

IDENTIFICATION

-----7----

PRODUCT CODE: MAINDEC-11-DZTAC-C-D
PRODUCT NAME: TA11 MANUAL INTERVENTION TEST
PRODUCT DATE: JULY, 1976
MAINTAINER: DIAGNOSTIC ENGINEERING

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 MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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, DIGITAL EQUIPMENT CORPORATION

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
MAINDEC-11-DZTAB

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 CONTINUOUS BLOCKS OF DATA
230 READ CONTINUOUS 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 TYPE INSTRUCTIONS ON THE TTY FOR THE OPERATOR TO FOLLOW. AFTER THE OPERATOR HAS PERFORMED THE REQUIRED OPERATION HE WILL PRESS "CARRIAGE RETURN" AND THE PROGRAM WILL CONTINUE.

*** NOTE: IF USING THE SOFTWARE SWITCH REGISTER THE PROGRAM WILL TYPE "SWR=XXXXXX NEW=" AFTER TYPING THE NAME OF THE PROGRAM.

4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF DRIVES "A" AND "B" TO BE TESTED.

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

STARTING THE PROGRAM AT 204, 210, OR 214 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED.

THE PROGRAM WILL TYPE "DRIVE(S)?".

EITHER OR BOTH DRIVES CAN BE SELECTED BY TYPING "A" AND/OR "B" FOLLOWED BY A CARRIAGE RETURN.

4.3.1.1 DRIVE SELECTION EXAMPLES

DRIVE(S)? A,B
DRIVE(S)? AB
DRIVE(S)? B,A
DRIVE(S)? B

4.3.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE "CONTROL AND STATUS" AND "DATA BUFFER" REGISTER ADDRESSES, THE VECTOR ADDRESS AND THE PRIORITY LEVEL.

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

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
 2. VECTOR ADDRESS
 3. PRIORITY LEVEL
- AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONDES 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.

5.2.2 TRAPCATCHER

A ".+2" - "HALT" SEQUENCE IS REPEATED FROM LOC. 0 TO LOC. 776 TO CATCH ANY UNEXPECTED TRAPS. THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2.

5.2.3 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6.)
*** THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
*** IF THE PROCESSOR HALTS (BIT 15=1), OPERATOR CAN RESET S/W SWITCH REGISTER BY HITTING A "CONTROL G" <"G"> BEFORE HITTING CONTINUE.

5.2.4 TRAP

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION. FOLLOWING IS THE CALLS USED AND THE 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

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 STYPEC

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 ASKADR

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

5.2.5.5 ASKQUES

THIS ROUTINE WILL TYPE DIRECTIONS ON THE TTY AND WAITS
FOR A RESPONSE OF A "Y", "N" OR "CARRIAGE RETURN.

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 MONO". THE THRESHOLD POT" AND THE "TAPE BLANK MONO".

5.2.6.6 SFFGSB

SPACE FORWARD FILE GAP FROM "BUT" TO "EOT"
START AT 244
THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
SPACE FORWARD (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 BSFGS

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

IS OPERATOR DEPENDENT BUT SHOULD TAKE NO MORE THAN 2 MINUTES.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

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

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 THROUGH OUT THE PROGRAM FOR "DRIVE", "TACS" AND "TADB"

9. PROGRAM DESCRIPTION

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

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

1. TEST "OFFLINE" WITH TU60 POWER OFF
2. TEST "OFFLINE" WITH CASSETTE REMOVED
3. TEST "WRITE LOCK" CIRCUITRY

12	GENERAL INFORMATION
60	OPERATIONAL SWITCH SETTINGS
73	BASIC DEFINITIONS
183	TA11 DEFINITIONS
224	STARTING ADDRESSES
225	TRAP CATCHER
234	STARTING ADDRESS(ES)
246	TOGGLE IN ROUTINES
293	LOAD SWITCH REGISTER INTO TACS
301	WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
327	READ FROM BOT TO EOT
344	COMMON TAGS
405	ERROR POINTER TABLE
466	START OF TEST
491	INITIALIZE THE COMMON TAGS
529	TYPE PROGRAM NAME
536	GET VALUE FOR SOFTWARE SWITCH REGISTER
702	T1 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
720	*****MANUAL INTERVENTION*****
721	T2 SETUP FOR MANUAL INTERVENTION
737	T3 SETUP FOR POWER DOWN TU60 TEST
753	T4 TEST "OFFLINE" WHEN TU60 IS POWERED DOWN
798	T5 POWER UP THE TU60
813	T6 TEST "OFFLINE" WHEN DRIVE IS EMPTY
829	T7 PUT DRIVE "ONLINE"
844	T10 PUT DATA ON TAPE FOR WRITE LOCK TEST
896	T11 TEST "WRITE LOCK" WHEN AT CLEAR LEADER
1000	T12 TEST "WRITE LOCK" WITH "CPC ERROR"
1101	T13 TEST "WRITE LOCK" WITHOUT ANY ERRORS
1194	T14 TEST "WRITE LOCK" WITH FILE GAP=1
1295	T15 TRY WRITING ON TAPE WHEN "WRITE LOCKED"
1341	T16 TRY "WFG" WHEN WRITE LOCKED
1372	T17 TEST "WFG" ON CLEAR LEADER AND WRITE LOCKED
1393	T20 TEST "WRITE" WITH WRITE LOCK ON AND AT CLEAR LEADER
1412	T21 TEST "WRITE ENABLE"
1430	T22 END OF TEST CODE
1440	END OF PASS ROUTINE
1476	SCOPE HANDLER ROUTINE
1540	ERROR HANDLER ROUTINE
1592	ERROR TIMEOUT ROUTINE
1628	ROUTINE TO WAIT ON THE READY BIT TO SET
1657	ROUTINE TO WAIT ON TRANSFER REQUEST
1686	ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
1721	ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
1786	TYPE DIRECTIONS TO OPERATOR AND WAIT FOR RESPONSE
1826	***** MANUAL ADJUSTMENT ROUTINES *****
1836	WRITE FILE GAPS FROM "BOT" TO "EOT"
1861	WRITE CONTINUOUS BLOCKS OF DATA
1893	READ CONTINUOUS BLOCKS OF DATA
1916	WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
1953	READ A BLOCK OF DATA AND A FILE GAP
1981	SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2008	BACK SPACE FILE GAP
2025	SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2059	WRITE ROUTINE FOR THE MANUAL OPERATIONS
2077	READ ROUTINE FOR THE MANUAL OPERATIONS

2102	ROUTINE TO CHANGE DRIVES
2118	ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2148	TYPE ROUTINE
2218	TTY INPUT ROUTINE
2357	READ AN OCTAL NUMBER FROM THE TTY
2395	BINARY TO OCIAL (ASCII) AND TYPE
2472	TRAP DECODER
2495	TRAP TABLE
2516	POWER DOWN AND UP ROUTINES

```

1 .TITLE TA11 MANUAL INTERVENTION TEST MAINDEC-11-DZTAC-C
2 ;*COPYRIGHT (C) 1973,1976
3 ;*DIGITAL EQUIPMENT CORP.
4 ;*MAYNARD, MASS. 01754
5 ;*
6 ;*PROGRAM BY JIM LACEY
7 ;*
8 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
9 ;*PACKAGE (MAINDEC-11-DZGAC-C1),MAR 24, 1976.
10 ;*
11 ;*****
12 ;*****
13 ;*****
14 .REM!

```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

```

21 -----
22 777500 TACS CONTROL AND STATUS REGISTER
23 777502 TADB DATA BUFFER REGISTER
24 260 IAVEC INTERRUPT VECTOR
25

```

1ACS REGISTER DESCRIPTION

```

26 -----
27
28
29
30 BIT NAME INIT READ AND/OR WRITE?
31 --- ----
32
33 15 EPROR ? READ ONLY
34 14 BLOCK CHECK ERROR 0 READ ONLY
35 13 CLEAR LEADER ? READ ONLY
36 12 WRITE LOCK ? READ ONLY
37 11 FILE GAP 0 READ ONLY
38 10 TIMING ERROR 0 READ ONLY
39 09 OFF LINE ? READ ONLY
40 08 UNIT SELECT 0 READ/WRITE
41 07 TPANSFER REQUEST 0 READ ONLY
42 06 INTERRUPT ENABLE 0 READ/WRITE
43 05 READY 1 READ ONLY
44 04 ILBS 0 READ/WRITE
45 03 FUNCTION BIT 02 0 READ/WRITE
46 02 FUNCTION BIT 01 0 READ/WRITE
47 01 FUNCTION BIT 00 0 READ/WRITE
48 0=WRITE-FILE-GAP
49 1=WRITE
50 2=READ
51 3=BACK SPACE FILE GAP
52 4=BACK SPACE BLOCK GAP
53 5=SPACE FORWARD FILE GAP
54 6=SPACE FORWARD BLOCK GAP
55 7=REWIND
56 00 GO BIT 0 WRITE ONLY!

```

GENERAL INFORMATION

```

57 ;*****
58 .SBTTL OPERATIONAL SWITCH SETTINGS
59 ;*
60 ;* SWITCH USE
61 ;* -----
62 ;* 15 HALT ON ERROR
63 ;* 14 LOOP ON TEST
64 ;* 13 INHIBIT ERROR TYPEOUTS
65 ;* 11 INHIBIT ITERATIONS
66 ;* 10 BELL ON ERROR
67 ;* 9 LOOP ON ERROR
68 ;* 8 LOOP ON TEST IN SWR<7:0>
69 ;* 7 LOCK ON CURRENT DRIVE (ONLY VALID WITH MANUAL LOOPING)
70 ;*****
71 .SBTTL BASIC DEFINITIONS
72
73 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
74 001100 STACK= 1100
75 .EQUIV EMT,EPROR ;;BASIC DEFINITION OF ERROR CALL
76 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
77
78 ;*MISCELLANEOUS DEFINITIONS
79 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
80 000012 LF= 12 ;;CODE FOR LINE FEED
81 000015 CF= 15 ;;CODE FOR CARRIAGE RETURN
82 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
83 177776 PS= 177776 ;;PROCESSOR STATUS WORD
84 .EQUIV PS,PSW
85 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
86 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
87 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
88 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
89
90 ;*GENERAL PURPOSE REGISTER DEFINITIONS
91 000000 R0= %0 ;;GENERAL REGISTER
92 000001 R1= %1 ;;GENERAL REGISTER
93 000002 R2= %2 ;;GENERAL REGISTER
94 000003 R3= %3 ;;GENERAL REGISTER
95 000004 R4= %4 ;;GENERAL REGISTER
96 000005 R5= %5 ;;GENERAL REGISTER
97 000006 R6= %6 ;;GENERAL REGISTER
98 000007 R7= %7 ;;GENERAL REGISTER
99 .EQUIV R6,SP ;;STACK POINTER
100 .EQUIV R7,PC ;;PROGRAM COUNTER
101
102 ;*PRIORITY LEVEL DEFINITIONS
103 000000 PR0= 0 ;;PRIORITY LEVEL 0
104 000040 PR1= 40 ;;PRIORITY LEVEL 1
105 000100 PR2= 100 ;;PRIORITY LEVEL 2
106 000140 PR3= 140 ;;PRIORITY LEVEL 3
107 000200 PR4= 200 ;;PRIORITY LEVEL 4
108 000240 PR5= 240 ;;PRIORITY LEVEL 5
109 000300 PR6= 300 ;;PRIORITY LEVEL 6
110 000340 PR7= 340 ;;PRIORITY LEVEL 7
111
112 ;*SWITCH REGISTER SWITCH DEFINITIONS

```

```

113      100000      SW15= 100000
114      040000      SW14= 400000
115      020000      SW13= 200000
116      010000      SW12= 100000
117      004000      SW11= 4000
118      002000      SW10= 2000
119      001000      SW09= 1000
120      000400      SW08= 400
121      000200      SW07= 200
122      000100      SW06= 100
123      000040      SW05= 40
124      000020      SW04= 20
125      000010      SW03= 10
126      000004      SW02= 4
127      000002      SW01= 2
128      000001      SW00= 1
129
130      .EQUIV SW09,SW9
131      .EQUIV SW08,SW8
132      .EQUIV SW07,SW7
133      .EQUIV SW06,SW6
134      .EQUIV SW05,SW5
135      .EQUIV SW04,SW4
136      .EQUIV SW03,SW3
137      .EQUIV SW02,SW2
138      .EQUIV SW01,SW1
139      .EQUIV SW00,SW0
140
141      100000      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
142      040000      BIT15= 100000
143      020000      BIT14= 400000
144      010000      BIT13= 200000
145      004000      BIT12= 100000
146      002000      BIT11= 4000
147      001000      BIT10= 2000
148      000400      BIT09= 1000
149      000200      BIT08= 400
150      000100      BIT07= 200
151      000040      BIT06= 100
152      000020      BIT05= 40
153      000010      BIT04= 20
154      000004      BIT03= 10
155      000002      BIT02= 4
156      000001      BIT01= 2
157
158      .EQUIV BIT09,BIT9
159      .EQUIV BIT08,BIT8
160      .EQUIV BIT07,BIT7
161      .EQUIV BIT06,BIT6
162      .EQUIV BIT05,BIT5
163      .EQUIV BIT04,BIT4
164      .EQUIV BIT03,BIT3
165      .EQUIV BIT02,BIT2
166      .EQUIV BIT01,BIT1
167      .EQUIV BIT00,BIT0
168
169      ;*BASIC "CPU" TRAP VECTOR ADDRESSES

```

```

169      000004      FRRVEC= 4      ;;TIME OUT AND OTHER ERRORS
170      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
171      000014      TRITVEC=14    ;;"T" BIT
172      000014      TPTVEC= 14    ;;TRACE TRAP
173      000014      BPTVEC= 14    ;;BREAKPOINT TRAP (BPT)
174      000020      IOTVEC= 20    ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
175      000024      PWRVEC= 24    ;;POWER FAIL
176      000030      EMTVEC= 30    ;;EMULATOR TRAP (EMT) **ERROR**
177      000034      TRAPVEC=34   ;;"TRAP" TRAP
178      000060      TKVEC= 60     ;;TTY KEYBOARD VECTOR
179      000064      TPVEC= 64     ;;TTY PRINTER VECTOR
180      000240      PIRQVEC=240  ;;PROGRAM INTERRUPT REQUEST VECTOR

```

```

181                                     ;TA11 FUNCTIONS
182 000000 WFG= 0 ;WRITE FILE GAP FUNCTION
183 000002 WRITE= 2 ;WRITE FUNCTION
184 000004 READ= 4 ;READ FUNCTION
185 000006 BSFG= 6 ;BACK SPACE FILE GAP FUNCTION
186 000010 BSBG= 10 ;BACK SPACE BLOCK GAP FUNCTION
187 000012 SFFG= 12 ;SPACE FWD FILE GAP FUNCTION
188 000014 SFBG= 14 ;SPACE FWD BLOCK GAP FUNCTION
189 000016 REWIND= 16 ;REWIND FUNCTION
190 ;*****
191
192 ;TA11 BIT ASSIGNMENT
193 100000 ERROR= BIT15
194 040000 CRCERR= BIT14
195 020000 LEADER= BIT13
196 010000 WRTLOCK=BIT12
197 004000 FGAP= BIT11
198 002000 TIMERR= BIT10
199 001000 OFFLINE=BIT09
200 000400 UNIT= BIT08
201 000200 TP,REQ= BIT07
202 000100 INT,EN= BIT06
203 000040 READY= BIT05
204 000020 ILBS= BIT04
205 000010 FUNC2= BIT03
206 000004 FUNC1= BIT02
207 000002 FUNC0= BIT01
208 000001 GO= BIT00
209 000016 FUNCTION= FUNC2+FUNC1+FUNC0
210 ;////////////////////////////////////
211 ;////////////////////////////////////
212 ;////////////////////////////////////
213 ;////////////////////////////////////
214
215 ;SPECIAL REGISTERS
216 000003 DRIVE= %3 ;R3 CONTAINS THE DRIVE UNDER TEST
217 000004 TACS= %4 ;R4 IS USED AS A POINTER TO THE TACS REGISTER
218 000005 TABD= %5 ;R5 IS USED AS A POINTER TO THE TABD REGISTER.
219 ;////////////////////////////////////
220 ;////////////////////////////////////
    
```

```

221 .SBTTL TRAP CATCHER
222
223 000000 .=0
224 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
225 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
226 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
227
228 000174 000000 .=174
229 000176 000000 DISPRG: ,WORD 0 ;SOFTWARE DISPLAY REGISTER
230 SWREG: ,WORD 0 ;SOFTWARE SWITCH REGISTER
231 .SBTTL STARTING ADDRESS(ES)
232 000200 000137 001326 JMP ##BEGIN1 ;JUMP TO STARTING ADDRESS OF PROGRAM
233 000204 000137 001360 JMP ##BEGIN2 ;SELECT DRIVE(S) BEFORE STARTING TEST
234 000210 000137 001366 JMP ##BEGIN3 ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
235 000214 000137 001730 JMP ##BEGINX ;SETUP FOR MANUAL LOOPING
236 000220 000137 007604 JMP ##WFGSUB ;WRITE FILE GAP FROM BOT TO EOT
237 000224 000137 007670 JMP ##WRDSUB ;WRITE CONTINUOUS BLOCKS OF DATA
238 000230 000137 007756 JMP ##RDSUB ;READ CONTINUOUS BLOCKS OF DATA
239 000234 000137 010036 JMP ##WGPBLK ;WRITE FILE GAP AND A BLOCK OF DATA
240 000240 000137 010140 JMP ##RGPBLK ;READ BLOCK OF DATA AND INTO A FILE GAP
241 000244 000137 010234 JMP ##SFFGSB ;SPACE FWD FILE GAP FROM BOT TO EOT
242 000250 000137 010320 JMP ##BSFGSB ;BACK SPACE FILE GAPS
    
```

```

242
243 ;////////////////////
244 ;////////////////////
245
246 ;THE FOLLOWING ROUTINES CAN BE TOGGLED IN.
247
248 ;////////////////////
249
250
251 .REM 1
252
253
254 THE FOLLOWING ROUTINES (LOOP1, LOOP2, & LOOP3) CAN BE TOGGLED
255 IN WHEN IT IS IMPOSSIBLE TO LOAD THE DIAGNOSTICS
256
257 NOTE: BEFORE USING THESE ROUTINES INSURE THAT R3,R4,& R5
258 ARE SETUP PROPERLY.
259 ** NOTE: IF USING SOFTWARE SWITCH REGISTER THE
260 LOCATION SWR (=1140) MUST CONTAIN
261 ADDRESS "SWREG" (=176).
262 *** PLACE VALUE INTO 176 ***
263 ** REGISTERS 3, 4, AND 5 **
264 ** MUST BE SETUP VIA MOVE INSTRUCTIONS **
265
266 R3= 0 IF USING DRIVE A
267 400 IF USING DRIVE B
268
269 R4= TA11 STATUS REG ADDRESS (TACS 177500)
270
271 R5= TA11 DATA BUFFER ADDRESS (TADB 177502)
272
273 LOOP1 WILL LOAD THE SWITCH REGISTER INTO THE TACS.
274
275 LOOP2 WILL WRITE THE CONTENTS OF THE SWITCH REGISTER
276 ALL THE WAY TO END-OF-TAPE(EOT).
277
278 LOOP3 WILL READ TO EOT. DATA WILL GO TO R0.
279
280 NOTE: LOOP2 AND LOOP3 WILL REWIND WHEN EOT IS REACHED AND
281 THEN START OVER.
282
283
284 ;*****
285 ;LOAD SWITCH REGISTER INTO THE TACS
286 ;*****
287
288
289 000500 . =500
290
291 000500 017714 000434 LOOP1: MOV @SWR,@TACS ;LOAD TACS
292 000504 000775 BR LOOP1 ;LOOP
293
294

```

```

295 ;*****
296 ;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
297 ;*****
298
299 000600 . =600
300
301 000600 000005 LOOP2: RESET ;CLEAR ALL FLAGS
302 000602 010314 MOV DRIVE,@TACS ;SELECT DRIVE
303 000604 112714 000017 MOVB #REWINDGO,@TACS ;GO TO BOT
304 000610 032714 000040 1s: BIT #READY,@TACS ;WAIT TILL READY COMES UP
305 000614 001775 BEQ 1s
306 000616 112714 000003 MOVB #WRITE!GO,@TACS ;START A WRITE
307 000622 105714 2s: TSTB @TACS ;CHECK FOR TRANSFER REQUEST
308 000624 100003 BPL 3s ;BR IF NOT SET
309 000626 017715 000306 MOV @SWR,@TADB ;SEND DATA TO TA11
310 000632 000773 BR 2s ;LOOP
311 000634 032714 000040 3s: BIT #READY,@TACS ;DID READY SET?
312 000640 001357 BNE LOOP2 ;START OVER IF YES
313 000642 000767 BR 2s ;LOOP
314
315 ;*****
316 ;READ FROM BOT TO EOT
317 ;*****
318
319
320
321 000700 . =700
322
323 000700 000005 LOOP3: RESET ;CLEAR ALL FLAGS
324 000702 010314 MOV DRIVE,@TACS ;SELECT DRIVE
325 000704 112714 000017 MOVB #REWINDGO,@TACS ;START A REWIND
326 000710 032714 000040 1s: BIT #READY,@TACS ;WAIT ON REWIND TO FINISH
327 000714 001775 BEQ 1s
328 000716 112714 000005 MOVB #READ!GO,@TACS ;START A READ
329 000722 105714 2s: TSTB @TACS ;CHECK TRANSFER REQ
330 000724 100002 BPL 3s ;BR IF NOT SET
331 000726 011500 MOV @TADB,R0 ;PICKUP THE DATA
332 000730 000774 BR 2s ;LOOP
333 000732 032714 000040 3s: BIT #READY,@TACS ;CHECK READY
334 000736 001360 BNE LOOP3 ;START OVER
335 000740 000770 BR 2s ;LOOP

