

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAB-C-D
PRODUCT NAME: TA11 LOGIC TEST (PART 2)
DATE REVISED: 21 JUNE 76
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: JIM LACEY

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

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

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

COPYRIGHT (C) 1973, 1976 BY DIGITAL EQUIPMENTS CORPORATION

MAIN DEC CHANGE NOTICE
MAY BE REQUIRED FOR
PROGRAM TO OPERATE

CONTENTS

- 1. ABSTRACT
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3. LOADING PROCEDURE
- 4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS
 - 4.3 PROGRAM & OPERATOR ACTION
- 5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUBROUTINE ABSTRACTS
- 6. ERRORS
- 7. RESTRICTIONS
- 8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 PASS COUNTER
 - 8.4 ITERATIONS
 - 8.5 SPECIAL REGISTERS
- 9. PROGRAM DESCRIPTION

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF BASIC LOGIC TESTS THAT CHECK THE TAIL FOR PROPER OPERATION.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TAIL CASSETTE STORAGE

2.2

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZTAA

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES OR A CASSETTE TAPE.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

4.2 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS
 204 SELECT DRIVE(S) BEFORE STARTING TEST
 210 SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST
 214 SETUP FOR MANUAL LOOPING
 220 WRITE FILE GAP FROM BOT TO EOT
 224 WRITE 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 LOOP & TTY BELL WILL RING ONCE EVERY
 PASS, IF SW<10>=0.
 *** NOTE: IF USING THE SOFTWARE SWITCH REGISTER
 PROGRAM WILL TYPE "SWR=XXXXX NEW=", TO ALLOW
 SETTING OF REGISTER BEFORE BEGINNING TEST.

4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC
 SELECTION OF DRIVES "A" AND "B" TO BE TESTED.
 NOTE: IF LOAD MEDIUM IS CASSETTE WITH STANDARD VECTOR
 PROGRAM WILL RESPOND AS IF STARTED AT 210.

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

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

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

4.3.1.1 DRIVE SELECTION EXAMPLES

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

4.3.2 ADDRESS SELECTION

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

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

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

;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

5.2.2 TRAPCATCHER

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

5.2.3 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6.)

;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

5.2.4 TRAP

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

5.2.4.1 TYPE (\$TYPE)

ROUTINE TO TYPE AN ASCII 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 "TAIL READY" BIT TO SET

5.2.4.5 WAITXFER (WAIT.FOR.XFER.REQ)

WAIT ON THE "TAIL TRANSFER REQUEST" BIT TO SET

5.2.4.6 \$TYPOC

CHANGE A BINARY NUMBER TO OCTAL ASCII AND TYPE IT.

5.2.5 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

5.2.5.1 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 SELADR

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

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

5.2.6.1 WFGSUB

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

5.2.6.2 WRTSUB

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

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

5.2.6.3 RDSUB

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

5.2.6.4 WGPBLK

WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO ECT
 START AT 234
 THE PROGRAM WILL HALT THREE (3) TIMES
 AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
 HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
 HALT 2 --- SWR<7:0> = PATTERN DESIRED
 HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
 THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND THE "GAP TIME MONO".
 ** IF USING SOFTWARE SWITCH REGISTER, AFTER EACH HALT OPERATOR WILL BE PROMPTED FOR THE VALUE WITH "SWR=XXXXXX NEW="

5.2.6.5 RGBLK

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

5.2.6.6 SFFGSB

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

5.2.6.7 BSFGSB

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

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 SERRTB FOR THE DIFFERENT ERRORS THAT CAN OCCUR.

7. RESTRICTIONS

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

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

A PROGRAM PASS THRU COUNT IS KEPT IN "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 FOR "DRIVE", "TACS" AND "TADB THROUGH OUT THE PROGRAM.

9. PROGRAM DESCRIPTION

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

THE TESTS CAN BE GROUPED INTO THE FOLLOWING GENERAL GROUPS.

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

TA11 BASIC LOGIC TEST (PART 2). MAINDEC-11-DZTAB-C
DZTABC.NEW
TABLE OF CONTENTS

12	GENERAL INFORMATION
60	OPERATIONAL SWITCH SETTINGS
73	BASIC DEFINITIONS
183	TA11 DEFINITIONS
224	STARTING ADDRESSES
225	TRAP CATCHER
234	STARTING ADDRESS(ES)
246	TOGGLE IN ROUTINES
293	LOAD SWITCH REGISTER INTO TACS
301	WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
327	READ FROM BOT TO EOT
345	COMMON TAGS
406	ERROR POINTER TABLE
473	START OF TEST
498	INITIALIZE THE COMMON TAGS
536	TYPE PROGRAM NAME
543	GET VALUE FOR SOFTWARE SWITCH REGISTER
710	ROUTINE TO DETERMINE TIME OF WAIT LOOPS
728	*****TIMING ERROR*****
733	TEST "TIMING ERROR" FOR "WRITE"
774	TEST "TIMING ERROR" FOR "READ"
827	TEST "ERROR" OUTPUT OF STATUS ROM USING A "TIMING ERROR"
898	*****INTERRUPTS*****
921	TEST INTERRUPT WITH READY = "1" AT LEVEL 0
970	TEST INTERRUPT WITH READY = "1" AT LEVEL 1
1019	TEST INTERRUPT WITH READY = "1" AT LEVEL 2
1068	TEST INTERRUPT WITH READY = "1" AT LEVEL 3
1117	TEST INTERRUPT WITH READY = "1" AT LEVEL 4
1166	TEST INTERRUPT WITH READY = "1" AT LEVEL 5
1215	TEST INTERRUPT WITH READY = "1" AT LEVEL 6
1264	TEST INTERRUPT WITH READY = "1" AT LEVEL 7
1306	TEST INTERRUPT WITH "TRANSFER REQUEST" = 1
1343	TEST INTERRUPT WITH "READY" = 0 AND "XFER REQ" = 0
1382	TEST INTERRUPT WITH "XFER REQ" = 1 & "TIMING ERROR" = 1
1450	*****SPACING FUNCTIONS*****
1455	TEST "BACK SPACE FILE GAP"
1486	TEST "SPACE FORWARD FILE GAP" FUNCTION
1515	TEST "BACK SPACE BLOCK GAP"
1543	TEST "SPACE FWD BLOCK GAP"
1576	TEST AUTOMATIC "WFG" WHEN WRITING OFF OF "CLEAR LEADER"
1602	TEST "READ" INTO FILE GAP CAUSES AN ERROR
1634	TEST "SFBG" INTO FILE GAP CAUSES AN ERROR
1666	TEST "ERROR" OUTPUT OF THE STATUS ROM WITH "FILE GAP=1"
1735	TEST BACK-SPACE-FILE-GAP INTO CLEAR LEADER
1783	TEST BACK-SPACE-BLOCK-GAP INTO CLEAR LEADER
1828	*****TADB*****
1833	TEST "WRITE" 377 & 0 "READ" 377 & 0
1869	TEST "WRITE & READ" A COUNT PATTERN
1916	*****CRC*****
1921	TEST "ERROR" WITH "CRCERR" = 1
1990	TEST "DATA OF 0 GIVES CRC OF 0"
2038	TEST "CRC" CIRCUIT USING A COUNT PATTERN
2112	TRY TO HANG "READY" ON "REWIND"
2166	TRY TO GLITCH THE "POWER SUPPLY"
2269	T41 END OF TEST CODE
2279	END OF PASS ROUTINE

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

```

1 .TITLE TA11 BASIC LOGIC TEST (PART 2) MAINDEC-11-DZTAB-C
2 ;*COPYRIGHT (C) 1976
3 ;*DIGITAL EQUIPMENT CORP.
4 ;*MAYNARD, MASS. 01754
5 ;*
6 ;*PROGRAM BY JAMES 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 TAVEC INTERRUPT VECTOR
25

```

TACS REGISTER DESCRIPTION

```

26 -----
27
28
29
30 BIT NAME INIT READ AND/OR WRITE?
31 --- --- STATE -----
32
33 15 ERROR ? 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 TRANSFER 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,ERROR ;;BASIC DEFINITION OF ERROR CALL
76 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
77
78 ;*MISCELLANEOUS DEFINITIONS
79 HT= 11 ;;CODE FOR HORIZONTAL TAB
80 000012 LF= 12 ;;CODE FOR LINE FEED
81 000015 CR= 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= 40000
118      002000      SW10= 20000
119      001000      SW09= 10000
120      000400      SW08= 4000
121      000200      SW07= 2000
122      000100      SW06= 1000
123      000040      SW05= 400
124      000020      SW04= 200
125      000010      SW03= 100
126      000004      SW02= 40
127      000002      SW01= 20
128      000001      SW00= 10
129      .FQUIV SW09,SW9
130      .EQUIV SW08,SW8
131      .EQUIV SW07,SW7
132      .EQUIV SW06,SW6
133      .EQUIV SW05,SW5
134      .EQUIV SW04,SW4
135      .EQUIV SW03,SW3
136      .EQUIV SW02,SW2
137      .EQUIV SW01,SW1
138      .EQUIV SW00,SW0
139
140      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
141      100000      BIT15= 100000
142      040000      BIT14= 400000
143      020000      BIT13= 200000
144      010000      BIT12= 100000
145      004000      BIT11= 40000
146      002000      BIT10= 20000
147      001000      BIT09= 10000
148      000400      BIT08= 4000
149      000200      BIT07= 2000
150      000100      BIT06= 1000
151      000040      BIT05= 400
152      000020      BIT04= 200
153      000010      BIT03= 100
154      000004      BIT02= 40
155      000002      BIT01= 20
156      000001      BIT00= 10
157      .EQUIV BIT09,BIT9
158      .EQUIV BIT08,BIT8
159      .EQUIV BIT07,BIT7
160      .EQUIV BIT06,BIT6
161      .EQUIV BIT05,BIT5
162      .EQUIV BIT04,BIT4
163      .EQUIV BIT03,BIT3
164      .EQUIV BIT02,BIT2
165      .EQUIV BIT01,BIT1
166      .EQUIV BIT00,BIT0
167
168      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
    
```

```

169      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
170      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
171      000014      TBITVEC=14    ;;"T" BIT
172      000014      TRTVEC= 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      BSG= 6                ;BACK SPACE FILE GAP FUNCTION
186      000010      BSBG= 10             ;BACK SPACE BLOCK GAP FUNCTION
187      000012      SFG= 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      TR,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      TADB= %5            ;R5 IS USED AS A POINTER TO THE TADB REGISTER.
219
220      ;////////////////////////////////////
221      ;////////////////////////////////////
    
```

```

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      000174      .=174
228      000174      000000      DISPREG: .WORD 0          ;SOFTWARE DISPLAY REGISTER
229      000176      000000      SWREG: .WORD 0           ;SOFTWARE SWITCH REGISTER
230
231      .SBTTL STARTING ADDRESS(ES)
232      000200      000137      001336      JMP @#BEGIN1          ;JUMP TO STARTING ADDRESS OF PROGRAM
233      000204      000137      001370      JMP @#BEGIN2          ;SELECT DRIVE(S) BEFORE STARTING TEST
234      000210      000137      001376      JMP @#BEGIN3          ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
235      000214      000137      001404      JMP @#BEGIN4          ;SETUP FOR MANUAL LOOPING
236      000220      000137      013776      JMP @#WFGSUB          ;WRITE FILE GAP FROM BOT TO EOT
237      000224      000137      014062      JMP @#WRFSUB          ;WRITE CONTINUOUS BLOCKS OF DATA
238      000230      000137      014150      JMP @#RDSUB          ;READ CONTINUOUS BLOCKS OF DATA
239      000234      000137      014230      JMP @#WGPBLK         ;WRITE FILE GAP AND A BLOCK OF DATA
240      000240      000137      014332      JMP @#RGPBLK         ;READ BLOCK OF DATA AND INTO A FILE GAP
241      000244      000137      014426      JMP @#SFFGSB        ;SPACE FWD FILE GAP FROM BOT TO EOT
242      000250      000137      014512      JMP @#BSFGSB        ;BACK SPACE FILE GAPS
    
```

242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294

```

;////////////////////////////////////
;////////////////////////////////////
;THE FOLLOWING ROUTINES CAN BE TOGGLED IN.
;////////////////////////////////////
;////////////////////////////////////
    
```

.REM !

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

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

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

R4= TA11 STATUS REG ADDRESS (TACS 177500)

R5= TA11 DATA BUFFER ADDRESS (TADB 177502)

LOOP1 WILL LOAD THE SWITCH REGISTER INTO THE TACS.

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

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

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

```

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

.=500

```

LOOP1: MOV @SWR,@TACS ;LOAD TACS
        BR LOOP1 ;LOOP
    
```

295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336

```

;*****
;WRITE SWITCH REGISTER ON TAPE FROM BOT TO EOT
;*****
.=600
LOOP2: RESET ;CLEAR ALL FLAGS
        MOV DRIVE,@TACS ;SELECT DRIVE
        MOVB #REWIND!GO,@TACS ;GO TO BOT
1$: BIT #READY,@TACS ;WAIT TILL READY COMES UP
        BEQ 1$
        MOVB #WRITE!GO,@TACS ;START A WRITE
2$: TSTB @TACS ;CHECK FOR TRANSFER REQUEST
        BPL 3$ ;BR IF NOT SET
        MOV #SWR,@TADB ;SEND DATA TO TA11
        BR 2$ ;LOOP
3$: BIT #READY,@TACS ;DID READY SET?
        BNE LOOP2 ;START OVER IF YES
        BR 2$ ;LOOP
    
```

```

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

.=700

```

LOOP3: RESET ;CLEAR ALL FLAGS
        MOV DRIVE,@TACS ;SELECT DRIVE
        MOVB #REWIND!GO,@TACS ;START A REWIND
1$: BIT #READY,@TACS ;WAIT ON REWIND TO FINISH
        BEQ 1$
        MOVB #READ!GO,@TACS ;START A READ
2$: TSTB @TACS ;CHECK TRANSFER REQ
        BPL 3$ ;BR IF NOT SET
        MOV @TADB,R0 ;PICKUP THE DATA
        BR 2$ ;LOOP
3$: BIT #READY,@TACS ;CHECK READY
        BNE LOOP3 ;START OVER
        BR 2$ ;LOOP
    
```

```

337                                     ,SBTTL COMMON TAGS
338
339                                     ;*****
340                                     ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
341                                     ;*USED IN THE PROGRAM.
342
343                                     ,=1100
344 001100 001100          $CMTAG:          ;START OF COMMON TAGS
345 001100 000000          $PASS: ,WORD 0      ;CONTAINS PASS COUNT
346 001102 000           $TSTNM: ,BYTE 0       ;CONTAINS THE TEST NUMBER
347 001103 000           $ERFLG: ,BYTE 0       ;CONTAINS ERROR FLAG
348 001104 000000          $ICNT: ,WORD 0       ;CONTAINS SUBTEST ITERATION COUNT
349 001106 000000          $LPADR: ,WORD 0      ;CONTAINS SCOPE LOOP ADDRESS
350 001110 000000          $LPERR: ,WORD 0     ;CONTAINS SCOPE RETURN FOR ERRORS
351 001112 000000          $ERTTL: ,WORD 0     ;CONTAINS TOTAL ERRORS DETECTED
352 001114 000           $ITEMB: ,BYTE 0      ;CONTAINS ITEM CONTROL BYTE
353 001115 001           $ERMAX: ,BYTE 1      ;CONTAINS MAX. ERRORS PER TEST
354 001116 000000          $ERRPC: ,WORD 0     ;CONTAINS PC OF LAST ERROR INSTRUCTION
355 001120 000000          $GADDR: ,WORD 0     ;CONTAINS ADDRESS OF "GOOD" DATA
356 001122 002000          $BDADR: ,WORD 0     ;CONTAINS ADDRESS OF "BAD" DATA
357 001124 000000          $GDDAT: ,WORD 0     ;CONTAINS "GOOD" DATA
358 001126 000000          $BDDAT: ,WORD 0     ;CONTAINS "BAD" DATA
359 001130 000000          ,WORD 0            ;RESERVED--NOT TO BE USED
360 001132 000000          ,WORD 0
361 001134 000           $AUTOB: ,BYTE 0      ;AUTOMATIC MODE INDICATOR
362 001135 000           $INTAG: ,BYTE 0      ;INTERRUPT MODE INDICATOR
363 001136 000000          ,WORD 0
364 001140 177570          SWR: ,WORD DSWR     ;ADDRESS OF SWITCH REGISTER
365 001142 177570          DISPLAY: ,WORD DDISP ;ADDRESS OF DISPLAY REGISTER
366 001144 177560          $TKS: 177560      ;TTY KBD STATUS
367 001146 177562          $TKB: 177562      ;TTY KBD BUFFER
368 001150 177564          $TPS: 177564      ;TTY PRINTER STATUS REG. ADDRESS
369 001152 177566          $TPB: 177566      ;TTY PRINTER BUFFER REG. ADDRESS
370 001154 000           $NULL: ,BYTE 0      ;CONTAINS NULL CHARACTER FOR FILLS
371 001155 002           $FILLS: ,BYTE 2      ;CONTAINS # OF FILLER CHARACTERS REQUIRED
372 001156 012           $FILLC: ,BYTE 12     ;INSERT FILL CHARS. AFTER A "LINE FEED"
373 001157 000           $TPFLG: ,BYTE 0     ;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
374 001160 000000          $REGAD: ,WORD 0    ;CONTAINS THE ADDRESS FROM
375                                     ;WHICH ($REG0) WAS OBTAINED
376 001162 000000          $REG0: ,WORD 0     ;CONTAINS (($REGAD)+0)
377 001164 000000          $REG1: ,WORD 0     ;CONTAINS (($REGAD)+2)
378 001166 000000          $TIMES: 0         ;MAX. NUMBER OF ITERATIONS
379 001170 000000          $ESCAPE: 0        ;ESCAPE ON ERROR ADDRESS
380 001172 177607 000377 $BELL: ,ASCIZ <207><377><377> ;CODE FOR BELL
381 001176 077           $QUES: ,ASCIZ /?/    ;QUESTION MARK
382 001177 015           $CRLF: ,ASCIZ <15>   ;CARRIAGE RETURN
383 001200 000012          $LF: ,ASCIZ <12>   ;LINE FEED
384                                     ;*****
385 001202 000000          SAVPC: ,WORD 0     ;STORAGE FOR THE PC
386 001204 000000          SAVPS: ,WORD 0     ;STORAGE FOR THE PS
387
388 001206 177500          TACSL: 177500      ;LOW BYTE ADDRESS OF TACS
389 001210 177501          TACSH: 177501      ;HIGH BYTE ADDRESS OF TACS
390 001212 177502          TADBL: 177502      ;LOW BYTE ADDRESS OF TADB
391 001214 177503          TADBH: 177503      ;HIGH BYTE ADDRESS OF TADB
392 001216 000260 000262 TAVEC: 260,262    ;TA11 VECTOR ADDRESS
    
```

```

393 001222 000300          TAPRIO: 300          ;TA11 BR LEVEL 6
394 001224 000000 000000 DRVKEY: 0,0      ;DRIVE SELECT KEY:
395 001230 001224          DRVPT: DRVKEY
396 001232 000000          ASKKEY: 0
397 001234 000000          CURDRV: 0          ;CURRENT DRIVE BEING TESTED
    
```

```

398 .SBTTL ERROR POINTER TABLE
399
400 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
401 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
402 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
403 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
404 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
405
406 ;* EM ;POINTS TO THE ERROR MESSAGE
407 ;* DH ;POINTS TO THE DATA HEADER
408 ;* DT ;POINTS TO THE DATA
409 ;* DF ;POINTS TO THE DATA FORMAT
410
411
412 001236 $ERRTB:
413
414 ;NOTE: ALL NUMBERS ARE TYPED AS 6-DIGIT OCTAL NUMBERS
415
416 ;ITEM 1
417 001236 017066 EM1 ;STATUS PROBLEM
418 001240 017256 DH1 ;PC TACS
419 001242 017420 DT1 ;$ERRPC $REG0
420 001244 000000 0
421
422 ;ITEM 2
423 001246 017105 EM2 ;READY FAILED TO SET
424 001250 017273 DH2 ;PC TACS WAIT ADDRESS
425 001252 017426 DT2 ;$ERRPC $REG0 SAVPC
426 001254 000000 0
427
428 ;ITEM 3
429 001256 017133 EM3 ;TRANSFER REQUEST FAILED TO SET
430 001260 017273 DH2 ;PC TACS WAIT ADDRESS
431 001262 017426 DT2 ;$ERRPC $REG0 SAVPC
432 001264 000000 0
433
434 ;ITEM 4
435 001266 017174 EM4 ;THE WRONG FLAG SET
436 001270 017273 DH2 ;PC TACS WAIT ADDRESS
437 001272 017426 DT2 ;$ERRPC $REG0 SAVPC
438 001274 000000 0
439
440 ;ITEM 5
441 001276 017217 EM5 ;DATA PROBLEM
442 001300 017330 DH5 ;PC TACS EXPECT RCV'D
443 001302 017436 DT5 ;$ERRPC $REG0 $GDDAT $BDDAT
444 001304 000000 0
445
446 ;ITEM 6
447 001306 017234 EM6 ;INTERRUPT PROBLEM
448 001310 017366 DH6 ;PC TACS BR LEVEL
449 001312 017450 DT6 ;$ERRPC $REG0 $GDDAT
450 001314 000000 0
451
452
    
```

```

453 001316 ITEMS2: ;ITEMS 201-202
454
455 001316 017472 EM201 ;TA11 FAILED TO RESPOND
456 001320 017544 DH201 ;PC TACS
457 001322 017460 DT201 ;$ERRPC TACS
458 001324 000000 0 ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
459
460 001326 017521 EM202 ;NO DRIVES AVAILABLE
461 001330 017561 DH202 ;PC
462 001332 017466 DT202 ;$ERRPC
463 001334 000000 0 ;
464
    
```

```

465 ;////////////////////////////////////
466 ;////////////////////////////////////
467 ;*****
468 ;
469 ;BEGIN1 IS FOR NORMAL START
470 ;BEGIN2 IS FOR DRIVE SELECTION
471 ;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
472 ;BEGIN4 IS FOR MANUAL OPERATION
473 ;
474 ;*****
475 ;
476 001336 005005 BEGIN1: CLR R5 ;NORMAL START
477 001340 012737 MOV #*AB,@#DRVKEY
478 001346 122737 CMPB #5,@#41 ;CASSETTE DDP?
479 001354 001015 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
480 001356 022737 CMP #260,@#TAVEC ;STANDARD VECTOR?
481 001364 001011 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
482 001366 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
483 001370 012705 BEGIN2: MOV #1,R5 ;ASK FOR DRIVES FLAG
484 001374 000405 BR BGNCMN ;BEGIN COMMON CODE
485 001376 012705 BEGIN3: MOV #2,R5 ;ASK FOR DRIVES AND ADDRESSES
486 001402 000402 BR BGNCMN
487 001404 012705 BEGIN4: MOV #3,R5
488 001410 BGNCMN:
489 .SBTTL INITIALIZE THE COMMON TAGS
490 ;;CLEAR THE COMMON TAGS (SCMTAG) AREA
491 001410 012706 001100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
492 001414 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
493 001416 022706 001140 CMP #SWR,R6 ;;DONE?
494 001422 001374 BNE -6 ;;LOOP BACK IF NO
495 001424 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
496 ;;INITIALIZE A FEW VECTORS
497 001430 012737 012264 000020 MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
498 001436 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7
499 001444 012737 012536 000030 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
500 001452 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
501 001460 012737 016476 000034 MOV #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
502 001466 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
503 001474 012737 016562 000024 MOV #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
504 001502 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;LEVEL 7
505 001510 016767 010476 010466 MOV SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
506 001516 005067 177444 CLR TIMES ;;INITIALIZE NUMBER OF ITERATIONS
507 001522 005067 177442 CLR ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
508 001526 112767 000001 177361 MOVB #1,SEMAX ;;ALLOW ONE ERROR PER TEST
509 001534 012767 001534 177344 MOV #.,SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
510 001542 012767 001542 177340 MOV #.,SLPERR ;;SETUP THE ERROR LOOP ADDRESS
511 ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
512 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
513 001550 013746 000004 MOV #ERRVEC,-(SP) ;;SAVE ERROR VECTOR
514 001554 012737 001610 000004 MOV #64,@#ERRVEC ;;SET UP ERROR VECTOR
515 001562 012767 177570 177350 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
516 001570 012767 177570 177344 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
517 001576 022777 177777 177334 CMP #1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
518 001604 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
519 BR 65$ ;;AND THE HARDWARE SWR IS NOT = -1
520 001606 000403 BR 65$ ;;BRANCH IF NO TIMEOUT

