   2505
NXTDRV  015002          2634    2666    2689    2720    2748    2776    2881#
OFFLIN= 001000           199#   2912
OK      002200           649     651#
PATTRN  014656          2821*   2827*   2835#   2848    2859
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      657*    914*    923*    926*    962*    971*    974*    1010*   1019*   1022*   1058*
                        1067*   1070*   1106*   1115*   1118*   1154*   1163*   1166*   1202*   1211*   1214*   1250*   1259*
                        1262*   1297*   1305*   1309*   1327*   1342*   1346*   1396*   1401*   1412*   1415*
PSW   = 177776            84#
PWRST   002206           653#   3330
PWRVEC= 000024           175#    503*    504*   3297*   3298*   3307*   3313*   3325*   3326*
RDBLK   014722          2694    2753    2858#
RDCHR = 104407          2561    3146    3288#
RDLIN = 104410          2482    3010    3289#
RDOCT = 104411          2514    2528    2537    3290#
RDSUB   014150           237    2684#   2688    2696
READ  = 000004           184#    328     784     860    1593    1672    1816    1861    1904    1969    2026    2168    2205
                        2860
READY = 000040           203#    304     311     326     333     694    1336    2093    2097    2102    2149    2201    2427
                        2457    2637    2669    2692    2723    2751    2779    2844    2862    2908
RESVEC= 000010           170#
REWIND= 000016           189#    303     325     693     710     731     771     782     823     880    1292    1335    1366
                        1437    1467    1479    1495    1522    1534    1554    1579    1610    1642    1656    1692    1710
                        1724    1757    1771    1804    1839    1889    1944    1957    2004    2086    2096    2101    2136
                        2142    2193    2230    2636    2668    2691    2722    2750    2778
RGPBLK  014332           239    2743#   2747    2755    2759
```



```
SAVPC   001202           385#   2424*   2425*   2441    2452*   2453*   2469    3399
SAVPS   001204           386#   2426*   2440    2454*   2468
SETBUF  014544          2660    2714    2808#
SFBG  = 000014           188#    876    1536    1591    1622    1624    1688    1940    2756
SFFG  = 000012           187#    872    1481    1658    1684    1726    1773    1936    2781
SFFGSB  014426           240    2771#   2775    2786
STACK = 001100            74#    495    2629    2661    2684    2715    2743    2771
START   002220           652     655#   2264
STKLMT= 177774            85#
SWR     001140           291     309     364#    493     515*    517     523*    537     685    2284    2298    2300    2306
                        2313    2354    2361    2366    2370    2810    2814    2816    2823    2827    2829    2881    2916
                        2918    3043    3080#   3305    3318#
SWREG   000176           229#    523     537     685    2810    2823    2829    3043    3056
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           389#   2522*   2523#
TACSL   001206           388#    583     658    2519*   2522    2549    2630    2662    2685    2716    2744    2772    2903
                        3405
TADBH   001214           391#   2525*   2526#
TADBL   001212           390#    659    2521*   2525    2552    2631    2663    2686    2717    2745    2773
TAPRIO  001222           393#    929     933     977     981    1025    1029    1073    1077    1121    1125    1169    1173
                        1217    1221    1265    1269   2547*   2558
TAVEC   001216           392#    480     597*    655*    656*    918*    919*    943*    944*    966*    967*    991*    992*
                        1014*   1015*   1039*   1040*   1062*   1063*   1087*   1088*   1110*   1111*   1135*   1136*   1158*