```

```

336                                     ,SBTTL COMMON TAGS
337
338                                     ;*****
339                                     ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
340                                     ;*USED IN THE PROGRAM.
341
342                                     . =1100
343 001100                                SCMTAG: .WORD 0                ;START OF COMMON TAGS
344 001100 000000                        SPASS: .WORD 0                ;CONTAINS PASS COUNT
345 001102 000                                STSTNM: .BYTE 0              ;CONTAINS THE TEST NUMBER
346 001103 000                                SERFLG: .BYTE 0              ;CONTAINS ERROR FLAG
347 001104 000000                        SICNT: .WORD 0                ;CONTAINS SUBTEST ITERATION COUNT
348 001106 000000                        SLPADR: .WORD 0              ;CONTAINS SCOPE LOOP ADDRESS
349 001110 000000                        SLPERR: .WORD 0              ;CONTAINS SCOPE RETURN FOR ERRORS
350 001112 000000                        SERTTL: .WORD 0              ;CONTAINS TOTAL ERRORS DETECTED
351 001114 000                                SITEMB: .BYTE 0              ;CONTAINS ITEM CONTROL BYTE
352 001115 001                                SERMAX: .BYTE 1              ;CONTAINS MAX. ERRORS PER TEST
353 001116 000000                        SERRPC: .WORD 0              ;CONTAINS PC OF LAST ERROR INSTRUCTION
354 001120 000000                        SGADR: .WORD 0                ;CONTAINS ADDRESS OF 'GOOD' DATA
355 001122 000000                        SBDADR: .WORD 0              ;CONTAINS ADDRESS OF 'BAD' DATA
356 001124 000000                        SGDDAT: .WORD 0              ;CONTAINS 'GOOD' DATA
357 001126 000000                        SBDDAT: .WORD 0              ;CONTAINS 'BAD' DATA
358 001130 000000                        .WORD 0                      ;RESERVED--NOT TO BE USED
359 001132 000000                        .WORD 0
360 001134 000                                SAUTOB: .BYTE 0              ;AUTOMATIC MODE INDICATOR
361 001135 000                                SINTAG: .BYTE 0              ;INTERRUPT MODE INDICATOR
362 001136 000000                        .WORD 0
363 001140 177570                        SWR: .WORD DSWR              ;ADDRESS OF SWITCH REGISTER
364 001142 177570                        DISPLAY: .WORD DDISP         ;ADDRESS OF DISPLAY REGISTER
365 001144 177560                        STKS: 177560                 ;TTY KBD STATUS
366 001146 177562                        STKB: 177562                 ;TTY KBD BUFFER
367 001150 177564                        STPS: 177564                 ;TTY PRINTER STATUS REG. ADDRESS
368 001152 177566                        STPB: 177566                 ;TTY PRINTER BUFFER REG. ADDRESS
369 001154 000                                SNULL: .BYTE 0                ;CONTAINS NULL CHARACTER FOR FILLS
370 001155 002                                SFILLS: .BYTE 2              ;CONTAINS # OF FILLER CHARACTERS REQUIRED
371 001156 012                                SFILLC: .BYTE 12             ;INSERT FILL CHARS. AFTER A "LINE FEED"
372 001157 000                                STPFLG: .BYTE 0              ;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
373 001160 000000                        SREGAD: .WORD 0              ;CONTAINS THE ADDRESS FROM
374                                     ;WHICH ($REG0) WAS OBTAINED
375 001162 000000                        SREG0: .WORD 0                ;CONTAINS (($REGAD)+0)
376 001164 000000                        SREG1: .WORD 0                ;CONTAINS (($REGAD)+2)
377 001166 000000                        STIMES: 0                    ;MAX. NUMBER OF ITERATIONS
378 001170 000000                        SESCPE: 0                    ;ESCAPE ON ERROR ADDRESS
379 001172 177607 000377                SBELL: .ASCIZ <207><377><377> ;CODE FOR BELL
380 001176 077                                SQUES: .ASCII /?/            ;QUESTION MARK
381 001177 015                                SCRLF: .ASCII <15>           ;CARRIAGE RETURN
382 001200 000012                        SLF: .ASCIZ <12>              ;LINE FEED
383                                     ;*****
384 001202 000000                        SAVPC: .WORD 0                ;STORAGE FOR THE PC
385 001204 000000                        SAVPS: .WORD 0                ;STORAGE FOR THE PS
386
387 001206 177500                        TACSL: 177500                 ;LOW BYTE ADDRESS OF TACS
388 001210 177501                        TACSH: 177501                 ;HIGH BYTE ADDRESS OF TACS
389 001212 177502                        TADBL: 177502                 ;LOW BYTE ADDRESS OF TADB
390 001214 177503                        TADBH: 177503                 ;HIGH BYTE ADDRESS OF TADB
391 001216 000260 000262                TAVEC: 260,262               ;TA11 VECTOR ADDRESS

```

```

392 001222 000300                        TAPRIO: 300                   ;TA11 BR LEVEL 6
393 001224 000000 000000                DRVKEY: 0,0                   ;DRIVE SELECT KEY:
394 001230 001224                        DRVPNT: DRVKEY
395 001232 000000                        ASKKEY: 0
396 001234 000000                        CURDRV: 0                       ;CURRENT DRIVE BEING TESTED

```

```

397          ,SBTTL ERROR POINTER TABLE
398
399          ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
400          ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
401          ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
402          ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
403          ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
404
405          ;*      EM          ;;POINTS TO THE ERROR MESSAGE
406          ;*      DH          ;;POINTS TO THE DATA HEADER
407          ;*      DT          ;;POINTS TO THE DATA
408          ;*      DF          ;;POINTS TO THE DATA FORMAT
409
410
411          $ERRTB:
412
413          ;NOTE: ALL NUMBERS ARE TYPED AS 6-DIGIT OCTAL NUMBERS
414
415          ;ITEM 1
416          001236 013061      EM1      ;STATUS PROBLEM
417          001240 013227      DH1      ;PC TACS
418          001242 013340      DT1      ;$ERRPC $REG0
419          001244 000000      0
420
421          ;ITEM 2
422          001246 013100      EM2      ;READY FAILED TO SET
423          001250 013244      DH2      ;PC TACS WAIT ADDRESS
424          001252 013346      DT2      ;$ERRPC $REG0 SAVPC
425          001254 000000      0
426
427          ;ITEM 3
428          001256 013126      EM3      ;TRANSFER REQUEST FAILED TO SET
429          001260 013244      DH2      ;PC TACS WAIT ADDRESS
430          001262 013346      DT2      ;$ERRPC $REG0 SAVPC
431          001264 000000      0
432
433          ;ITEM 4
434          001266 013167      EM4      ;THE WRONG FLAG SET
435          001270 013244      DH2      ;PC TACS WAIT ADDRESS
436          001272 013346      DT2      ;$ERRPC $REG0 SAVPC
437          001274 000000      0
438
439          ;ITEM 5
440          001276 013212      EM5      ;DATA PROBLEM
441          001300 013301      DH5      ;PC TACS EXPECT RCV'D
442          001302 013356      DT5      ;$ERRPC $REG0 $GGDAT $BDDAT
443          001304 000000      0
444
445
    
```

```

446          001306          ITEMS2:      ;ITEMS 201-202
447
448          001306 013402      EM201     ;TA11 FAILED TO RESPOND
449          001310 013454      DH201     ;PC TACS
450          001312 013370      DT201     ;$ERRPC TACS
451          001314 000000      0          ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
452
453          001316 013431      EM202     ;NO DRIVES AVAILABLE
454          001320 013471      DH202     ;PC
455          001322 013376      DT202     ;$ERRPC
456          001324 000000      0
457
    
```

```

458 ;////////////////////////////////////
459 ;////////////////////////////////////
460 ;////////////////////////////////////
461 ;*****
462 ;BEGIN1 IS FOR NORMAL START
463 ;BEGIN2 IS FOR DRIVE SELECTION
464 ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
465 ;BEGIN4 IS FOR MANUAL OPERATION
466
467 ;*****
468
469 001326 005005 BEGIN1: CLR R5 ;NORMAL START
470 001330 012737 041101 001224 MOV #*AB,@DRVKEY
471 001336 122737 000005 000041 CMPB #5,@41 ;CASSETTE DDP?
472 001344 001015 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
473 001346 022737 000260 001216 CMP #260,@TAVEC ;STANDARD VECTOR?
474 001354 001011 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
475 001356 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
476 001360 012705 000001 BEGIN2: MOV #1,R5 ;ASK FOR DRIVES FLAG
477 001364 000405 BR BGNCMN ;BEGIN COMMON CODE
478 001366 012705 000002 BEGIN3: MOV #2,R5 ;ASK FOR DRIVES AND ADDRESSES
479 001372 000402 BR BGNCMN
480 001374 012705 000003 BEGIN4: MOV #3,R5
481 001400 BGNCMN:
482 ;SBTTL INITIALIZE THE COMMON TAGS
483 ;CLEAP THE COMMON TAGS ($CMTAG) AREA
484 001400 012706 001100 MOV #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
485 001404 005026 CLR (R6)+ ;CLEAR MEMORY LOCATION
486 001406 022706 001140 CMP #SWR,R6 ;DONE?
487 001412 001374 BNE -6 ;LOOP BACK IF NO
488 001414 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER
489 ;INITIALIZE A FEW VECTORS
490 001420 012737 006046 000020 MOV #SSCOPE,@IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
491 001426 012737 000340 000022 MOV #340,@IOTVEC+2 ;LEVEL 7
492 001434 012737 006320 000030 MOV #SEERROR,@EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
493 001442 012737 000340 000032 MOV #340,@EMTVEC+2 ;LEVEL 7
494 001450 012737 012304 000034 MOV #STRAP,@TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
495 001456 012737 000340 000036 MOV #340,@TRAPVEC+2 ;LEVEL 7
496 001464 012737 012370 000024 MOV #SPWRDN,@PWRVEC ;POWER FAILURE VECTOR
497 001472 012737 000340 000026 MOV #340,@PWRVEC+2 ;LEVEL 7
498 001500 016767 004270 004260 MOV SENDCT,$EOPCT ;SETUP END-OF-PROGRAM COUNTER
499 001506 005067 177454 CLR STIMES ;INITIALIZE NUMBER OF ITERATIONS
500 001512 005067 177452 CLR ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
501 001516 112767 000001 177371 MOVB #1,SERMAX ;ALLOW ONE ERROR PER TEST
502 001524 012767 001524 177354 MOV #,$SLPADR ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
503 001532 012767 001532 177350 MOV #,$SLPERR ;SETUP THE ERROR LOOP ADDRESS
504 ;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
505 ;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
506 001540 013746 000004 MOV #ERRVEC,-(SP) ;SAVE ERROR VECTOR
507 001544 012737 001600 000004 MOV #64$,@ERRVEC ;SET UP ERROR VECTOR
508 001552 012767 177570 177360 MOV #DSWR,SWR ;SETUP FOR A HARDWARE SWICH REGISTER
509 001560 012767 177570 177354 MOV #DDISP,DISPLAY ;AND A HARDWARE DISPLAY REGISTER
510 001566 022777 177777 177344 CMP #-1,@SWR ;TRY TO REFERENCE HARDWARE SWR
511 001574 001012 BNE 66$ ;BRANCH IF NO TIMEOUT TRAP OCCURRED
512 ;AND THE HARDWARE SWR IS NOT = -1
513 001576 000403 BR 65$ ;BRANCH IF NO TIMEOUT

```

```

514 001600 012716 001606 64$: MOV #65$,(SP) ;SET UP FOR TRAP RETURN
515 001604 000002 RTI
516 001606 012767 000176 177324 65$: MOV #SWREG,SWR ;POINT TO SOFTWARE SWR
517 001614 012767 000174 177320 MOV #DISPREG,DISPLAY
518 001622 012637 000004 66$: MOV (SP)+,@ERRVEC ;RESTORE ERROR VECTOR
519
520 ;SBTTL TYPE PROGRAM NAME
521 ;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
522 001626 005227 177777 INC #1 ;FIRST TIME?
523 001632 001036 BNE HERE ;BRANCH IF NO
524 001634 022737 006014 000042 CMP #SENDAD,@42 ;ACT-11?
525 001642 001432 BEQ HERE ;BRANCH IF YES
526 001644 104401 001702 TYPE ,MSGID ;TYPE ASCIZ STRING
527 ;SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
528 001650 005737 000042 TST #42 ;ARE WE RUNNING UNDER XXDP/ACT?
529 001654 001006 BNE 67$ ;BRANCH IF YES
530 001656 026727 177256 000176 CMP SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
531 001664 001005 BNE 68$ ;BRANCH IF NO
532 001666 104405 GTSWR ;GET SOFT-SWR SETTINGS
533 001670 000403 BR 68$
534 001672 112767 000001 177234 67$: MOVB #1,$AUTOB ;SET AUTO-MODE INDICATOR
535 001700 68$:
536 001700 000413 BR HERE ;GET OVER THE ASCIZ
537 ;MSGID: .ASCIZ <CRLF>/MAINDEC-11-DZTAC-C/<CRLF>
538 001730 HERE:
539 ;*****
540 ;*****
541 ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
542 ;
543 ;
544 ; R5=3 MANUAL OPERATIONS
545 ; R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
546 ; R5=1 ASK FOR DRIVE(S)
547 ; R5=0 DON'T ASK FOR ANYTHING
548 ;
549 ;*****
550 001730 010504 BEGINX: MOV R5,R4 ;COPY R5
551 001732 005305 DEC R5 ;ASK FOR DRIVES?
552 001734 002406 BLT CHKADR ;BR IF NO
553 001736 004737 007036 JSR PC,@ASKDRV ;GO GET DRIVES TO BE TESTED
554 001742 005305 DEC R5 ;ASK FOR ADDRESSES?
555 001744 002402 BLT CHKADR ;BR IF NO
556 001746 004737 007146 JSR PC,@ASKADR ;GO GET TA11 ADDRESSES
557 ;*****
558 ;*****
559 ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
560 ;
561 ;
562 ; I. TIMEOUT OCCURRED
563 ; A. TYPE ERROR MESSAGE
564 ; B. EXAMINE R4
565 ;
566 ; 1. R4>0 GOTO BEGINX
567 ; 2. R4=0 EXAMINE (42)
568 ; A. (42)=0 GOTO BEGINX
569 ; B. (42)>0 GOTO SENDAD

```



```

570 ;II. TIMEOUT DIDN'T OCCUR
571 ; A. CONTINUE
572 ;
573 ;*****
574 001752 012737 001770 000004 CHKADR: MOV #15,0#ERRVEC ;IN CASE OF TIMEOUTS
575 001760 005000 CLR R0 ;USE AS A SWITCH
576 001762 005777 177220 TST @TACSL ;SEE IF TA11 RESPONDS
577 001766 000402 BR 28 ;BR IF NO TIMEOUT
578 001770 005200 16: INC R0 ;COME HERE ON TIMEOUT
579 001772 022626 CMP (SP)+,(SP)+ ;CLEANUP THE STACK
580 001774 012737 000006 000004 26: MOV #ERRVEC+2,0#ERRVEC ;RESTORE TIMEOUT VECTOR
581 002002 005700 TST R0 ;DID A TIMEOUT OCCUR?
582 002004 001412 BEQ 38 ;BR IF NO
583 002006 104201 ERROR 201 ;TA11 FAILED TO RESPOND
584 002010 012705 000002 MOV #2,R5 ;DRIVES & ADDRESSES
585 002014 005704 TST R4 ;OPERATOR INPUTS?
586 002016 001344 BNE BEGINX ;BR IF YES
587 002020 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
588 002024 001741 BEQ BEGINX ;BR IF NO MONITOR
589 002026 000137 006014 JMP 0#SENDAD ;GO TO END
590 002032 38:

```

```

591 ;*****
592 ;*****
593 ;
594 ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
595 ;
596 ;I. DESIRED DRIVES CAN NOT BE TESTED
597 ; A. TYPE ERROR MESSAGE
598 ; B. EXAMINE R4
599 ; 1. R4>0 GOTO BEGINX
600 ; 2. R4=0 EXAMINE (42)
601 ; A. (42)=0 GOTO BEGINX
602 ; B. (42)>0 GOTO SENDAD
603 ;
604 ;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
605 ; A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
606 ; B. CONTINUE IN PROGRAM
607 ;
608 ;III. DESIRED DRIVE(S) CAN BE TESTED
609 ; A. CONTINUE IN PROGRAM
610 ;
611 ;*****
612 002032 012700 001224 CHKDRV: MOV #DRVKEY,R0 ;PICKUP ADDRESS OF ASCII DRIVE KEY
613 002036 004737 010654 JSR PC,#EXAM ;GO EXAMINE FIRST DRIVE
614 002042 000410 BR 18 ;OK TO TEST---GO CHECK NEXT
615 002044 116010 000001 MOV#B 1(R0),(R0) ;REPLACE 1ST WITH 2ND
616 002050 001412 BEQ 28 ;BR IF NO 2ND DRIVE SELECTED
617 002052 004737 010654 JSR PC,#EXAM ;GO EXAMINE DRIVE
618 002056 000407 BR 28 ;OK TO TEST
619 002060 005010 CLR (R0) ;CLEAR DRIVE CODES
620 002062 000405 BR 28
621 002064 005200 18: INC R0 ;POINT TO 2ND
622 002066 004737 010654 JSR PC,#EXAM ;GO EXAMINE DRIVE
623 002072 000401 BR 28 ;OK TO TEST
624 002074 105010 CLR#B (R0) ;CLEAR 2ND
625 002076 012700 001224 28: MOV #DRVKEY,R0 ;RESET ADDRESS POINTERS
626 002102 010037 001230 MOV R0,#DRVPT
627 002106 121060 000001 CMP#B (R0),1(R0) ;1ST = 2ND?
628 002112 001002 BNE 38 ;BR IF NO
629 002114 105060 000001 CLR#B 1(R0) ;YES---CLEAR 2ND
630 002120 005710 38: TST (R0) ;ANY DRIVES?
631 002122 001401 BEQ 58 ;BR IF NO
632 002124 000412 BR MANUAL
633 002126 104202 ERROR 202 ;NO DRIVES AVAILABLE
634 002130 012705 000002 MOV #2,R5 ;DRIVES & ADDRESS
635 002134 005704 TST R4 ;OPERATOR INPUTS?
636 002136 001274 BNE BEGINX ;BR IF YES
637 002140 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
638 002144 001671 BEQ BEGINX ;NO MONITOR
639 002146 000137 006014 JMP 0#SENDAD ;GO TO END
640 002152 020427 000003 MANUAL: CMP R4,#3
641 002156 001002 BNE OK
642 002160 016704 175613 MOV =1,R4
643 002164 010437 001232 OK: MOV R4,#ASKKEY
644 002170 000405 BR START
645 002172 104401 001702 PWRST: TYPE #MSGID ;POWER FAIL RESTART
646 002176 012737 001224 MOV #DRVKEY,#DRVPT