```

```

521 001610 012716 001616 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
522 001614 000002 RTI
523 001616 012767 000176 177314 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
524 001624 012767 000174 177310 66$: MOV #DISPREG,DISPLAY
525 001632 012637 000004 (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
526
527 .SBTTL TYPE PROGRAM NAME
528 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
529 001636 005227 177777 INC #-1 ;;FIRST TIME?
530 001642 001036 BNE HERE ;;BRANCH IF NO
531 001644 022737 012232 000042 CMP #SENDAD,@#42 ;;ACT-11?
532 001652 001432 BEQ HERE ;;BRANCH IF YES
533 001654 104401 001712 TYPE ,MSGID ;;TYPE ASCIZ STRING
534 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
535 001660 005737 000042 TST #42 ;;ARE WE RUNNING UNDER XXDP/ACT?
536 001664 001006 BNE 67$ ;;BRANCH IF YES
537 001666 026727 177246 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
538 001674 001005 BNE 68$ ;;BRANCH IF NO
539 001676 104405 GTSWR ;;GET SOFT-SWR SETTINGS
540 001700 000403 BR 68$
541 001702 112767 000001 177224 67$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
542 001710 68$: BR HERE ;;GET OVER THE ASCIZ
543 001710 000413 ;;MSGID: .ASCIZ <CRLF>/MAINDEC-11-DZTAB-C/<CRLF>
544 HERE:
545 001740 ;
546 ;*****
547 ;
548 ;
549 ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
550 ;
551 ; R5=3 MANUAL OPERATIONS
552 ; R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
553 ; R5=1 ASK FOR DRIVE(S)
554 ; R5=0 DON'T ASK FOR ANYTHING
555 ;
556 ;*****
557 001740 010504 BEGINX: MOV R5,R4 ;COPY R5
558 001742 005305 DEC R5 ;ASK FOR DRIVES?
559 001744 002406 BLT CHKADR ;BR IF NO
560 001746 004737 013254 JSR PC,@#ASKDRV ;GO GET DRIVES TO BE TESTED
561 001752 005305 DEC R5 ;ASK FOR ADDRESSES?
562 001754 002402 BLT CHKADR ;BR IF NO
563 001756 004737 013364 JSR PC,@#ASKADR ;GO GET T11 ADDRESSES
564 ;*****
565 ;
566 ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
567 ;
568 ;
569 ;I. TIMEOUT OCCURRED
570 ; A. TYPE ERROR MESSAGE
571 ; B. EXAMINE R4
572 ; 1. R4>0 GOTO BEGINX
573 ; 2. R4=0 EXAMINE (42)
574 ; A. (42)=0 GOTO BEGINX
575 ; B. (42)>0 GOTO SENDAD
576 ;

```

```

577 ;II. TIMEOUT DIDN'T OCCUR
578 ; A. CONTINUE
579 ;
580 ;*****
581 001762 012737 002000 000004 CHKADR: MOV #1S,@ERRVEC ;IN CASE OF TIMEOUTS
582 001770 005000 CLR R0 ;USE AS A SWITCH
583 001772 005777 177210 TST @TACSL ;SEE IF TA11 RESPONDS
584 001776 000402 BR 2S ;BR IF NO TIMEOUT
585 002000 005200 1S: INC R0 ;COME HERE ON TIMEOUT
586 002002 022626 CMP (SP)+,(SP)+ ;CLEANUP THE STACK
587 002004 012737 000006 000004 2S: MOV #ERRVEC+2,@ERRVEC ;RESTORE TIMEOUT VECTOR
588 002012 005700 TST R0 ;DID A TIMEOUT OCCUR?
589 002014 001414 BEQ 3S ;BR IF NO
590 002016 104201 ERROR 201 ;TA11 FAILED TO RESPOND
591 002020 012705 000002 MOV #2,R5 ;DRIVES & ADDRESSES
592 002024 005704 TST R4 ;OPERATOR INPUTS?
593 002026 001344 BNE BEGINX ;BR IF YES
594 002030 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
595 002034 001741 BEQ BEGINX ;BR IF NO MONITOR
596 002036 000137 012232 JMP #SENDAD ;GO TO END
597 002042 005077 177152 CLR @TAVEC+2
598 002046 3S:

```

```

599 ;*****
600 ;*****
601 ;
602 ;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
603 ;
604 ;I. DESIRED DRIVES CAN NOT BE TESTED
605 ; A. TYPE ERROR MESSAGE
606 ; B. EXAMINE R4
607 ; 1. R4>0 GOTO BEGINX
608 ; 2. R4=0 EXAMINE (42)
609 ; A. (42)=0 GOTO BEGINX
610 ; B. (42)>0 GOTO SENDAD
611 ;
612 ;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
613 ; A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
614 ; B. CONTINUE IN PROGRAM
615 ;
616 ;III. DESIRED DRIVE(S) CAN BE TESTED
617 ; A. CONTINUE IN PROGRAM
618 ;
619 ;*****
620 002046 012700 001224 CHKDRV: MOV #DRVKEY,R0 ;PICKUP ADDRESS OF ASCII DRIVE KEY
621 002052 004737 015046 JSR PC,@EXAM ;GO EXAMINE FIRST DRIVE
622 002056 000410 BR 1S ;OK TO TEST--GO CHECK NEXT
623 002060 116010 000001 MOVB 1(R0),(R0) ;REPLACE 1ST WITH 2ND
624 002064 001412 BEQ 2S ;BR IF NO 2ND DRIVE SELECTED
625 002066 004737 015046 JSR PC,@EXAM ;GO EXAMINE DRIVE
626 002072 000407 BR 2S ;OK TO TEST
627 002074 005010 CLR (R0) ;CLEAR DRIVE CODES
628 002076 000405 BR 2S
629 002100 005200 1S: INC R0 ;POINT TO 2ND
630 002102 004737 015046 JSR PC,@EXAM ;GO EXAMINE DRIVE
631 002106 000401 BR 2S ;OK TO TEST
632 002110 105010 CLRB (R0) ;CLEAR 2ND
633 002112 012700 001224 2S: MOV #DRVKEY,R0 ;RESET ADDRESS POINTERS
634 002116 010037 001230 MOV R0,@DRVPT
635 002122 121060 000001 CMPB (R0),1(R0) ;1ST = 2ND?
636 002126 001002 BNE 3S ;BR IF NO
637 002130 105060 000001 CLRB 1(R0) ;YES--CLEAR 2ND
638 002134 005710 3S: TST (R0) ;ANY DRIVES?
639 002136 001401 BEQ 5S ;BR IF NO
640 002140 000412 BR MANUAL
641 002142 104202 ERROR 202 ;NO DRIVES AVAILABLE
642 002144 012705 000002 MOV #2,R5 ;DRIVES & ADDRESS
643 002150 005704 TST R4 ;OPERATOR INPUTS?
644 002152 001272 BNE BEGINX ;BR IF YES
645 002154 013700 000042 MOV #42,R0 ;GET MONITOR RETURN ADDRESS
646 002160 001667 BEQ BEGINX ;NO MONITOR
647 002162 000137 012232 JMP #SENDAD ;GO TO END
648 002166 020427 000003 MANUAL: CMP R4,#3
649 002172 001002 BNE OK
650 002174 016704 175577 MOV -1,R4
651 002200 010437 001232 OK: MOV R4,@ASKKEY
652 002204 000405 BR START
653 002206 104401 001712 PWRST: TYPE #MSGID ;POWER FAIL RESTART
654 002212 012737 001224 001230 MOV #DRVKEY,@DRVPT

```

```

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

```

```

696 ;////////////////////////////////////
697 ;////////////////////////////////////
698 ;*****
699 ;THIS ISN'T A REAL TEST BUT A SMALL ROUTINE TO DETERMINE THE MAX.
700 ;TIME FOR THE WAIT LOOPS (WAIT FOR "READY" AND "TRANSFER REQUEST")
701 ;*****
702 ;*TEST 1 ROUTINE TO DETERMINE TIME OF WAIT LOOPS
703 ;*****
704 002436 000004 TST1: SCOPE
705 002440 012767 000001 176520 MOV #1,$TIMES ;DO 1 ITERATION
706 002446 005737 001100 TST @#SPASS ;IS THIS THE FIRST PASS?
707 002452 001021 BNE TST2 ;BR IF NO
708 002454 000005 RESET
709 002456 010314 MOV DRIVE,@TACS ;SELECT THE DRIVE
710 002460 112714 000017 MOVB #REWIND!GO,@TACS ;START A REWIND
711 002464 104412 WAITREADY ;WAIT FOR READY
712 002466 112714 000001 MOVB #WFG!GO,@TACS ;WRITE A FILE GAP
713 002472 104412 WAITREADY ;WAIT ON READY
714 002474 163737 013124 013150 SUB @#HGHTIM,@#MAXCNT ;GET THE TIME IT TOOK
715 002502 005237 013150 INC @#MAXCNT ;MAKE IT BIGGER
716 002506 006137 013150 ROL @#MAXCNT
717 002512 006137 013150 ROL @#MAXCNT
718 ;////////////////////////////////////
719 ;////////////////////////////////////
720 ;THE FOLLOWING TEST WILL CHECK TIMING ERRORS FOR BOTH "WRITE" AND "READ".
721 ;////////////////////////////////////
722 ;*****
723 ;*TEST 2 TEST "TIMING ERROR" FOR "WRITE"
724 ;*****
725 002516 000004 TST2: SCOPE
726 002520 012767 000012 176440 MOV #10,$TIMES ;DO 10 ITERATIONS
727 002526 012767 002554 176352 MOV #7$,SLPADR ;SET SCOPE LOOP ADDRESS
728 002534 012767 002656 176426 MOV #TST3,$ESCAPE ;ESCAPE TO TEST 3 ON ERROR
729 002542 000005 RESET
730 002544 010314 MOV DRIVE,@TACS ;SELECT DRIVE
731 002546 112714 000017 MOVB #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
732 002552 104412 WAITREADY ;WAIT ON "READY"
733 002554 112714 000003 7$: MOVB #WRITE!GO,@TACS ;WRITE FUNCTION
734 002560 104413 WAITXFER ;WAIT ON "XFER REQ"
735 002562 105714 TSTB @TACS ;IS "XFER REQ" = 1
736 002564 100401 BMI 1$ ;BR IF YES
737 002566 104401 ERROR 1 ;NO "XFER REQ" OCCURRED
738 002570 005015 CLR @TADB ;KNOCK DOWN "XFER REQ"
739 002572 005000 CLR R0 ;CLEAR COUNTER
740 002574 005001 CLR R1
741 002576 105714 2$: TSTB @TACS ;WAIT FOR "XFER REQ"
742 002600 100405 BMI 3$
743 002602 062700 000010 ADD #10,R0 ;KEEP TRACK OF HOW LONG
744 002606 005501 ADC R1 ;IT TAKES "XFER REQ" TO SET
745 002610 100372 BPL 2$
746 002612 104001 ERROR 1 ;"XFER REQ" FAILED TO SET
747 002614 105714 3$: TSTB @TACS ;SAMPLE "XFER REQ"
748 002616 100005 BPL 4$ ;IF "XFER REQ" = 0 SOMETHING IS WRONG
749 002620 162700 000001 SUB #1,R0 ;DOWN COUNT THE COUNTER
750 002624 005601 SBC R1 ;SO A "TIMING ERROR WILL
751 002626 100372 BPL 3$ OCCUR