                        1159*   1183*   1184*   1206*   1207*   1231*   1232*   1254*   1255*   1279*   1280*   1299*   1300*
                        1315*   1316*   1331*   1332*   1353*   1354*   1391*   1392*   1410*   1420*   1421*   2533*   2535*
                        2555
TBITVE= 000014           171#
TDRV    002310           667     669#
TIMERR= 002000           198#    759     811     849
TKVEC = 000060           178#
TPVEC = 000064           179#
```

```
TRAPVE= 000034          177#    501*    502*
TRTVEC= 000014          172#
TR,REQ= 000200          201#   1336
TST1    002436          691     704#
TST10   004150         1053#
TST11   004352         1101#
TST12   004554         1149#
TST13   004756         1197#
TST14   005160         1245#
TST15   005362         1286#
TST16   005554         1322#
TST17   005750         1360#
TST2    002516          707     725#
TST20   006246         1431#
TST21   006370         1434    1456    1461#
TST22   006510         1464    1484    1489#
TST23   006622         1492    1511    1516#
TST24   006742         1519    1539    1548#
TST25   007050         1551    1568    1573#
TST26   007176         1576    1599    1604#
TST27   007324         1607    1630    1635#
TST3    002656          728     760     765#
TST30   007574         1638    1694    1703#
TST31   010000         1706    1745    1750#
TST32   010172         1753    1789    1798#
TST33   010330         1801    1833#
TST34   010516         1836    1874    1883#
TST35   010776         1886    1946    1951#
TST36   011160         1954    1993    1998#
TST37   011412         2001    2053    2071#
TST4    003062          768     812     817#
TST40   011552         2074    2076    2124#
TST41   012134         2127    2129    2221    2226#
TST5    003342          820     882     909#
TST6    003544          957#
TST7    003746         1005#
TYPE  = 104401          533     653     670     675     676    2254    2356    2364    2386    2395    2397    2400    2402
                       2406    2414    2480    2502    2513    2527    2536    2548    2551    2554    2557    2560    2567
                       2569    2959    3054    3055    3058    3071    3082    3101    3150    3153    3157    3232    3280#
                       3327
TYPERR  012730          2363   2386#
TYPOC = 104402          2411    2550    2553    2556    2559    3057    3281#
TYPON = 104404          3283#
TYPOS = 104403          3282#
UNIT  = 000400           200#    668    2079    2080    2132    2133    2887    2907
WAITRE= 104412           711     713     732     755     772     781     783     807     824     826     845    1293    1296
                        1330    1367    1369    1409    1438    1440    1446    1448    1468    1470    1476    1478    1480
                        1482    1496    1498    1504    1506    1523    1525    1531    1533    1535    1537    1555    1561
                        1563    1580    1586    1588    1590    1592    1594    1611    1617    1619    1621    1623    1625
                        1641    1643    1645    1653    1655    1657    1659    1709    1711    1713    1721    1723    1725
                        1727    1732    1740    1756    1758    1760    1768    1770    1772    1774    1779    1784    1805
                        1813    1815    1829    1840    1842    1855    1860    1872    1890    1892    1901    1903    1912
                        1958    1966    1968    1988    2005    2007    2020    2025    2047    2083    2085    2105    2137
                        2139    2141    2159    2164    2182    2187    2192    2204    2219    2231    2640    2726    2757
                        2782    2798    2851    2872    3291#
WAITXF= 104413           734     775     777     785     828    1304    1372    1442    1444    1472    1474    1500    1502
```



```