```

```
647 002204 013777 001220 177004 START: MOV @TAVEC+2,@TAVEC ;SETUP TA11 TRAP VECTOR
648 002212 005077 177002 CLR @TAVEC+2
649 002216 012737 000340 177776 MOV #340,@PS ;LOCKOUT ALL I/O INT
650 002224 013704 001206 MOV @TACSL,TACS ;SETUP TACS
651 002230 013705 001212 MOV @TADBL,TADB ;SETUP TADB
652 002234 005737 001100 TST @SPASS ;IF FIRST PASS SETUP FOR EXTRA LONG WAIT LOOPS
653 002240 001003 BNE 1$ ;OTHERWISE USE OLD VALUES
654 002242 012737 077777 006732 MOV #CBIT15,@MAXCNT
655 002250 005037 001100 1$: MOV @TSTNM ;ZERO THE TEST NUMBER
656 002254 005003 CLR DRIVE ;SET DRIVE TO "A"
657 002256 013701 001230 MOV @DRVPT,R1 ;GET DRIVE POINTER
658 002262 121127 000101 CMPB (R1),#A ;IS IT DRIVE "A"?
659 002266 001402 BEQ TDRV ;BR IF YES
660 002270 012703 000400 MOV #UNIT,DRIVE ;SET DRIVE TO "B"
661 002274 TDRV:
662 002274 104401 002302 TYPE ,65$ ;TYPE ASCIZ STRING
663 002300 000411 BR 64$ ;GET OVER THE ASCIZ
664 64$: .ASCIZ <15><12>*TESTING DRIVE *
665 002324
666 002324 112167 176704 MOVB (R1)+,CURDRV ;SETUP TO TYPE CURRENT DRIVE
667 002330 104401 001234 TYPE ,CURDRV
668 002334 104401 001177 TYPE ,$CRLF ;TYPE A CR & LF
669 002340 105711 TSTB (R1) ;LAST DRIVE BEEN SELECTED
670 002342 001002 BNE 1$ ;BR IF NO
671 002344 012701 001224 MOV @DRVKEY,R1 ;RESET DRIVE POINTER
672 002350 010137 001230 R1,@DRVPT ;SAVE DRIVE POINTER FOR NEXT TIME
673 002354 005737 001232 TST @ASKKEY ;GO START TESTING IF NO MANUAL
674 002360 002007 BGE 2$ ; OPERATIONS REQUESTED
675 002362 005000 R0
676 002364 000000 HALT ;GIVE CONTROL TO THE OPERATOR
677 002366 022767 000176 176544 CMP #SWREG,SWR ;USING S/W SWITCH REG?
678 002374 001001 BNE 20$ ;NO- GET OUT
679 002376 104405 GTSWR ;GET VALUE
680 002400 20$: ;CONTINUE
681 ;THIS CODE IS FOR ACT11 & DDP
682 002400 005737 000042 2$: TST @#42 ;IS THERE A MONITOR?
683 002404 001406 BEQ TST1 ;GO START TESTING IF NO
684 002406 010314 MOV DRIVE,@TACS ;IF YES SELECT DRIVE
685 002410 112714 000017 MOVB @REWIND!GO,@TACS ;SEND TAPE TO BOT
686 002414 032714 000040 BIT #READY,@TACS ;WAIT ON READY
687 002420 001775 BEQ 3$ ;FALL THRU IF READY=1
```

```
688 ////////////////////////////////////////////////////////////////////
689 ////////////////////////////////////////////////////////////////////
690 ;*****
691 ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX.
692 ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
693 ;*****
694 *TEST 1 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
695 ;*****
696 002422 000004 TST1: SCOPE
697 002424 012767 000001 176534 MOV #1,$TIMES ;DO 1 ITERATION
698 002432 005737 001100 TST @SPASS ;IS THIS THE FIRST PASS?
699 002436 001021 BNE TST2 ;BR IF NO
700 002440 000005 RESET
701 002442 010314 MOV DRIVE,@TACS ;SELECT THE DRIVE
702 002444 112714 000017 MOVB @REWIND!GO,@TACS ;START A REWIND
703 002450 104412 WAITREADY ;WAIT FOR READY
704 002452 112714 000001 MOVB @WFG!GO,@TACS ;WRITE A FILE GAP
705 002456 104412 WAITREADY ;WAIT ON READY
706 002460 163737 006706 006732 SUB @#HGHTIM,@#MAXCNT ;GET THE TIME IT TOOK
707 002466 005237 006732 INC @#MAXCNT ;MAKE IT BIGGER
708 002472 006137 006732 ROL @#MAXCNT
709 002476 006137 006732 ROL @#MAXCNT
710 ;*****
711 *TEST 2 SETUP FOR MANUAL INTERVENTION
712 ;*****
713 002502 000004 TST2: SCOPE
714 002504 012767 000001 176454 MOV #1,$TIMES ;DO 1 ITERATION
715 002512 005767 176362 TST $PASS ;IS THIS THE FIRST PASS?
716 002516 001053 BNE TST3 ;SKIP TYPE OUT IF YES
717 002520 104401 002526 TYPE ,65$ ;TYPE ASCIZ STRING
718 002524 000423 BR 64$ ;GET OVER THE ASCIZ
719 64$: .ASCIZ <15><12>"DIRECTIONS WILL BE TYPED ON THE TTY"
720 002574 TYPE ,67$ ;TYPE ASCIZ STRING
721 002574 104401 002602 BR 66$ ;GET OVER THE ASCIZ
722 002600 000422 66$: .ASCIZ <15><12>"HIT 'CR' WHEN READY TO CONTINUE"<15><12>
723 66$:
724 002646 ;*****
725 *TEST 3 SETUP FOR POWER DOWN TU60 TEST
726 ;*****
727 TST3: SCOPE
728 002646 000004 MOV #10,$TIMES ;DO 10. ITERATIONS
729 002650 012767 000012 176310 MOV #15,$LPADR ;SET SCOPE LOOP ADDRESS
730 002656 012767 002700 176222 JSR R0,@ASKQUES
731 002664 004037 007436 MTUPWR
732 002670 012674 BR 1$ ;DO THIS TEST
733 002672 000402 JMP @TST6 ;SKIP POWER FAIL TESTS
734 002674 000137 003120 1$: RESET ;CLEAR INTERFACE
735 002700 000005 MOV DRIVE,@TACS ;SELECT DRIVE
736 002702 010314 BIT #OFFLINE,@TACS ;IS "OFFLINE" ON A ONE?
737 002704 032714 001000 BNE TST4 ;BR IF YES
738 002710 001001 ERROR 1 ;"OFFLINE" BIT ISN'T ON A ONE
739 002712 104001 ;*****
740 *TEST 4 TEST "OFFLINE" WHEN TU60 IS POWERED DOWN
741 ;*****
742 TST4: SCOPE
743 002714 000004
```

```

744 002716 012767 003062 176244      MOV  #TST5,$ESCAPE  ;;ESCAPE TO TEST 5 ON ERROR
745 002724 112714 000001      MOVB #WFG!GO,@TACS  ;TRY TO DO A "WFG"
746 002730 104412      WAITREADY           ;WAIT IN "READY"
747 002732 005714      TST  @TACS          ;CHECK FOR AN "ERROR"
748 002734 100401      BMI  1$             ;BR IF "ERROR"=1
749 002736 104001      ERROR 1            ;"ERROR"=0
750 002740 112714 000003 1$:  MOV  #WRITE!GO,@TACS ;TRY TO DO A "WRITE"
751 002744 104412      WAITREADY           ;WAIT ON "READY"
752 002746 005714      TST  @TACS          ;CHECK ERROR FLAG
753 002750 100401      BMI  2$             ;"ERROR"=0
754 002752 104001      ERROR 1            ;"ERROR"=0
755 002754 112714 000005 2$:  MOVB #READ!GO,@TACS ;TRY TO DO A "READ"
756 002760 104412      WAITREADY           ;WAIT ON "READY"
757 002762 005714      TST  @TACS          ;CHECK ERROR FLAG
758 002764 100401      BMI  3$             ;"ERROR"=0
759 002766 104001      ERROR 1            ;"ERROR"=0
760 002770 112714 000007 3$:  MOVB #BSFG!GO,@TACS ;TRY TO DO A "BSFG"
761 002774 104412      WAITREADY           ;WAIT ON "READY"
762 002776 005714      TST  @TACS          ;CHECK ERROR FLAG
763 003000 100401      BMI  4$             ;"ERROR"=0
764 003002 104001      ERROR 1            ;"ERROR"=0
765 003004 112714 000011 4$:  MOVB #BSBG!GO,@TACS ;TRY TO DO A "BSBG"
766 003010 104412      WAITREADY           ;WAIT ON "READY"
767 003012 005714      TST  @TACS          ;CHECK ERROR FLAG
768 003014 100401      BMI  5$             ;"ERROR"=0
769 003016 104001      ERROR 1            ;"ERROR"=0
770 003020 112714 000013 5$:  MOVB #SFFG!GO,@TACS ;TRY TO DO A "SFFG"
771 003024 104412      WAITREADY           ;WAIT ON "READY"
772 003026 005714      TST  @TACS          ;CHECK ERROR FLAG
773 003030 100401      BMI  6$             ;"ERROR"=0
774 003032 104001      ERROR 1            ;"ERROR"=0
775 003034 112714 000015 6$:  MOVB #SFBG!GO,@TACS ;TRY TO DO A "SFBG"
776 003040 104412      WAITREADY           ;WAIT ON "READY"
777 003042 005714      TST  @TACS          ;CHECK ERROR FLAG
778 003044 100401      BMI  7$             ;"ERROR"=0
779 003046 104001      ERROR 1            ;"ERROR"=0
780 003050 112714 000017 7$:  MOVB #REWIND!GO,@TACS ;TRY TO DO A REWIND
781 003054 005714      TST  @TACS          ;CHECK ERROR FLAG
782 003056 100401      BMI  TST5           ;BR IF ERROR=1
783 003060 104001      ERROR 1            ;"ERROR"=0
784                                     ;*****
785                                     ;*TEST 5 POWER UP THE TU60
786                                     ;*****
787 003062 000004      TST5: SCOPE
788 003064 012767 003104 176014      MOV  #2$, $LPADR  ;;SET SCOPE LOOP ADDRESS
789 003072 004037 007436      JSR  R0, @ASKQUES
790 003076 012714      MPWRUP           ;MESSAGE POINTER
791 003100 000240      NOP
792 003102 000005 1$:  RESET
793 003104 010314 2$:  MOV  DRIVE,@TACS  ;SELECT DRIVE
794 003106 104412      WAITREADY           ;WAIT ON READY
795 003110 032714 001000      BIT  #OFFLINE,@TACS ;DID OFFLINE CLEAR?
796 003114 001401      BEQ  TST6         ;BR IF YES
797 003116 104001      ERROR 1           ;OFFLINE = 1
798                                     ;*****
799                                     ;*TEST 6 TEST "OFFLINE" WHEN DRIVE IS EMPTY
    
```

```

800                                     ;*****
801 003120 000004      TST6: SCOPE
802 003122 012767 003146 175756      MOV  #2$, $LPADR  ;;SET SCOPE LOOP ADDRESS
803 003130 004037 007436      JSR  R0, @ASKQUES
804 003134 012732      MOFFLN           ;MESSAGE POINTER
805 003136 000402      BR  1$            ;DO TEST
806 003140 000137 003224      JMP  @TST10       ;SKIP THIS TEST
807 003144 000005 1$:  RESET           ;CLEAR THE WORLD
808 003146 010314 2$:  MOV  DRIVE,@TACS  ;SELECT DRIVE
809 003150 042714 001000      BIC  #OFFLINE,@TACS ;TRY TO CLEAR OFFLINE INDICATION
810 003154 032714 001000      BIT  #OFFLINE,@TACS ;TEST FOR "OFFLINE"
811 003160 001001      RNE  TST7        ;BR IF "OFFLINE"=1
812 003162 104001      ERROR 1          ;"OFFLINE"=0
813                                     ;*****
814                                     ;*TEST 7 PUT DRIVE "ONLINE"
815                                     ;*****
816 003164 000004      TST7: SCOPE
817 003166 012767 003212 175712      MOV  #2$, $LPADR  ;;SET SCOPE LOOP ADDRESS
818 003174 004037 007436      JSR  R0, @ASKQUES
819 003200 013000      MTAPE
820 003202 000402      BP  1$            ;DO TEST
821 003204 000137 003224      JMP  @TST10       ;SKIP THIS TEST
822 003210 000005 1$:  RESET           ;CLEAR THE WORLD
823 003212 010314 2$:  MOV  DRIVE,@TACS  ;SELECT DRIVE
824 003214 032714 001000      BIT  #OFFLINE,@TACS ;IS DRIVE OFFLINE?
825 003220 001401      BEQ  TST10       ;BR IF NO
826 003222 104001      ERROR 1          ;DRIVE IS OFFLINE
827                                     ;*****
828                                     ;*TEST 10 PUT DATA ON TAPE FOR WRITE LOCK TEST
829                                     ;*****
830 003224 000004      TST10: SCOPE
831 003226 012767 000001 175732      MOV  #1, $TIMES  ;;DO 1 ITERATION
832 003234 012767 003242 175726      MOV  #1$, $ESCAPE ;ESCAPE TO 1$ ON ERROR
833                                     ;*****
834                                     ;HANG IN THIS TEST UNTIL DATA IS ON TAPE WITH NO ERRORS
835 003242 000005 1$:  RESET           ;CLEAR THE WORLD
836 003244 010314      MOV  DRIVE,@TACS  ;SELECT DRIVE
837 003246 112714 000017      MOVB #REWIND!GO,@TACS ;WAIT ON READY
838 003252 104412      WAITREADY           ;WAIT ON READY
839 003254 005714      TST  @TACS          ;CHECK FOR ERROR
840 003256 100001      BPL  2$           ;"ERROR" = 1
841 003260 104001      ERROR 1           ;"ERROR" = 1
842 003262      ;WRITE A BLOCK OF 6 BYTES
843      MOV  #6, 65$
844 003262 012767 000006 000014      MOV  #6, 65$      ;SETUP FOR 6 BYTES
845 003270 112714 000003      MOVB #WRITE!GO,@TACS ;START A WRITE
846 003274 104413 64$:  WAITXFER           ;WAIT ON TRANSFER REQUEST
847 003276 112715 000377      MOVB #377, @TADB  ;LOAD DATA BUFFER
848 003302 005327      DEC  (PC)+        ;MORE TO DO?
849 003304 000000 65$:  0                ;NUMBER OF BYTES TO WRITE GOES HERE
850 003306 003372      BGT  64$         ;BR IF YES
851 003310 104413      WAITXFER           ;WAIT ON TRANSFER REQUEST
852 003312 052714 000020      BIS  #ILBS,@TACS  ;WRITE CRC
853 003316 104412      WAITREADY           ;WAIT ON READY
854      ;WRITE A BLOCK OF 6 BYTES
855 003320 012767 000006 000014      MOV  #6, 67$      ;SETUP FOR 6 BYTES
    
```

```

856 003326 112714 000003      MOVB  #WRITE!GO,@TACS      ;START A WRITE
857 003332 104413              WAITXFER                    ;WAIT ON TRANSFER REQUEST
858 003334 112715 000377      MOVB  #377,@TADB           ;LOAD DATA BUFFER
859 003340 005327              DEC    (PC)+                ;MORE TO DO?
860 003342 000000              0                          ;NUMBER OF BYTES TO WRITE GOES HERE
861 003344 003372              BGT   666                  ;BR IF YES
862 003346 104413              WAITXFER                    ;WAIT ON TRANSFER REQUEST
863 003350 052714 000020      BIS   #ILBS,@TACS         ;WRITE CRC
864 003354 104412              WAITREADY                   ;WAIT ON READY
865 003356 112714 000001      MOVB  #WFG!GO,@TACS       ;WRITE A FILE GAP
866 003362 104412              WAITREADY                   ;WAIT ON "READY"
867                                ;WRITE A BLOCK OF 6 BYTES
868 003364 012767 000006 000014  MOV   #6,69$              ;SETUP FOR 6 BYTES
869 003372 112714 000003      MOVB  #WRITE!GO,@TACS     ;START A WRITE
870 003376 104413              WAITXFER                    ;WAIT ON TRANSFER REQUEST
871 003400 112715 000377      MOVB  #377,@TADB           ;LOAD DATA BUFFER
872 003404 005327              DEC    (PC)+                ;MORE TO DO?
873 003406 000000              0                          ;NUMBER OF BYTES TO WRITE GOES HERE
874 003410 003372              BGT   60$                  ;BR IF YES
875 003412 104413              WAITXFER                    ;WAIT ON TRANSFER REQUEST
876 003414 052714 000020      BIS   #ILBS,@TACS         ;WRITE CRC
877 003420 104412              WAITREADY                   ;WAIT ON READY
878                                ;*****
879                                ;*TEST 11 TEST "WRITE LOCK" WHEN AT CLEAR LEADER
880                                ;*****
881 003422 000004              TST11: SCOPE
882 003424 012767 000001 175534  MOV   #1,$TIMES            ;DO 1 ITERATION
883 003432 012767 003462 175446  MOV   #1$,SLPADR          ;SET SCOPE LOOP ADDRESS
884 003440 012767 003772 175522  MOV   #TST12,$ESCAPE     ;ESCAPE TO TEST 12 ON ERROR
885 003446 004037 007436      JSR   R0,#ASKQUES        ;HAVE OPERATOR SET "WRITE LOCK"
886 003452 013021              MWRTLK
887 003454 000402              BR    1$                  ;DO TEST
888 003456 000137 005716      JMP   0*TST22             ;SKIP WRITE LOCK TESTS
889                                ;
890 003462 000005              1$: RESET                  ;CLEAR ALL
891 003464 010314              MOV   DRIVE,@TACS         ;SELECT DRIVE
892 003466 112714 000017      MOVB  #REWIND!GO,@TACS    ;GO TO "BOT"
893 003472 104412              WAITREADY
894 003474 005714              TST   @TACS                ;IS "ERROR"=1
895 003476 100001              BPL   2$                  ;BR IF NO
896 003500 104001              ERROR 1                    ;"ERROR"=1
897 003502 032714 020000      2$: BIT   #LEADER,@TACS    ;CHECK FOR CLEAR LEADER
898 003506 001001              BNE   3$                  ;BR IF ON CLEAR LEADER
899 003510 104001              ERROR 1                    ;REWIND DIDN'T GO TO BOT
900 003512              3$:
901 003512 112714 000000      MOVB  #WFG,@TACS          ;LOAD A "WFG"
902 003516 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "1"
903 003522 001001              BNE   64$                 ;BR IF YES
904 003524 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "1"
905 003526              64$:
906 003526 112714 000002      MOVB  #WRITE,@TACS        ;LOAD A "WRITE"
907 003532 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "1"
908 003536 001001              BNE   65$                 ;BR IF YES
909 003540 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "1"
910 003542              65$:
911 003542 112714 000004      MOVB  #READ,@TACS         ;LOAD A "READ"
    
```

```

912 003546 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
913 003552 001401              BEQ   66$                 ;BR IF YES
914 003554 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
915 003556              66$:
916 003556 112714 000006      MOVB  #BSFG,@TACS        ;LOAD A "BSFG"
917 003562 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
918 003566 001401              BEQ   67$                 ;BR IF YES
919 003570 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
920 003572              67$:
921 003572 112714 000010      MOVB  #BSBG,@TACS        ;LOAD A "BSBG"
922 003576 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
923 003602 001401              BEQ   68$                 ;BR IF YES
924 003604 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
925 003606              68$:
926 003606 112714 000012      MOVB  #SFFG,@TACS        ;LOAD A "SFFG"
927 003612 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
928 003616 001401              BEQ   69$                 ;BR IF YES
929 003620 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
930 003622              69$:
931 003622 112714 000014      MOVB  #SFBG,@TACS        ;LOAD A "SFBG"
932 003626 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
933 003632 001401              BEQ   70$                 ;BR IF YES
934 003634 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
935 003636              70$:
936 003636 112714 000016      MOVB  #REWIND,@TACS      ;LOAD A "REWIND"
937 003642 032714 010000      BIT   #WRTLOCK,@TACS     ;IS "WRITE LOCK ERROR" = "0"
938 003646 001401              BEQ   71$                 ;BR IF YES
939 003650 104001              ERROR 1                    ;"WRITE LOCK ERROR" NOT EQUAL "0"
940 003652              71$:
941 003652 112714 000000      MOVB  #WFG,@TACS        ;CHECK "ERROR" WITH "WFG"
942 003656 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
943 003660 100401              BMI   72$                 ;BR IF "ERROR" = 1
944 003662 104001              ERROR 1                    ;"ERROR" NOT = 1
945 003664              72$:
946 003664 112714 000002      MOVB  #WRITE,@TACS       ;CHECK "ERROR" WITH "WRITE"
947 003670 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
948 003672 100401              BMI   73$                 ;BR IF "ERROR" = 1
949 003674 104001              ERROR 1                    ;"ERROR" NOT = 1
950 003676              73$:
951 003676 112714 000004      MOVB  #READ,@TACS        ;CHECK "ERROR" WITH "READ"
952 003702 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
953 003704 100401              BMI   74$                 ;BR IF "ERROR" = 1
954 003706 104001              ERROR 1                    ;"ERROR" NOT = 1
955 003710              74$:
956 003710 112714 000006      MOVB  #BSFG,@TACS        ;CHECK "ERROR" WITH "BSFG"
957 003714 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
958 003716 100401              BMI   75$                 ;BR IF "ERROR" = 1
959 003720 104001              ERROR 1                    ;"ERROR" NOT = 1
960 003722              75$:
961 003722 112714 000010      MOVB  #BSBG,@TACS        ;CHECK "ERROR" WITH "BSBG"
962 003726 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
963 003730 100401              BMI   76$                 ;BR IF "ERROR" = 1
964 003732 104001              ERROR 1                    ;"ERROR" NOT = 1
965 003734              76$:
966 003734 112714 000012      MOVB  #SFFG,@TACS        ;CHECK "ERROR" WITH "SFFG"
967 003740 005714              TST   @TACS                ;SAMPLE THE "ERROR" BIT
    
```