```

```

752 002630 000401          BR      5$
753 002632 104001          4$:  ERROR 1          ;"XFER REQ" SHOULDN'T HAVE CLEARED
754 002634 105715          5$:  TSTB  @TADB        ;CLEAP "XFER REQ"
755 002636 104412          WAITREADY          ;WAIT FOR "READY"
756 002640 005714          TST  @TACS         ;IS "ERROR" BIT SET?
757 002642 104401          BMI  6$            ;BR IF YES
758 002644 104001          ERROR 1            ;"ERROR" BIT NOT SET
759 002646 032714 002000  6$:  BIT  #TIMERR,@TACS ;IS "TIMING ERROR" = 1?
760 002652 001001          BNE  TST3          ;;BR IF YES
761 002654 104001          ERROR 1            ;"TIMING ERROR" NOT SET
762                                     ;*****
763                                     ;*TEST 3 TEST "TIMING ERROR" FOR "READ"
764                                     ;*****
765 002656 000004          TST3: SCOPE
766 002660 012767 000012 176300  MOV  #10,,$TIMES ;DO 10. ITERATIONS
767 002666 012767 002714 176212  MOV  #7$,SLPADR  ;;SET SCOPE LOOP ADDRESS
768 002674 012767 003062 176266  MOV  #TST4,$ESCAPE ;;ESCAPE TO TEST 4 ON ERROR
769 002702 000005          RESET
770 002704 010314          MOV  DRIVE,@TACS ;SELECT DRIVE
771 002706 112714 000017  MOVB #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
772 002712 104412          WAITREADY          ;WAIT FOR "READY"
773 002714 112714 000003  7$:  MOVB #WRITE!GO,@TACS ;WRITE
774 002720 012700 000006  MOV  #6,R0          ;WRITE THIS MANY BYTES ON TAPE
775 002724 104413          WAITXFER          ;WAIT FOR "XFER REQ"
776 002726 112715 000377  8$:  MOVB #377,@TADB    ;WRITE ONE BYTE
777 002732 104413          WAITXFER          ;WAIT FOR "XFER REQ"
778 002734 005300          DEC  R0             ;LAST BYTE OUT?
779 002736 003373          BGT  8$            ;BR IF NO
780 002740 052714 000020  BIS  #ILBS,@TACS   ;WRITE "CRC"
781 002744 104412          WAITREADY          ;WAIT FOR "READY"
782 002746 112714 000017  MOVB #REWIND!GO,@TACS ;BACK OVER THE BLOCK
783 002752 104412          WAITREADY          ;WAIT FOR "READY"
784 002754 112714 000005  MOVB #READ!GO,@TACS ;START A "READ"
785 002760 104413          WAITXFER          ;WAIT FOR "XFER REQ"
786 002762 105714          TSTB @TACS         ;IS "XFER REQ" SET
787 002764 100401          BMI  1$            ;BR IF YES
788 002766 104001          ERROR 1            ;"XFER REQ" DIDN'T SET
789 002770 105715          1$:  TSTB @TADB        ;KNOCK DOWN "XFER REQ"
790 002772 005000          CLR  R0             ;CLEAR COUNTERS
791 002774 005001          CLR  R1
792 002776 105714          2$:  TSTB @TACS         ;FIND OUT HOW LONG IT
793 003000 100405          BMI  3$            ;TAKES FOR "XFER REQ"
794 003002 062700 000012  ADD  #10,,$R0      ;TO SET
795 003006 005501          ADC  R1
796 003010 100372          BPL  2$
797 003012 104001          ERROR 1            ;"XFER REQ" FAILED TO SET
798 003014 105714          3$:  TSTB @TACS         ;NOW WASTE MORE THAN THE ABOVE
799 003016 100005          BPL  4$            ;AMOUNT OF TIME SO THAT
800 003020 162700 000001  SUB  #1,R0          ;A "TE" WILL OCCUR
801 003024 005601          SBC  R1
802 003026 100372          BPL  3$
803 003030 000401          BR   5$
804 003032 104001          4$:  ERROR 1            ;"XFER REQ" CLEARED SOME HOW
805 003034 105715          5$:  TSTB @TADB        ;KNOCK DOWN "XFER REQ"
806 003036 052714 000020  BIS  #ILBS,@TACS   ;STOP THE READ
807 003042 104412          WAITREADY          ;WAIT ON "READY"
    
```

```

808 003044 005714          TST  @TACS         ;IS "ERROR" SET?
809 003046 100401          BMI  6$            ;BR IF YES
810 003050 104001          ERROR 1            ;"ERROR" NOT = 1
811 003052 032714 002000  6$:  BIT  #TIMERR,@TACS ;IS THERE A "TIMING ERROR"
812 003056 001001          BNE  TST4          ;;GO TO NEXT TEST IF YES
813 003060 104001          ERROR 1            ;"TIMING ERROR" NOT SET
814                                     ;*****
815                                     ;*TEST 4 TEST "ERROR" OUTPUT OF STATUS ROM USING A "TIMING ERROR"
816                                     ;*****
817 003062 000004          TST4: SCOPE
818 003064 012767 000012 176074  MOV  #10,,$TIMES ;DO 10. ITERATIONS
819 003072 012767 003126 176006  MOV  #15,$LPADR  ;;SET SCOPE LOOP ADDRESS
820 003100 012767 003342 176062  MOV  #TST5,$ESCAPE ;;ESCAPE TO TEST 5 ON ERROR
821 003106 000005          RESET
822 003110 010314          MOV  DRIVE,@TACS ;SELECT DRIVE
823 003112 112714 000017  MOVB #REWIND!GO,@TACS ;GO TO BEGINNING OF TAPE
824 003116 104412          WAITREADY          ;WAIT ON "READY" TO SET
825 003120 112714 000001  MOVB #WFG!GO,@TACS ;GET ON THE OXIDE
826 003124 104412          WAITREADY          ;WAIT ON "READY" TO SET
827 003126 112714 000003  1$:  MOVB #WRITE!GO,@TACS ;START A WRITE
828 003132 104413          WAITXFER          ;WAIT FOR "TRANSFER REQ"
829 003134 112715 000377  MOVB #377,@TADB    ;WRITE ONE BYTE
830 003140 005000          CLR  R0             ;CLEAR THE DOUBLE LENGTH COUNTER
831 003142 005001          CLR  R1
832 003144 105714          2$:  TSTB @TACS         ;CHECK FOR "XFER REQ"
833 003146 100405          BMI  3$            ;BR IF SET
834 003150 062700 000012  ADD  #10,,$R0      ;COUNT UP UNTIL "XFER REQ"
835 003154 005501          ADC  R1             ;SETS OR OVERFLOW OCCURS
836 003156 100372          BPL  2$
837 003160 104001          ERROR 1            ;"XFER REQ" FAILED TO SET
838 003162 105714          3$:  TSTB @TACS         ;CHECK "XFER REQ"
839 003164 100401          BMI  4$            ;BR IF SET
840 003166 104001          ERROR 1            ;"XFER REQ" SHOULDN'T HAVE CLEARED
841 003170 162700 000001  4$:  SUB  #1,R0          ;COUNT DOWN TO CAUSE A "TE"
842 003174 005601          SBC  R1
843 003176 100371          BPL  3$
844 003200 105715          TSTB @TADB        ;CLEAR "XFER REQ"
845 003202 104412          WAITREADY          ;WAIT FOR READY
846 003204 005714          TST  @TACS         ;MAKE SURE "ERROR" IS SET
847 003206 100401          BMI  5$
848 003210 104001          ERROR 1            ;"ERROR" DIDN'T SET
849 003212 032714 002000  5$:  BIT  #TIMERR,@TACS ;MAKE SURE "TE" IS SET
850 003216 001001          BNE  6$
851 003220 104001          ERROR 1            ;"TE" FAILED TO SET
852 003222 112714 000000  6$:  MOVB #WFG,@TACS   ;CHECK "ERROR" WITH "WFG"
853 003226 005714          TST  @TACS         ;SAMPLE THE "ERROR" BIT
854 003230 100001          BPL  7$            ;BR IF "ERROR" = 0
855 003232 104001          ERROR 1            ;"ERROR" NOT = 0
856 003234 112714 000002  7$:  MOVB #WRITE,@TACS ;CHECK "ERROR" WITH "WRITE"
857 003240 005714          TST  @TACS         ;SAMPLE THE "ERROR" BIT
858 003242 100401          BMI  8$            ;BR IF "ERROR" = 1
859 003244 104001          ERROR 1            ;"ERROR" NOT = 1
860 003246 112714 000004  8$:  MOVB #READ,@TACS  ;CHECK "ERROR" WITH "READ"
861 003252 005714          TST  @TACS         ;SAMPLE THE "ERROR" BIT
862 003254 100401          BMI  9$            ;BR IF "ERROR" = 1
863 003256 104001          ERROR 1            ;"ERROR" NOT = 1
    
```



```

864 003260 112714 000006 9S: MOVB #BSFG,@TACS ;CHECK "ERROR" WITH "BSFG"
865 003264 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
866 003266 100001 BPL 10S ;BR IF "ERROR" = 0
867 003270 104001 ERROR 1 ;"ERROR" NOT = 0
868 003272 112714 000010 10S: MOVB #BSBG,@TACS ;CHECK "ERROR" WITH "BSBG"
869 003276 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
870 003300 100001 BPL 11S ;BR IF "ERROR" = 0
871 003302 104001 ERROR 1 ;"ERROR" NOT = 0
872 003304 112714 000012 11S: MOVB #SFFG,@TACS ;CHECK "ERROR" WITH "SFFG"
873 003310 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
874 003312 100001 BPL 12S ;BR IF "ERROR" = 0
875 003314 104001 ERROR 1 ;"ERROR" NOT = 0
876 003316 112714 000014 12S: MOVB #SFBG,@TACS ;CHECK "ERROR" WITH "SFBG"
877 003322 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
878 003324 100001 BPL 13S ;BR IF "ERROR" = 0
879 003326 104001 ERROR 1 ;"ERROR" NOT = 0
880 003330 112714 000016 13S: MOVB #REWIND,@TACS ;CHECK "ERROR" WITH "REWIND"
881 003334 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
882 003336 100001 BPL TST5 ;BR IF "ERROR" = 0
883 003340 104001 ERROR 1 ;"ERROR" NOT = 0
884
885 ;////////////////////////////////////
886 ;THE FOLLOWING TESTS CHECK THE INTERRUPT CIRCUITRY FOR:
887 ;
888 ;1) INTERRUPT OCCURS AT THE PROPER "BR" LEVEL
889 ;2) "READY" WILL INTERRUPT
890 ;3) "TRANSFER REQUEST" WILL INTERRUPT
891 ;4) IMPROPER INTERRUPTS DO NOT OCCUR
892 ;5) WITH AN ERROR PENDING AND "XFER REQ." SET THE FOLLOWING SEQUENCE OCCURS
893 ; A) "XFER REQ" CAUSES AN INTERRUPT
894 ; B) CLEARING "XFER REQ" ALLOWS "READY" TO SET
895 ; C) "READY" CAUSES AN INTERRUPT
896 ;
897 ;////////////////////////////////////
898
899 ;*****
900 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
901 ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
902 ;IT WILL THEN SET THE PRIORITY LEVEL TO 0 AND WASTE ENOUGH TIME
903 ;FOR AN INTERRUPT TO OCCUR.
904 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
905 ;AND CHECK IF AN INTERRUPT OCCURRED.
906 ;*****
907 ;*TEST 5 TEST INTERRUPT WITH READY = "1" AT LEVEL 0
908 ;*****
909 003342 000004 TST5: SCOPE
910 003344 012767 003362 175534 MOV #1S,SFPADR ;;SET SCOPE LOOP ADDRESS
911 003352 012767 003520 175610 MOV #6S,SESCAPE ;;ESCAPE TO 6S ON ERROR
912 003360 000005 RESET
913 003362 010314 1S: MOV DRIVE,@TACS
914 003364 012737 000340 177776 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
915 003372 012700 000000 MOV #0,R0 ;TEST AT PRIORITY LEVEL 0
916 003376 005001 CLR R1 ;CLEAR INTERRUPT KEY
917 003400 012702 000001 MOV #1,R2 ;SETUP TO WASTE A LITTLE TIME
918 003404 012777 003510 175604 MOV #4S,@TAVEC ;INTERRUPT RETURN
919 003412 012777 000340 175600 MOV #340,@TAVEC+2 ;LOCK OUT THE WORLD IF INTERRUPT
    
```

```

920 003420 012737 000137 000000 MOV #137,#0 ;SETUP TO CATCH INTERRUPT IF
921 003426 012737 003514 000002 MOV #5S,#2 ; IT GOES TO LOCATION 0
922 003434 112714 000100 MOVB #INT.EN,@TACS ;SET INTERRUPT ENABLE
923 003440 012737 000000 177776 MOV #0#40,@#PS ;SET TO PRIORITY LEVEL 0
924 003446 006302 2S: ASL R2 ;KILL SOME TIME
925 003450 001376 BNE 2S
926 003452 012737 000340 177776 MOV #340,@#PS ;LOCK OUT THE WORLD
927 003460 005701 TST R1 ;DID AN INTERRUPT OCCUR?
928 003462 001005 BNE 3S ;BR IF YES
929 003464 022737 000000 001222 CMP #0#40,@#TAPRIO ;WAS AN INTERRUPT EXPECTED?
930 003472 002012 BGE 6S ;BR IF NO
931 003474 104006 ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 0
932
933 003476 022737 000000 001222 3S: CMP #0#40,@#TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
934 003504 002405 BLT 6S ;BR IF YES
935 003506 104006 ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 0
936
937 003510 005201 4S: INC R1 ;SET INTERRUPT KEY
938 003512 000002 RTI ;RETURN AFTER INTERRUPT
939 003514 022626 5S: CMP (SP)+,(SP)+ ;POP THE STACK
940 003516 104006 ERROR 6 ;INTERRUPT WENT TO LOCATION 0
941
942 003520 005014 6S: CLR @TACS ;CLEAR INTERRUPT ENABLE
943 003522 013777 001220 175466 MOV #@TAVEC+2,@TAVEC ;SET TRAP CATCHER
944 003530 005077 175464 CLR @TAVEC+2
945 003534 005037 000000 CLR #0
946 003540 005037 000002 CLR #02
947
948 ;*****
949 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
950 ;A JMP @(PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
951 ;IT WILL THEN SET THE PRIORITY LEVEL TO 1 AND WASTE ENOUGH TIME
952 ;FOR AN INTERRUPT TO OCCUR.
953 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
954 ;AND CHECK IF AN INTERRUPT OCCURRED.
955 ;*****
956 ;*TEST 6 TEST INTERRUPT WITH READY = "1" AT LEVEL 1
957 ;*****
958 003544 000004 TST6: SCOPE
959 003546 012767 003564 175332 MOV #1S,SFPADR ;;SET SCOPE LOOP ADDRESS
960 003554 012767 003722 175406 MOV #6S,SESCAPE ;;ESCAPE TO 6S ON ERROR
961 003562 000005 RESET
962 003564 010314 1S: MOV DRIVE,@TACS
963 003566 012737 000340 177776 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
964 003574 012700 000001 MOV #1,R0 ;TEST AT PRIORITY LEVEL 1
965 003600 005001 CLR R1 ;CLEAR INTERRUPT KEY
966 003602 012702 000001 MOV #1,R2 ;SETUP TO WASTE A LITTLE TIME
967 003606 012777 003712 175402 MOV #4S,@TAVEC ;INTERRUPT RETURN
968 003614 012777 000340 175376 MOV #340,@TAVEC+2 ;LOCK OUT THE WORLD IF INTERRUPT
969 003622 012737 000137 000000 MOV #137,#0 ;SETUP TO CATCH INTERRUPT IF
970 003630 012737 003716 000002 MOV #5S,#2 ; IT GOES TO LOCATION 0
971 003642 012737 000040 177776 MOVB #INT.EN,@TACS ;SET INTERRUPT ENABLE
972 003650 006302 2S: ASL R2 ;SET TO PRIORITY LEVEL 1
973 003652 001376 BNE 2S ;KILL SOME TIME
974 003654 012737 000340 177776 MOV #340,@#PS ;LOCK OUT THE WORLD
975 003662 005701 TST R1 ;DID AN INTERRUPT OCCUR?
    
```

```

976 003664 001005          BNE 3$ ;BR IF YES
977 003666 022737 000040 001222 CMP #1*40,@TAPRIO ;WAS AN INTERRUPT EXPECTED?
978 003674 002012          BGE 6$ ;BR IF NO
979 003676 104006          ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 1
980
981 003700 022737 000040 001222 3$: CMP #1*40,@TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
982 003706 002405          BLT 6$ ;BR IF YES
983 003710 104006          ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 1
984
985 003712 005201          4$: INC R1 ;SET INTERRUPT KEY
986 003714 000302          RTI ;RETURN AFTER INTERRUPT
987 003716 022626          5$: CMP (SP)+,(SP)+ ;POP THE STACK
988 003720 104006          ERROR 6 ;INTERRUPT WENT TO LOCATION 0
989
990 003722 005014          6$: CLR @TACS ;CLEAR INTERRUPT ENABLE
991 003724 013777 001220 175264 MOV @TAVEC+2,@TAVEC ;SET TRAP CATCHER
992 003732 005077 175262 CLR @TAVEC+2
993 003736 005037 000000 CLR @#0
994 003742 005037 000002 CLR @#2
995
996 ;*****
997 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
998 ;A JMP @PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
999 ;IT WILL THEN SET THE PRIORITY LEVEL TO 2 AND WASTE ENOUGH TIME
1000 ;FOR AN INTERRUPT TO OCCUR.
1001 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1002 ;AND CHECK IF AN INTERRUPT OCCURRED.
1003 ;*****
1004 ;*TEST 7 TEST INTERRUPT WITH READY = "1" AT LEVEL 2
1005 ;*****
1006 TST7: SCOPE
1007 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1008 MOV #6$,SESCAPE ;;ESCAPE TO 6$ ON ERROR
1009 RESET
1010 MOV DRIVE,@TACS
1011 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
1012 CLR R1 ;TEST AT PRIORITY LEVEL 2
1013 MOV #1,R2 ;CLEAR INTERRUPT KEY
1014 MOV #4$,@TAVEC ;SETUP TO WASTE A LITTLE TIME
1015 MOV #340,@TAVEC+2 ;INTERRUPT RETURN
1016 MOV #137,@#0 ;LOCK OUT THE WORLD IF INTERRUPT
1017 MOV #5$,@#2 ;SETUP TO CATCH INTERRUPT IF
1018 MOV @INT.EN,@TACS ; IT GOES TO LOCATION 0
1019 MOV #2*40,@#PS ;SET INTERRUPT ENABLE
1020 ASL R2 ;SET TO PRIORITY LEVEL 2
1021 BNE 2$ ;KILL SOME TIME
1022 MOV #340,@#PS ;LOCK OUT THE WORLD
1023 TST R1 ;DID AN INTERRUPT OCCUR?
1024 BNE 3$ ;BR IF YES
1025 CMP #2*40,@TAPRIO ;WAS AN INTERRUPT EXPECTED?
1026 BGE 6$ ;BR IF NO
1027 ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 2
1028
1029 004102 022737 000100 001222 3$: CMP #2*40,@TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1030 004110 002405          BLT 6$ ;BR IF YES
1031 004112 104006          ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 2
    
```

```

1032
1033 004114 005201          4$: INC R1 ;SET INTERRUPT KEY
1034 004116 000302          RTI ;RETURN AFTER INTERRUPT
1035 004120 022626          5$: CMP (SP)+,(SP)+ ;POP THE STACK
1036 004122 104006          ERROR 6 ;INTERRUPT WENT TO LOCATION 0
1037
1038 004124 005014          6$: CLR @TACS ;CLEAR INTERRUPT ENABLE
1039 004126 013777 001220 175062 MOV @TAVEC+2,@TAVEC ;SET TRAP CATCHER
1040 004134 005077 175060 CLR @TAVEC+2
1041 004140 005037 000000 CLR @#0
1042 004144 005037 000002 CLR @#2
1043
1044 ;*****
1045 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1046 ;A JMP @PC)+ IS STORED INCASE THE VECTOR IS READ AS 0.
1047 ;IT WILL THEN SET THE PRIORITY LEVEL TO 3 AND WASTE ENOUGH TIME
1048 ;FOR AN INTERRUPT TO OCCUR.
1049 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1050 ;AND CHECK IF AN INTERRUPT OCCURRED.
1051 ;*****
1052 ;*TEST 10 TEST INTERRUPT WITH READY = "1" AT LEVEL 3
1053 ;*****
1054 TST10: SCOPE
1055 MOV #1$,SLPADR ;;SET SCOPE LOOP ADDRESS
1056 MOV #6$,SESCAPE ;;ESCAPE TO 6$ ON ERROR
1057 RESET
1058 MOV DRIVE,@TACS
1059 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
1060 CLR R1 ;TEST AT PRIORITY LEVEL 3
1061 MOV #1,R2 ;CLEAR INTERRUPT KEY
1062 MOV #4$,@TAVEC ;SETUP TO WASTE A LITTLE TIME
1063 MOV #340,@TAVEC+2 ;INTERRUPT RETURN
1064 MOV #137,@#0 ;LOCK OUT THE WORLD IF INTERRUPT
1065 MOV #5$,@#2 ;SETUP TO CATCH INTERRUPT IF
1066 MOV @INT.EN,@TACS ; IT GOES TO LOCATION 0
1067 MOV #3*40,@#PS ;SET INTERRUPT ENABLE
1068 ASL R2 ;SET TO PRIORITY LEVEL 3
1069 BNE 2$ ;KILL SOME TIME
1070 MOV #340,@#PS ;LOCK OUT THE WORLD
1071 TST R1 ;DID AN INTERRUPT OCCUR?
1072 BNE 3$ ;BR IF YES
1073 CMP #3*40,@TAPRIO ;WAS AN INTERRUPT EXPECTED?
1074 BGE 6$ ;BR IF NO
1075 ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 3
1076
1077 004304 022737 000140 001222 3$: CMP #3*40,@TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1078 004312 002405          BLT 6$ ;BR IF YES
1079 004314 104006          ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 3
1080
1081 004316 005201          4$: INC R1 ;SET INTERRUPT KEY
1082 004320 000302          RTI ;RETURN AFTER INTERRUPT
1083 004322 022626          5$: CMP (SP)+,(SP)+ ;POP THE STACK
1084 004324 104006          ERROR 6 ;INTERRUPT WENT TO LOCATION 0
1085
1086 004326 005014          6$: CLR @TACS ;CLEAR INTERRUPT ENABLE
1087 004330 013777 001220 174660 MOV @TAVEC+2,@TAVEC ;SET TRAP CATCHER
    
```

```

1088 004336 005077 174656 CLR @TAVEC+2
1089 004342 005037 000000 CLR @#0
1090 004346 005037 000002 CLR @#2
1091 ;*****
1092 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1093 ;A JMP @PC)+ IS STORED IN CASE THE VECTOR IS READ AS 0.
1094 ;IT WILL THEN SET THE PRIORITY LEVEL TO 4 AND WASTE ENOUGH TIME
1095 ;FOR AN INTERRUPT TO OCCUR.
1096 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1097 ;AND CHECK IF AN INTERRUPT OCCURRED.
1098 ;*****
1099 ;*TEST 11 TEST INTERRUPT WITH READY = "1" AT LEVEL 4
1100 ;*****
1101 004352 000004 TST11: SCOPE
1102 004354 012767 004372 174524 MOV #16,$LPADR ;;SET SCOPE LOOP ADDRESS
1103 004362 012767 004530 174600 MOV #66,$ESCAPE ;;ESCAPE TO 66 ON ERROR
1104 004370 000005 RESET
1105 004372 010314 1s: MOV DRIVE,@TACS
1106 004374 012737 000340 177776 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
1107 004402 012700 000004 MOV #4,R0 ;TEST AT PRIORITY LEVEL 4
1108 004406 005001 CLR R1 ;CLEAR INTERRUPT KEY
1109 004410 012702 000001 MOV #1,R2 ;SETUP TO WASTE A LITTLE TIME
1110 004414 012777 004520 174574 MOV #45,@TAVEC ;INTERRUPT RETURN
1111 004422 012777 000340 174570 MOV #340,@TAVEC+2 ;LOCK OUT THE WORLD IF INTERRUPT
1112 004430 012737 000137 000000 MOV #137,@#0 ;SETUP TO CATCH INTERRUPT IF
1113 004436 012737 004524 000002 MOV #55,@#2 ; IT GOES TO LOCATION 0
1114 004444 112714 000100 MOV #INT,EN,@TACS ;SET INTERRUPT ENABLE
1115 004450 012737 000200 177776 MOV #4*40,@#PS ;SET TO PRIORITY LEVEL 4
1116 004456 006302 2s: ASL R2 ;KILL SOME TIME
1117 004460 001376 BNE 25
1118 004462 012737 000340 177776 MOV #340,@#PS ;LOCK OUT THE WORLD
1119 004470 005701 TST R1 ;DID AN INTERRUPT OCCUR?
1120 004472 001005 BNE 35 ;BR IF YES
1121 004474 022737 000200 001222 CMP #4*40,@#TAPRIO ;WAS AN INTERRUPT EXPECTED?
1122 004502 002012 BGE 65 ;BR IF NO
1123 004504 104006 ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 4
1124
1125 004506 022737 000200 001222 3s: CMP #4*40,@#TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1126 004514 002405 BLT 65 ;BR IF YES
1127 004516 104006 ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 4
1128
1129 004520 005201 4s: INC R1 ;SET INTERRUPT KEY
1130 004522 000002 RTI ;RETURN AFTER INTERRUPT
1131 004524 022626 5s: CMP (SP)+,(SP)+ ;POP THE STACK
1132 004526 104006 ERROR 6 ;INTERRUPT WENT TO LOCATION 0
1133
1134 004530 005014 6s: CLR @TACS ;CLEAR INTERRUPT ENABLE
1135 004532 013777 001220 174456 MOV @#TAVEC+2,@TAVEC ;SET TRAP CATCHER
1136 004540 005077 174454 CLR @TAVEC+2
1137 004544 005037 000000 CLR @#0
1138 004550 005037 000002 CLR @#2
1139 ;*****
1140 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1141 ;A JMP @PC)+ IS STORED IN CASE THE VECTOR IS READ AS 0.
1142 ;IT WILL THEN SET THE PRIORITY LEVEL TO 5 AND WASTE ENOUGH TIME
1143 ;FOR AN INTERRUPT TO OCCUR.
    
```

```

1144 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1145 ;AND CHECK IF AN INTERRUPT OCCURRED.
1146 ;*****
1147 ;*TEST 12 TEST INTERRUPT WITH READY = "1" AT LEVEL 5
1148 ;*****
1149 004554 000004 TST12: SCOPE
1150 004556 012767 004574 174322 MOV #16,$LPADR ;;SET SCOPE LOOP ADDRESS
1151 004564 012767 004732 174376 MOV #66,$ESCAPE ;;ESCAPE TO 66 ON ERROR
1152 004572 000005 RESET
1153 004574 010314 1s: MOV DRIVE,@TACS
1154 004576 012737 000340 177776 MOV #340,@#PS ;LOCK OUT ALL INTERRUPTS
1155 004604 012700 000005 MOV #5,R0 ;TEST AT PRIORITY LEVEL 5
1156 004610 005001 CLR R1 ;CLEAR INTERRUPT KEY
1157 004612 012702 000001 MOV #1,R2 ;SETUP TO WASTE A LITTLE TIME
1158 004616 012777 004722 174372 MOV #45,@TAVEC ;INTERRUPT RETURN
1159 004624 012777 000340 174366 MOV #340,@TAVEC+2 ;LOCK OUT THE WORLD IF INTERRUPT
1160 004632 012737 000137 000000 MOV #137,@#0 ;SETUP TO CATCH INTERRUPT IF
1161 004640 012737 004726 000002 MOV #55,@#2 ; IT GOES TO LOCATION 0
1162 004646 112714 000100 MOV #INT,EN,@TACS ;SET INTERRUPT ENABLE
1163 004652 012737 000240 177776 MOV #5*40,@#PS ;SET TO PRIORITY LEVEL 5
1164 004660 006302 2s: ASL R2 ;KILL SOME TIME
1165 004662 001376 BNE 25
1166 004664 012737 000340 177776 MOV #340,@#PS ;LOCK OUT THE WORLD
1167 004672 005701 TST R1 ;DID AN INTERRUPT OCCUR?
1168 004674 001005 BNE 35 ;BR IF YES
1169 004676 022737 000240 001222 CMP #5*40,@#TAPRIO ;WAS AN INTERRUPT EXPECTED?
1170 004704 002012 BGE 65 ;BR IF NO
1171 004706 104006 ERROR 6 ;INTERRUPT FAILED TO OCCUR AT PRIORITY LEVEL 5
1172
1173 004710 022737 000240 001222 3s: CMP #5*40,@#TAPRIO ;INTERRUPT OCCURRED--SHOULD IT HAVE?
1174 004716 002405 BLT 65 ;BR IF YES
1175 004720 104006 ERROR 6 ;INTERRUPT OCCURRED AT PRIORITY LEVEL 5
1176
1177 004722 005201 4s: INC R1 ;SET INTERRUPT KEY
1178 004724 000002 RTI ;RETURN AFTER INTERRUPT
1179 004726 022626 5s: CMP (SP)+,(SP)+ ;POP THE STACK
1180 004730 104006 ERROR 6 ;INTERRUPT WENT TO LOCATION 0
1181
1182 004732 005014 6s: CLR @TACS ;CLEAR INTERRUPT ENABLE
1183 004734 013777 001220 174254 MOV @#TAVEC+2,@TAVEC ;SET TRAP CATCHER
1184 004742 005077 174252 CLR @TAVEC+2
1185 004746 005037 000000 CLR @#0
1186 004752 005037 000002 CLR @#2
1187 ;*****
1188 ;THIS TEST WILL SETUP THE TA11'S VECTOR ADDRESS AND AT ADDRESS 0
1189 ;A JMP @PC)+ IS STORED IN CASE THE VECTOR IS READ AS 0.
1190 ;IT WILL THEN SET THE PRIORITY LEVEL TO 6 AND WASTE ENOUGH TIME
1191 ;FOR AN INTERRUPT TO OCCUR.
1192 ;AFTER THE TIME IS UP IT WILL RESTORE THE PS TO LEVEL 7
1193 ;AND CHECK IF AN INTERRUPT OCCURRED.
1194 ;*****
1195 ;*TEST 13 TEST INTERRUPT WITH READY = "1" AT LEVEL 6
1196 ;*****
1197 004756 000004 TST13: SCOPE
1198 004760 012767 004776 174120 MOV #16,$LPADR ;;SET SCOPE LOOP ADDRESS
1199 004766 012767 005134 174174 MOV #66,$ESCAPE ;;ESCAPE TO 66 ON ERROR
    
```

```

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