                        1527    1529    1557    1559    1582    1584    1613    1615    1647    1649    1651    1715    1717
                        1719    1762    1764    1766    1807    1809    1811    1817    1823    1845    1847    1863    1870
                        1895    1906    1960    1962    1964    1970    1975    1979    1983    2010    2012    2030    2039
                        2153    2155    2157    2169    2175    2180    2206    2212    2217    2843    2861    3292#
WAIT,F  013152          2449#   3292
WAIT,O  013046          2421#   3291
WAIT1   013076          2427#   2435    2438
WAIT2   013116          2422*   2434#
WAIT3   013132          2428    2432    2440#
WAIT4   013114          2430    2433#
WFG   = 000000           182#    712     825     852    1295    1368    1439    1469    1477    1497    1524    1532    1587
                        1618    1644    1654    1664    1712    1722    1759    1769    1841    1891    1920    2006    2138
                        2639    2725
WFGSUB  013776           235    2629#   2633    2644
WGBLOP  014234          2715#   2719    2733
WGPBLK  014230           238    2714#
WLOOP   014066          2661#   2665    2675
WRITE = 000002           183#    306     733     773     827     856    1303    1371    1441    1471    1499    1526    1556
                        1581    1612    1646    1668    1714    1761    1806    1844    1893    1924    1959    2009    2152
                        2842
WRTBLK  014660          2671    2729    2841#
WRTLOC= 010000           196#   2914
WRTSUB  014062           236    2660#
$AUTOB  001134           361#    541*   3051    3172
$BDADR  001122           356#
$BDDAT  001126           358#   2350*   3401
$BELL   001172           380#   2356    2382
$CHARC  015366          2961*   2971*   2978    2987*   2992#
$CKSWR  015474          3043#   3287
$CMTAG  001100           344#    490     491     499     505     506     507
$CM1  = 000002           376#    377#    378#
$CM2  = 000004           376#    377#    378#
$CM3  = 000002           374#    376
$CNTLG  016221          3054    3167#
$CNTLU  016214          3071    3166#
$CRLF   001177           382#    676    2364    2382    2386    2397    2402    2406    2960    2995    3082    3166
$DOAGN  012242          2250    2256    2262#
$ENDAD  012232           531     596     647    2258#   2377
$ENDCT  012212           505    2252#
$ENDMG  012251          2254    2266#
$ENULL  012246          2265#
$EOP    012156          2242#
$EOPCT  012204           505*   2249#   2253
$ERFLG  001103           347#   2273    2302    2304    2310*   2331    2351*   2382
$ERMAX  001115           353#    508*   2304    2326*   2331
$ERROR  012536           499    2345#
$ERRPC  001116           354#   2358*   2359*   2360    2382    3398    3399    3401    3403    3405    3407
$ERRTB  001236           412#   2393
$ERTTL  001112           351#   2357*   2382
$ESCAP  001170           379#    507*    728*    768*    820*    911*    959*   1007*   1055*   1103*   1151*   1199*   1247*
                       1289*   1324*   1363*   1434*   1464*   1492*   1519*   1551*   1576*   1607*   1638*   1706*   1753*
                       1801*   1836*   1886*   1954*   2001*   2074*   2127*   2325*   2373    2375    2382
$FILLC  001156           372#   2964    2995
$FILLS  001155           371#   2995
$GDADR  001120           355#
$GDDAT  001124           357#   2349*   3401    3403
```

```
$GET42  012222          2255#
$GTSWR  015544          3055#   3285
$HD   = 000000            11
$HIOCT  015472          3027*   3032#
$ICNT   001104           348#   2317*   2318    2320*   2330
$ILLUP  016726          3297    3313    3332#
$INTAG  001135           362#   3083    3172
$ITEMB  001114           352#   2360*   2382    2388
$LF     001200           383#   2382    2995    3157    3166