DZTACC,NEW	T11	TEST "WRITE LOCK" WHEN AT CLEAR LEADER			
969	003742	100401		BMI	770
969	003744	104001		ERROR	1
970	003746		770:		
971	003746	112714	000014	MOVB	#SFBG,@TACS
972	003752	005714		TST	@TACS
973	003754	100401		BMI	708
974	003756	104001		ERROR	1
975	003760		780:		
976	003760	112714	000016	MOVB	#REWIND,@TACS
977	003764	005714		TST	@TACS
978	003766	100001		BPL	798
979	003770	104001		ERROR	1
980	003772		790:		
981				;*****	
982				;*TEST 12 TEST "WRITE LOCK" WITH "CRC ERROR"	
983				;*****	
984	003772	000004		TST12:	SCOPE
985	003774	012767	004052	MOV	#16,\$LPADR ;SET SCOPE LOOP ADDRESS
986	004002	012767	004332	MOV	#TST13,\$ESCAPE ;ESCAPE TO TEST 13 ON ERROR
987	004010	000005	175104	RESET	
988	004012	010314	175160	MOV	DRIVE,@TACS ;INIT. THE INTERFACE
989	004014	112714		MOVB	#REWINDIG0,@TACS ;SELECT DRIVE
990	004020	104412	000017	WAITREADY	
991	004022	112714		MOVB	#READIG0,@TACS ;GO TO BOT
992	004026	104413	000005	WAITXFER	
993	004030	104413		TSTB	@TADB ;WAIT ON READY
994	004032	104413		WAITXFER	
995	004034	052714	000020	BIS	#ILBS,@TACS ;GET OVER FIRST BLOCK OF DATA
996	004040	104412		WAITREADY	
997	004042	032714	040000	BIT	#CRCERR,@TACS ;KNOCK DOWN "XFER REQ"
998	004046	001001		BNE	1\$;SHUT DOWN GENERATE CRC ERR
999	004050	104001		ERROR	1 ;IS CRC ERROR FLAG=1?
1000	004052		1\$:		
1001	004052	112714	000000	MOVB	#WFG,@TACS ;BR IF YES
1002	004056	005714		TST	@TACS ;NO CRC ERROR
1003	004060	100401		BMI	64\$
1004	004062	104001		ERROR	1 ;CHECK "ERROR" WITH "WFG"
1005	004064		64\$:		
1006	004064	112714	000002	MOVB	#WRITE,@TACS ;SAMPLE THE "ERROR" BIT
1007	004070	005714		TST	@TACS ;BR IF "ERROR" = 1
1008	004072	100401		BMI	65\$
1009	004074	104001		ERROR	1 ;"ERROR" NOT = 1
1010	004076		65\$:		
1011	004076	112714	000004	MOVB	#READ,@TACS ;CHECK "ERROR" WITH "WRITE"
1012	004102	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1013	004104	100401		BMI	66\$
1014	004106	104001		ERROR	1 ;BR IF "ERROR" = 1
1015	004110		66\$:		
1016	004110	112714	000006	MOVB	#BSFG,@TACS ;CHECK "ERROR" WITH "READ"
1017	004114	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1018	004116	100001		BPL	67\$
1019	004120	104001		ERROR	1 ;BR IF "ERROR" = 0
1020	004122		67\$:		
1021	004122	112714	000010	MOVB	#BSBG,@TACS ;CHECK "ERROR" WITH "BSFG"
1022	004126	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1023	004130	100001		BPL	68\$;BR IF "ERROR" = 0

DZTACC,NEW	T12	TEST "WRITE LOCK" WITH "CRC ERROR"			
1024	004132	104001		ERROR	1
1025	004134		68\$:		
1026	004134	112714	000012	MOVB	#SFFG,@TACS ;CHECK "ERROR" WITH "WRITE"
1027	004140	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1028	004142	100001		BPL	69\$
1029	004144	104001		ERROR	1 ;BR IF "ERROR" = 0
1030	004146		69\$:		
1031	004146	112714	000014	MOVB	#SFBG,@TACS ;CHECK "ERROR" WITH "BSFG"
1032	004152	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1033	004154	100001		BPL	70\$
1034	004156	104001		ERROR	1 ;BR IF "ERROR" = 0
1035	004160		70\$:		
1036	004160	112714	000016	MOVB	#REWIND,@TACS ;CHECK "ERROR" WITH "BSFG"
1037	004164	005714		TST	@TACS ;SAMPLE THE "ERROR" BIT
1038	004166	100001		BPL	71\$
1039	004170	104001		ERROR	1 ;BR IF "ERROR" = 0
1040	004172		71\$:		
1041	004172	112714	000000	MOVB	#WFG,@TACS ;LOAD A "WRITE"
1042	004176	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "1"
1043	004202	001001		BNE	72\$;BR IF YES
1044	004204	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "1"
1045	004206		72\$:		
1046	004206	112714	000002	MOVB	#WRITE,@TACS ;LOAD A "READ"
1047	004212	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "1"
1048	004216	001001		BNE	73\$;BR IF YES
1049	004220	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "1"
1050	004222		73\$:		
1051	004222	112714	000004	MOVB	#READ,@TACS ;LOAD A "BSFG"
1052	004226	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1053	004232	001401		BEQ	74\$;BR IF YES
1054	004234	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"
1055	004236		74\$:		
1056	004236	112714	000006	MOVB	#BSFG,@TACS ;LOAD A "BSBG"
1057	004242	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1058	004246	001401		BEQ	75\$;BR IF YES
1059	004250	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"
1060	004252		75\$:		
1061	004252	112714	000010	MOVB	#BSBG,@TACS ;LOAD A "SFFG"
1062	004256	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1063	004262	001401		BEQ	76\$;BR IF YES
1064	004264	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"
1065	004266		76\$:		
1066	004266	112714	000012	MOVB	#SFFG,@TACS ;LOAD A "BSFG"
1067	004272	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1068	004276	001401		BEQ	77\$;BR IF YES
1069	004300	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"
1070	004302		77\$:		
1071	004302	112714	000014	MOVB	#SFBG,@TACS ;LOAD A "REWIND"
1072	004306	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1073	004312	001401		BEQ	78\$;BR IF YES
1074	004314	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"
1075	004316		78\$:		
1076	004316	112714	000016	MOVB	#REWIND,@TACS ;LOAD A "BSBG"
1077	004322	032714	010000	BIT	#WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1078	004326	001401		BEQ	79\$;BR IF YES
1079	004330	104001		ERROR	1 ;"WRITE LOCK ERROR" NOT EQUAL "0"

```

1080 004332          798:
1081                ;*****
1082                ;*TEST 13  TEST "WRITE LOCK" WITHOUT ANY ERRORS
1083                ;*****
1084 004332 000004    TST13: SCOPE
1085 004334 012767 004364 174544    MOV      #18,$LPADR      ;;SET SCOPE LOOP ADDRESS
1086 004342 012767 004644 174620    MOV      #TST14,$ESCAPE ;;ESCAPE TO TEST 14 ON ERROR
1087 004350 000005    RESET
1088 004352 010314    MOV      DRIVE,@TACS      ;SELECT DRIVE
1089 004354 032714 067000    BIT      #CRCERR|LEADER|FGAP|TIMER|OFFLINE,@TACS ;CHECK STATUS
1090 004360 001401    BEQ      18                ;BR IF OK TO TEST "WRITE LOCK"
1091 004362 104001    ERROR   1                ;TAPE ISN'T POSITIONED PROPERLY
1092 004364
1093 004364 112714 000000    18:    MOV      #WFG,@TACS      ;LOAD A "WFG"
1094 004370 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "1"
1095 004374 001001    BNE      648             ;BR IF YES
1096 004376 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "1"
1097 004400
1098 004400 112714 000002    648:   MOV      #WRITE,@TACS     ;LOAD A "WRITE"
1099 004404 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "1"
1100 004410 001001    BNE      658             ;BR IF YES
1101 004412 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "1"
1102 004414
1103 004414 112714 000004    658:   MOV      #READ,@TACS     ;LOAD A "READ"
1104 004420 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1105 004424 001401    BEQ      668             ;BR IF YES
1106 004426 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1107 004430
1108 004430 112714 000006    668:   MOV      #BSFG,@TACS     ;LOAD A "BSFG"
1109 004434 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1110 004440 001401    BEQ      678             ;BR IF YES
1111 004442 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1112 004444
1113 004444 112714 000010    678:   MOV      #BSBG,@TACS     ;LOAD A "BSBG"
1114 004450 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1115 004454 001401    BEQ      688             ;BR IF YES
1116 004456 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1117 004460
1118 004460 112714 000012    688:   MOV      #SFFG,@TACS     ;LOAD A "SFFG"
1119 004464 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1120 004470 001401    BEQ      698             ;BR IF YES
1121 004472 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1122 004474
1123 004474 112714 000014    698:   MOV      #SFBG,@TACS     ;LOAD A "SFBG"
1124 004500 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1125 004504 001401    BEQ      708             ;BR IF YES
1126 004506 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1127 004510
1128 004510 112714 000016    708:   MOV      #REWIND,@TACS    ;LOAD A "REWIND"
1129 004514 032714 010000    BIT      #WRTLOCK,@TACS ;IS "WRITE LOCK ERROR" = "0"
1130 004520 001401    BEQ      718             ;BR IF YES
1131 004522 104001    ERROR   1                ;"WRITE LOCK ERROR" NOT EQUAL "0"
1132 004524
1133 004524 112714 000000    718:   MOV      #WFG,@TACS      ;CHECK "ERROR" WITH "WFG"
1134 004530 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1135 004532 104001    BMI      728             ;BR IF "ERROR" = 1

```

```

1136 004534 104001    ERROR   1                ;"ERROR" NOT = 1
1137 004536
1138 004536 112714 000002    728:   MOV      #WRITE,@TACS     ;CHECK "ERROR" WITH "WRITE"
1139 004542 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1140 004544 104001    BMI      738             ;BR IF "ERROR" = 1
1141 004546 104001    ERROR   1                ;"ERROR" NOT = 1
1142 004550
1143 004550 112714 000004    738:   MOV      #READ,@TACS     ;CHECK "ERROR" WITH "READ"
1144 004554 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1145 004556 100001    BPL      748             ;BR IF "ERROR" = 0
1146 004560 104001    ERROR   1                ;"ERROR" NOT = 0
1147 004562
1148 004562 112714 000006    748:   MOV      #BSFG,@TACS     ;CHECK "ERROR" WITH "BSFG"
1149 004566 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1150 004570 100001    BPL      758             ;BR IF "ERROR" = 0
1151 004572 104001    ERROR   1                ;"ERROR" NOT = 0
1152 004574
1153 004574 112714 000010    758:   MOV      #BSBG,@TACS     ;CHECK "ERROR" WITH "BSBG"
1154 004600 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1155 004602 100001    BPL      768             ;BR IF "ERROR" = 0
1156 004604 104001    ERROR   1                ;"ERROR" NOT = 0
1157 004606
1158 004606 112714 000012    768:   MOV      #SFFG,@TACS     ;CHECK "ERROR" WITH "SFFG"
1159 004612 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1160 004614 100001    BPL      778             ;BR IF "ERROR" = 0
1161 004616 104001    ERROR   1                ;"ERROR" NOT = 0
1162 004620
1163 004620 112714 000014    778:   MOV      #SFBG,@TACS     ;CHECK "ERROR" WITH "SFBG"
1164 004624 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1165 004626 100001    BPL      788             ;BR IF "ERROR" = 0
1166 004630 104001    ERROR   1                ;"ERROR" NOT = 0
1167 004632
1168 004632 112714 000016    788:   MOV      #REWIND,@TACS    ;CHECK "ERROR" WITH "REWIND"
1169 004636 005714    TST      @TACS            ;SAMPLE THE "ERROR" BIT
1170 004640 100001    BPL      798             ;BR IF "ERROR" = 0
1171 004642 104001    ERROR   1                ;"ERROR" NOT = 0
1172 004644
1173                798:
1174                ;*****
1175                ;*TEST 14  TEST "WRITE LOCK" WITH FILE GAP=1
1176                ;*****
1176 004644 000004    TST14: SCOPE
1177 004646 012767 004726 174232    MOV      #18,$LPADR      ;;SET SCOPE LOOP ADDRESS
1178 004654 012767 005206 174306    MOV      #TST15,$ESCAPE ;;ESCAPE TO TEST 15 ON ERROR
1179 004662 000005    RESET ;CLEAR THE INTERFACE
1180 004664 010314    MOV      DRIVE,@TACS     ;SELECT @TACS
1181 004666 112714 000017    MOV      #REWIND|GO,@TACS
1182 004672 104412    WAITREADY
1183 004674 112714 000005    MOV      #READ!GO,@TACS  ;START A READ
1184 004700 104413    WAITXFER
1185 004702 052714 000020    BIS      #ILBS,@TACS     ;SHUT DOWN
1186 004706 104412    WAITREADY
1187 004710 112714 000013    MOV      #SFFG!GO,@TACS ;GO TO THE FILE GAP
1188 004714 104412    WAITREADY ;WAIT ON "READY"
1189 004716 032714 004000    BIT      #FGAP,@TACS     ;CHECK FOR FILE GAP
1190 004722 001001    BNE      18                ;BR IF IN A GAP
1191 004724 104001    ERROR   1                ;FILE GAP = 0

```

1192	004726			15:			
1193	004726	112714	000000		MOVB	#WFG,@TACS	;LOAD A "WFG"
1194	004732	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "1"
1195	004736	001001			BNE	64s	;BR IF YES
1196	004740	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1197	004742			64s:			
1198	004742	112714	000002		MOVB	#WRITE,@TACS	;LOAD A "WRITE"
1199	004746	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "1"
1200	004752	001001			BNE	65s	;BR IF YES
1201	004754	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "1"
1202	004756			65s:			
1203	004756	112714	000004		MOVB	#READ,@TACS	;LOAD A "READ"
1204	004762	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1205	004766	001401			BEG	66s	;BR IF YES
1206	004770	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1207	004772			66s:			
1208	004772	112714	000006		MOVB	#BSFG,@TACS	;LOAD A "BSFG"
1209	004776	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1210	005002	001401			BEG	67s	;BR IF YES
1211	005004	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1212	005006			67s:			
1213	005006	112714	000010		MOVB	#BSBG,@TACS	;LOAD A "BSBG"
1214	005012	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1215	005016	001401			BEG	68s	;BR IF YES
1216	005020	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1217	005022			68s:			
1218	005022	112714	000012		MOVB	#SFFG,@TACS	;LOAD A "SFFG"
1219	005026	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1220	005032	001401			BEG	69s	;BR IF YES
1221	005034	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1222	005036			69s:			
1223	005036	112714	000014		MOVB	#SFBG,@TACS	;LOAD A "SFBG"
1224	005042	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1225	005046	001401			BEG	70s	;BR IF YES
1226	005050	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1227	005052			70s:			
1228	005052	112714	000016		MOVB	#REWIND,@TACS	;LOAD A "REWIND"
1229	005056	032714	010000		BIT	#WRTLOCK,@TACS	;IS "WRITE LOCK ERROR" = "0"
1230	005062	001401			BEG	71s	;BR IF YES
1231	005064	104001			ERROR	1	; "WRITE LOCK ERROR" NOT EQUAL "0"
1232	005066			71s:			
1233	005066	112714	000000		MOVB	#WFG,@TACS	;CHECK "ERROR" WITH "WFG"
1234	005072	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1235	005074	100401			BMI	72s	;BR IF "ERROR" = 1
1236	005076	104001			ERROR	1	; "ERROR" NOT = 1
1237	005100			72s:			
1238	005100	112714	000002		MOVB	#WRITE,@TACS	;CHECK "ERROR" WITH "WRITE"
1239	005104	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1240	005106	100401			BMI	73s	;BR IF "ERROR" = 1
1241	005110	104001			ERROR	1	; "ERROR" NOT = 1
1242	005112			73s:			
1243	005112	112714	000004		MOVB	#READ,@TACS	;CHECK "ERROR" WITH "READ"
1244	005116	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1245	005120	100401			BMI	74s	;BR IF "ERROR" = 1
1246	005122	104001			ERROR	1	; "ERROR" NOT = 1
1247	005124			74s:			

1248	005124	112714	000006		MOVB	#BSFG,@TACS	;CHECK "ERROR" WITH "BSFG"
1249	005130	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1250	005132	100001			BPL	75s	;BR IF "ERROR" = 0
1251	005134	104001			ERROR	1	; "ERROR" NOT = 0
1252	005136			75s:			
1253	005136	112714	000010		MOVB	#BSBG,@TACS	;CHECK "ERROR" WITH "BSBG"
1254	005142	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1255	005144	100001			BPL	76s	;BR IF "ERROR" = 0
1256	005146	104001			ERROR	1	; "ERROR" NOT = 0
1257	005150			76s:			
1258	005150	112714	000012		MOVB	#SFFG,@TACS	;CHECK "ERROR" WITH "SFFG"
1259	005154	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1260	005156	100001			BPL	77s	;BR IF "ERROR" = 0
1261	005160	104001			ERROR	1	; "ERROR" NOT = 0
1262	005162			77s:			
1263	005162	112714	000014		MOVB	#SFBG,@TACS	;CHECK "ERROR" WITH "SFBG"
1264	005166	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1265	005170	100401			BMI	78s	;BR IF "ERROR" = 1
1266	005172	104001			ERROR	1	; "ERROR" NOT = 1
1267	005174			78s:			
1268	005174	112714	000016		MOVB	#REWIND,@TACS	;CHECK "ERROR" WITH "REWIND"
1269	005200	005714			TST	@TACS	;SAMPLE THE "ERROR" BIT
1270	005202	100001			BPL	79s	;BR IF "ERROR" = 0
1271	005204	104001			ERROR	1	; "ERROR" NOT = 0
1272	005206			79s:			
1273					;*****		
1274					;*TEST 15 TRY WRITING ON TAPE WHEN "WRITE LOCKED"		
1275					;*****		
1276	005206	000004			TST15: SCOPE		
1277	005210	012767	000001	173750	MOV	#1,\$TIMES	;DO 1 ITERATION
1278	005216	012767	005266	173662	MOV	#36,\$LPADR	;SET SCOPE LOOP ADDRESS
1279	005224	012767	005362	173736	MOV	#TST16,\$ESCAPE	;ESCAPE TO TEST 16 ON ERROR
1280	005232	000005			RESET		;CLEAR ALL
1281	005234	010314			MOV	DRIVE,@TACS	;SELECT DRIVE
1282	005236	112714	000017		MOVB	#REWIND!GO,@TACS	;GO TO BOT
1283	005242	104412			WAITREADY		;WAIT ON READY FLAG
1284	005244	112714	000005		MOVB	#READ!GO,@TACS	;START A READ
1285	005250	104413			WAITXFER		;WAIT ON "XFER REQ"
1286	005252	052714	000020		BIS	#ILBS,@TACS	;SHUT DOWN
1287	005256	104412			WAITREADY		;WAIT ON READY
1288	005260	112714	000015		MOVB	#SFBG!GO,@TACS	;SPACE OVER THE 2ND BLOCK
1289	005264	104412			WAITREADY		;WAIT ON READY FLAG
1290	005266	000005			RESET		;CLEAR ALL FLAGS
1291	005270	010314			MOV	DRIVE,@TACS	;SELECT DRIVE
1292	005272	104412			WAITREADY		;WAIT ON READY
1293	005274	112714	000003		MOVB	#WRITE!GO,@TACS	;START A "WRITE"
1294	005300	005001			CLR	R1	;TIMER
1295	005302	105201			INCB	R1	;COUNT THE COUNTER GIVING
1296	005304	001376			BNE	4s	; "XFER REQ" TIME TO SET IF IT
1297							; IS GOING TO
1298	005306	105714			TSTB	@TACS	;CHECK "XFER REQ"
1299	005310	100001			BPL	5s	;BR IF IT DIDN'T SET
1300	005312	104001			ERROR	1	; "XFER REQ" SET
1301	005314	104412			WAITREADY		;WAIT ON READY
1302	005316	005714			TST	@TACS	;CHECK ERROR FLAG
1303	005320	100401			BMI	6s	;BR IF IT IS SET

```

DZTACC_NEW T15 TRY WRITING ON TAPE WHEN "WRITE LOCKED"
1304 005322 104001          ERROR 1          ;ERROR ISN'T SET
1305 005324 112714 000017 6s:  MOVB #REWIND!GO,@TACS ;GO TO BOT
1306 005330 104412          WAITREADY        ;WAIT ON READY FLAG
1307 005332 012701 000002          MOV #2,R1        ;GET OVER THE TWO DATA BLOCKS
1308 005336 112714 000005 7s:  MOVB #READ!GO,@TACS ;START A READ
1309 005342 005301          DEC R1           ;COUNT THIS BLOCK
1310 005344 002405          BLT R6           ;BR IF OVER THE 2ND BLOCK?
1311 005346 104413          WAITXFER        ;WAIT ON "XFER REQ"
1312 005350 052714 000020          BIS #ILBS,@TACS ;SHUT DOWN
1313 005354 104412          WAITREADY        ;WAIT ON READY
1314 005356 000767          BR 7s           ;GO DO ANOTHER BLOCK
1315 005360 104412 8s:  WAITREADY        ;IF "XFER REQ" COMES UP THERE
; IS MORE THAN TWO BLOCKS ON TAPE
; "WRITE" CAUSED DATA TO BE WRITTEN ON TAP
;*****
;*TEST 16 TRY "WFG" WHEN WRITE LOCKED
;*****
TST16: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
MOV #TST17,$ESCAPE ;ESCAPE TO TEST 17 ON ERROR
RESET
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #REWIND!GO,@TACS ;GO TO BOT
WAITREADY ;WAIT ON READY
MOVB #READ!GO,@TACS ;GET PAST THE FIRST BLOCK
WAITXFER ;WAIT ON TRANSFER REQ.
BIS #ILBS,@TACS ;SHUT DOWN
WAITREADY ;WAIT ON READY
RESET ;CLEAR THE WORLD
MOV DRIVE,@TACS ;SELECT DRIVE
WAITREADY ;WAIT ON THE READY FLAG
MOVB #WFG!GO,@TACS ;START A "WFG"
WAITREADY ;WAIT ON READY
MOVB #REWIND!GO,@TACS ;GO TO BOT
WAITREADY ;WAIT ON READY
MOVB #READ!GO,@TACS ;GET PAST 1ST BLOCK
WAITXFER ;WAIT ON TRANSFER REQ.
BIS #ILBS,@TACS ;SHUT DOWN
WAITREADY ;WAIT ON READY
MOVB #READ!GO,@TACS ;SEE IF 2ND BLOCK IS STILL ON TAPE
WAITXFER ;IF READY COMES UP THE "WFG"
; COMMAND WROTE ON THE TAPE
;SHUT DOWN
;WAIT ON READY
;*****
;*TEST 17 TEST "WFG" ON CLEAR LEADER AND WRITE LOCKED
;*****
TST17: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
MOV #1,$SLPADR ;SET SCOPE LOOP ADDRESS
MOV #TST20,$ESCAPE ;ESCAPE TO TEST 20 ON ERROR
RESET
MOV DRIVE,@TACS ;SELECT DRIVE
WAITREADY ;WAIT ON THE READY FLAG
MOVB #REWIND!GO,@TACS ;GO TO BOT
WAITREADY