```

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

```

1312 005524 104006          ERROR 6          ;INTERRUPT WENT TO LOCATION 0
1313 005526 022626          2s:  CMP (SP)+,(SP)+ ;POP STACK
1314 005530 000005          3s:  RESET          ;CLEAR ALL
1315 005532 013777 001220 173456  MOV @TAVEC+2,@TAVEC ;SET TRAP CATCHER
1316 005540 005077 173454  CLR @TAVEC+2
1317 005544 005037 000000  CLR #0
1318 005550 005037 000002  CLR #2
1319
1320 ;*****
;*TEST 16 TEST INTERRUPT WITH "READY" = 0 AND "XFER REQ" = 0
;*****
1321 TST16: SCOPE
1322 005554 000004          MOV #1,$TIMES ;DO 1 ITERATION
1323 005556 012767 000001 173402  MOV #56,$ESCAPE ;ESCAPE TO 5S ON ERROR
1324 005564 012767 005724 173376  MOV #56,3S
1325 005572 012767 005724 000064  MOV #1,R1
1326 005600 012701 000001  MOV #340,@#PS ;LOCK OUT THE WORLD
1327 005604 012737 000340 177776  RESET          ;CLEAR THE WORLD
1328 005612 000005          MOV DRIVE,@TACS ;SELECT DRIVE
1329 005614 010314          WAITREADY     ;GO WAIT FOR "READY" TO SET
1330 005616 104412          MOV #4s,@TAVEC ;RETURN IF INTERRUPT OCCURS
1331 005620 012777 005714 173370  MOV #340,@TAVEC+2 ;LEVEL "7" IF INTERRUPT
1332 005626 012777 000340 173364  MOV #137,#0 ;CATCH INTERRUPT IF IT GOES TO
1333 005634 012737 000137 000000  MOV #7s,@#2 ; LOCATION 0
1334 005642 012737 005720 000002  MOV #REWIND!INT.EN!GO,@TACS
1335 005650 112714 000117  MOV #TR.REQ!READY,@TACS ;GIVE READY TIME TO CLEAR
1336 005654 032714 000240 1s:  BIT
1337 005660 001404          BEQ 2s
1338 005662 005327          DEC (PC)+
1339 005664 000000          3s:  0
1340 005666 001372          BNE 1s
1341 005670 104001          ERROR 1 ;READY OR XFER REQ = "1"
1342 005672 005037 177776  2s:  CLR @#PS ;ALLOW INTERRUPT
1343 005676 005000          CLR R0 ;AT LEVEL 0
1344 005700 006301          6s:  ASL R1 ;GIVE INTERRUPT TIME TO COME IN
1345 005702 001376          BNE 6s
1346 005704 012737 000340 177776  MOV #340,@#PS ;LOCK OUT INTERRUPT
1347 005712 000404          BR 5s ;EXIT WITH NO ERRORS
1348 005714 022626          4s:  CMP (SP)+,(SP)+ ;POP STACK
1349 005716 104006          ERROR 6 ;INTERRUPT OCCURRED
1350 005720 022626          7s:  CMP (SP)+,(SP)+ ;POP THE STACK
1351 005722 104006          ERROR 6 ;INTERRUPTED TO LOCATION 0
1352 005724 000005          5s:  RESET          ;CLEAR THE WORLD
1353 005726 013777 001220 173262  MOV @TAVEC+2,@TAVEC ;SET TRAP CATCHER
1354 005734 005077 173260  CLR @TAVEC+2
1355 005740 005037 000000  CLR #0
1356 005744 005037 000002  CLR #2
1357
1358 ;*****
;*TEST 17 TEST INTERRUPT WITH "XFER REQ" = 1 & "TIMING ERROR" = 1
;*****
1359 TST17: SCOPE
1360 005750 000004          MOV #10,$TIMES ;DO 10 ITERATIONS
1361 005752 012767 000012 173206  MOV #1s,$LPADR ;SET SCOPE LOOP ADDRESS
1362 005760 012767 006014 173120  MOV #11s,$ESCAPE ;ESCAPE TO 11s ON ERROR
1363 005766 012767 006222 173174  RESET          ;CLEAR ALL
1364 005774 000005          MOV DRIVE,@TACS ;SELECT DRIVE
1365 005776 010314          MOV #REWIND!GO,@TACS ;GO TO "BOT"
1366 006000 112714 000017  WAITREADY     ;WAIT ON READY
1367 006004 104412

```

```

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

```

```
1424 ;////////////////////////////////////
1425 ;////////////////////////////////////
1426 ;THE FOLLOWING TEST ARE USED TO CHECK THE BASIC OPERATION OF THE SPACING FUNCTIONS
1427 ;////////////////////////////////////
1428 ;*****
1429 ;*TEST 20 TEST "BACK SPACE FILE GAP"
1430 ;*****
1431 TST20: SCOPE
1432 MOV #10,,STIMES ;DO 10. ITERATIONS
1433 MOV #38,$LPADR ;SET SCOPE LOOP ADDRESS
1434 MOV #TST21,$ESCAPE ;ESCAPE TO TEST 21 ON ERROR
1435 RESET
1436 MOV DRIVE,@TACS ;SELECT DRIVE
1437 MOV #REWIND!GO,@TACS ;POSITION TAPE ON CLEAR LEADER
1438 WAITREADY ;GO WAIT FOR "READY" TO SET
1439 MOV #WFG!GO,@TACS ;WRITE A FILE GAP OFF OF CLEAR LEADER
1440 WAITREADY ;GO WAIT FOR "READY" TO SET
1441 MOV #WRITE!GO,@TACS ;PUT A DATA BLOCK ON THE TAPE
1442 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1443 MOV #377,@TADB ;KNOCK DOWN XFER REQ
1444 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1445 BIS #ILBS,@TACS ;WRITE "CRC"
1446 WAITREADY ;GO WAIT FOR "READY" TO SET
1447 MOV #BSFG!GO,@TACS ;BACK UP OVER THE DATA
1448 WAITREADY ;GO WAIT FOR "READY" TO SET
1449 BIT #FGAP,@TACS ;CHECK FOR FILE GAP
1450 BNE 1$ ;BR IF "FILE GAP" =1
1451 ERROR 1 ;ERROR--NO FILE GAP INDICATION
1452 TST @TACS ;IS "ERROR" =1?
1453 BPL 2$ ;BR IF NO
1454 ERROR 1 ;"ERROR" =1
1455 BIT #LEADER,@TACS ;IS "CLEAR LEADER" =1?
1456 BEQ TST21 ;BR IF NO
1457 ERROR 1 ;"WFG" OR "BSFG" FAILED
1458 ;*****
1459 ;*TEST 21 TEST "SPACE FORWARD FILE GAP" FUNCTION
1460 ;*****
1461 TST21: SCOPE
1462 MOV #5,STIMES ;DO 5 ITERATIONS
1463 MOV #15,$LPADR ;SET SCOPE LOOP ADDRESS
1464 MOV #TST22,$ESCAPE ;ESCAPE TO TEST 22 ON ERROR
1465 RESET
1466 MOV DRIVE,@TACS ;SELECT A DRIVE
1467 MOV #REWIND!GO,@TACS ;POSITION THE TAPE
1468 WAITREADY ;GO WAIT FOR "READY" TO SET
1469 MOV #WFG!GO,@TACS ;WRITE FILE GAP OFF OF CLEAR LEADER
1470 WAITREADY ;GO WAIT FOR "READY" TO SET
1471 MOV #WRITE!GO,@TACS ;PUT SOME DATA ON THE TAPE
1472 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1473 MOV #377,@TADB ;KNOCK DOWN XFER REQ
1474 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1475 BIS #ILBS,@TACS ;WRITE CRC AND SHUT DOWN
1476 WAITREADY ;GO WAIT FOR "READY" TO SET
1477 MOV #WFG!GO,@TACS ;WRITE FILE GAP AFTER THE RECORD
1478 WAITREADY ;GO WAIT FOR "READY" TO SET
1479 MOV #REWIND!GO,@TACS ;REPOSITION THE TAPE
```

```
1480 WAITREADY ;GO WAIT FOR "READY" TO SET
1481 MOV #SFFG!GO,@TACS ;SPACE OVER THE RECORD
1482 WAITREADY ;GO WAIT FOR "READY" TO SET
1483 BIT #FGAP,@TACS ;SETTING IN A FILE GAP?
1484 BNE TST22 ;BR IF YES
1485 ERROR 1 ;NO FILE GAP INDICATION
1486 ;*****
1487 ;*TEST 22 TEST "BACK SPACE BLOCK GAP"
1488 ;*****
1489 TST22: SCOPE
1490 MOV #5,STIMES ;DO 5 ITERATIONS
1491 MOV #15,$LPADR ;SET SCOPE LOOP ADDRESS
1492 MOV #TST23,$ESCAPE ;ESCAPE TO TEST 23 ON ERROR
1493 RESET
1494 MOV DRIVE,@TACS ;SELECT DRIVE
1495 MOV #REWIND!GO,@TACS ;GO TO CLEAR LEADER
1496 WAITREADY ;GO WAIT FOR "READY" TO SET
1497 MOV #WFG!GO,@TACS ;WRITE A FILE GAP
1498 WAITREADY ;GO WAIT FOR "READY" TO SET
1499 MOV #WRITE!GO,@TACS ;WRITE A BLOCK OF DATA
1500 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1501 MOV #377,@TADB ;WRITE ONE BYTE
1502 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1503 BIS #ILBS,@TACS ;WRITE "CRC"
1504 WAITREADY ;GO WAIT FOR "READY" TO SET
1505 MOV #BSBG!GO,@TACS ;BACK OVER THE DATA
1506 WAITREADY ;GO WAIT FOR "READY" TO SET
1507 TST @TACS ;CHECK FOR ERRORS
1508 BPL 2$ ;BR IF NONE
1509 ERROR 1 ;ERROR OCCURRED
1510 BIT #LEADER,@TACS ;IS CLEAR LEADER=1?
1511 BEQ TST23 ;BR IF NO
1512 ERROR 1 ;ON CLEAR LEADER
1513 ;*****
1514 ;*TEST 23 TEST "SPACE FWD BLOCK GAP"
1515 ;*****
1516 TST23: SCOPE
1517 MOV #5,STIMES ;DO 5 ITERATIONS
1518 MOV #15,$LPADR ;SET SCOPE LOOP ADDRESS
1519 MOV #TST24,$ESCAPE ;ESCAPE TO TEST 24 ON ERROR
1520 RESET
1521 MOV DRIVE,@TACS ;GO TO CLEAR LEADER
1522 MOV #REWIND!GO,@TACS ;GO WAIT FOR "READY" TO SET
1523 WAITREADY ;GO WAIT FOR "READY" TO SET
1524 MOV #WFG!GO,@TACS ;PUT FILE GAP ON TAPE
1525 WAITREADY ;GO WAIT FOR "READY" TO SET
1526 MOV #WRITE!GO,@TACS ;WRITE ONE BYTE BLOCK
1527 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1528 MOV #377,@TADB ;GO WAIT ON "TRANSFER REQUEST" TO SET
1529 WAITXFER ;GO WAIT ON "TRANSFER REQUEST" TO SET
1530 BIS #ILBS,@TACS ;WRITE "CRC"
1531 WAITREADY ;GO WAIT FOR "READY" TO SET
1532 MOV #WFG!GO,@TACS ;WRITE A FILE GAP AFTER THE DATA
1533 WAITREADY ;GO WAIT FOR "READY" TO SET
1534 MOV #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
1535 WAITREADY ;GO WAIT FOR "READY" TO SET
```

```

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