$LPADR  001106           349#    509*    727*    767*    819*    910*    958*   1006*   1054*   1102*   1150*   1198*   1246*
                        1288*   1362*   1433*   1463*   1491*   1518*   1550*   1575*   1606*   1637*   1705*   1752*   1800*
                        1835*   1885*   1953*   2000*   2073*   2126*   2308*   2323*   2328    2330
$LPERR  001110           350#    510*   2308    2324*   2330    2372    2633*   2665*   2688*   2719*   2747*   2775*   2795*
$MAIL = ****** U          527     537   2323    2366    2948
$MNEW   016237          3058    3170#
$MSWR   016226          3055    3168#
$MXCNT  012534          2321    2330#
$NULL   001154           370#   2966    2995
$NWTST= 000001           701#    722#    762#    814#    906#    954#   1002#   1050#   1098#   1146#   1194#   1242#   1283#
                        1319#   1357#   1428#   1458#   1486#   1513#   1545#   1570#   1601#   1632#   1700#   1747#   1795#
                        1830#   1880#   1948#   1995#   2068#   2121#   2223#
$OCNT   016472          3204*   3233*   3246#
$OMODE  016474          3199*   3203*   3208    3211*   3222*   3248#
$OVER   012520          2285    2301    2309    2319    2327#
$PASS   001100           345#    660     706   2246*   2247*   2265    2315    2331
$POWER  016734          3328    3335#
$PWRAD  016722          3330#
$PWRDN  016562           503    3297#   3325
$PWRMG  016716          3328#
$PWRUP  016634          3307    3313#
$QUES   001176           381#   2382    2502    2567    2995    3101    3150    3166
$RDCHR  015756          3114#   3288
$RDDEC= ****** U        3291
$RDLIN  016076          3142#   3289
$RDOCT  015372          3005#   3290
$RDSZ = 000010          3135#
$REGAD  001160           374#
$REG0   001162           376#   2347*   3398    3399    3401    3403
$REG1   001164           377#   2348*
$RTNAD  012244          2264#
$R2A  = ****** U        3291
$SAVRE= ****** U        3291
$SAVR6  016732          3306*   3314    3315*   3316*   3334#
$SCOPE  012264           497    2282#
$SETUP= 000137           476#    496     497     499     501     503     505     506     507     509     531     534    2244
                        2283    2346    2369    2377    3038    3172
$STUP = 177777           476#
$SVLAD  012472          2293    2322#
$SWR  = 167400             1#     11      62      63      64      65      66      67      68     378     379     380     506
                         507     509     510     705     726     766     818     910     958    1006    1054    1102    1150
                        1198    1246    1287    1323    1361    1432    1462    1490    1517    1549    1574    1605    1636
                        1704    1751    1799    1834    1884    1952    1999    2072    2125    2227    2237    2245    2257
                        2263    2265    2274    2275    2276    2277    2278    2284    2296    2298    2299    2302    2303
                        2304    2311    2312    2313    2324    2327    2330    2337    2338    2339    2340    2341    2354
                        2361    2366    2370    2382    3331
$SWRMK= 000000            68      69    2278    2279    2300
```



```
$TIMES  001166           378#    506*    705*    726*    766*    818*   1287*   1323*   1361*   1432*   1462*   1490*   1517*
                        1549*   1574*   1605*   1636*   1704*   1751*   1799*   1834*   1884*   1952*   1999*   2072*   2125*
                        2227*   2245*   2311*   2318    2321*   2330
$TKB    001146           367#   3036    3047    3064    3118    3124
$TKS    001144           366#   3036    3045    3061    3085*   3116    3122
$TN   = 000042             1#     11     701     705#    707     722     726#    728     760     762     766#    768     812
                         814     818#    820     882     906     910#    954     958#   1002   1006#   1050   1054#   1098