```

```

DZTACC_NEW T17 TEST "WFG" ON CLEAR LEADER AND WRITE LOCKED
1360 005540 112714 000001          MOVB #WFG!GO,@TACS ;START A "WFG"
1361 005544 104412          WAITREADY        ;WAIT FOR READY
1362 005546 005714          TST @TACS        ;CHECK ERROR
1363 005550 104001          BMI 2s          ;BR IF ERROR=1
1364 005552 104001          ERROR 1         ;ERROR ISN'T ON AN ONE
1365 005554 032714 020000 2s:  BIT #LEADER,@TACS ;CHECK IF STILL ON CLEAR LEADER
1366 005560 001001          BNE TST20       ;GO TO THE NEXT TEST IF IT IS
1367 005562 104001          ERROR 1         ;NOT ON CLEAR LEADER
;*****
;*TEST 20 TEST "WRITE" WITH WRITE LOCK ON AND AT CLEAR LEADER
;*****
TST20: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
MOV #1,$SLPADR ;SET SCOPE LOOP ADDRESS
MOV #TST21,$ESCAPE ;ESCAPE TO TEST 21 ON ERROR
RESET
MOV DRIVE,@TACS ;SELECT THE DRIVE
WAITREADY ;WAIT ON READY
MOVB #REWIND!GO,@TACS ;GO TO BOT
WAITREADY ;WAIT TILL READY = 1
MOVB #WRITE!GO,@TACS ;TRY TO START A WRITE
WAITREADY ;IF TRANSFER REQ. COMES UP
;THERE IS A PROBLEM WITH WRITE LOCK
;MAKE SURE STILL A CLEAR LEADER
BNE TST21 ;BR IF AT CLEAR LEADER
ERROR 1 ;TAPE MOVED OFF OF CLEAR LEADER
;*****
;*TEST 21 TEST "WRITE ENABLE"
;*****
TST21: SCOPE
MOV #10,$TIMES ;DO 10. ITERATIONS
MOV #1,$SLPADR ;SET SCOPE LOOP ADDRESS
JSR R0,@ASKQUES
MLAST
NOP
RESET
MOV DRIVE,@TACS
MOVB #REWIND!GO,@TACS
WAITREADY
MOVB #WFG,@TACS ;LOAD A "WFG" COMMAND
BIT #WRTLOCK!OFFLINE,@TACS ;MAKE SURE THE CASSETTE IS
BEQ TST22 ; "WRITE ENABLED" AND "ON LINE"
ERROR 1
;*****
;*TEST 22 END OF TEST CODE
;*****
TST22: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
RESET
MOV DRIVE,@TACS ;SELECT DRIVE
MOVB #REWIND!GO,@TACS
WAITREADY
;SBTTL END OF PASS ROUTINE
;*****
;*INCREMENT THE PASS NUMBER ($PASS)