```

```

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

```

```

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

```

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



```

1747 ;*****
1748 ;*TEST 31 TEST BACK-SPACE-BLOCK-GAP INTO CLEAR LEADER
1749 ;*****
1750 TST31: SCOPE
1751 MOV #10.,STIMES ;DO 10. ITERATIONS
1752 MOV #1$,SLPADR ;SET SCOPE LOOP ADDRESS
1753 MOV #TST32,SESCAPE ;ESCAPE TO TEST 32 ON ERROR
1754 RESET
1755 MOV DRIVE,@TACS ;SELECT DRIVE
1756 WAITREADY ;MAKE SURE READY IS SET
1757 MOV #REWIND!GO,@TACS ;GO TO BEGINNING OF TAPE
1758 WAITREADY ;WAIT ON "READY"
1759 MOV #WFG!GO,@TACS ;GET OFF OF CLEAR LEADER
1760 WAITREADY ;WAIT FOR READY
1761 MOV #WRITE!GO,@TACS ;START A WRITE
1762 WAITXFER ;WAIT ON "XFER REQ"
1763 MOV #377,@TADB ;WRITE ONE BYTE
1764 WAITXFER ;WAIT ON "XFER REQ"
1765 MOV #377,@TADB ;WRITE
1766 WAITXFER ;WAIT ON "XFER REQ"
1767 BIS #ILBS,@TACS ;WRITE "CRC"
1768 WAITREADY ;WAIT ON "READY"
1769 MOV #WFG!GO,@TACS ;WRITE A "FILE GAP"
1770 WAITREADY ;WAIT ON "READY"
1771 1s: MOV #REWIND!GO,@TACS ;BACK OVER THE DATA
1772 WAITREADY ;WAIT ON "READY"
1773 MOV #SFFG!GO,@TACS ;DO A SPACE FWD FILE GAP
1774 WAITREADY
1775 BIT #FGAP,@TACS ;IS FILE GAP=1?
1776 BNE 2$ ;BR IF YES
1777 ERROR 1 ;FILE GAP NOT=1
1778 2s: MOV #BSBG!GO,@TACS ;GO TO FIRST FILE GAP ON TAPE
1779 WAITREADY ;GO WAIT FOR "READY" TO SET
1780 BIT #ERROR!FGAP!LEADER,@TACS ;DID WE GET ANY ERROR?
1781 BEQ 3$ ;BR IF NO
1782 ERROR 1 ;AN ERROR OCCURRED
1783 3s: MOV #BSBG!GO,@TACS ;BSBG WHILE IN A FILE GAP
1784 WAITREADY ;GO WAIT FOR "READY" TO SET
1785 TST @TACS ;"ERROR" BIT SHOULD BE SET
1786 BMI 4$ ;BR IF IT IS
1787 ERROR 1 ;"ERROR" BIT WASN'T SET
1788 4s: BIT #LEADER,@TACS ;"CLEAR LEADER" SHOULD BE SET
1789 BNE TST32 ;"BR IF "CLEAR LEADER" = 1
1790 ERROR 1 ;"CLEAR LEADER" NOT SET
1791 ;//
1792 ;//
1793 ;//
1794 ;//
1795 ;//
1796 ;//
1797 ;//
1798 ;//
1799 ;//
1800 ;//
1801 ;//
1802 ;//
1803 ;//
1804 ;//
1805 ;//
1806 ;//
1807 ;//
1808 ;//
1809 ;//
1810 ;//
1811 ;//
1812 ;//
1813 ;//
1814 ;//
1815 ;//
1816 ;//
1817 ;//
1818 ;//
1819 ;//
1820 ;//
1821 ;//
1822 ;//
1823 ;//
1824 ;//
1825 ;//
1826 ;//
1827 ;//
1828 ;//
1829 ;//
1830 ;//
1831 ;//
1832 ;//
1833 ;//
1834 ;//
1835 ;//
1836 ;//
1837 ;//
1838 ;//
1839 ;//
1840 ;//
1841 ;//
1842 ;//
1843 ;//
1844 ;//
1845 ;//
1846 ;//
1847 ;//
1848 ;//
1849 ;//
1850 ;//
1851 ;//
1852 ;//
1853 ;//
1854 ;//
1855 ;//
1856 ;//
1857 ;//
1858 ;//
1859 ;//
1860 ;//
1861 ;//
1862 ;//
1863 ;//
1864 ;//
1865 ;//
1866 ;//
1867 ;//
1868 ;//
1869 ;//
1870 ;//
1871 ;//
1872 ;//
1873 ;//
1874 ;//
1875 ;//
1876 ;//
1877 ;//
1878 ;//
1879 ;//
1880 ;//
1881 ;//
1882 ;//
1883 ;//
1884 ;//
1885 ;//
1886 ;//
1887 ;//
1888 ;//
1889 ;//
1890 ;//
1891 ;//
1892 ;//
1893 ;//
1894 ;//
1895 ;//
1896 ;//
1897 ;//
1898 ;//
1899 ;//
1900 ;//
1901 ;//
1902 ;//
1903 ;//
1904 ;//
1905 ;//
1906 ;//
1907 ;//
1908 ;//
1909 ;//
1910 ;//
1911 ;//
1912 ;//
1913 ;//
1914 ;//
1915 ;//
1916 ;//
1917 ;//
1918 ;//
1919 ;//
1920 ;//
1921 ;//
1922 ;//
1923 ;//
1924 ;//
1925 ;//
1926 ;//
1927 ;//
1928 ;//
1929 ;//
1930 ;//
1931 ;//
1932 ;//
1933 ;//
1934 ;//
1935 ;//
1936 ;//
1937 ;//
1938 ;//
1939 ;//
1940 ;//
1941 ;//
1942 ;//
1943 ;//
1944 ;//
1945 ;//
1946 ;//
1947 ;//
1948 ;//
1949 ;//
1950 ;//
1951 ;//
1952 ;//
1953 ;//
1954 ;//
1955 ;//
1956 ;//
1957 ;//
1958 ;//
1959 ;//
1960 ;//
1961 ;//
1962 ;//
1963 ;//
1964 ;//
1965 ;//
1966 ;//
1967 ;//
1968 ;//
1969 ;//
1970 ;//
1971 ;//
1972 ;//
1973 ;//
1974 ;//
1975 ;//
1976 ;//
1977 ;//
1978 ;//
1979 ;//
1980 ;//
1981 ;//
1982 ;//
1983 ;//
1984 ;//
1985 ;//
1986 ;//
1987 ;//
1988 ;//
1989 ;//
1990 ;//
1991 ;//
1992 ;//
1993 ;//
1994 ;//
1995 ;//
1996 ;//
1997 ;//
1998 ;//
1999 ;//
2000 ;//
    
```

```

1803 MOV DRIVE,@TACS ;SELECT DRIVE
1804 MOV #REWIND!GO,@TACS ;GO TO "CLEAR LEADER"
1805 WAITREADY ;WAIT ON "READY"
1806 3s: MOV #WRITE!GO,@TACS ;WRITE DATA
1807 WAITXFER ;WAIT ON "XFER REQ"
1808 MOV #377,@TADB ;1ST BYTE
1809 WAITXFER ;WAIT ON "XFER REQ"
1810 MOV #0,@TADB ;2ND BYTE
1811 WAITXFER ;WAIT ON "XFER REQ"
1812 BIS #ILBS,@TACS ;WRITE "CRC"
1813 WAITREADY ;WAIT ON "READY"
1814 MOV #BSBG!GO,@TACS ;BACK OVER THE DATA
1815 WAITREADY ;WAIT ON "READY"
1816 MOV #READ!GO,@TACS ;START A "READ"
1817 WAITXFER ;WAIT ON "XFER REQ"
1818 MOV @TADB,R1 ;PUT THIS BYTE IN R1
1819 MOV #377,R0 ;PUT WHAT IT SHOULD BE IN R0
1820 CMP R1,R0 ;DID DATA READ GOOD?
1821 BEQ 1$ ;BR IF YES
1822 ERROR 5 ;DATA WASN'T = 377
1823 1s: WAITXFER ;WAIT ON "XFER"
1824 CLR R0
1825 MOV @TADB,R1 ;READ
1826 BEQ 2$ ;BR IF DATA = 000
1827 ERROR 5 ;DATA WASN'T = 000
1828 2s: BIS #ILBS,@TACS ;SHUT DOWN
1829 WAITREADY ;WAIT ON "READY"
1830 ;*****
1831 ;*TEST 33 TEST "WRITE & READ" A COUNT PATTERN
1832 ;*****
1833 TST33: SCOPE
1834 MOV #10.,STIMES ;DO 10. ITERATIONS
1835 MOV #13$,SLPADR ;SET SCOPE LOOP ADDRESS
1836 MOV #TST34,SESCAPE ;ESCAPE TO TEST 34 ON ERROR
1837 RESET ;CLEAR ALL
1838 MOV DRIVE,@TACS ;SELECT DRIVE
1839 MOV #REWIND!GO,@TACS ;GO TO "BOT"
1840 WAITREADY ;WAIT ON "READY" TO SET
1841 MOV #WFG!GO,@TACS ;GET ON OXIDE
1842 WAITREADY ;WAIT ON "READY" TO SET
1843 MOV #377,R0 ;FIRST DATA PATTERN AND COUNTER
1844 MOV #WRITE!GO,@TACS ;START A WRITE
1845 WAITXFER ;WAIT ON "XFER REQ" TO SET
1846 MOV R0,@TADB ;WRITE A BYTE ON TAPE
1847 WAITXFER ;WAIT ON "XFER REQ" TO SET
1848 MOV @TADB,R1 ;GET BACK THE LAST BYTE WRITTEN
1849 CMP R0,R1 ; AND CHECK IT
1850 BEQ 2$ ;BR IF IT LOOKS GOOD
1851 ERROR 5 ;THE DATA IN TADB WAS BAD
1852 2s: DEC R0 ;NEXT PATTERN
1853 BGE 1$ ;BR IF MORE TO DO
1854 BIS #ILBS,@TACS ;WRITE THE "CRC"
1855 WAITREADY ;WAIT ON "READY" TO SET
1856 TST @TACS ;ANY ERRORS OCCUR?
1857 BPL 3$ ;BR IF NO
1858 ERROR 1 ;ERROR OCCURRED DURING WRITE
    
```

```

1859 010442 112714 000011 3s: MOVB #BSBG:GO,@TACS ;GO TO BEGINNING OF BLOCK
1860 010446 104412 WAITREADY ;WAIT ON "READY" TO SET
1861 010450 112714 000005 MOVB #READ:GO,@TACS ;START A "READ"
1862 010454 012700 000377 MOV #377,R0 ;FIRST DATA PATTERN AND COUNTER
1863 010460 104413 4s: WAITXFER ;WAIT ON "XFER REQ" TO SET
1864 010462 011501 MOV @TADB,R1 ;READ A BYTE FROM TAPE
1865 010464 020401 CMP R0,R1 ;IS IT VALID?
1866 010466 001401 BQ 5s ;BR IF YES
1867 010470 104005 ERROR 5 ;BAD DATA READ FROM TAPE
1868 010472 005300 DEC R0 ;NEXT PATTERN
1869 010474 002371 BGE 4s ;BR IF MORE TO READ
1870 010476 104413 WAITXFER ;WAIT ON "XFER REQ" TO SET
1871 010500 052714 000020 BIS #ILBS,@TACS ;SHUT DOWN THE "READ" OPERATION
1872 010504 104412 WAITREADY ;WAIT ON "READY" TO SET
1873 010506 032714 140000 BIT #ERROR!CRCERR,@TACS ;ERROR AND CRCERR SHOULD BE = 0
1874 010512 001401 BEQ TST34 ;;BR IF THEY ARE
1875 010514 104001 ERROR 1 ;(ERROR 1 CRCERR) = 1
1876 ;
1877 ;
1878 ;
1879 ;THE FOLLOWING TESTS ARE USED TO INSURE THAT THE "CRC" CIRCUITRY FUNCTIONS PROPERLY
1880 ;
1881 ;*****
1882 ;*TEST 34 TEST "ERROR" WITH "CRCERR" = 1
1883 ;*****
1884 010516 000004 TST34: SCOPE
1885 010520 012767 000012 170440 MOV #10, $TIMES ;;DO 10. ITERATIONS
1886 010524 012767 010562 170352 MOV #20$, $LPADR ;;SET SCOPE LOOP ADDRESS
1887 010534 012767 010776 170426 MOV #TST35, $ESCAPE ;;ESCAPE TO TEST 35 ON ERROR
1888 010544 010314 RESET
1889 010546 112714 000017 MOVB DRIVE,@TACS ;SELECT DRIVE
1890 010552 104412 WAITREADY ;GO TO CLEAR LEADER
1891 010554 112714 000001 MOVB #WFG:GO,@TACS ;WAIT ON "READY"
1892 010560 104412 WAITREADY ;GET OFF OF CLEAR LEADER
1893 010562 112714 000003 MOVB #WRITE:GO,@TACS ;WAIT FOR READY
1894 010566 012700 000006 MOV #6,R0 ;START A WRITE
1895 010572 104413 1s: WAITXFER ;SETUP FOR 6 BYTES
1896 010574 005300 DEC R0 ;WAIT ON "XFER REQ"
1897 010576 002403 BLT 2s ;DOWN COUNT
1898 010600 112715 000377 MOVB #377,@TADB ;DONE?
1899 010604 000772 BR 1s ;NO--WRITE ON TAPE
1900 010606 052714 000020 BIS #ILBS,@TACS ;LOOP
1901 010612 104412 WAITREADY ;WRITE "CRC"
1902 010614 112714 000011 MOVB #BSBG:GO,@TACS ;WAIT ON "READY"
1903 010620 104412 WAITREADY ;BACK OVER THE DATA BLOCK
1904 010622 112714 000005 MOVB #READ:GO,@TACS ;WAIT ON "READY"
1905 010626 012700 000003 MOV #3,R0 ;START A "READ"
1906 010632 104413 3s: WAITXFER ;DO 3 BYTES
1907 010634 005300 DEC R0 ;WAIT FOR "XFER REQ"
1908 010636 002402 BLT 4s ;COUNT # OF BYTES
1909 010640 105715 TSTB @TADB ;CLEAR "XFER REQ"
1910 010642 000773 BR 3s
1911 010644 052714 4s: BIS #ILBS,@TACS ;DO "ILBS"
1912 010650 104412 WAITREADY ;WAIT ON "READY"
1913 010652 005714 TST @TACS ;CHECK FOR "ERROR"
1914 010654 104401 BMI 5s ;BR IF "ERROR"
    
```

```

1915 010656 104001 5s: ERROR 1 ;"ERROR" BIT NOT SET
1916 010660 032714 040000 BIT #CRCERR,@TACS ;CHECK FOR "CRC" ERROR
1917 010664 001001 RNE 6s ;BR IF "CRC" ERROR
1918 010666 104001 ERROR 1 ;NO "CRC" ERROR
1919 ;
1920 ;THE FOLLOWING CODE IS USED TO CHECK THE ROM
1921 010670 112714 000000 6s: MOVB #WFG,@TACS ;CHECK "ERROR" WITH "WFG"
1922 010674 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1923 010676 100001 BPL 7s ;BR IF "ERROR" = 0
1924 010680 104001 ERROR 1 ;"ERROR" NOT = 0
1925 010682 112714 000002 7s: MOVB #WRITE,@TACS ;CHECK "ERROR" WITH "WRITE"
1926 010686 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1927 010690 104001 BMI 8s ;BR IF "ERROR" = 1
1928 010694 112714 000006 8s: ERROR 1 ;"ERROR" NOT = 1
1929 010698 104001 MOVB #BSFG,@TACS ;CHECK "ERROR" WITH "BSFG"
1930 010702 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1931 010706 100001 BPL 9s ;BR IF "ERROR" = 0
1932 010710 104001 ERROR 1 ;"ERROR" NOT = 0
1933 010714 112714 000010 9s: MOVB #BSBG,@TACS ;CHECK "ERROR" WITH "BSBG"
1934 010718 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1935 010722 100001 BPL 10s ;BR IF "ERROR" = 0
1936 010726 104001 ERROR 1 ;"ERROR" NOT = 0
1937 010730 112714 000012 10s: MOVB #SFFG,@TACS ;CHECK "ERROR" WITH "SFFG"
1938 010734 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1939 010738 100001 BPL 11s ;BR IF "ERROR" = 0
1940 010742 104001 ERROR 1 ;"ERROR" NOT = 0
1941 010746 112714 000014 11s: MOVB #SFBG,@TACS ;CHECK "ERROR" WITH "SFBG"
1942 010750 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1943 010754 100001 BPL 12s ;BR IF "ERROR" = 0
1944 010758 104001 ERROR 1 ;"ERROR" NOT = 0
1945 010762 112714 000016 12s: MOVB #REWIND,@TACS ;CHECK "ERROR" WITH "REWIND"
1946 010766 005714 TST @TACS ;SAMPLE THE "ERROR" BIT
1947 010770 100001 BPL TST35 ;;BR IF "ERROR" = 0
1948 010774 104001 ERROR 1 ;"ERROR" NOT = 0
1949 ;
1950 ;*****
1951 ;*TEST 35 TEST "DATA OF 0 GIVES CRC OF 0"
1952 ;*****
1953 010776 000004 TST35: SCOPE
1954 010800 012767 000005 170160 MOV #5, $TIMES ;;DO 5 ITERATIONS
1955 010804 012767 011034 170072 MOV #6$, $LPADR ;;SET SCOPE LOOP ADDRESS
1956 010808 012767 011160 170146 MOV #TST36, $ESCAPE ;;ESCAPE TO TEST 36 ON ERROR
1957 010812 000005 RESET
1958 010816 010314 MOVB DRIVE,@TACS ;SELECT DRIVE
1959 010820 112714 000017 MOVB #REWIND,@TACS ;GO TO "CLEAR LEADER"
1960 010824 104412 WAITREADY ;WAIT ON "READY"
1961 010828 112714 000003 MOVB #WRITE:GO,@TACS ;WRITE THE DATA
1962 010832 104413 WAITXFER ;WAIT ON "XFER REQ"
1963 010836 105015 CLRB @TADB ;1ST BYTE = 0
1964 010840 104413 WAITXFER ;WAIT ON "XFER REQ"
1965 010844 105015 CLRB @TADB ;2ND BYTE = 0
1966 010848 104413 WAITXFER ;WAIT ON "XFER REQ"
1967 010852 052714 000020 BIS #ILBS,@TACS ;WRITE "CRC"
1968 010856 104412 WAITREADY ;WAIT ON "READY"
1969 010860 112714 000011 MOVB #BSBG:GO,@TACS ;BACK OVER THE DATA
1970 010864 104412 WAITREADY ;WAIT ON "READY"
1971 010868 112714 000005 MOVB #READ:GO,@TACS ;START A "READ"
1972 010872 104413 WAITXFER ;WAIT ON "XFER REQ" FIRST
    
```

```

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