                        1102#   1146   1150#   1194   1198#   1242   1246#   1283   1287#   1319   1323#   1357   1361#
                        1428   1432#   1434   1456   1458   1462#   1464   1484   1486   1490#   1492   1511   1513
                        1517#   1519   1539   1545   1549#   1551   1568   1570   1574#   1576   1599   1601   1605#
                        1607   1630   1632   1636#   1638   1694   1700   1704#   1706   1745   1747   1751#   1753
                        1789   1795   1799#   1801   1830   1834#   1836   1874   1880   1884#   1886   1946   1948
                        1952#   1954   1993   1995   1999#   2001   2053   2068   2072#   2074   2076   2121   2125#
                        2127   2129   2221   2223   2227#
$TPB    001152           369#   2984*   2995
$TPFLG  001157           373#   2942    2995
$TPS    001150           368#   2982    2995
$TRAP   016476           501    3257#
$TRAP2  016520          3268#   3279
$TRP  = 000014          3272#   3281#   3282#   3283#   3284#   3285    3286#   3287    3288#   3289#   3290#   3291#   3292#
                        3293#
$TRPAD  016532          3262    3279#
$TSTNM  001102           346#    663*   2244*   2273    2300    2322*   2327    2331    2353    2382
$TTYIN  016204          3143    3144    3161    3165#
$TYPBN= ****** U        3284
$TYPDS= ****** U        3284
$TYPE   015152          2942#   3272    3280
$TYPEC  015322          2963    2970    2977    2982#   2983    3087
$TYPEX  015370          2988    2990    2993#
$TYPOC  016274          3202#   3281
$TYPON  016310          3201    3204#   3283
$TYPOS  016250          3197#   3282
$XTSTR  012276          2287#
$$GET4= 000000          2257#
$0FILL  016473          3198*   3202*   3212    3247#
$40CAT= ****** U        2284    2363
.     = 017564           223#    227#    289#    299#    321#    343#    384     494     509     510     545#    673#   2265
                        2268    2330    2331    2382   2417#   2995    3036   3165#   3166    3172    3309    3333   3397#
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

*TABC,TABC/SOL/CRF:SYM=DZTABC.NEW
RUN-TIME: 50 39 3 SECONDS
RUN-TIME RATIO: 346/94=3.6
CORE USED:  25K  (49 PAGES)
```

digital

ENGINEERING CHANGE ORDER

ORIGINATOR O. Choate
TEL EXT 4088 DATE 17 AUG 76
LOCATION ML21-4/E10
COST CENTER NO. 301

ECO NO. MD-11-DZTAB-C1/01
SHEET 1 OF 2
DATE RECEIVED 18-Aug-76
ISSUE DATE
FINAL RELEASE

PROBLEM

Tests # 37 and # 40 of DZTABC fail when run on 11/70 with CACHE Memory enabled.

UNIT TO BE CHANGED

MD-11-DZTABC

PRODUCT AFFECTED

TA11

CORRECTION /DEPO

Install patch to diagnostic

Conditional

TYPE OF ECO

- ☐ HARDWARE
- ☒ SOFTWARE
- ☐ PURCHASE SPEC.

BREAK-IN/EFFECTIVITY

Immediately

FIELD SERVICE AFFECTED

☐ YES ☐ NO

F D, P, R DISTR.

L.O.U. CODE

WHERE USED

QUICK-CHECK

TEST

	DOCUMENT/PART NO.	OLD REV	NEW REV	DESCRIPTION OF CHANGE/DISPOSITION OF MATERIAL
1	MD-11-DZTAB	C	N/A	No document change
2	MD-11-DZTAB	C	N/A	CHANGE LOCATION FROM TO

CHANGE LOCATION	FROM	TO
11500	6301	4737
11502	1001	17600
11506	32714	240
11510	40	240
11512	1372	240
11652	6301	4737
11654	1003	17600
11660	32714	240
11662	40	240

APPROVAL SIGNATURE (TYPE NAME and SIGN)

PROJECT ENGR. O. Choate/

ENG. MGR. M. Horovitz/

TEL. EXT. 4088 COST CENTER NO. 301

DISC. PROJ. NO. V20-07846 DATE 8/18/76

COORD. NO. REF. NO.

REVIEW SIGNATURES (SEE INSTRUCTIONS FOR APPLIC.)

FIELD SERVICE

DIAGNOSTIC ENGR.

digital

ENGINEERING
CHANGE ORDER

CONTINUATION

SHEET

ECO NO. DZTAB-C1/01

SHEET 2 OF 2

ITEM NO.	DOCUMENT/PART NO.	OLD REV	NEW REV	DISP CODE	DESCRIPTION OF CHANGE
2	(CONTINUED)				CHANGE LOCATION FROM TO
					11664 1372 240
					17600 6301
					17602 1376
					17604 32714
					17606 40
					17610 1002
					17612 062716
					17614 2
					17616 207
3	MD-11-DZTAB	C	N/A	-	NO OTHER PATCHES REQUIRED