```



```

1416 ;*TYPE "END PASS"
1417 ;*IF THERES A MONITOR GO TO IT
1418 ;*IF THERE ISN'T JUMP TO START
1419 ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
1420 ;*$ENDMG CAN BE CHANGED TO 7.
1421
1422 005740 ;EOP:
1423 005740 000004 SCOPE
1424 005742 005067 173134 CLR $TSTNM ;;ZERO THE TEST NUMBER
1425 005746 005067 173214 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
1426 005752 005267 173122 INC $PASS ;;INCREMENT THE PASS NUMBER
1427 005756 042767 100000 173114 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
1428 005764 005327 DEC (PC)+ ;;LOOP?
1429 005766 000001 $EOPCT: ,WORD 1
1430 005770 003015 BGT $DOAGN ;;YES
1431 005772 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
1432 005774 000001 $ENDCT: ,WORD 1
1433 005776 005766 $EOPCT
1434 006000 104401 006033 TYPE , $ENDMG ;;TYPE "END PASS"
1435 006004 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
1436 006010 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
1437 006012 000005 RESET ;;CLEAR THE WORLD
1438 006014 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
1439 006016 000240 NOP ;;SAVE ROOM
1440 006020 000240 NOP ;;FOR
1441 006022 000240 NOP ;;ACT11
1442 006024 $DOAGN:
1443 006024 000137 JMP @ (PC)+ ;;RETURN
1444 006026 002204 $RTNAD: ,WORD START
1445 006030 377 000 $ENULL: ,BYTE -1,-1,0
1446 006033 015 042412 042116 $ENDMG: ,ASCIZ <15><12>/END PASS/
1447 006040 050040 051501 000123
    
```

```

1448 .SBTTL SCOPE HANDLER ROUTINE
1449
1450 ;*****
1451 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1452 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1453 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1454 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1455 ;*SW14=1 LOOP ON TEST
1456 ;*SW11=1 INHIBIT ITERATIONS
1457 ;*SW09=1 LOOP ON ERROR
1458 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
1459 ;*CALL
1460 ;* SCOPE ;;SCOPE=IOT
1461
1462 006046 $SCOPE:
1463 006046 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1464 006050 032777 040000 173062 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
1465 006056 001111 BNE $OVER ;;YES IF SW14=1
1466 ;****START OF CODE FOR THE XOR TESTER****
1467 006060 000416 6$: XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1468 ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1469 006062 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1470 006066 012737 006106 000004 MOV #5@,$ERRVEC ;;SET FOR TIMEOUT
1471 006074 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
1472 006100 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
1473 006104 000463 BR $SVLAD ;;GO TO THE NEXT TEST
1474 006106 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
1475 006110 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
1476 006114 000423 BR 7$ ;;LOOP ON THE PRESENT TEST
1477 006116 6$: ;****END OF CODE FOR THE XOR TESTER****
1478 006116 032777 000400 173014 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
1479 006124 001404 BEQ 2$ ;;BR IF NO
1480 006126 127767 173006 172746 CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
1481 006134 001462 BEQ $OVER ;;BR IF YES
1482 006136 105767 172741 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
1483 006142 001421 BEQ 3$ ;;BR IF NO
1484 006144 126767 172745 172731 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1485 006152 101015 BHI 3$ ;;BR IF NO
1486 006154 032777 001000 172756 BIT #BIT09,@SWR ;;LOOP ON ERROR?
1487 006162 001404 BEQ 4$ ;;BR IF NO
1488 006164 016767 172720 172714 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
1489 006172 000443 BR $OVER
1490 006174 105067 172703 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
1491 006200 005067 172762 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1492 006204 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
1493 006206 032777 004000 172724 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
1494 006214 001011 BNE 1$ ;;BR IF YES
1495 006216 005767 172656 TST $PASS ;;IF FIRST PASS OF PROGRAM
1496 006222 001406 BEQ 1$ ;; INHIBIT ITERATIONS
1497 006224 005267 172654 INC $ICNT ;;INCREMENT ITERATION COUNT
1498 006230 026767 172732 172646 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
1499 006236 002021 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
1500 006240 012767 000001 172636 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
1501 006246 016767 000044 172712 $SVLAD: MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
1502 006254 105267 172622 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
1503 006260 011667 172622 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
    
```

```

1504 006264 011667 172620      MOV      (SP),&LPERR      ;;SAVE ERROR LOOP ADDRESS
1505 006270 005067 172674      CLR      $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1506 006274 112767 000001 172613  MOV      #1,&ERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1507 006302 016777 172574 172632 $OVER:  MOV      $TSTNM,&DISPLAY  ;;DISPLAY TEST NUMBER
1508 006310 016716 172572      MOV      $LPADR,(SP)   ;;FUDGE RETURN ADDRESS
1509 006314 000002      RTI      ;;FIXES PS
1510 006316 003720      $MXCNT: 2000.        ;;MAX. NUMBER OF ITERATIONS
    
```

```

1511      .SBTTL  ERROR HANDLER ROUTINE
1512
1513      ;;*****
1514      ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1515      ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1516      ;;AND GO TO TYPERR ON ERROR
1517      ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1518      ;;SW15=1      HALT ON ERROR
1519      ;;SW13=1      INHIBIT ERROR TYPEOUTS
1520      ;;SW10=1      BELL ON ERROR
1521      ;;SW09=1      LOOP ON ERROR
1522      ;;CALL
1523      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1524
1525      $ERROR:
1526 006320 104406      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
1527 006322 011437 001162      MOV      @TACS,@$$REG0  ;;SAVE THE STATUS REG.
1528 006326 011537 001164      MOV      @TADB,@$$REG1  ;;SAVE THE DATA BUFFER
1529 006332 010037 001124      MOV      R0,@$$GDDAT    ;;R0 WILL CONTAIN THE GOOD DATA
1530 006336 010137 001126      MOV      R1,@$$BDDAT    ;;R1 WILL CONTAIN THE BAD DATA
1531 006342 105267 172535      7s:      INC      $ERFLG      ;;SET THE ERROR FLAG
1532 006346 001775      BEQ      7s            ;;DON'T LET THE FLAG GO TO ZERO
1533 006350 016777 172526 172564      MOV      $TSTNM,&DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
1534 006356 032777 002000 172554      BIT      $BIT10,@$SWR   ;;BELL ON ERROR?
1535 006364 001402      BEQ      1s            ;;NO - SKIP
1536 006366 104401 001172      TYPE    $SBELL        ;;RING BELL
1537 006372 005267 172514      1s:      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
1538 006376 011667 172514      MOV      (SP),&ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION
1539 006402 162767 000002 172506      SUB      #2,&ERRPC
1540 006410 117767 172502 172476      MOV      @&ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
1541 006416 032777 020000 172514      BIT      $BIT13,@$SWR   ;;SKIP TYPEOUT IF SET
1542 006424 001004      BNE      20s          ;;SKIP TYPEOUTS
1543 006426 004767 000060      JSR      PC,TYPERR    ;;GO TO USER ERROR ROUTINE
1544 006432 104401 001177      TYPE    $SCRFL
1545 006436
1546 006436 005777 172476      20s:     TST      @SWR          ;;HALT ON ERROR
1547 006442 100002      BPL      3s            ;;SKIP IF CONTINUE
1548 006444 000000      HALT
1549 006446 104406      CKSWR      ;;HALT ON ERROR!
1550 006450 032777 001000 172462 3s:      BIT      $BIT09,@$SWR  ;;TEST FOR CHANGE IN SOFT-SWR
1551 006456 001402      BEQ      4s            ;;LOOP ON ERROR SWITCH SET?
1552 006460 016716 172424      BEQ      4s            ;;BR IF NO
1553 006464 005767 172500      MOV      $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
1554 006470 001402      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
1555 006472 016716 172472      BEQ      5s            ;;BR IF NONE
1556 006476      MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
1557 006476 022737 006014 000042 5s:      CMP      $$ENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
1558 006504 001001      BNE      6s            ;;BRANCH IF NO
1559 006506 000000      HALT
1560 006510      6s:
1561 006510 000002      RTI
1562
    
```

```

1563 ;*****
1564 ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
1565
1566 006512 104401 001177 TYPERR: TYPE ,SCLRF ;TYPE A CARRIAGE RETURN & LINE FEED
1567 006516 010046 MOV R0,-(SP) ;SAVE R0
1568 006520 113700 001114 MOVB @#$ITEMB,R0 ;PICKUP THE ITEM INDEX
1569 006524 005300 DEC R0 ;ADJUST THE INDEX
1570 006526 006300 ASL R0 ;SO IT WILL WORK FOR
1571 006530 006300 ASL R0 ;THE ERROR TABLE
1572 006532 006300 ASL R0
1573 006534 062700 001236 ADD $ERRTB,R0 ;FORM THE TABLE POINTER
1574 006540 012067 000002 MOV (R0)+,1$ ;PICKUP "ERROR MESSAGE" POINTER
1575 006544 104401 TYPE ;TYPE "ERROR MESSAGE"
1576 006546 000000 1$: 0 ;"ERROR MESSAGE POINTER" GOES HERE
1577 006550 104401 TYPE ,SCLRF
1578 006554 012067 000004 MOV (R0)+,2$ ;PICKUP "DATA HEADER" POINTER
1579 006560 001404 BEQ 3$ ;IF "0" DON'T TYPE
1580 006562 104401 TYPE ;TYPE "DATA HEADER"
1581 006564 000000 2$: 0 ;"DATA HEADER" POINTER GOES HERE
1582 006566 104401 001177 TYPE ,SCLRF
1583 006572 012000 3$: MOV (R0)+,R0 ;PICKUP "DATA POINTER"
1584 006574 001004 BNE 5$ ;IF THERE IS DATA TO TYPE GO DO IT
1585 006576 012600 4$: MOV (SP)+,R0 ;RESTORE R0
1586 006600 104401 001177 TYPE ,SCLRF ;TYPE A CARRIAGE RETURN&LINE FEED
1587 006604 000207 RTS PC ;RETURN
1588 006606
1589 006606 013046 5$: MOV @ (R0)+,-(SP) ;SAVE @ (R0)+ FOR TYPEOUT
1590 ;TYPE DATA
1591 006610 104402 TYPDC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1592 006612 005710 TST (R0) ;TERMINATOR?
1593 006614 001770 BEQ 4$ ;BR IF YES
1594 006616 104401 006624 TYPE ,6$ ;TYPE 2 SPACES
1595 006622 000771 BR 5$ ;LOOP
1596 006624 020040 000 6$: ,ASCIZ / / ;ASCII STRING OF 2 SPACES
1597 006630 ,EVEN
    
```

```

1598 ;*****
1599 ;ROUTINE TO WAIT ON THE READY BIT
1600
1601 006630 WAIT,ON,READY:
1602 006630 005067 000044 CLR WAIT2 ;SETUP MAX. TIME TO WAIT ON "READY"
1603 006634 016767 000072 000044 MOV MAXCNT,HGHTIM
1604 006642 012637 001202 MOV (SP)+,@$SAVPC ;GET THE PC OF THE WAITREADY INSTRUCTION
1605 006646 162737 000002 001202 SUB #2,@$SAVPC
1606 006654 012637 001204 MOV (SP)+,@$SAVPS ;SAVE THE PS
1607 006660 032714 000040 WAIT1: BIT $READY,@TACS ;READY=1?
1608 006664 001013 BNE WAIT3 ;GO ON IF YES
1609 006666 105714 TSTB @TACS ;CHECK TRANSFER REQUEST
1610 006670 100002 BPL WAIT4
1611 006672 104004 ERROR 4 ;"TRANSFER REQUEST" SET WHILE WAITING ON "READY"
1612 006674 000407 BR WAIT3
1613 006676 005227 WAIT4: INC (PC)+ ;COUNT FAST COUNTER
1614 006700 000000 WAIT2: 0
1615 006702 001366 BNE WAIT1 ;GO CHECK "READY" AGAIN
1616 006704 005327 DEC (PC)+ ;COUNT LOOP COUNTER
1617 006706 000000 HGHTIM: 0
1618 006710 003363 BGT WAIT1 ;GO LOOP AGAIN
1619 006712 104002 ERROR 2 ;"READY" FAILED TO SET
1620 006714 013746 001204 WAIT3: MOV @$SAVPS,-(SP) ;GET THE STATUS BACK
1621 006720 013746 001202 MOV @$SAVPC,-(SP) ;GET THE PC
1622 006724 062716 000002 ADD #2,(SP)
1623 006730 000002 RTI
1624 006732 000000 MAXCNT: 0
1625
1626 ;*****
1627 ;ROUTINE TO WAIT ON TRANSFER REQUEST
1628
1629 006734 WAIT,FOR,XFER,REQ:
1630 006734 005067 000044 CLR 2$ ;SETUP WASTE TIME LOOP
1631 006740 013767 006732 000044 MOV @MAXCNT,3$
1632 006746 012637 001202 MOV (SP)+,@$SAVPC ;GET THE PC OF THE WAITXFER INSTRUCTION
1633 006752 162737 000002 001202 SUB #2,@$SAVPC
1634 006760 012637 001204 MOV (SP)+,@$SAVPS ;SAVE THE PS
1635 006764 105714 1$: TSTB @TACS ;CHECK XFER REQ
1636 006766 100414 BMI 4$ ;EXIT IF SET
1637 006770 032714 000040 BIT $READY,@TACS ;LOOK AT READY
1638 006774 001402 BEQ 5$ ;BR IF "READY" ISN'T SET
1639 006776 104004 ERROR 4 ;"READY" SET WHILE WAITING FOR "XFER REQ"
1640 007000 000407 BR 4$
1641 007002 005227 5$: INC (PC)+ ;COUNT
1642 007004 000000 2$: 0
1643 007006 001366 BNE 1$ ;BR IF MORE TO DO
1644 007010 005327 DEC (PC)+
1645 007012 000000 3$: 0
1646 007014 003363 BGT 1$
1647 007016 104003 ERROR 3 ;"TRANSFER REQUEST" FAILED TO SET
1648 007020 013746 001204 4$: MOV @$SAVPS,-(SP) ;GET THE STATUS BACK
1649 007024 013746 001202 MOV @$SAVPC,-(SP) ;GET THE PC
1650 007030 062716 000002 ADD #2,(SP)
1651 007034 000002 RTI ;GO BACK
    
```

```

1652 ;*****
1653
1654 .SBTTL ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
1655
1656 ;CALL
1657 ; JSR PC,@#ASKDRV
1658 ; RETURN ;NOTE: R0 AND R1 ARE DESTROYED
1659
1660 007036 104401 012552 ASKDRV: TYPE ,MSGDRV ;<CRLF>"DRIVE(S)? "
1661 007042 005067 172156 CLR DRVKEY
1662 007046 104410 RDLIN ;GO GET A DRIVE
1663 007050 012600 MOV (SP)+,R0 ;SETUP TO CHECK FOR VALID DRIVE(S)
1664 007052 105710 TSTB @R0 ;WAS A DRIVE SELECTED?
1665 007054 001425 BEQ NOTLGL ;BR IF NO
1666 007056 012701 001224 MOV #DRVKEY,R1
1667 007062 122710 000101 LOOP: CMPB #*A,@R0 ;WAS DRIVE "A" SELECTED?
1668 007066 001002 BNE NOTA ;BR IF NO
1669 007070 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "A"
1670 007072 000411 BR NEXT
1671 007074 122710 000102 NOTA: CMPB #*B,@R0 ;WAS DRIVE "B" SELECTED?
1672 007100 001002 BNE NOTB ;BR IF NO
1673 007102 112021 MOVB (R0)+,(R1)+ ;SET KEY FOR DRIVE "B"
1674 007104 000404 BR NEXT
1675 007106 122710 000054 NOTB: CMPB #54,@R0 ;WAS A COMMA TYPED?
1676 007112 001006 BNE NOTLGL ;BR IF NO
1677 007114 105720 TSTB (R0)+ ;DUMP THE COMMA
1678 007116 105710 NEXT: TSTB @R0 ;TERMINATOR?
1679 007120 001406 BEQ EXIT ;BR IF YES
1680 007122 022701 001226 CMP #DRVKEY+2,R1 ;TWO DRIVES SELECTED?
1681 007126 101355 BHI LOOP ;BR IF NO
1682 007130 104401 001176 NOTLGL: TYPE ,$QUES ;ILLEGAL INPUT DETECTED
1683 007134 000740 BR ASKDRV ;GO TRY AGAIN
1684 007136 005767 172062 EXIT: TST DRVKEY ;ANY DRIVE SELECTED?
1685 007142 001772 BEQ NOTLGL ;BR IF NO
1686 007144 000207 RTS PC
1687
    
```

```

1688 ;*****
1689 ;CALL
1690 ;JSR PC,@#ASKADR
1691
1692 007146 010046 ASKADR: MOV R0,-(SP) ;SAVE R0
1693 007150 104401 012566 1s: TYPE ,MSGASK ;"TACS?"
1694 007154 104411 RDOCT ;GET VALUE
1695 007156 012600 MOV (SP)+,R0 ;PICK UP THE OCTAL NUMBER
1696 007160 001423 BEQ 3s ;IF "0" USE OLD VALUES
1697 007162 020027 160000 CMP R0,#160000 ;MAKE SURE IT IS A BUS ADDRESS
1698 007166 103770 BLO 1s
1699 007170 010037 001206 MOV R0,@#TACSL ;SAVE TOE TACS
1700 007174 062700 000002 ADD #2,R0 ;STEP TO TADB ADDRESS
1701 007200 010037 001212 MOV R0,@#TADBL ;AND SAVE IT
1702 007204 013737 001206 001210 MOV @#TACSL,@#TACSH ;SET UP TACS UPPER
1703 007212 005237 001210 INC @#TACSH ;BYTE POINTER
1704
1705 007216 013737 001212 001214 MOV @#TADBL,@#TADBH ;SET UP TADB UPPER
1706 007224 005237 001214 INC @#TADBH ;BYTE POINTER
1707 007230 104401 012576 3s: TYPE ,MSGVEC ;"VECTOR?"
1708 007234 104411 RDOCT
1709 007236 012600 MOV (SP)+,R0
1710 007240 001411 BEQ 5s
1711 007242 020027 001000 CMP R0,#1000 ;MAKE SURE ADDRESS IS IN VECTOR AREA
1712 007246 103370 BHIS 3s
1713 007250 010037 001216 MOV R0,@#TAVEC ;SAVE AS VECTOR ADDRESS
1714 007254 062700 000002 ADD #2,R0
1715 007260 010037 001220 MOV R0,@#TAVEC+2
1716 007264 104401 012606 5s: TYPE ,MSGPRI ;ASK FOR PRIORITY
1717 007270 104411 RDOCT
1718 007272 012600 MOV (SP)+,R0
1719 007274 001413 BEQ 6s ;IF "0" USE OLD VALUE
1720 007276 020027 000007 CMP R0,#7 ;MAKE SURE ITS VALID
1721 007302 101370 BHI 5s
1722 007304 000300 SWAB R0 ;PUT INTO HIGH BYTE
1723 007306 006200 ASR R0 ;AND SHIFT
1724 007310 006200 ASR R0 ;INTO PROPER
1725 007312 006200 ASR R0 ;POSITION
1726 007314 042700 177437 BIC #<C340>,R0 ;SAVE ONLY PRIORITY BITS
1727 007320 010037 001222 MOV R0,@#TAPRIO ;STORE IT AWAY
1728 007324 104401 012620 6s: TYPE ,MTACS ;TACS="
1729 007330 016746 171652 MOV TACSL,-(SP) ;;SAVE TACSL FOR TYPEOUT
1730 007334 104402 TPOCT ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1731 007336 104401 012626 TYPE ,MTADB ;"TADB="
1732 007342 016746 171644 TADBL,-(SP) ;;SAVE TADBL FOR TYPEOUT
1733 007346 104402 TPOCT ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1734 007350 104401 012635 TYPE ,MTAVEC ;"VECTOR="
1735 007354 016746 171636 MOV TAVEC,-(SP) ;;SAVE TAVEC FOR TYPEOUT
1736 007360 104402 TPOCT ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1737 007362 104401 012646 TYPE ,MTAPRIO ;"PRIORITY="
1738 007366 016746 171630 MOV TAPRIO,-(SP) ;;SAVE TAPRIO FOR TYPEOUT
1739 007372 104402 TPOCT ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1740 007374 104401 012661 TYPE ,MSGOK ;"OK?"
1741 007400 104407 RDCHR ;GO READ ONE CHARACTER
1742 007402 012600 MOV (SP)+,R0 ;GET IT
1743 007404 022700 000015 CMP #15,R0 ;IS IT "CR"?
    
```

```

1744 007410 001406      BEQ      78      ;BRANCH IF YES
1745 007412 022700 000131    CMP      #'Y,R0    ;IS IT "Y"?
1746 007416 001403      BEQ      78      ;IT WAS
1747 007420 104401 001176    TYPE    ,SQUES    ;TYPE "?"
1748 007424 000651      BR       18      ;AND LET HIM CORRECT THEM
1749 007426 104401 012667    78:    TYPE    ,MYES ;TYPE OUT "YES"
1750 007432 012600      MOV      (SP)+,R0 ;RESTORE R0
1751 007434 000207      RTS      PC      ;AND RETURN
    
```

```

1752                                     ;*****
1753                                     ;*****
1754 ;ROUTINE TO TYPE DIRECTIONS TO OPERATOR AND WAIT FOR RESPONSE
1755 ;CALL:
1756 ;       JSR      R0,#ASKQUES
1757 ;       RETURN  HERE ON A RESPONSE OF "Y"
1758 ;       RETURN  HERE ON A RESPONSE OF "N"
1759
1760 007436      ASKQUES:
1761 007436 010146      MOV      R1,-(SP)      ;SAVE R1
1762 007440 012067 000006    MOV      (R0)+,2S    ;PICKUP THE MESSAGE POINTER
1763 007444 104401 001177    TYPE    ,SCLRF     ;TYPE A "CR" AND "LF"
1764 007450 104401      1S:    TYPE    ,TYPE THE MESSAGE
1765 007452 000000      2S:    0          ;MESSAGE POINTER GOES HERE
1766 007454 104407      RDCHR     ;GET ONE CHARACTER FROM THE TTY
1767 007456 012601      MOV      (SP)+,R1    ;AND PUT IT IN R1
1768 007460 020127 000015    CMP      R1,#15     ;CHECK THIS CHARACTER
1769 007464 001443      BEQ      4S          ;BR IF ITS A "CR"
1770 007466 020127 000016    CMP      R1,#"N"    ;IS IT A "N"?
1771 007472 001415      BEQ      3S          ;BR IF YES
1772 007474 020127 000013    CMP      R1,#"Y"    ;IS IT A "Y"
1773 007500 001007      BNE      5S          ;BR IF NOT "Y"
1774 007502 104401 007510    TYPE    ,65$        ;;TYPE ASCIZ STRING
1775 007506 000403      BR       64$        ;;GET OVER THE ASCIZ
1776
1777 007516      ;;65$: .ASCIZ / YES/
1778 007516 000426      64$:    BR       4S          ;GO TO EXIT
1779 007520 104401 001176    5S:    TYPE    ,SQUES    ;UNKNOWN REQUEST
1780 007524 000751      BR       1S          ;GO ASK AGAIN
1781 007526
1782 007526 104401 007534    3S:    TYPE    ,67$        ;;TYPE ASCIZ STRING
1783 007532 000415      BR       66$        ;;GET OVER THE ASCIZ
1784
1785 007566      ;;67$: .ASCIZ * NO---SKIPPING THIS TEST*
1786 007566 005720 .      66$:    TST      (R0)+      ;STEP OVER "CR" RETURN
1787 007570 005037 001166    CLR      0$TIMES    ;CLEAR ITERATIONS
1788 007574 104401 001177    4S:    TYPE    ,SCLRF
1789 007600 012601      MOV      (SP)+,R1    ;RESTORE R1
1790 007602 000200      RTS      R0        ;RETURN
1791
    
```

```

1792 ;////////////////////////////////////
1793 ;////////////////////////////////////
1794 ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TUGO
1795 ;NOTE: *** BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ***
1796 ;////////////////////////////////////
1797 ;////////////////////////////////////
1798 ;////////////////////////////////////
1799 ;////////////////////////////////////
1800 ;*****
1801 ; WRITE FILE GAPS FROM "BOT" TO "EOT"
1802 ;
1803 ;START AT 220
1804 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
1805 ;THE "WRITE DELAY MONO".
1806 ;*****
1806 007604 012706 001100 WFGSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1807 007610 013704 001206 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
1808 007614 013705 001212 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
1809 007620 000005 RESET ;RESET THE WORLD
1810 007622 012737 007604 001110 MOV #WFGSUB,##$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
1811 007630 004737 010610 JSR PC,##NXTDRV ;GO SETUP FOR NEXT DRIVE
1812 007634 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1813 007636 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
1814 007642 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1815 007646 001775 BEQ 100s
1816 007650 112714 000001 1s: MOV# #WFG!GO,@TACS ;WRITE A FILE GAP
1817 007654 104412 WAITREADY ;WAIT ON READY
1818 007656 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1819 007662 001772 BEQ 1s ;BR IF NO
1820 007664 000000 HALT ;STOP IF YES
1821 007666 000746 BR WFGSUB ;LOOP ON CONT.
1822
1823
1824 ;*****
1825 ; WRITE CONTINUOUS BLOCKS OF DATA
1826 ;
1827 ;START AT 224
1828 ;THE PROGRAM WILL HALT THREE(3) TIMES
1829 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
1830 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
1831 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
1832 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
1833 ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
1834 ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
1835 ; EACH HALT OPERATOR WILL BE PROMPTED
1836 ; FOR THE VALUE WITH "SWR=XXXXXX NEW="
1837 ;*****
1837 007670 004737 010352 WRTSUB: JSR PC,##SETBUF ;GET BLOCK SIZE AND PATTERN
1838 007674 012706 001100 WLOOP: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1839 007700 013704 001206 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
1840 007704 013705 001212 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
1841 007710 000005 RESET ;RESET THE WORLD
1842 007712 012737 007674 001110 MOV #WLOOP,##$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
1843 007720 004737 010610 JSR PC,##NXTDRV ;GO SETUP FOR NEXT DRIVE
1844 007724 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1845 007726 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
1846 007732 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1847 007736 001775 BEQ 100s
    
```

```

1848 007740 004737 010466 1s: JSR PC,##WRTBLK ;WRITE A BLOCK
1849 007744 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1850 007750 001773 BEQ 1s ;BR IF NO
1851 007752 000000 HALT ;STOP IF "EOT"
1852 007754 000747 BR WLOOP ;LOOP IF CONT.
1853
1854
1855 ;*****
1856 ; READ CONTINUOUS BLOCKS OF DATA
1857 ;
1858 ;START AT 230
1859 ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
1860 ;AND THE "THRESHOLD POT".
1861 ;*****
1861 007756 012706 001100 RDSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1862 007762 013704 001206 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
1863 007766 013705 001212 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
1864 007772 000005 RESET ;RESET THE WORLD
1865 007774 012737 007756 001110 MOV #RDSUB,##$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
1866 010002 004737 010610 JSR PC,##NXTDRV ;GO SETUP FOR NEXT DRIVE
1867 010006 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1868 010010 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
1869 010014 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1870 010020 001775 BEQ 100s
1871 010022 004737 010530 1s: JSR PC,##RDBLK ;READ A BLOCK
1872 010026 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1873 010032 001351 BNE RDSUB ;BR IF YES---LOOP
1874 010034 000772 BR 1s
1875
1876
1877 ;*****
1878 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
1879 ;
1880 ;START AT 234
1881 ;THE PROGRAM WILL HALT THREE(3) TIMES
1882 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
1883 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
1884 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
1885 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
1886 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
1887 ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
1888 ; EACH HALT OPERATOR WILL BE PROMPTED
1889 ; FOR THE VALUE WITH "SWR=XXXXXX NEW="
1890 ;AND THE "GAP TIME MONO".
1891 ;*****
1891 010036 004737 010352 WGPBLK: JSR PC,##SETBUF ;GET BLOCK SIZE AND PATTERN
1892 010042 012706 001100 WGBLOP: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1893 010046 013704 001206 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
1894 010052 013705 001212 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
1895 010056 000005 RESET ;RESET THE WORLD
1896 010060 012737 010042 001110 MOV #WGBLOP,##$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
1897 010066 004737 010610 JSR PC,##NXTDRV ;GO SETUP FOR NEXT DRIVE
1898 010072 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1899 010074 112714 000017 MOV# #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
1900 010100 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1901 010104 001775 BEQ 100s
1902 010106 112714 000001 1s: MOV# #WFG!GO,@TACS ;WRITE A FILE GAP
1903 010112 104412 WAITREADY ;WAIT ON READY
    
```

```

1904 010114 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1905 010120 001005 BNE 26 ;BR IF YES
1906 010122 004737 010466 JSR PC,@#WRTBLK ;WRITE A BLOCK
1907 010126 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1908 010132 001765 BEQ 18 ;BR IF NO
1909 010134 000000 26: HALT ;STOP AT "EOT"
1910 010136 000741 BR WGBLOP ;START OVER ON CONT.
1911
1912
1913 ;*****
1914 ; READ A BLOCK OF DATA AND A FILE GAP
1915 ;START AT 240
1916 ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
1917 ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
1918 ;AND THE "TAPE BLANK MONO".
1919 ;*****
1920 010140 012706 001100 RGPBLK: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1921 010144 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
1922 010150 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
1923 010154 000005 RESET ;RESET THE WORLD
1924 010156 012737 010140 001110 MOV #RGPBLK,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
1925 010164 004737 010610 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
1926 010170 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1927 010172 112714 000017 MOV #REWINDIGO,@TACS ;SEND TAPE TO "BOT"
1928 010176 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1929 010202 001775 BEQ 100s
1930 010204 004737 010530 15: JSR PC,@#RDBLK ;READ A BLOCK OF DATA
1931 010210 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1932 010214 001351 BNE RGPBLK ;BR IF YES
1933 010216 112714 000015 MOV #SFBG!GO,@TACS ;GET INTO A FILE GAP
1934 010222 104412 WAITREADY
1935 010224 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1936 010230 001343 BNE RGPBLK ;BR IF YES
1937 010232 000764 BR 15 ;LOOP
1938
1939
1940 ;*****
1941 ; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
1942 ;START AT 244
1943 ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
1944 ;SPACE FORWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
1945 ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
1946 ;(SIGNAL MONO CAN BE CHECKED).
1947 ;*****
1948 010234 012706 001100 SFFGSB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
1949 010240 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
1950 010244 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
1951 010250 000005 RESET ;RESET THE WORLD
1952 010252 012737 010234 001110 MOV #SFFGSB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
1953 010260 004737 010610 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
1954 010264 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1955 010266 112714 000017 MOV #REWINDIGO,@TACS ;SEND TAPE TO "BOT"
1956 010272 032714 000040 100s: BIT #READY,@TACS ;WAIT ON READY
1957 010276 001775 BEQ 100s
1958 010300 112714 000013 15: MOV #SFFG!GO,@TACS ;SPACE INTO A FILE GAP
1959 010304 104412 WAITREADY ;WAIT ON READY
    
```

```

1960 010306 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1961 010312 001772 BEQ 18 ;BR IF NO
1962 010314 000000 HALT ;STOP AT "EOT"
1963 010316 000746 BR SFFGSB ;LOOP ON CONT.
1964
1965
1966 ;*****
1967 ; BACK SPACE FILE GAP
1968 ;START AT 250
1969 ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
1970 ;*****
1971 010320 000005 BSFGSB: RESET ;RESET THE WORLD
1972 010322 012737 010320 001110 MOV #BSFGSB,@#SLPERR ;LOOP ON ERROR ADDRESS
1973 010330 010314 MOV DRIVE,@TACS ;SELECT DRIVE
1974 010332 112714 000007 15: MOV #BSFG!GO,@TACS ;BACK SPACE A FILE GAP
1975 010336 104412 WAITREADY ;WAIT ON READY
1976 010340 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
1977 010344 001772 BEQ 18 ;BR IF NO
1978 010346 000000 HALT ;STOP AT BOT
1979 010350 000763 BR BSFGSB ;START OVER ON CONT.
1980
1981
1982 ;*****
1983 ; SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
1984 ;*****
1985 010352 005000 SETBUF: CLR R0
1986 010354 000000 HALT
1987 010356 022767 000176 170554 CMP #SWREG,SWR ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
1988 010364 001001 BNE 20s ;USING S/W SWITCH REG?
1989 010366 104405 GTSWR ;NO- GET OUT
1990 010370 000000 ;GET VALUE
1991 010370 157700 170544 20s: BISB @SWR,R0 ;PICKUP THE BYTE COUNT
1992 010374 001006 BNE 26 ;BR IF NON-ZERO
1993 010376 105777 170537 TSTB @SWR+1 ;CHECK IF GREATER THAN 377
1994 010402 001402 BEQ 18 ;BR IF NO
1995 010404 012700 000376 MOV #376,R0 ;SET FOR MAX ALLOWED
1996 010410 005200 18: INC R0 ;MAKE IT 377 OR 1
1997 010412 010037 010462 2s: MOV R0,@#BLKLIM ;SETUP THE BLOCK LIMIT
1998 010416 005037 010464 CLR @PATTRN
1999 010422 000000 HALT
2000 010424 022767 000176 170506 CMP #SWREG,SWR ;OPERATOR PUTS PATTERN IN SWR<7:0>
2001 010432 001001 BNE 21s ;USING S/W SWITCH REG?
2002 010434 104405 GTSWR ;NO- GET OUT
2003 010436 000000 ;GET VALUE
2004 010436 117737 170476 010464 21s: MOV #SWR,@#PATTRN ;PICK UP THE PATTERN
2005 010444 000000 HALT ;SET OPERATIONAL SWITCHES
2006 010446 022767 000176 170464 22s: CMP #SWREG,SWR ;USING S/W SWITCH REG?
2007 010454 001001 BNE 22s ;NO- GET OUT
2008 010456 104405 GTSWR ;GET VALUE
2009 010460 000000 22s: ;CONTINUE
2010 010460 000207 RTS PC ;RETURN
2011 010462 000000 BLKLIM: 0 ;READ AND WRITE BLOCK SIZE
2012 010464 000000 PATTRN: 0 ;PATTERN TO GO ON THE TAPE
2013
2014
2015 ;*****
    
```

```

2016 ; WRITE ROUTINE FOR THE MANUAL OPERATIONS
2017 ;*****
2018 WRTBLK: MOV @#BLKLM,R1 ;PICKUP THE BLOCK SIZE
2019 MOVB #WRITE!GO,@TACS ;START A WRITE
2020 1S: WAITXFER ;WAIT ON TRANSFER REQUEST
2021 BIT #READY,@TACS ;DID READY SET?
2022 BNE 3S ;BR IF YES
2023 DEC R1 ;COUNT THIS REQUEST
2024 BLT 2S ;BR IF TIME FOR ILBS
2025 MOVB @#PATRN,@TADB ;PUT DATA ON TAPE
2026 BR 1S ;LOOP
2027 2S: BIS #ILBS,@TACS ;WRITE CRC AND SHUT DOWN
2028 WAITREADY ;WAIT ON THE READY FLAG
2029 3S: RTS PC
2030
2031
2032 ;*****
2033 ; READ ROUTINE FOR THE MANUAL OPERATIONS
2034 ;*****
2035 RDBLK: MOV @#BLKLM,R2 ;PICKUP THE BLOCK SIZE
2036 MOV @#PATRN,R0 ;USE THIS DATA PATTERN TO COMPARE TO
2037 MOVB #READ!GO,@TACS ;START A READ
2038 1S: WAITXFER ;WAIT ON TRANSFER REQUEST
2039 BIT #READY,@TACS ;IS READY SET?
2040 BNE 3S ;BR IF YES
2041 DEC R2 ;COUNT THIS REQUEST
2042 BLT 2S ;BR IF TIME FOR ILBS
2043 MOV @TADB,R1 ;READ THE DATA BUFFER
2044 CMPB R0,R1 ;CHECK THE DATA
2045 BEQ 1S ;BR IF OK
2046 ERROR 5 ;BAD DATA
2047 BR 4S ;GET OUT
2048 2S: BIS #ILBS,@TACS ;READ ILBS
2049 WAITREADY ;WAIT ON READY
2050 TST @TACS ;CHECK FOR ERROR
2051 BPL 4S ;BR IF NONE
2052 ERROR 1 ;ERROR OCCURRED
2053 RTS PC ;RETURN
2054
2055 ;*****
2056 ; ROUTINE TO CHANGE DRIVES
2057 NXTDRV: TSTB @SWR ;IS SW07 ON A (1)?
2058 BMI 3S ;BR IF YES
2059 CLK DRIVE ;SET DRIVE TO "A"
2060 MOV @#DRVPT,R1 ;GET DRIVE POINTER
2061 CMPB (R1)+,#"A" ;IS IT DRIVE "A"?
2062 BEQ 1S ;BR IF YES
2063 MOV #UNIT,DRIVE ;SET DRIVE TO "B"
2064 1S: TSTB (R1) ;LAST DRIVE BEEN SELECTED
2065 BNE 2S ;BR IF NO
2066 MOV @#DRVKEY,R1 ;RESET DRIVE POINTER
2067 2S: MOV R1,@#DRVPT ;SAVE DRIVE POINTER FOR NEXT TIME
2068 3S: RTS PC ;GO BACK
2069

```

```

2070 ;*****
2071 ;
2072 ;SBTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2073
2074 ;CALL:
2075 ; MOV #DRVKEY,R0
2076 ; JSR PC,@#EXAM ;R1 IS DESTROYED
2077 ; NORMAL RETURN
2078 ; ERROR RETURN
2079
2080 EXAM: MOV @#TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2081 CLR (R1) ;DRIVE="A", FUNCTION="WFG"
2082 CMPB #"A,(R0) ;EXAMINE DRIVE "A"?
2083 BEQ 1S ;BR IF YES
2084 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2085 1S: BIT #READY,(R1) ;WAIT ON READY
2086 BEQ 1S
2087 TST (R1) ;ANY ERROR?
2088 BPL 4S ;BR IF NO
2089 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2090 BNE 3S ;BR IF YES
2091 BIT #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2092 BEQ 2S ;BR IF NO
2093 CMPB #BIT07!BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
2094 BEQ 4S ;BR IF YES
2095 CMPB #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
2096 BEQ 4S ;BR IF YES
2097 BR 3S ;TAKE THE ERROR EXIT
2098 2S: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2099 BNE 4S ;BR IF YES
2100 3S: ADD #2,(SP) ;TAKE ERROR RETURN
2101 4S: RTS PC ;RETURN