```

```

2027 011304 012700 000377 MOV #377,R0 ;FIRST DATA PATTERN AND COUNTER
2028 011310 005037 013774 CLR @CRC.WD ;INITIALIZES THE "CRC WORD"
2029 011314 004737 013654 4$: JSR PC,@DO.CRC ;COMBINE THIS BYTE [R0] WITH THE "CRC WORD"
2030 011320 104413 WAITXFER ;WAIT ON "XFER REQ" TO SET
2031 011322 011501 MOV @TADB,R1 ;READ A BYTE FROM TAPE
2032 011324 020001 CMP R0,R1 ;IS IT VALID?
2033 011326 001401 BEQ 5$ ;BR IF YES
2034 011330 104005 EPROR 5 ;BAD DATA READ FROM TAPE
2035 011332 005300 DEC R0 ;NEXT PATTERN
2036 011334 002367 BGE 4$ ;BR IF MORE TO READ
2037 011336 005000 CLR R0 ;GET LOW BYTE OF "CRC WORD"
2038 011340 153700 013774 BISB @*CRC.WD,R0
2039 011344 104413 WAITXFER ;WAIT ON "XFER REQ" TO SET
2040 011346 052714 000020 BIS #ILBS,@TACS ;SHUT DOWN THE "READ" OPERATION
2041 011352 011501 MOV @TADB,R1 ;PICK FIRST BYTE OF TU60 "CRC"
2042 011354 020001 CMP R0,R1 ;IS IT GOOD?
2043 011356 001401 BEQ 6$ ;BR IF YES
2044 011360 104005 ERROR 5 ;FIRST BYTE OF TU60 "CRC" IS BAD
2045 011362 005000 CLR R0 ;GET HIGH BYTE OF "CRC WORD"
2046 011364 153700 013775 BISB @*CRC.WD+1,R0
2047 011370 104412 WAITREADY ;WAIT ON "READY" TO SET
2048 011372 011501 MOV @TADB,R1 ;READ THE 2ND BYTE OF TU60 "CRC"
2049 011374 020001 CMP R0,R1 ;IS IT GOOD?
2050 011376 001401 BEQ 7$ ;BR IF YES
2051 011400 104005 ERROR 5 ;HIGH BYTE OF "CRC" IS BAD
2052 011402 032714 140000 7$: BIT #ERROR!CRCERR,@TACS ;ERROR AND CRCERR SHOULD BE = 0
2053 011406 001401 BEQ TST37 ;;BR IF THEY ARE
2054 011410 104001 ERROR 1 ;(ERROR ! CRCERR) = 1
;*****
;THIS TEST REQUIRES BOTH DRIVES ,
;FOR THE FOLLOWING DESCRIPTION DRIVE "1" IS THE DRIVE UNDER TEST
;AND DRIVE "2" IS THE OTHER DRIVE
;
;TEST DESCRIPTION
2061 1) REWIND DRIVE 1
2062 2) WAIT FOR READY TO SET
2063 3) START DRIVE 2 REWINDING
2064 4) WAIT FOR READY TO CLEAR
2065 5) SELECT DRIVE 1 AND CHECK THAT READY IS STILL SET
2066 6) SELECT DRIVE 2 AND CHECK THAT READY IS STILL CLEAR
;
;*****
;*TEST 37 TRY TO HANG "READY" ON "REWIND"
;*****
TST37: SCOPE
2071 011412 000004 MOV #5, $TIMES ;;DO 5 ITERATIONS
2072 011414 012767 000005 167544 MOV #1$, $LPADR ;;SET SCOPE LOOP ADDRESS
2073 011422 012767 011436 167456 MOV #TST40, $ESCAPE ;;ESCAPE TO TEST 40 ON ERROR
2074 011430 012767 011552 167532 1$: TSTB @*DRVKEY+1 ;TESTING TWO DRIVES?
2075 011436 105737 001225 BEQ TST40 ;BR IF NO
2076 011442 001443 MOV #1,R1 ;SET TIMER
2077 011444 012701 000001 MOV DRIVE,R2 ;SET R2 TO THE OTHER DRIVE
2078 011450 010302 ADD #UNIT,R2
2079 011452 062702 000400 BIC #*CUNIT,R2
2080 011456 042702 173777 RESET ;CLEAR ALL
2081 011462 000005 MOV DRIVE,@TACS ;SELECT 1ST DRIVE
2082 011464 010314

```

```
2083 011466 104412 WAITREADY ;WAIT ON "READY"
2084 011470 010214 MOV R2,@TACS ;SELECT 2ND DRIVE
2085 011472 104412 WAITREADY ;WAIT ON "READY"
2086 011474 112714 000017 MOVB #REWIND!GO,@TACS ;SEND 2ND DRIVE TO BOT
2087 011500 006301 2s: ASL R1 ;GIVE "READY" TIME TO CLEAR
2088 011502 001001 BNE 3s
2089 011504 104001 ERROR 1 ;"READY" FAILED TO CLEAR
2090 ;NOTE: THIS ERROR OCCURRED ON THE
2091 ;2ND DRIVE, I.E. IF TESTING DRIVE A
2092 ;THEN DRIVE B FAILED
2093 011506 032714 000040 3s: BIT #READY,@TACS ;WAIT FOR "READY" TO CLEAR
2094 011512 001372 BNE 2s
2095 011514 010314 MOV DRIVE,@TACS ;SELECT 1ST DRIVE
2096 011516 112714 000016 MOVB #REWIND,@TACS ;LOAD A "REWIND"
2097 011522 032714 000040 BIT #READY,@TACS ;CHECK "READY"
2098 011526 001001 BNE 4s ;BR IF STILL SET
2099 011530 104001 ERROR 1 ;"READY" WAS CLEAR
2100 011532 010214 4s: MOV R2,@TACS ;SELECT 2ND DRIVE
2101 011534 112714 000016 MOVB #REWIND,@TACS ;LOAD A REWIND
2102 011540 032714 000040 BIT #READY,@TACS ;CHECK "READY"
2103 011544 001401 BEQ 5s ;BR IF STILL CLEAR
2104 011546 104001 ERROR 1 ;"READY" WAS SET
2105 011550 104412 5s: WAITREADY ;WAIT ON "READY"
2106 ;*****
2107 ;THIS TEST REQUIRES BOTH DRIVES .
2108 ;FOR THE FOLLOWING DESCRIPTION DRIVE "1" IS THE DRIVE UNDER TEST
2109 ;AND DRIVE "2" IS THE OTHER DRIVE
2110 ;
2111 ;TEST DESCRIPTION
2112 ;1) REWIND DRIVE 1
2113 ;2) WFG ON DRIVE 1
2114 ;3) START A REWIND ON DRIVE 2
2115 ;4) WHILE DRIVE 2 IS REWINDING WRITE DATA ON DRIVE 1
2116 ;5) CHECK FOR ERRORS
2117 ;6) START A REWIND ON DRIVE 2
2118 ;7) WHILE DRIVE 2 IS REWINDING READ DATA FROM DRIVE 1
2119 ;8) CHECK FOR ERRORS
2120 ;
2121 ;*****
2122 ;*TEST 40 TRY TO GLITCH THE "POWER SUPPLY"
2123 ;*****
2124 TST40: SCOPE
2125 011554 012767 000005 167404 MOV #5,STIMES ;DO 5 ITERATIONS
2126 011562 012767 011576 167316 MOV #1s,SLPADR ;SET SCOPE LOOP ADDRESS
2127 011570 012767 012134 167372 MOV #TST41,$ESCAPE ;ESCAPE TO TEST 41 ON ERROR
2128 011576 105737 001225 1s: TSTB @DRVKEY+1 ;TESTING TWO DRIVES?
2129 011602 001554 BEQ TST41 ;BR IF NO
2130 011604 012701 000001 MOV #1,R1 ;SET TIMER
2131 011610 010302 MOV DRIVE,R2 ;SET R2 TO THE OTHER DRIVE
2132 011612 062702 000400 ADD #UNIT,R2
2133 011616 042702 177377 BIC #*CUNIT,R2
2134 011622 000005 RESET ;CLEAR ALL
2135 011624 010314 MOV DRIVE,@TACS ;SELECT 1ST DRIVE
2136 011626 112714 000017 MOVB #REWIND!GO,@TACS ;REWIND TO BOT
2137 011632 104412 WAITREADY ;WAIT ON "READY"
2138 011634 112714 000001 MOVB #WFG!GO,@TACS ;GET ON OXIDE
```

```
2139 011640 104412 WAITREADY ;WAIT ON "READY"
2140 011642 010214 MOV R2,@TACS ;SELECT 2ND DRIVE
2141 011644 104412 WAITREADY ;WAIT ON "READY"
2142 011646 112714 000017 MOVB #REWIND!GO,@TACS ;START A "REWIND"
2143 011652 006301 2s: ASL R1 ;GIVE "READY" TIME TO CLEAR
2144 011654 001001 BNE 3s
2145 011656 104001 ERROR 1 ;"READY" FAILED TO CLEAR
2146 ;NOTE: THIS ERROR OCCURRED ON THE
2147 ;2ND DRIVE, I.E. IF TESTING DRIVE A
2148 ;THEN DRIVE B FAILED
2149 011660 032714 000040 3s: BIT #READY,@TACS ;WAIT FOR "READY" TO CLEAR
2150 011664 001372 BNE 2s
2151 011666 010314 MOV DRIVE,@TACS ;SELECT 1ST DRIVE
2152 011670 112714 000003 MOVB #WRITE!GO,@TACS ;WRITE WHILE THE OTHER DRIVE IS "REWINDING"
2153 011674 104413 WAITXFER ;WAIT ON "XFER REQ"
2154 011676 112715 000377 MOVB #377,@TADB ;WRITE 1ST BYTE
2155 011702 104413 WAITXFER ;WAIT ON "XFER REQ"
2156 011704 105015 CLR B @TADB ;WRITE 2ND BYTE
2157 011706 104413 WAITXFER ;WAIT ON "XFER REQ"
2158 011710 052714 000020 BIS #ILBS,@TACS ;WRITE "CRC" & SHUT DOWN
2159 011714 104412 WAITREADY ;WAIT ON "READY"
2160 011716 005714 TST @TACS ;ANY ERROR?
2161 011720 100001 BPL 4s ;BR IF NO
2162 011722 104001 ERROR 1 ;ERROR OCCURRED DURING "WRITE"
2163 011724 112714 000011 4s: MOVB #BSBG!GO,@TACS ;POSITION TAPE FOR "READ"
2164 011730 104412 WAITREADY ;WAIT ON "READY"
2165 011732 005714 TST @TACS ;ANY ERROR?
2166 011734 100001 BPL 5s ;BR IF NO
2167 011736 104001 ERROR 1 ;ERROR OCCURRED DURING "BSBG"
2168 011740 112714 000005 5s: MOVB #READ!GO,@TACS ;START A "READ"
2169 011744 104413 WAITXFER ;WAIT ON "XFER REQ"
2170 011746 012700 MOV #377,R0 ;FIRST DATA PATTERN
2171 011752 011501 MOV @TADB,R1 ;PICKUP FIST BYTE
2172 011754 020001 CMP R0,R1 ;IS IT GOOD?
2173 011756 001401 BEQ 6s ;BR IF NO
2174 011760 104005 ERROR 5 ;1ST DATA BYTE IS BAD
2175 011762 104413 WAITXFER ;WAIT ON "XFER REQ"
2176 011764 005000 CLR R0 ;2ND DATA PATTERN
2177 011766 011501 MOV @TADB,R1 ;PICKUP 2ND BYTE
2178 011770 001401 BEQ 7s ;BR IF IT IS GOOD
2179 011772 104005 ERROR 5 ;2ND BYTE IS BAD
2180 011774 104413 WAITXFER ;WAIT ON "XFER REQ"
2181 011776 052714 000020 BIS #ILBS,@TACS ;SHUT DOWN
2182 012002 104412 WAITREADY ;WAIT ON "READY"
2183 012004 005714 TST @TACS ;ERROR?
2184 012006 100001 BPL 8s ;BR IF NO
2185 012010 104001 ERROR 1 ;ERROR OCCURRED
2186 012012 112714 000011 8s: MOVB #BSBG!GO,@TACS ;POSITION FOR NEXT "READ"
2187 012016 104412 WAITREADY ;WAIT ON "READY"
2188 012020 005714 TST @TACS ;ERROR?
2189 012022 100001 BPL 9s ;BR IF NO
2190 012024 104001 ERROR 1 ;ERROR OCCURRED DURING "BSBG"
2191 012026 010214 MOV R2,@TACS ;SELECT 2ND DRIVE
2192 012030 104412 WAITREADY ;WAIT ON "READY"
2193 012032 112714 000017 MOVB #REWIND!GO,@TACS ;START A "REWIND"
2194 012036 012701 000001 MOV #1,R1 ;SET THE TIMER
```

```

2195 012042 005201          10s:  INC  R1          ;GIVE "READY" TIME TO CLEAR
2196 012044 001001          BNE  11s          ;"READY" FAILED TO CLEAR
2197 012046 104001          ERROR 1          ;NOTE: THIS ERROR OCCURRED ON THE
2198                                     ;2ND DRIVE. I.E. IF TESTING DRIVE A
2199                                     ;THEN DRIVE B FAILED
2200                                     ;WAIT FOR "READY" TO CLEAR
2201 012050 032714 000040    11s:  BIT   #READY,@TACS
2202 012054 001372          BNE  10s          ;SELECT 1ST DRIVE
2203 012056 010314          MOV   DRIVE,@TACS ;WAIT ON "READY"
2204 012060 104412          WAITREADY        ;"READ" WHILE OTHER DRIVE IS "REWINDING"
2205 012062 112714 000005    MOVB  #READ!GO,@TACS ;WAIT ON "XFER REQ"
2206 012066 104413          WAITXFER        ;1ST DATA PATTERN
2207 012070 012700 000377    MOV   #377,R0     ;READ 1ST BYTE
2208 012074 011501          MOV   @TADB,R1    ;IS IT GOOD?
2209 012076 020001          CMP  R0,R1        ;BR IF NO
2210 012100 001401          BEQ  12s          ;1ST BYTE FAILED
2211 012102 104005          ERROR 5          ;WAIT ON "XFER REQ"
2212 012104 104413          WAITXFER        ;2ND DATA PATTERN
2213 012106 005000          CLR  R0           ;READ 2ND BYTE
2214 012110 011501          MOV   @TADB,R1    ;BR IF IT IS GOOD
2215 012112 001401          BEQ  13s          ;2ND BYTE FAILED
2216 012114 104005          ERROR 5          ;WAIT ON "XFER REQ"
2217 012116 104413          WAITXFER        ;SHUT DOWN
2218 012120 052714 000020    BIS   #LBS,@TACS ;WAIT ON "READY"
2219 012124 104412          WAITREADY        ;ERROR?
2220 012126 005714          TST  @TACS        ;BR IF NO
2221 012130 100001          BPL  TST41        ;ERROR OCCURRED
2222 012132 104001          ERROR 1          ;*****
2223                                     ;*TEST 41      END OF TEST CODE
2224                                     ;*****
2225                                     ;*****
2226 012134 000004          IST41: SCOPE
2227 012136 012767 000001 167022  MOV   #1,$TIMES   ;;DO 1 ITERATION
2228 012144 000005          RESET
2229 012146 010314          MOV   DRIVE,@TACS ;SELECT DRIVE
2230 012150 112714 000017    MOVB  #REWIND!GO,@TACS
2231 012154 104412          WAITREADY
2232                                     .SBTTL  END OF PASS ROUTINE
2233
2234                                     ;*****
2235                                     ;*INCREMENT THE PASS NUMBER ($PASS)
2236                                     ;*TYPE "END PASS"
2237                                     ;*IF THERES A MONITOR GO TO IT
2238                                     ;*IF THERE ISN'T JUMP TO START
2239                                     ;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
2240                                     ;*SENDMG CAN BE CHANGED TO 7.
2241
2242 012156                                     SEOP:
2243 012156 000004          SCOPE
2244 012160 005067 166716    CLR   $STNM       ;;ZERO THE TEST NUMBER
2245 012164 005067 166776    CLR   $TIMES      ;;ZERO THE NUMBER OF ITERATIONS
2246 012170 005267 166704    INC   $PASS       ;;INCREMENT THE PASS NUMBER
2247 012174 042767 100000 166676    BIC   #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
2248 012202 005327          DEC   (PC)+       ;;LOOP?
2249 012204 000001          SEOPCT: .WORD 1
2250 012206 003015          BGT   $DOAGN     ;;YES
    
```

```

2251 012210 012737          MOV   (PC)+,@(PC)+ ;RESTORE COUNTER
2252 012212 000001          SENDCT: .WORD 1
2253 012214 012204          $EOPCT
2254 012216 104401 012251    TYPE   ,SENDMG    ;;TYPE "END PASS"
2255 012222 013700 000042    SGET42: MOV   @#42,R0 ;;GET MONITOR ADDRESS
2256 012226 001405          BEQ   $DOAGN      ;;BRANCH IF NO MONITOR
2257 012230 000005          RESET           ;;CLEAR THE WORLD
2258 012232 004710          SENDAD: JSR   PC,(R0) ;;GO TO MONITOR
2259 012234 000240          NOP             ;;SAVE ROOM
2260 012236 000240          NOP             ;;FOR
2261 012240 000240          NOP             ;;ACT11
2262 012242
2263 012242 000137          $DOAGN: JMP   @ (PC)+     ;;RETURN
2264 012244 002220          SRINAD: .WORD START
2265 012246 377 377 000    SENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
2266 012251 015 042412 042116 SENDMG: .ASCIZ <15><12>/END PASS/
2267 012256 050040 051501 000123
    
```

```

2268                ,SBTTL SCOPE HANDLER ROUTINE
2269                ;*****
2270                ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS, IT WILL INCREMENT
2271                ;AND LOAD THE TEST NUMBER(STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2272                ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2273                ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2274                ;*SW14=1 LOOP ON TEST
2275                ;*SW11=1 INHIBIT ITERATIONS
2276                ;*SW09=1 LOOP ON ERROR
2277                ;*SW08=1 LOOP ON TEST IN SWR<7:0>
2278                ;*CALL SCOPE ;SCOPE=IOT
2279                ;*
2280
2281                $SCOPE:
2282                CKSWR                ;;TEST FOR CHANGE IN SOFT-SWR
2283                BIT #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
2284                BNE $OVER            ;;YES IF SW14=1
2285                012274 001111
2286                ;*****START OF CODE FOR THE XOR TESTER*****
2287                SXTSTR: BR 6S        ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2288                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
2289                MOV @ERRVEC,=(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2290                MOV #5S,@ERRVEC      ;;SET FOR TIMEOUT
2291                TST @#177060         ;;TIME OUT ON XOR?
2292                MOV (SP)+,@ERRVEC    ;;RESTORE THE ERROR VECTOR
2293                BR $SVLAD            ;;GO TO THE NEXT TEST
2294                CMP (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
2295                MOV (SP)+,@ERRVEC    ;;RESTORE THE ERROR VECTOR
2296                BR 7S                ;;LOOP ON THE PRESENT TEST
2297                6S:;****END OF CODE FOR THE XOR TESTER*****
2298                BIT #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
2299                BEQ 2S                ;;BR IF NO
2300                CMPB @SWR,$STSTNM   ;;ON THE RIGHT TEST? SWR<7:0>
2301                BEQ $OVER            ;;BR IF YES
2302                TSTB $ERFLG         ;;HAS AN ERROR OCCURRED?
2303                BEQ 3S                ;;BR IF NO
2304                CMPB $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2305                BEQ 3S                ;;BR IF NO
2306                BIT #BIT09,@SWR     ;;LOOP ON ERROR?
2307                BEQ 4S                ;;BR IF NO
2308                MOV $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
2309                BR $OVER
2310                CLRFB $ERFLG        ;;ZERO THE ERROR FLAG
2311                CLR $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2312                BR 1S                ;;ESCAPE TO THE NEXT TEST
2313                BIT #BIT11,@SWR     ;;INHIBIT ITERATIONS?
2314                BNE 1S                ;;BR IF YES
2315                TST $PASS           ;;IF FIRST PASS OF PROGRAM
2316                BEQ 1S                ;; INHIBIT ITERATIONS
2317                INC $ICNT           ;;INCREMENT ITERATION COUNT
2318                CMP $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
2319                BEQ $OVER            ;;BR IF MORE ITERATION REQUIRED
2320                MOV #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
2321                MOV $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
2322                INCB $STSTNM        ;;COUNT TEST NUMBERS
2323                MOV (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS

```

```

2324                MOV $LPERR        ;;SAVE ERROR LOOP ADDRESS
2325                CLR $ESCAPE        ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
2326                MOVB #1,$ERMAX     ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2327                MOV $STSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
2328                MOV $LPADR,(SP)    ;;FUDGE RETURN ADDRESS
2329                RTI                ;;FIXES PS
2330                $MXCNT: 2000.      ;;MAX. NUMBER OF ITERATIONS

```

```

2331          ,SBTTL  ERROR HANDLER ROUTINE
2332
2333          ;*****
2334          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2335          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2336          ;*AND GO TO TYPERR ON ERROR
2337          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2338          ;*SW15=1      HALT ON ERROR
2339          ;*SW13=1      INHIBIT ERROR TYPEOUTS
2340          ;*SW10=1      BELL ON ERROR
2341          ;*SW09=1      LOOP ON ERROR
2342          ;*CALL
2343          ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2344
2345          SERRUR:
2346          012536      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
2347          012536      104406      MOV      @TACS,@#SREG0      ;;SAVE THE STATUS REG.
2348          012544      011437      001162      MOV      @TADB,@#SREG1      ;;SAVE THE DATA BUFFER
2349          012550      010037      001124      MOV      R0,@#SDDAT      ;;R0 WILL CONTAIN THE GOOD DATA
2350          012554      010137      001126      MOV      R1,@#SDDAT      ;;R1 WILL CONTAIN THE BAD DATA
2351          012560      105267      166317      7S:      INCB      $ERFLG      ;;SET THE ERROR FLAG
2352          012564      001775      BEQ      7S      ;;DON'T LET THE FLAG GO TO ZERO
2353          012566      016777      166310      166346      MOV      $TSTNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
2354          012574      032777      002000      166336      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
2355          012602      001402      BEQ      1S      ;;NO - SKIP
2356          012604      104401      001172      TYPE     $SBELL      ;;RING BELL
2357          012610      005267      166276      1S:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
2358          012614      011667      166276      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
2359          012620      162767      000002      166270      SUB      #2,$ERRPC
2360          012626      117767      166264      166260      MOV      @ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
2361          012634      032777      020000      166276      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
2362          012642      001004      BNE      20S      ;;SKIP TYPEOUTS
2363          012644      004767      000060      JSR      PC,TYPEPR      ;;GO TO USER ERROR ROUTINE
2364          012650      104401      001177      TYPE     $CRLF
2365          012654
2366          012654      005777      166260      20S:      TST      @SWR      ;;HALT ON ERROR
2367          012660      100002      BPL      3S      ;;SKIP IF CONTINUE
2368          012662      000000      HALT
2369          012664      104406      CKSWR
2370          012666      032777      001000      166244      3S:      BIT      #BIT09,@SWR      ;;TEST FOR CHANGE IN SOFT-SWR
2371          012674      001402      BEQ      4S      ;;LOOP ON ERROR SWITCH SET?
2372          012676      016716      166206      MOV      $LPERR,(SP)      ;;BR IF NO
2373          012702      005767      166262      4S:      TST      $ESCAPE      ;;FUDDGE RETURN FOR LOOPING
2374          012706      001402      BEQ      5S      ;;CHECK FOR AN ESCAPE ADDRESS
2375          012710      016716      166254      MOV      $ESCAPE,(SP)      ;;BR IF NONE
2376          012714
2377          012714      022737      012232      000042      5S:      CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
2378          012722      001001      BNE      6S      ;;BRANCH IF NO
2379          012724      000000      HALT      ;;YES
2380          012726
2381          012726      000002      RTI
2382

```

```

2383          ;*****
2384          ;THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2385
2386          012730      104401      001177      TYPERR:  TYPE     $CRLF      ;;TYPE A CARRIAGE RETURN & LINE FEED
2387          012734      010046      MOV      R0,-(SP)      ;;SAVE R0
2388          012736      113700      001114      MOV      @#$ITEMB,R0      ;;PICKUP THE ITEM INDEX
2389          012742      005300      DEC      R0      ;;ADJUST THE INDEX
2390          012744      006300      ASL      R0      ;;SO IT WILL WORK FOR
2391          012746      006300      ASL      R0      ;;THE ERROR TABLE
2392          012750      006300      ASL      R0
2393          012752      062700      001236      ADD      #$ERRTB,R0      ;;FORM THE TABLE POINTER
2394          012756      012067      000002      MOV      (R0)+,1S      ;;PICKUP "ERROR MESSAGE" POINTER
2395          012762      104401      TYPE     TYPE      ;;TYPE "ERROR MESSAGE"
2396          012764      000000      1S:      0      ;;"ERROR MESSAGE POINTER" GOES HERE
2397          012766      104401      001177      TYPE     $CRLF
2398          012772      012067      000004      MOV      (R0)+,2S      ;;PICKUP "DATA HEADER" POINTER
2399          012776      001404      BEQ      3S      ;;IF "0" DON'T TYPE
2400          013000      104401      TYPE     TYPE      ;;TYPE "DATA HEADER"
2401          013002      000000      2S:      0      ;;"DATA HEADER" POINTER GOES HERE
2402          013004      104401      TYPE     $CRLF
2403          013010      012000      3S:      MOV      (R0)+,R0      ;;PICKUP "DATA POINTER"
2404          013012      001004      BNE      5S      ;;IF THERE IS DATA TO TYPE GO DO IT
2405          013014      012600      4S:      MOV      (SP)+,R0      ;;RESTORE R0
2406          013016      104401      TYPE     $CRLF      ;;TYPE A CARRIAGE RETURN&LINE FEED
2407          013022      000207      RTS      PC      ;;RETURN
2408          013024
2409          013024      013046      5S:      MOV      @-(R0)+,-(SP)      ;;SAVE @-(R0)+ FOR TYPEOUT
2410          013026
2411          013026      104402      TYPCC      ;;TYPE DATA
2412          013030      005710      TST      (R0)      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2413          013032      001770      BEQ      4S      ;;TERMINATOR?
2414          013034      104401      TYPE     $S      ;;BR IF YES
2415          013040      000771      BR      5S      ;;TYPE 2 SPACES
2416          013042      020040      6S:      .ASCIZ  / /      ;;LOOP
2417          013046      013046      .EVEN      ;;ASCII STRING OF 2 SPACES

```

```

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

```

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



```

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

```

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

```

2615 ;////////////////////////////////////
2616 ;////////////////////////////////////
2617 ;THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TUGO
2618 ;NOTE: *** BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 ***
2619 ;////////////////////////////////////
2620
2621
2622
2623 ;*****J*****
2624 ; WRITE FILE GAPS FROM "BOT" TO "EOT"
2625 ;START AT 220
2626 ;THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO" AND
2627 ;THE "WRITE DELAY MONO".
2628
2629 WFGSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2630 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
2631 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
2632 RESET ;RESET THE WORLD
2633 MOV #WFGSUB,@$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2634 JSR PC,@$NXTDRV ;GO SETUP FOR NEXT DRIVE
2635 MOV DRIVE,@TACS ;SELECT DRIVE
2636 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2637 100$: BIT #READY,@TACS ;WAIT ON READY
2638 BEQ 100$
2639 1$: MOVB #WFG!GO,@TACS ;WRITE A FILE GAP
2640 WAITREADY ;WAIT ON READY
2641 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2642 BEQ 1$ ;BR IF NO
2643 HALT ;STOP IF YES
2644 BR WFGSUB ;LOOP ON CONT.
2645
2646
2647 ;*****
2648 ; WRITE CONTINUOUS BLOCKS OF DATA
2649 ;START AT 224
2650 ;THE PROGRAM WILL HALT THREE(3) TIMES
2651 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2652 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2653 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2654 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2655 ;THIS ROUTINE CAN BE USED TO ADJUST THE "GAP TIME MONO"
2656 ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
2657 ; EACH HALT OPERATOR WILL BE PROMPTED
2658 ; FOR THE VALUE WITH "SWR=XXXXX NEW="
2659
2660 WRTSUB: JSR PC,#$SETBUF ;GET BLOCK SIZE AND PATTERN
2661 WLOOP: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2662 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
2663 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
2664 RESET ;RESET THE WORLD
2665 MOV #WLOOP,@$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2666 JSR PC,@$NXTDRV ;GO SETUP FOR NEXT DRIVE
2667 MOV DRIVE,@TACS ;SELECT DRIVE
2668 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2669 100$: BIT #READY,@TACS ;WAIT ON READY
2670 BEQ 100$
    
```

```

2671 1$: JSR PC,@$WRIBLK ;WRITE A BLOCK
2672 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2673 BEQ 1$ ;BR IF NO
2674 HALT ;STOP IF "EOT"
2675 BR WLOOP ;LOOP IF CONT.
2676
2677
2678 ;*****
2679 ; READ CONTINUOUS BLOCKS OF DATA
2680 ;START AT 230
2681 ;THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
2682 ;AND THE "THRESHOLD POT".
2683
2684 RDSUB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2685 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
2686 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
2687 RESET ;RESET THE WORLD
2688 MOV #RDSUB,@$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2689 JSR PC,@$NXTDRV ;GO SETUP FOR NEXT DRIVE
2690 MOV DRIVE,@TACS ;SELECT DRIVE
2691 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2692 100$: BIT #READY,@TACS ;WAIT ON READY
2693 BEQ 100$
2694 1$: JSR PC,@$RDBLK ;READ A BLOCK
2695 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2696 BNE RDSUB ;BR IF YES---LOOP
2697 BR 1$
2698
2699
2700 ;*****
2701 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2702 ;START AT 234
2703 ;THE PROGRAM WILL HALT THREE(3) TIMES
2704 ;AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2705 ;HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2706 ;HALT 2 --- SWR<7:0> = PATTERN DESIRED
2707 ;HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2708 ;** IF USING SOFTWARE SWITCH REGISTER, AFTER
2709 ; EACH HALT OPERATOR WILL BE PROMPTED
2710 ; FOR THE VALUE WITH "SWR=XXXXX NEW="
2711 ; AND THE "GAP TIME MONO".
2712 ;THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
2713
2714 WGPBLK: JSR PC,#$SETBUF ;GET BLOCK SIZE AND PATTERN
2715 WGBLOP: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2716 MOV @TACSL,TACS ;SETUP THE TA11 STATUS AND
2717 MOV @TADBL,TADB ;DATA BUFFER REGISTERS
2718 RESET ;RESET THE WORLD
2719 MOV #WGBLOP,@$SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2720 JSR PC,@$NXTDRV ;GO SETUP FOR NEXT DRIVE
2721 MOV DRIVE,@TACS ;SELECT DRIVE
2722 MOVB #REWIND!GO,@TACS ;SEND TAPE TO "BOT"
2723 100$: BIT #READY,@TACS ;WAIT ON READY
2724 BEQ 100$
2725 1$: MOVB #WFG!GO,@TACS ;WRITE A FILE GAP
2726 WAITREADY ;WAIT ON READY
    
```

```

2727 014306 03:714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2728 014312 00:005 BNE 2S ;BR IF YES
2729 014314 00:737 014660 JSR PC,@#WRTBLK ;WRITE A BLOCK
2730 014320 03:714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2731 014324 00:765 BEQ 1S ;BR IF NO
2732 014326 00:000 2S: HALT ;STOP AT "EOT"
2733 014330 00:741 BR WGBLOP ;START OVER ON CONT.
2734
2735
2736 ;*****
2737 ; READ A BLOCK OF DATA AND A FILE GAP
2738 ;START AT 240
2739 ;THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
2740 ;IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
2741 ;AND THE "TAPE BLANK MONO".
2742 ;*****
2743 014332 012706 001100 RGPBLK: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2744 014336 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2745 014342 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2746 014346 000005 RESET ;RESET THE WORLD
2747 014350 012737 014332 001110 MOV #RGPBLK,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2748 014356 004737 015002 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2749 014362 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2750 014364 112714 000017 MOVB #REWIND:GO,@TACS ;SEND TAPE TO "BOT"
2751 014370 032714 000040 100S: BIT #READY,@TACS ;WAIT ON READY
2752 014374 001775 BEQ 100S
2753 014376 004737 014722 1S: JSR PC,@#RDBLK ;READ A BLOCK OF DATA
2754 014402 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2755 014406 001351 BNE RGPBLK ;BR IF YES
2756 014410 112714 000015 MOVB #SFBG:GO,@TACS ;GET INTO A FILE GAP
2757 014414 104412 WAITREADY
2758 014416 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2759 014422 001343 BNE RGPBLK ;BR IF YES
2760 014424 000764 BR 1S ;LOOP
2761
2762 ;*****
2763 ; SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"
2764 ;START AT 244
2765 ;THIS ROUTINE CAN BE USED AFTER "WRITE FILE GAP" FOR LOW SPEED
2766 ;SPACE FOWARD (TAPE BLANK MONO CAN BE ADJUSTED), OR AFTER READ OR
2767 ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
2768 ;(SIGNAL MONO CAN BE CHECKED).
2769 ;*****
2770 014426 012706 001100 SFFGSB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2771 014432 013704 001206 MOV #TACSL,TACS ;SETUP THE TA11 STATUS AND
2772 014436 013705 001212 MOV #TADBL,TADB ;DATA BUFFER REGISTERS
2773 014442 000005 RESET ;RESET THE WORLD
2774 014444 012737 014426 001110 MOV #SFFGSB,@#SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
2775 014452 004737 015002 JSR PC,@#NXTDRV ;GO SETUP FOR NEXT DRIVE
2776 014456 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2777 014460 112714 000017 MOVB #REWIND:GO,@TACS ;SEND TAPE TO "BOT"
2778 014464 032714 000040 100S: BIT #READY,@TACS ;WAIT ON READY
2779 014470 001775 BEQ 100S
2780 014472 112714 000013 1S: MOVB #SFFG:GO,@TACS ;SPACE INTO A FILE GAP
2781 014476 104412 WAITREADY ;WAIT ON READY
2782

```

```

2783 014500 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2784 014504 001772 BEQ 1S ;BR IF NO
2785 014506 000000 HALT ;STOP AT "EOT"
2786 014510 000764 BR SFFGSB ;LOOP ON CONT.
2787
2788 ;*****
2789 ; BACK SPACE FILE GAP
2790 ;START AT 250
2791 ;THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
2792 ;*****
2793 014512 000005 BSFGSB: RESET ;RESET THE WORLD
2794 014514 012737 014512 001110 MOV #BSFGSB,@#SLPERR ;LOOP ON ERROR ADDRESS
2795 014522 010314 MOV DRIVE,@TACS ;SELECT DRIVE
2796 014524 112714 000007 1S: MOVB #BSFG:GO,@TACS ;BACK SPACE A FILE GAP
2797 014530 104412 WAITREADY ;WAIT ON READY
2798 014532 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2799 014536 001772 BEQ 1S ;BR IF NO
2800 014540 000000 HALT ;STOP AT BOT
2801 014542 000763 BR BSFGSB ;START OVER ON CONT.
2802
2803 ;*****
2804 ; SETUP BLOCK SIZE AND PATTERN FOR SUBROUTINES
2805 ;*****
2806 SETBUF: CLR R0
2807 014544 005000 2808: HALT ;OPERATOR PUTS BYTE COUNT IN SWR<7:0>
2809 014546 000000 CMP #SWREG,SWR ;USING S/W SWITCH REG.?
2810 014550 022767 000176 164362 BNE 20S ;NO- GET OUT
2811 014556 001001 GTSWR ;LET HIM CHANGE IT
2812 014560 104405 20S: ;CONTINUE
2813 014562 BISB @SWR,R0 ;PICKUP THE BYTE COUNT
2814 014566 157700 164352 BNE 2S ;BR IF NON-ZERO
2815 014570 105777 164345 TSTB @SWR+1 ;CHECK IF GREATER THAN 377
2816 014574 001402 BEQ 1S ;BR IF NO
2817 014576 012700 000376 MOV #376,R0 ;SET FOR MAX ALLOWED
2818 014602 005200 1S: INC R0 ;MAKE IT 377 OR 1
2819 014604 010037 014654 2S: MOV R0,@#BLKLM ;SETUP THE BLOCK LIMIT
2820 014610 005037 014656 CLR @PATTRN
2821 014614 000000 HALT ;OPERATOR PUTS PATTERN IN SWR<7:0>
2822 014616 022767 000176 164314 CMP #SWREG,SWR ;USING S/W SWITCH REG.?
2823 014624 001001 BNE 21S ;NO- GET OUT
2824 014626 104405 21S: GTSWR ;LET HIM CHANGE IT
2825 014630 ;CONTINUE
2826 014634 117737 164304 014656 MOVB @SWR,@#PATTRN ;PICK UP THE PATTERN
2827 014636 000000 HALT ;SET OPERATIONAL SWITCHES
2828 014640 022767 000176 164272 CMP #SWREG,SWR ;USING S/W SWITCH REG.?
2829 014644 001001 BNE 22S ;NO- GET OUT
2830 014650 104405 22S: GTSWR ;LET HIM CHANGE IT
2831 014652 ;CONTINUE
2832 014654 000207 RTS PC ;RETURN
2833 014656 000000 BLKLM: 0 ;READ AND WRITE BLOCK SIZE
2834 014658 000000 PATRN: 0 ;PATTERN TO GO ON THE TAPE
2835
2836 ;*****
2837
2838