```



```

2102                .SBTTL TYPE ROUTINE
2103                ;*****
2104                ;*ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE.
2105                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2106                ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2107                ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2108                ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2109                ;*
2110                ;*CALL:
2111                ;*1) USING A TRAP INSTRUCTION
2112                ;*      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2113                ;*OR
2114                ;*      TYPE      ,MESADR
2115                ;*
2116                ;*
2117                ;*
2118                ;*
2119 010760 105767 170173 STYPE: TSTB  $TPFLG          ;;IS THERE A TERMINAL?
2120 010764 100002          BPL      1$          ;;BR IF YES
2121 010766 000000          HALT          ;;HALT HERE IF NO TERMINAL
2122 010770 000407          BR      3$          ;;LEAVE
2123 010772 010046 1$:    MOV      R0,-(SP)      ;;SAVE R0
2124 010774 017600 000002 MOV      02(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
2125 011000 112046 2$:    MOVB     (R0)+,(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2126 011002 001005          BNE     4$          ;;BR IF IT ISN'T THE TERMINATOR
2127 011004 005726          TST     (SP)+        ;;IF TERMINATOR POP IT OFF THE STACK
2128 011006 012600 60$:   MOV      (SP)+,R0      ;;RESTORE R0
2129 011010 062716 000002 3$:    ADD     #2,(SP)        ;;ADJUST RETURN PC
2130 011014 000002          RTI          ;;RETURN
2131 011016 122716 4$:    CMPB     #HT,(SP)        ;;BRANCH IF <HT>
2132 011022 001430          BEQ     8$          ;;BRANCH IF NOT <CRLF>
2133 011024 122716 000200 CMPB     #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
2134 011030 001006          BNE     5$          ;;BRANCH IF NOT <CRLF>
2135 011032 005726          TST     (SP)+        ;;POP <CR><LF> EQUIV
2136 011034 104401          TYPE          ;;TYPE A CR AND LF
2137 011036 001177          $CRLF          ;;TYPE A CR AND LF
2138 011040 105067 000130 CLRBR   $CHARCNT      ;;CLEAR CHARACTER COUNT
2139 011044 000755          BR      2$          ;;GET NEXT CHARACTER
2140 011046 004767 000056 5$:    JSR     PC,$STYPEC      ;;GO TYPE THIS CHARACTER
2141 011052 126726 170100 6$:    CMPB     $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
2142 011056 001350          BNE     2$          ;;IF NO GO GET NEXT CHAR.
2143 011060 016746 170070 MOV      $NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
2144                ;;AND THE NULL CHAR.
2145 011064 105366 000001 7$:    DECB     1(SP)        ;;DOES A NULL NEED TO BE TYPED?
2146 011070 002770          BLT     6$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
2147 011072 004767 000032 JSR     PC,$STYPEC      ;;GO TYPE A NULL
2148 011076 105367 000072 DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
2149 011102 000770          BR      7$          ;;LOOP
2150
2151                ;HORIZONTAL TAB PROCESSOR
2152
2153 011104 112716 000040 8$:    MOVBR   #' ,(SP)        ;;REPLACE TAB WITH SPACE
2154 011110 004767 000014 9$:    JSR     PC,$STYPEC      ;;TYPE A SPACE
2155 011114 132767 000007 000052 BITBR   #7,$CHARCNT      ;;BRANCH IF NOT AT
2156 011122 001372          BNE     9$          ;;TAB STOP
2157 011124 005726          TST     (SP)+        ;;POP SPACE OFF STACK
    
```

```

2158 011126 000724          BR      2$          ;;GET NEXT CHARACTER
2159 011130 105777 170014 STYPE: TSTB  $STPS          ;;WAIT UNTIL PRINTER IS READY
2160 011134 100375          BPL     $STYPEC      ;;WAIT UNTIL PRINTER IS READY
2161 011136 116677 000002 170006 MOVBR   2(SP),$STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2162 011144 122766 000015 000002 CMPBR   #CR,2(SP)        ;;IS CHARACTER A CARRIAGE RETURN?
2163 011152 001003          BNE     1$          ;;BRANCH IF NO
2164 011154 105067 000014 CLRBR   $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
2165 011160 000406          BR      $TYPEX        ;;EXIT
2166 011162 122766 000012 000002 1$:    CMPBR   #LF,2(SP)        ;;IS CHARACTER A LINE FEED?
2167 011170 001402          BEQ     $TYPEX        ;;BRANCH IF YES
2168 011172 105227          INCB     (PC)+        ;;COUNT THE CHARACTER
2169 011174 000000          SCHARCNT:,$WORD 0     ;;CHARACTER COUNT STORAGE
2170 011176 000207          STYPEX: RTS     PC
2171
2172                .SBTTL TTY INPUT ROUTINE
2173
2174                ;*****
2175                .ENABL LSB
2176
2177                ;*****
2178                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2179                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2180                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2181                ;*WHEN OPERATING IN TTY FLAG MODE.
2182 011200 022767 000176 167732 SCKSWR: CMP     $SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
2183 011206 001074          BNE     15$          ;;BRANCH IF NO
2184 011210 105777 167730 TSTB   $STKS          ;;CHAR THERE?
2185 011214 100071          BPL     15$          ;;IF NO, DON'T WAIT AROUND
2186 011216 117746 167724 MOVBR   $STKB,-(SP)      ;;SAVE THE CHAR
2187 011222 042716 177600 BIC     #'C177,(SP)      ;;STRIP-OFF THE ASCII
2188 011226 022726 000007 CMP     #7,(SP)+        ;;IS IT A CONTROL G?
2189 011232 001062          BNE     15$          ;;NO, RETURN TO USER
2190 011234 126727 167674 000001 CMPBR   $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
2191 011242 001456          BEQ     15$          ;;BRANCH IF YES
2192
2193 011244 104401 011725 SGTSWR: TYPE     $CNTLG      ;;ECHO THE CONTROL-G ("G")
2194 011250 104401 011732 TYPE     $MSWR          ;;TYPE CURRENT CONTENTS
2195 011254 016746 166716 MOV      SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
2196 011260 104402          TYPEPC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2197 011262 104401 011743 TYPE     $MNEW          ;;PROMPT FOR NEW SWR
2198 011266 005046 19$:    CLR     -(SP)        ;;CLEAR COUNTER
2199 011270 005046          CLR     -(SP)        ;;THE NEW SWR
2200 011272 105777 167646 7$:    TSTB   $STKS          ;;CHAR THERE?
2201 011276 100375          BPL     7$          ;;IF NOT TRY AGAIN
2202
2203 011300 117746 167642 MOVBR   $STKB,-(SP)      ;;PICK UP CHAR
2204 011304 042716 177600 BIC     #'C177,(SP)      ;;MAKE IT 7-BIT ASCII
2205
2206
2207
2208 011310 021627 000025 9$:    CMP     (SP),#25        ;;IS IT A CONTROL-U?
2209 011314 001005          BNE     10$          ;;BRANCH IF NOT
2210 011316 104401 011720 TYPE     $CNTLU        ;;YES, ECHO CONTROL-U ("U")
2211 011322 062706 000006 20$:   ADD     #6,SP          ;;IGNORE PREVIOUS INPUT
2212 011326 000757          BR      19$          ;;LET'S TRY IT AGAIN
    
```

```

2214
2215 011330 021627 000015      10s:  CMP      (SP),#15      ;;IS IT A <CR>?
2216 011334 001022              BNE      16s              ;;BRANCH IF NO
2217 011336 005766 000004      TST      4(SP)           ;;YES, IS IT THE FIRST CHAR?
2218 011342 001403              BEQ      11s              ;;BRANCH IF YES
2219 011344 016677 000002 167566  MOV      2(SP),@SWR      ;;SAVE NEW SWR
2220 011352 062706 000006      ADD      #6,SP           ;;CLEAR UP STACK
2221 011356 104401 001177      TYPE    #,SCLF          ;;ECHO <CR> AND <LF>
2222 011362 126727 167547 000001  CMPB    @INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
2223 011370 001003              BNE      15s              ;;BRANCH IF NOT
2224 011372 012777 000100 167544  MOV      #100,@STKS     ;;RE-ENABLE TTY KBD INTERRUPTS
2225 011400 000002              RTI                      ;;RETURN
2226 011402 004767 177522      JSR     PC,$TYPEC       ;;ECHO CHAR
2227 011406 021627 000060      CMP      (SP),#60       ;;CHAR < 0?
2228 011412 002420              BLT     18s              ;;BRANCH IF YES
2229 011414 021627 000067      CMP      (SP),#67       ;;CHAR > 7?
2230 011420 003015              BGT     18s              ;;BRANCH IF YES
2231 011422 042726 000060      BIC     #60,(SP)+       ;;STRIP-OFF ASCII
2232 011426 005766 000002      TST     2(SP)           ;;IS THIS THE FIRST CHAR
2233 011432 001403              BEQ     17s              ;;BRANCH IF YES
2234 011434 006316              ASL     (SP)            ;;NO, SHIFT PRESENT
2235 011436 006316              ASL     (SP)            ;; CHAR OVER TO MAKE
2236 011440 006316              ASL     (SP)            ;; ROOM FOR NEW ONE.
2237 011442 005266 000002      INC     2(SP)           ;;KEEP COUNT OF CHAR
2238 011446 056616 177776      BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
2239 011452 000707              BR      7s              ;;GET THE NEXT ONE
2240 011454 104401 001176      TYPE    #,SQUES        ;;TYPE ?<CR><LF>
2241 011460 000720              BR      20s             ;;SIMULATE CONTROL-U
2242
2243      .DSABL  LSB
2244
2245      ;*****
2246      ;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2247      ;CALL:
2248      ;*  RDCHR              ;;INPUT A SINGLE CHARACTER FROM THE TTY
2249      ;*  RETURN HERE      ;;CHARACTER IS ON THE STACK
2250      ;*                    ;;WITH PARITY BIT STRIPPED OFF
2251      ;
2252
2253 011462 011646              SRDCHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC
2254 011464 016666 000004 000002  MOV      4(SP),2(SP)    ;;SAVE THE PS
2255 011472 105777 167446      1s:  TSTB    @STKS        ;;WAIT FOR
2256 011476 100375              BPL     1s              ;;A CHARACTER
2257 011500 117766 167442 000004  MOVB    @STKB,4(SP)     ;;READ THE TTY
2258 011506 042766 177600 000004  BIC     #"C<177>,4(SP) ;;GET RID OF JUNK IF ANY
2259 011514 026627 000004 000023  CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2260 011522 001013              BNE     3s              ;;BRANCH IF NO
2261 011524 105777 167414      2s:  TSTB    @STKS        ;;WAIT FOR A CHARACTER
2262 011530 100375              BPL     2s              ;;LOOP UNTIL ITS THERE
2263 011532 117746 167410      MOVB    @STKB,-(SP)    ;;GET CHARACTER
2264 011536 042716 177600      BIC     #"C<177>,(SP)  ;;MAKE IT 7-BIT ASCII
2265 011542 026627 000021      CMP     (SP)+,#21      ;;IS IT A CONTROL-Q?
2266 011546 001366              BNE     2s              ;;IF NOT DISCARD IT
2267 011550 000750              BR      1s              ;;YES, RESUME
2268 011552 026627 000004 000140  3s:  CMP     4(SP),#140    ;;IS IT UPPER CASE?
2269 011560 002407              BLT     4s              ;;BRANCH IF YES

```

```

2270 011562 026627 000004 000175      CMP     4(SP),#175     ;;IS IT A SPECIAL CHAR?
2271 011570 003003              BGT     4s              ;;BRANCH IF YES
2272 011572 042766 000040 000004  BIC     #40,4(SP)     ;;MAKE IT UPPER CASE
2273 011600 000002      4s:  RTI                      ;;GO BACK TO USER
2274      ;*****
2275      ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2276      ;CALL:
2277      ;*  RDLIN              ;;INPUT A STRING FROM THE TTY
2278      ;*  RETURN HERE      ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2279      ;*                    ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2280
2281 011602 010346              SRDLIN: MOV      R3,-(SP) ;;SAVE R3
2282 011604 012703 011710      1s:  MOV     $TTYIN,R3   ;;GET ADDRESS
2283 011610 022703 011720      2s:  CMP     $TTYIN+8,,R3 ;;BUFFER FULL?
2284 011614 101405              BLOS    4s              ;;BR IF YES
2285 011616 104407              RDCHR   ;;GO READ ONE CHARACTER FROM THE TTY
2286 011620 112613              MOVB    (SP)+,(R3)     ;;GET CHARACTER
2287 011622 122713 000177      10s:  CMPB    #177,(R3)   ;;IS IT A RUBOUT
2288 011626 001003              BNE     3s              ;;SKIP IF NOT
2289 011630 104401 001176      4s:  TYPE    #,SQUES      ;;TYPE A '?'
2290 011634 000763              BR      1s              ;;CLEAR THE BUFFER AND LOOP
2291 011636 111367 000044      3s:  MOVB    (R3),9s     ;;ECHO THE CHARACTER
2292 011642 104401 011706              TYPE    #,9s
2293 011646 122723 000015      CMPB    #15,(R3)+     ;;CHECK FOR RETURN
2294 011652 001356              BNE     2s              ;;LOOP IF NOT RETURN
2295 011654 105063 177777      CLRB    -1(R3)        ;;CLEAR RETURN (THE 15)
2296 011660 104401 001200      TYPE    #,SLF         ;;TYPE A LINE FEED
2297 011664 012603              MOV     (SP)+,R3      ;;RESTORE R3
2298 011666 011646              MOV     (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2299 011670 016666 000004 000002  MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
2300 011676 012766 011710 000004  MOV     $TTYIN,4(SP)
2301 011704 000002              RTI                      ;;RETURN
2302 011706 000              9s:  .BYTE    0            ;;STORAGE FOR ASCII CHAR. TO TYPE
2303 011707 000              .BYTE    0            ;;TERMINATOR
2304 011710 000010      $TTYIN: .BLKB    8     ;;RESERVE 8 BYTES FOR TTY INPUT
2305 011720 005236 005015 000      $CNTLU: .ASCIZ  /"U/<15><12> ;;CONTROL "U"
2306 011725 136 006507 000012      $CNTLG: .ASCIZ  /"G/<15><12> ;;CONTROL "G"
2307 011732 005015 053523 020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
2308 011740 020075 000
2309 011743 040 047040 053505  $MNEW:  .ASCIZ  / NEW = /
2310 011750 036440 000040
2311      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2312
2313      ;*****
2314      ;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2315      ;CHANGE IT TO BINARY.
2316      ;CALL:
2317      ;*  RDOCT              ;;READ AN OCTAL NUMBER
2318      ;*  RETURN HERE      ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2319      ;*                    ;;HIGH ORDER BITS ARE IN SHIOCT
2320
2321 011754 011646              SRDOCT: MOV      (SP),-(SP) ;;PROVIDE SPACE FOR THE
2322 011756 016666 000004 000002  MOV     4(SP),2(SP)    ;;INPUT NUMBER
2323 011764 010046              MOV     R0,-(SP)      ;;PUSH R0 ON STACK
2324 011766 010146              MOV     R1,-(SP)      ;;PUSH R1 ON STACK
2325 011770 010246              MOV     R2,-(SP)      ;;PUSH R2 ON STACK

```

```

2326 011772 104410          1s:  RDLIN          ;;READ AN ASCIZ LINE
2327 011774 012600          MOV          (SP)+,R0  ;;GET ADDRESS OF 1ST CHARACTER
2328 011776 005001          CLR          R1        ;;CLEAR DATA WORD
2329 012000 005002          CLR          R2
2330 012002 112046          2s:  MOVVB         (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
2331 012004 001412          BEQ          3$        ;;IF ZERO GET OUT
2332 012006 006301          ASL          R1        ;;*2
2333 012010 006102          ROL          R2
2334 012012 006301          ASL          R1        ;;*4
2335 012014 006102          ROL          R2
2336 012016 006301          ASL          R1        ;;*8
2337 012020 006102          ROL          R2
2338 012022 042716 177770    BIC          #'C7,(SP)  ;;STRIP THE ASCII JUNK
2339 012026 062601          ADD          (SP)+,R1  ;;ADD IN THIS DIGIT
2340 012030 000764          BR          2$        ;;LOOP
2341 012032 005726          3s:  TST          (SP)+  ;;CLEAN TERMINATOR FROM STACK
2342 012034 010166 000012    MOV          R1,12(SP)  ;;SAVE THE RESULT
2343 012040 010267 000010    MOV          R2,$HIOCT
2344 012044 012602          MOV          (SP)+,R2  ;;POP STACK INTO R2
2345 012046 012601          MOV          (SP)+,R1  ;;POP STACK INTO R1
2346 012050 012600          MOV          (SP)+,R0  ;;POP STACK INTO R0
2347 012052 000002          RTI           ;;RETURN
2348 012054 000000          $HIOCT: .WORD 0        ;;HIGH ORDER BITS GO HERE
2349          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2350
2351          ;;*****
2352          ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2353          ;;OCTAL (ASCII) NUMBER AND TYPE IT.
2354          ;;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2355          ;;CALL:
2356          ;;*   MOV          NUM,-(SP)          ;;NUMBER TO BE TYPED
2357          ;;*   TYPOS          ;;CALL FOR TYPEOUT
2358          ;;*   .BYTE N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2359          ;;*   .BYTE M          ;;M=1 OR 0
2360          ;;*   ;;1=TYPE LEADING ZEROS
2361          ;;*   ;;0=SUPPRESS LEADING ZEROS
2362          ;;*
2363          ;;$TYPON---ENTER HERF TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2364          ;;$TYPOS OR $TYPOC
2365          ;;CALL:
2366          ;;*   MOV          NUM,-(SP)          ;;NUMBER TO BE TYPED
2367          ;;*   TYPON          ;;CALL FOR TYPEOUT
2368          ;;*
2369          ;;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2370          ;;CALL:
2371          ;;*   MOV          NUM,-(SP)          ;;NUMBER TO BE TYPED
2372          ;;*   TYPOC          ;;CALL FOR TYPEOUT
2373
2374 012056 017646 000000    $TYPOS: MOV          0(SP),-(SP)  ;;PICKUP THE MODE
2375 012062 116667 000001 000211  MOVVB        1(SP),$0FILL  ;;LOAD ZERO FILL SWITCH
2376 012070 112667 000207  MOVVB        (SP)+,$0MODE+1  ;;NUMBER OF DIGITS TO TYPE
2377 012074 062716 000002  ADD          #2,(SP)      ;;ADJUST RETURN ADDRESS
2378 012100 000406          BR          $TYPON
2379 012102 112767 000001 000171  $TYPOC: MOVVB        #1,$0FILL  ;;SET THE ZERO FILL SWITCH
2380 012110 112767 000006 000165  MOVVB        #6,$0MODE+1  ;;SET FOR SIX(6) DIGITS
2381 012116 112767 000005 000154  $TYPON: MOVVB        #5,$0CNT  ;;SET THE ITERATION COUNT

```

```

2382 012124 010346          MOV          R3,-(SP)  ;;SAVE R3
2383 012126 010446          MOV          R4,-(SP)  ;;SAVE R4
2384 012130 010546          MOV          R5,-(SP)  ;;SAVE R5
2385 012132 116704 000145    MOVVB        $0MODE+1,R4  ;;GET THE NUMBER OF DIGITS TO TYPE
2386 012136 005404          NEG          R4
2387 012140 062704 000006    ADD          #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
2388 012144 110467 000132    MOVVB        R4,$0MODE  ;;SAVE IT FOR USE
2389 012150 116704 000125    MOVVB        $0FILL,R4  ;;GET THE ZERO FILL SWITCH
2390 012154 016605 000012    MOV          12(SP),R5  ;;PICKUP THE INPUT NUMBER
2391 012160 005003          CLR          R3        ;;CLEAR THE OUTPUT WORD
2392 012162 006105          1s:  ROL          R5        ;;ROTATE MSB INTO "C"
2393 012164 000404          BR          3$        ;;GO DO MSB
2394 012166 006105          2s:  ROL          R5        ;;FORM THIS DIGIT
2395 012170 006105          ROL          R5
2396 012172 006105          ROL          R5
2397 012174 010503          MOV          R5,R3
2398 012176 006103          3s:  ROL          R3        ;;GET LSB OF THIS DIGIT
2399 012200 105367 000076    DECB        $0MODE      ;;TYPE THIS DIGIT?
2400 012204 100016          BPL          7$        ;;BR IF NO
2401 012206 042703 177770    BIC          #177770,R3  ;;GET RID OF JUNK
2402 012212 001002          BNE          4$        ;;TEST FOR 0
2403 012214 005704          TST          R4        ;;SUPPRESS THIS 0?
2404 012216 001403          BEQ          5$        ;;BR IF YES
2405 012220 005204          4s:  INC          R4        ;;DON'T SUPPRESS ANYMORE 0'S
2406 012222 052703 000060    BIS          #'0,R3     ;;MAKE THIS DIGIT ASCII
2407 012226 052703 000040    BIS          #' ,R3     ;;MAKE ASCII IF NOT ALREADY
2408 012232 110367 000040    MOVVB        R3,$8     ;;SAVE FOR TYPING
2409 012236 104401 012276    TYPE        ,#8        ;;GO TYPE THIS DIGIT
2410 012242 105367 000032    7s:  DECB        $0CNT     ;;COUNT BY 1
2411 012246 003347          BGT          2$        ;;BR IF MORE TO DO
2412 012250 002402          BLT          6$        ;;BR IF DONE
2413 012252 005204          INC          R4        ;;INSURE LAST DIGIT ISN'T A BLANK
2414 012254 000744          BR          2$        ;;GO DO THE LAST DIGIT
2415 012256 012605          6s:  MOV          (SP)+,R5  ;;RESTORE R5
2416 012260 012604          MOV          (SP)+,R4  ;;RESTORE R4
2417 012262 012603          MOV          (SP)+,R3  ;;RESTORE R3
2418 012264 016666 000002 000004  MOV          2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
2419 012272 012616          MOV          (SP)+,(SP)
2420 012274 000002          RTI           ;;RETURN
2421 012276 000          .BYTE 0          ;;STORAGE FOR ASCII DIGIT
2422 012277 000          .BYTE 0          ;;TERMINATOR FOR TYPE ROUTINE
2423 012300 000          $0CNT: .BYTE 0    ;;OCTAL DIGIT COUNTER
2424 012301 000          $0FILL: .BYTE 0   ;;ZERO FILL SWITCH
2425 012302 000000          $0MODE: .WORD 0   ;;NUMBER OF DIGITS TO TYPE

```

```
2426          ,SBTTL TRAP DECODER
2427
2428          ;*****
2429          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2430          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2431          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2432          ;*GO TO THAT ROUTINE.
2433
2434 012304 010046          STRAP: MOV R0,-(SP)          ;;SAVE R0
2435 012306 016600 000002 MOV 2(SP),R0          ;;GET TRAP ADDRESS
2436 012312 005740          TST -(R0)          ;;BACKUP BY 2
2437 012314 111000          MOVB (R0),R0          ;;GET RIGHT BYTE OF TRAP
2438 012316 006300          ASL R0          ;;POSITION FOR INDEXING
2439 012320 016000 012340 MOV STRPAD(R0),R0      ;;INDEX TO TABLE
2440 012324 000200          RTS R0          ;;GO TO ROUTINE
2441
2442
2443          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2444
2445 012326 011646          STRAP2: MOV (SP),-(SP)      ;;MOVE THE PC DOWN
2446 012330 016666 000004 000002 MOV 4(SP),2(SP)      ;;MOVE THE PSW DOWN
2447 012336 000002          RTI          ;;RESTORE THE PSW
2448
2449          ,SBTTL TRAP TABLE
2450
2451          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2452          ;*BY THE "TRAP" INSTRUCTION.
2453
2454          ; ROUTINE
2455          ;-----
2456 012340 012326          STRPAD: .WORD STRAP2
2457 012342 010760          STYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2458 012344 012102          STYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2459 012346 012056          STYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2460 012350 012116          STYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2461
2462 012352 011250          SGTSWR ;;CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
2463
2464 012354 011200          SCKSWR ;;CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
2465 012356 011462          SRDCHR ;;CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
2466 012360 011602          SRDLIN ;;CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE
2467 012362 011754          SRDOCT ;;CALL=RDOCT TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
2468 012364 006630          WAIT_ON_READY ;;CALL=WAITREADY TRAP+12(104412) WAIT ON THE READY BIT TO
2469 012366 006734          WAIT_FOR_XFER ;;CALL=WAITXFER TRAP+13(104413) WAIT ON XFER REQ.
```

```
2470          ,SBTTL POWER DOWN AND UP ROUTINES
2471
2472          ;*****
2473          ;POWER DOWN ROUTINE
2474 012370 012737 012534 000024 SPWRDN: MOV $SILLUP,@PWRVEC ;;SET FOR FAST UP
2475 012376 012737 000340 000026 MOV $340,@PWRVEC+2 ;;PRIO:7
2476 012404 010046          MOV R0,-(SP)          ;;PUSH R0 ON STACK
2477 012406 010146          MOV R1,-(SP)          ;;PUSH R1 ON STACK
2478 012410 010246          MOV R2,-(SP)          ;;PUSH R2 ON STACK
2479 012412 010346          MOV R3,-(SP)          ;;PUSH R3 ON STACK
2480 012414 010446          MOV R4,-(SP)          ;;PUSH R4 ON STACK
2481 012416 010546          MOV R5,-(SP)          ;;PUSH R5 ON STACK
2482 012420 017746 166514          MOV @SWR,-(SP)      ;;PUSH @SWR ON STACK
2483 012424 010667 000110          MOV SP,$SAVR6      ;;SAVE SP
2484 012430 012737 012442 000024          MOV $SPWRUP,@PWRVEC ;;SET UP VECTOR
2485 012436 000000          HALT
2486 012440 000776          BR -2          ;;HANG UP
2487
2488          ;*****
2489          ;POWER UP ROUTINE
2490 012442 012737 012534 000024 SPWRUP: MOV $SILLUP,@PWRVEC ;;SET FOR FAST DOWN
2491 012450 016706 000064          MOV $SAVR6,SP          ;;GET SP
2492 012454 005067 000060          CLR $SAVR6          ;;WAIT LOOP FOR THE TTY
2493 012460 005267 000054          1$: INC $SAVR6          ;;WAIT FOR THE INC
2494 012464 001375          BNE 1$          ;;OF WORD
2495 012466 012677 166446          MOV (SP)+,@SWR      ;;POP STACK INTO @SWR
2496 012472 012605          MOV (SP)+,R5          ;;POP STACK INTO R5
2497 012474 012604          MOV (SP)+,R4          ;;POP STACK INTO R4
2498 012476 012603          MOV (SP)+,R3          ;;POP STACK INTO R3
2499 012500 012602          MOV (SP)+,R2          ;;POP STACK INTO R2
2500 012502 012601          MOV (SP)+,R1          ;;POP STACK INTO R1
2501 012504 012600          MOV (SP)+,R0          ;;POP STACK INTO R0
2502 012506 012737 012370 000024          MOV $SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
2503 012514 012737 000340 000026          MOV $340,@PWRVEC+2 ;;PRIO:7
2504 012522 104401          TYPE          ;;REPORT THE POWER FAILURE
2505 012524 012542          SPWRMG: .WORD $POWER          ;;POWER FAIL MESSAGE POINTER
2506 012526 012716          MOV (PC)+,(SP)      ;;RESTART AT PWRST
2507 012530 002172          SPWRAD: .WORD PWRST          ;;RESTART ADDRESS
2508 012532 000002          RTI
2509 012534 000000          $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
2510 012536 000776          BR -2          ;; BEFORE THE POWER DOWN WAS COMPLETE
2511 012540 000000          $SAVR6: 0          ;;PUT THE SP HERE
2512 012542 005015 047520 042527          $POWER: .ASCIZ <15><12>"POWER"
2513 012550 000122
2514          ,EVEN
```

DZTACC,NEW	POWER	DOWN	AND	UP	ROUTINES
2515	012552	042200	044522	042526	MSGDRV: .ASCIZ <CRLF>"DRIVE(S)? "
2516	012560	051450	037451	000040	
2517	012566	005015	040524	051503	MSGASK: .ASCIZ <15><12>/TACS?/
2518	012574	000077			
2519	012576	042526	052103	051117	MSGVEC: .ASCIZ /VECTOR?/
2520	012604	000077			
2521	012606	051120	047511	044522	MSGPRI: .ASCIZ /PRIORITY?/
2522	012614	054524	000077		
2523	012620	040524	051503	000075	MTACS: .ASCIZ /TACS=/
2524	012626	052040	042101	036502	MTADB: .ASCIZ / TADB=/
2525	012634	000			
2526	012635	040	042526	052103	MTAVEC: .ASCIZ / VECTOR=/
2527	012642	051117	000075		
2528	012646	050040	044522	051117	MTAPRI: .ASCIZ / PRIORITY=/
2529	012654	052111	036531	000	
2530	012661	015	047412	037513	MSGOK: .ASCIZ <15><12>/OK?/
2531	012666	000			
2532	012667	131	051505	000200	MYES: .ASCIZ /YES/<CRLF>
2533	012674	047520	042527	020122	MTUPWR: .ASCIZ /POWER DOWN TU60/
2534	012702	047504	047127	052040	
2535	012710	033125	000060		
2536	012714	047520	042527	020122	MPWRUP: .ASCIZ /POWER UP TU60/
2537	012722	050125	052040	033125	
2538	012730	000060			
2539	012732	042522	047515	042526	MOFFLN: .ASCIZ /REMOVE CASSETTE FROM DRIVE UNDER TEST/
2540	012740	041440	051501	042523	
2541	012746	052124	020105	051106	
2542	012754	046517	042040	044522	
2543	012762	042526	052440	042116	
2544	012770	051105	052040	051505	
2545	012776	000124			
2546	013000	042522	046120	041501	MTAPE: .ASCIZ /REPLACE CASSETTE/
2547	013006	020105	040503	051523	
2548	013014	052105	042524	000	
2549	013021	123	052105	053440	MWRTLK: .ASCIZ /SET WRITE LOCK/
2550	013026	044522	042524	046040	
2551	013034	041517	000113		
2552	013040	042523	020124	051127	MLAST: .ASCIZ /SET WRITE ENABLE/
2553	013046	052111	020105	047105	
2554	013054	041101	042514	000	
2555	013061	123	040524	052524	EM1: .ASCIZ /STATUS PROBLEM/
2556	013066	020123	051120	041117	
2557	013074	042514	000115		
2558	013100	051042	040505	054504	EM2: .ASCIZ /"READY" FAILED TO SET/
2559	013106	020042	040506	046111	
2560	013114	042105	052040	020117	
2561	013122	042523	000124		
2562	013126	052042	040522	051516	EM3: .ASCIZ /"TRANSFER REQUEST" FAILED TO SET/
2563	013134	042506	020122	042522	
2564	013142	052521	051505	021124	
2565	013150	043040	044501	042514	
2566	013156	020104	047524	051440	
2567	013164	052105	000		
2568	013167	124	042510	053440	EM4: .ASCIZ /THE WRONG FLAG SET/
2569	013174	047522	043516	043040	
2570	013202	040514	020107	042523	

DZTACC,NEW	POWER	DOWN	AND	UP	ROUTINES
2571	013210	000124			
2572	013212	040504	040524	050040	EM5: .ASCIZ /DATA PROBLEM/
2573	013220	047522	046102	046505	
2574	013226	000			
2575	013227	120	020103	020040	DH1: .ASCIZ /PC TACS/
2576	013234	020040	052040	041501	
2577	013242	000123			
2578	013244	041520	020040	020040	DH2: .ASCIZ /PC TACS WAIT ADDRESS/
2579	013252	020040	040524	051503	
2580	013260	020040	020040	040527	
2581	013266	052111	040440	042104	
2582	013274	042522	051523	000	
2583	013301	120	020103	020040	DH5: .ASCIZ /PC TACS EXPECT RCVD/
2584	013306	020040	052040	041501	
2585	013314	020123	020040	042440	
2586	013322	050130	041505	020124	
2587	013330	051040	053103	042047	
2588	013336	000			
2589		013340			.EVEN
2590	013340	001116	001162	000000	DT1: .WORD SERRPC,\$REG0,0
2591	013346	001116	001162	001202	DT2: .WORD SERRPC,\$REG0,\$AVPC,0
2592	013354	000000			
2593	013356	001116	001162	001124	DT5: .WORD SERRPC,\$REG0,\$GDDAT,\$SDDAT,0
2594	013364	001126	000000		
2595	013370	001116	001206	000000	DT201: .WORD SERRPC,TACSL,0
2596					
2597	013376	001116	000000		DT202: .WORD SERRPC,0
2598					
2599	013402	040524	030461	043040	EM201: .ASCIZ "TA11 FAILED TO RESPOND"
2600	013410	044501	042514	020104	
2601	013416	047524	051040	051505	
2602	013424	047520	042116	000	
2603	013431	116	020117	051104	EM202: .ASCIZ "NO DRIVE AVAILABLE"
2604	013436	053111	020105	053101	
2605	013444	044501	040514	046102	
2606	013452	000105			
2607	013454	041520	020040	020040	DH201: .ASCIZ /PC TACS/
2608	013462	020040	040524	051503	
2609	013470	000			
2610	013471	120	000103		DH202: .ASCIZ /PC/
2611		000001			.END

DZTACC.NEW		CROSS REFERENCE		TABLE -- USER SYMBOLS											
TST22	005716	888	1401	1406#											
TST3	002646	716	728#												
TST4	002714	738	743#												
TST5	003062	744	782	787#											
TST6	003120	734	796	801#											
TST7	003164	811	816#												
TYPE =	104401	526	645	662	667	668	717	721	1434	1536	1544	1566	1575	1577	
		1580	1582	1586	1594	1660	1682	1693	1707	1716	1728	1731	1734	1737	
		1740	1747	1749	1763	1764	1774	1779	1782	1788	2136	2193	2194	2197	
		2210	2221	2240	2289	2292	2296	2409	2457#	2504					
TYPERR	006512	1543	1566#												
TYPOC =	104402	1591	1730	1733	1736	1739	2196	2458#							
TYPON =	104404	2460#													
TYPOS =	104403	2459#													
UNIT =	000400	200#	660	2064	2084										
WAITRE=	104412	703	705	746	751	756	761	766	771	776	794	838	853	864	
		866	877	893	990	996	1182	1186	1188	1283	1287	1289	1292	1301	
		1306	1313	1315	1327	1331	1334	1336	1338	1342	1347	1357	1359	1361	
		1377	1379	1381	1398	1411	1817	1903	1934	1959	1975	2028	2049	2468#	
WAITXF=	104413	846	851	857	862	870	875	992	994	1184	1285	1311	1329	1340	
		1344	2020	2038	2469#										
WAIT.F	006734	1629#	2469												
WAIT.O	006630	1601#	2468												
WAITI	006660	1607#	1615	1618											
WAIT2	006700	1602*	1614#												
WAIT3	006714	1608	1612	1620#											
WAIT4	006676	1610	1613#												
WFG	= 000000	182#	704	745	865	901	941	1001	1041	1093	1133	1193	1233	1335	
		1360	1399	1816	1902										
WFGSUB	007604	235	1806#	1810	1821										
WGBL0P	010042	1892#	1896	1910											
WGPBLK	010036	238	1891#												
WLOOP	007674	1838#	1842	1852											
WRITE =	000002	183#	306	750	845	856	869	906	946	1006	1046	1098	1138	1198	
		1238	1293	1380	2019										
WRtblK	010466	1848	1906	2018#											
WRtLOC=	010000	196#	902	907	912	917	922	927	932	937	1042	1047	1052	1057	
		1062	1067	1072	1077	1094	1099	1104	1109	1114	1119	1124	1129	1194	
		1199	1204	1209	1214	1219	1224	1229	1400	2091					
WRISUB	007670	236	1837#												
SAUTOB	001134	360#	534*	2190	2311										
SBDADR	001122	355#													
SBDDAT	001126	357#	1530*	2593											
SBELL	001172	379#	1536	1562											
SCHARC	011174	2138#	2148*	2155	2164*	2169#									
SCKSWR	011200	2182#	2464												
SCMTAG	001100	343#	483	484	492	498	499	500							
SCM1 =	000002	375#	376#	377#											
SCM2 =	000004	375#	376#	377#											
SCM3 =	000002	373#	375												
SCNTLG	011725	2193	2306#												
SCNTLU	011720	2210	2305#												
SCRFL	001177	381#	668	1544	1562	1566	1577	1582	1586	1763	1788	2137	2172	2221	
		2305													
SDOAGN	006024	1430	1436	1442#											
SENDAD	006014	524	589	639	1438#	1557									

DZTACC.NEW		CROSS REFERENCE		TABLE -- USER SYMBOLS											
SENDCT	005774	498	1432#												
SENDMG	006033	1434	1446#												
SENULL	006030	1445#													
SEOP	005740	1422#													
SEOPCT	005766	498#	1429#	1433											
SERFLG	001103	346#	1453	1482	1484	1490*	1511	1531*	1562						
SERMAX	001115	352#	501*	1484	1506*	1511									
SERR0R	006320	492	1525#												
SERRPC	001116	353#	1538*	1539*	1540	1562	2590	2591	2593	2595	2597				
SERRTB	001236	411#	1573												
SERTTL	001112	350#	1537*	1562											
SESCAP	001170	378#	500*	744*	832*	884*	986*	1086*	1178*	1279*	1323*	1354*	1374*	1505*	
		1553	1555	1562											
SFILLC	001156	371#	2141	2172											
SFILLS	001155	370#	2172												
SGDADR	001120	354#													
SGDDAT	001124	356#	1529*	2593											
SGET42	006004	1435#													
SCTSWR	011250	2194#	2462												
SHD =	000000	11													
SHIOCT	012054	2343#	2348#												
SICNT	001104	347#	1497*	1498	1500*	1510									
SILLUP	012534	2474	2490	2509#											
SINTAG	001135	361#	2222	2311											
SITEMB	001114	351#	1540*	1562	1568										
SIF	001200	382#	1562	2172	2296	2305									
SLPADR	001106	348#	502*	730*	788*	802*	817*	883*	985*	1085*	1177*	1278*	1353*	1373*	
		1391#	1488*	1503*	1508	1510									
		349#	503*	1488	1504*	1510	1552	1810*	1842*	1865*	1896*	1924*	1952*	1972*	
SLPERR	001110	520	530	1503	1546	2125									
SMAIL =	***** U	2197	2309#												
SMNEW	011743	2194	2307#												
SMSWR	011732	1501	1510#												
SMXCNT	006316	369#	2143	2172											
SNULL	001154	693#	710#	725#	740#	784#	798#	813#	827#	878#	981#	1081#	1173#	1273#	
SNWTST=	000001	1318#	1348#	1368#	1386#	1403#									
		2381*	2410*	2423#											
SOCNT	012300	2376*	2380*	2385	2388*	2399*	2425#								
SOMODE	012302	1465	1481	1489	1499	1507#									
SOVER	006302	344#	652	698	715	1426*	1427*	1445	1495	1511					
SPASS	001100	2505	2512#												
SP0WER	012542	2507#													
SPWRAD	012530	496	2474#	2502											
SPWRDN	012370	2505#													
SPWRMG	012524	2484	2490#												
SPWRUP	012442	380#	1562	1682	1747	1779	2172	2240	2289	2305					
SQUES	001176	2253#	2465												
SRDCHR	011462	2468													
SRDDEC=	***** U	2281#	2466												
SRDLIN	011602	2321#	2467												
SRDOCT	011754	2274#													
SRDSZ =	000010	373#													
SREGAD	001160	375#	1527*	2590	2591	2593									
SREG0	001162	376#	1528*												
SREG1	001164	1444#													
SRTNAD	006026	2468													
SR2A =	***** U														

\$SAVRE= ***** U	2460																
\$SAVR6 012540	2483*	2491	2492*	2493*	2511*												
\$SCOPE 006046	490	462#															
\$SETUP= 000137	469#	489	490	492	494	496	498	499	500	502	524	527	1424				
	1463	1526	1549	1557	2177	2311											
\$STUP = 177777	469#																
\$SVLAD 006254	1473	1502#															
\$SWR = 167400	1#	11	62	63	64	65	66	67	68	377	378	379	499				
	500	502	503	697	714	729	744	788	802	817	831	882	985				
	1085	1177	1277	1322	1352	1372	1390	1407	1417	1425	1437	1443	1445				
	1454	1455	1456	1457	1458	1464	1476	1478	1479	1482	1483	1484	1491				
	1492	1493	1504	1507	1510	1517	1518	1519	1520	1521	1534	1541	1546				
	1550	1562	1787	2508													
\$SWRKM= 000000	68	69	1458	1459	1480												
\$TIMES 001166	377#	499#	697*	714*	729*	831*	882*	1277*	1322*	1352*	1372*	1390*	1407*				
	1425*	1491*	1490	1501*	1510	1787*											
\$TKB 001146	366#	2175	2186	2203	2257	2263											
\$TKS 001144	365#	2175	2184	2200	2224*	2255											
\$TN = 000023	1#	11	693	697#	699	710	714#	716	725	729#	734	738	740				
	744#	782	784	788#	796	798	802#	806	811	813	817#	821	825				
	827#	831#	878	882#	884	888	981	985#	986	1081	1085#	1086	1173				
	1177#	1178	1273	1277#	1279	1318	1322#	1323	1348	1352#	1354	1366	1368				
	1372#	1374	1384	1386	1390#	1401	1403	1407#									
\$TPB 001152	368#	2161*	2172														
\$TPFLG 001157	372#	2119	2172														
\$TPS 001150	367#	2159	2172														
\$TRAP 012304	494	2434#															
\$TRAP2 012326	2445#	2456															
\$TRP = 000014	2449#	2458#	2459*	2460#	2461#	2462	2463#	2464	2465#	2466#	2467#	2468#	2469#				
	2470#																
\$TRPAD 012340	2439	2456#															
\$TSTNM 001102	345#	655*	1424*	1453	1480	1502*	1507	1511	1533	1562							
\$TYIN 011710	2282	2283	2300	2304#													
\$TYPN= ***** U	2461																
\$TYPDS= ***** U	2461																
\$TYPE 010760	2119#	2449	2457														
\$TYPEC 011130	2140	2147	2154	2159#	2160	2226											
\$TYPEX 011176	2165	2167	2170#														
\$TYPOC 012102	2379#	2458															
\$TYPON 012116	2378	2381#	2460														
\$TYPOS 012056	2374#	2459															
\$XTSTR 006060	1467#																
\$SET4= 000000	1437#																
\$FILL 012301	2375#	2379*	2389	2424#													
\$40CAT= ***** U	1464	1543															
, = 013474	223#	227#	289#	299#	321#	342#	383	487	502	503	538#	665#	1445				
	1448	1510	1511	1562	1597#	1777#	1785#	2172	2175	2304#	2305	2311	2486				
	2510	2589#															

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

*TACC, TACC/SOL/CRF; SYM=DZTACC,NEW

RUN-TIME: 44 29 2 SECONDS
 RUN-TIME RATIO: 171/77=2.2
 CORE USED: 25K (49 PAGES)