```

```

2839 ; WRITE ROUTINE FOR THE MANUAL OPERATIONS
2840 ;*****
2841 014660 013701 014654 WRTBLK: MOV @#BLKLIM,R1 ;PICKUP THE BLOCK SIZE
2842 014664 112714 000003 ;WRITE!GO,@TACS ;START A WRITE
2843 014670 104413 1$: WAITXFER ;WAIT ON TRANSFER REQUEST
2844 014672 032714 000040 BIT #READY,@TACS ;DID READY SET?
2845 014676 001010 BNE 3$ ;BR IF YES
2846 014700 005301 DEC R1 ;COUNT THIS REQUEST
2847 014702 002403 BLT 2$ ;BR IF TIME FOR ILBS
2848 014704 113715 014656 MOV# @#PATRN,@TADB ;PUT DATA ON TAPE
2849 014710 000767 BR 1$ ;LOOP
2850 014712 052714 000020 2$: BIS #ILBS,@TACS ;WRITE CRC AND SHUT DOWN
2851 014716 104412 WAITREADY ;WAIT ON THE READY FLAG
2852 014720 000207 3$: RTS PC
2853
2854
2855 ;*****
2856 ; READ ROUTINE FOR THE MANUAL OPERATIONS
2857 ;*****
2858 014722 013702 014654 RDBLK: MOV @#BLKLIM,R2 ;PICKUP THE BLOCK SIZE
2859 014726 013700 014656 MOV @#PATRN,R0 ;USE THIS DATA PATTERN TO COMPARE TO
2860 014732 112714 000005 MOV# $READ!GO,@TACS ;START A READ
2861 014736 104413 1$: WAITXFER ;WAIT ON TRANSFER REQUEST
2862 014740 032714 000040 BIT #READY,@TACS ;IS READY SET?
2863 014744 001012 BNE 3$ ;BR IF YES
2864 014746 005302 DEC R2 ;COUNT THIS REQUEST
2865 014750 002405 BLT 2$ ;BR IF TIME FOR ILBS
2866 014752 011501 MOV @TADB,R1 ;READ THE DATA BUFFER
2867 014754 120001 CMPB R0,R1 ;CHECK THE DATA
2868 014756 001767 BEQ 1$ ;BR IF OK
2869 014760 104005 ERROR 5 ;BAD DATA
2870 014762 000406 HR 4$ ;GET OUT
2871 014764 052714 000020 2$: BIS #ILBS,@TACS ;READ ILBS
2872 014770 104412 WAITREADY ;WAIT ON READY
2873 014772 005714 3$: TST @TACS ;CHECK FOR ERROR
2874 014774 100001 BPL 4$ ;BR IF NONE
2875 014776 104001 ERROR 1 ;ERROR OCCURRED
2876 015000 000207 4$: KTS PC ;RETURN
2877
2878
2879 ;*****
2880 ; ROUTINE TO CHANGE DRIVES
2881 015002 105777 164132 NXTDRV: TSTB @SWR ;IS SW07 ON A (1)?
2882 015006 100416 BMI 3$ ;BR IF YES
2883 015010 005003 CLR DRIVE ;SET DRIVE TO "A"
2884 015012 013701 001230 MOV @#DRVPT,R1 ;GET DRIVE POINTER
2885 015016 122127 000101 CMPB (R1)+,#"A" ;IS IT DRIVE "A"?
2886 015022 001402 BEQ 1$ ;BR IF YES
2887 015024 012703 000400 MOV #UNIT,DRIVE ;SET DRIVE TO "B"
2888 015030 105711 1$: TSTB (R1) ;LAST DRIVE BEEN SELECTED
2889 015032 001002 BNE 2$ ;BR IF NO
2890 015034 012701 001224 MOV #DRVKEY,R1 ;RESET DRIVE POINTER
2891 015040 010137 001230 2$: MOV R1,@#DRVPT ;SAVE DRIVE POINTER FOR NEXT TIME
2892 015044 000207 3$: RTS PC ;GO BACK
    
```

```

2893 ;*****
2894 ;
2895 ;SBTTL ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2896
2897 ;CALL:
2898 ;
2899 ; MOV #DRVKEY,R0
2900 ; JSR PC,@#EXAM ;R1 IS DESTROYED
2901 ;
2902 ; NORMAL RETURN
2903 ; ERROR RETURN
2903 015046 013701 001206 EXAM: MOV @#TACSL,R1 ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2904 015052 005011 CLR (R1) ;DRIVE="A", FUNCTION="WFG"
2905 015054 122710 000101 CMPB #"A,(R0) ;EXAMINE DRIVE "A"?
2906 015060 001402 BEQ 1$ ;BR IF YES
2907 015062 052711 000400 BIS #UNIT,(R1) ;SELECT DRIVE "B"
2908 015066 032711 000040 1$: BIT #READY,(R1) ;WAIT ON READY
2909 015072 001775 BEQ 1$
2910 015074 005711 TST (R1) ;ANY ERROR?
2911 015076 100024 BPL 4$ ;BR IF NO
2912 015100 032711 001000 BIT #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2913 015104 001017 BNE 3$ ;BR IF YES
2914 015106 032711 010000 BIT #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2915 015112 001411 BEQ 2$ ;BR IF NO
2916 015114 122777 000201 164016 CMPB #BIT07!BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
2917 015122 001412 BEQ 4$ ;BR IF YES
2918 015124 122777 000203 164006 CMPB #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
2919 015132 001406 BEQ 4$ ;BR IF YES
2920 015134 000403 BR 3$ ;TAKE THE ERROR EXIT
2921 015136 032711 020000 2$: BIT #LEADER,(R1) ;ERROR DUE TO "CLEAR LEADER"?
2922 015142 001002 BNE 4$ ;BR IF YES
2923 015144 062716 000002 3$: ADD #2,(SP) ;TAKE ERROR RETURN
2924 015150 000207 4$: RTS PC ;RETURN
    
```

```

2925          ,SBTTL TYPE ROUTINE
2926
2927          ;*****
2928          ;*ROUTINE TO TYPE ASCIZ MESSAGE, MESSAGE MUST TERMINATE WITH A 0 BYTE.
2929          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2930          ;*NOTE1:  SNULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2931          ;*NOTE2:  SFILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2932          ;*NOTE3:  SFILLC CONTAINS THE CHARACTER TO FILL AFTER.
2933          ;*
2934          ;*CALL:
2935          ;*1) USING A TRAP INSTRUCTION
2936          ;*      TYPE      ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2937          ;*OR
2938          ;*      TYPE
2939          ;*      MESADR
2940          ;*
2941
2942          015152 105767 164001      STYPE: TSTB  STPFLG          ;;IS THERE A TERMINAL?
2943          015156 100002              BPL  1$              ;;BR IF YES
2944          015160 000000              HALT                    ;;HALT HERE IF NO TERMINAL
2945          015162 000407              BR  3$              ;;LEAVE
2946          015164 010046              1$:  MOV  R0,-(SP)      ;;SAVE R0
2947          015166 017600 000002      MOV  02(SP),R0        ;;GET ADDRESS OF ASCIZ STRING
2948          015172 112046              2$:  MOV  (R0)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2949          015174 010045              MOV  BNE  4$          ;;BR IF IT ISN'T THE TERMINATOR
2950          015176 005726              TST  (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
2951          015200 012600              60$: MOV  (SP)+,R0    ;;RESTORE R0
2952          015202 062716 000002      3$:  ADD  #2,(SP)     ;;ADJUST RETURN PC
2953          015206 000002              RTI                    ;;RETURN
2954          015210 122716 000011      4$:  CMPB #HT,(SP)    ;;BRANCH IF <HT>
2955          015214 001430              BEQ  8$              ;;BRANCH IF NOT <CRLF>
2956          015216 122716 000200      CMPB #CRLF,(SP)     ;;BRANCH IF NOT <CRLF>
2957          015222 001006              BNE  5$              ;;BRANCH IF NOT <CR><LF> EQUIV
2958          015224 005726              TST  (SP)+           ;;TYPE A CR AND LF
2959          015226 104401              $CRLF
2960          015230 001177              CLR  CLRB            ;;CLEAR CHARACTER COUNT
2961          015232 105067 000130      BR  2$              ;;GET NEXT CHARACTER
2962          015236 000755              JSR  PC,$TYPEPC     ;;GO TYPE THIS CHARACTER
2963          015240 004767 000056      5$:  JSR  PC,$TYPEPC     ;;IS IT TIME FOR FILLER CHARS.?
2964          015244 126726 163706      6$:  CMPB #FILLC,(SP)+ ;;IF NO GO GET NEXT CHAR.
2965          015250 001350              BNE  2$              ;;IF NO GO GET NEXT CHAR.
2966          015252 016746 163676      MOV  #NULL,-(SP)    ;;GET # OF FILLER CHARS. NEEDED
2967
2968          015256 105366 000001      7$:  DECB 1(SP)        ;;AND THE NULL CHAR.
2969          015262 002770              BLT  6$              ;;DOES A NULL NEED TO BE TYPED?
2970          015264 004767 000032      JSR  PC,$TYPEPC     ;;BR IF NO--GO POP THE NULL OFF OF STACK
2971          015270 105367 000072      DECB $CHARCNT       ;;GO TYPE A NULL
2972          015274 000770              BR  7$              ;;DO NOT COUNT AS A COUNT
2973
2974          ;HORIZONTAL TAB PROCESSOR
2975
2976          015276 112716 000040      8$:  MOV  #' ,(SP)     ;;REPLACE TAB WITH SPACE
2977          015302 004767 000014      9$:  JSR  PC,$TYPEPC     ;;TYPE A SPACE
2978          015306 132767 000007 000052 BITB #7,$CHARCNT    ;;BRANCH IF NOT AT
2979          015314 001372              BNE  9$              ;;TAB STOP
2980          015316 005726              TST  (SP)+           ;;POP SPACE OFF STACK
    
```

```

2981          015320 000724              BR  2$              ;;GET NEXT CHARACTER
2982          015322 105777 163622      STYPE: TSTB  #STPS    ;;WAIT UNTIL PRINTER IS READY
2983          015326 100375              BPL  $TYPEPC
2984          015330 116677 000002 163614  MOV  2(SP),#STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2985          015336 122766 000015 000002  CMPB #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
2986          015344 001003              BNE  1$              ;;BRANCH IF NO
2987          015346 105067 000014      CLR  $CHARCNT       ;;YES--CLEAR CHARACTER COUNT
2988          015352 000406              BR  $TYPEPC         ;;EXIT
2989          015354 122766 000012 000002  1$:  CMPB #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
2990          015362 001402              BEQ  $TYPEPC         ;;BRANCH IF YES
2991          015364 105227              INCB (PC)+          ;;COUNT THE CHARACTER
2992          015366 000000              $CHARCNT: WORD 0    ;;CHARACTER COUNT STORAGE
2993          015370 000207              $TYPEPC: RTS  PC
2994
2995          ,SBTTL READ AN OCTAL NUMBER FROM THE TTY
2996
2997          ;*****
2998          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2999          ;*CHANGE IT TO BINARY.
3000          ;*CALL:
3001          ;*      RDOCT          ;;READ AN OCTAL NUMBER
3002          ;*      RETURN HERE    ;;LOW ORDER BITS ARE ON TOP OF THE STACK
3003          ;*
3004
3005          015372 011646              SRDOCT: MOV  (SP),-(SP) ;;PROVIDE SPACE FOR THE
3006          015374 016666 000004 000002  MOV  4(SP),2(SP)    ;;INPUT NUMBER
3007          015402 010046              MOV  R0,-(SP)       ;;PUSH R0 ON STACK
3008          015404 010146              MOV  R1,-(SP)       ;;PUSH R1 ON STACK
3009          015406 010246              MOV  R2,-(SP)       ;;PUSH R2 ON STACK
3010          015410 104410              1$:  RDLIN (SP)+,R0    ;;READ AN ASCIZ LINE
3011          015412 012600              MOV  (SP)+,R0       ;;GET ADDRESS OF 1ST CHARACTER
3012          015414 005001              CLR  R1              ;;CLEAR DATA WORD
3013          015416 005002              CLR  R2
3014          015420 112046              2$:  MOV  (R0)+,-(SP)  ;;PICKUP THIS CHARACTER
3015          015422 001412              BEQ  3$              ;;IF ZERO GET OUT
3016          015424 006301              ASL  R1              ;;*2
3017          015426 006102              ROL  R2
3018          015430 006301              ASL  R1              ;;*4
3019          015432 006102              ROL  R2
3020          015434 006301              ASL  R1              ;;*8
3021          015436 006102              ROL  R2
3022          015440 042716 177770      BIC  #'C7,(SP)     ;;STRIP THE ASCII JUNK
3023          015444 062601              ADD  (SP)+,R1       ;;ADD IN THIS DIGIT
3024          015446 000764              BR  2$              ;;LOOP
3025          015450 005726              3$:  TST  (SP)+       ;;CLEAN TERMINATOR FROM STACK
3026          015452 010166 000012      MOV  R1,12(SP)     ;;SAVE THE RESULT
3027          015456 010267 000010      MOV  R2,$SHIOCT
3028          015462 012602              MOV  (SP)+,R2       ;;POP STACK INTO R2
3029          015464 012601              MOV  (SP)+,R1       ;;POP STACK INTO R1
3030          015466 012600              MOV  (SP)+,R0       ;;POP STACK INTO R0
3031          015470 000002              RTI                    ;;RETURN
3032          015472 000000              $SHIOCT: WORD 0     ;;HIGH ORDER BITS GO HERE
3033
3034          ,SBTTL TTY INPUT ROUTINE
3035
3036          ;*****
          ,ENABL LSB
    
```

```

3037
3038
3039
3040
3041
3042
3043 015474 022767 000176 163436
3044 015502 001074
3045 015504 105777 163434
3046 015510 100071
3047 015512 117746 163430
3048 015516 042716 177600
3049 015522 022726 000007
3050 015526 001062
3051 015530 126727 163400 000001
3052 015536 001456
3053
3054 015540 104401 016221
3055 015544 104401 016226
3056 015550 016746 162422
3057 015554 104402
3058 015556 104401 016237
3059 015562 005046
3060 015564 005046
3061 015566 105777 163352
3062 015572 100375
3063
3064 015574 117746 163346
3065 015600 042716 177600
3066
3067
3068
3069 015604 021627 000025
3070 015610 001005
3071 015612 104401 016214
3072 015616 062706 000006
3073 015622 000757
3074
3075
3076 015624 021627 000015
3077 015630 001022
3078 015632 005766 000004
3079 015636 001403
3080 015640 016577 000002 163272
3081 015646 062706 000006
3082 015652 104401 001177
3083 015656 126727 163253 000001
3084 015664 001003
3085 015666 012777 000100 163250
3086 015674 000002
3087 015676 004767 177420
3088 015702 021627 000060
3089 015706 002420
3090 015710 021627 000067
3091 015714 003015
3092 015716 042726 000060

;*****
;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TR/P CALL
;WHEN OPERATING IN TTY FLAG MODE.
SCKSWR: CMP #SWREG,SWR ;IS THE SOFT-SWR SELECTED?
        BNE 15$ ;BRANCH IF NO
        TSTB @STKS ;CHAR THERE?
        BPL 15$ ;IF NO, DON'T WAIT AROUND
        MOV #STKB,-(SP) ;SAVE THE CHAR
        BIC #'C177,(SP) ;STRIP-OFF THE ASCII
        CMP #7,(SP)+ ;IS IT A CONTROL G?
        BNE 15$ ;NO, RETURN TO USER
        CMPR SAUTOB,#1 ;ARE WE RUNNING IN AUTO-MODL?
        BEQ 15$ ;BRANCH IF YES

        TYPE ,SCNTLG ;ECHO THE CONTROL-G (^G)
        TYPE ,SMSWR ;TYPE CURRENT CONTENTS
        MOV SWREG,-(SP) ;SAVE SWREG FOR TYPEOUT
        TYPOC ;GO TYPE-OCTAL ASCII(ALL DIGITS)
        TYPE ,SMNEW ;PROMPT FOR NEW SWR
19$: CLR -(SP) ;CLEAR COUNTER
    CLR -(SP) ;THE NEW SWR
7$: TSTB @STKS ;CHAR THERE?
    BPL 7$ ;IF NOT TRY AGAIN

        MOV #STKB,-(SP) ;PICK UP CHAR
        BIC #'C177,(SP) ;MAKE IT 7-BIT ASCII

98: CMP (SP),#25 ;IS IT A CONTROL-U?
    BNE 10$ ;BRANCH IF NOT
    TYPE ,SCNTLU ;YES, ECHO CONTROL-U (^U)
20$: ADD #6,SP ;IGNORE PREVIOUS INPUT
    BR 19$ ;LET'S TRY IT AGAIN

10$: CMP (SP),#15 ;IS IT A <CR>?
    BNE 16$ ;BRANCH IF NO
    TST 4(SP) ;YES, IS IT THE FIRST CHAR?
    BEQ 11$ ;BRANCH IF YES
    MOV 2(SP),@SWR ;SAVE NEW SWR
    ADD #6,SP ;CLEAR UP STACK
11$: TYPE ,SCLF ;ECHO <CR> AND <LF>
14$: CMPC $INTAG,#1 ;RE-ENABLE TTY KBD INTERRUPTS?
    BNE 15$ ;BRANCH IF NOT
    MOV #100,@STKS ;RE-ENABLE TTY KBD INTERRUPTS
15$: RTI ;RETURN
16$: JSP PC,$TYPEC ;ECHO CHAR
    CMP (SP),#60 ;CHAR < 0?
    BPL 18$ ;BRANCH IF YES
    CMP (SP),#67 ;CHAR > 7?
    BGT 18$ ;BRANCH IF YES
    BIC #60,(SP)+ ;STRIP-OFF ASCII

```

```

3093 015722 005766 000002
3094 015726 001403
3095 015730 006316
3096 015732 006316
3097 015734 006316
3098 015736 005266 000002
3099 015742 056616 177776
3100 015746 000707
3101 015750 104401 001176
3102 015754 000720
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114 015756 011646
3115 015760 016666 000004 000002
3116 015766 105777 163152
3117 015772 100375
3118 015774 117766 163146 000004
3119 016002 042766 177600 000004
3120 016010 026627 000004 000023
3121 016016 001013
3122 016020 105777 163120
3123 016024 100375
3124 016026 117746 163114
3125 016032 042716 177600
3126 016036 022627 000021
3127 016042 001366
3128 016044 000750
3129 016046 026627 000004 000140
3130 016054 002407
3131 016056 026627 000004 000175
3132 016064 003003
3133 016066 042766 000040 000004
3134 016074 000002
3135
3136
3137
3138
3139
3140
3141
3142 016076 010346
3143 016100 012703 016204
3144 016104 022703 016214
3145 016110 101405
3146 016112 104407
3147 016114 112613
3148 016116 122713 000177

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
;RDCHR ;INPUT A SINGLE CHARACTER FROM THE TTY
;RETURN HERE ;CHARACTER IS ON THE STACK
; ;WITH PARITY BIT STRIPPED OFF
;

SRDCHR: MOV (SP),-(SP) ;PUSH DOWN THE PC
        MOV 4(SP),2(SP) ;SAVE THE PS
1$: TSTB @STKS ;WAIT FOR
    BPL 1$ ;A CHARACTER
    MOV #STKB,4(SP) ;READ THE TTY
    BIC #'C<177>,4(SP) ;GET RID OF JUNK IF ANY
    CMP 4(SP),#23 ;IS IT A CONTROL-S?
    BNE 3$ ;BRANCH IF NO
    TSTB @STKS ;WAIT FOR A CHARACTER
    BPL 2$ ;LOOP UNTIL ITS THERE
    MOV #STKB,-(SP) ;GET CHARACTER
    BIC #'C177,(SP) ;MAKE IT 7-BIT ASCII
    CMP (SP)+,#21 ;IS IT A CONTROL-Q?
    BNE 2$ ;IF NOT DISCARD IT
    BR 1$ ;YES, RESUME
3$: CMP 4(SP),#140 ;IS IT UPPER CASE?
    BLT 4$ ;BRANCH IF YES
    CMP 4(SP),#175 ;IS IT A SPECIAL CHAR?
    BGT 4$ ;BRANCH IF YES
    BIC #40,4(SP) ;MAKE IT UPPER CASE
4$: RTI ;GO BACK TO USER
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;RDLIN ;INPUT A STRING FROM THE TTY
;RETURN HERE ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
; ;TERMINATOR WILL BE A BYTE OF ALL 0'S
;

SRDLIN: MOV R3,-(SP) ;SAVE R3
        MOV @TTYIN,R3 ;GET ADDRESS
2$: CMP @TTYIN+8,,R3 ;BUFFER FULL?
    BLOS 4$ ;BR IF YES
    RDCHR ;GO READ ONE CHARACTER FROM THE TTY
    MOV (SP)+,(R3) ;GET CHARACTER
10$: CMPC #177,(R3) ;IS IT A RUBOUT

```

```

3149 016122 001003          BNE      38          ;;SKIP IF NOT
3150 016124 104401 001176    4$:      TYPE     ,SQUES      ;;TYPE A "7"
3151 016130 000763          BR       15          ;;CLEAR THE BUFFER AND LOOP
3152 016132 111367 000044    3$:      MOVVB    (R3),98      ;;ECHO THE CHARACTER
3153 016136 104401 016202    TYPE     ,98
3154 016142 122723 000015    CMPB    #15,(R3)+    ;;CHECK FOR RETURN
3155 016146 001356          BNE      28          ;;LOOP IF NOT RETURN
3156 016150 105063 177777    CLRBB   -1(R3)      ;;CLEAR RETURN (THE 15)
3157 016154 104401 001200    TYPE     ,SLF       ;;TYPE A LINE FEED
3158 016160 012603          MOV      (SP)+,R3    ;;RESTORE R3
3159 016162 011646          MOV      (SP),-(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3160 016164 016666 000004 000002 MOV      4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
3161 016172 012766 016204 000004 MOV      #STTYIN,4(SP)
3162 016200 000002          RTI              ;;RETURN
3163 016202 000          .BYTE     0        ;;STORAGE FOR ASCII CHAR. TO TYPE
3164 016203 000          .BYTE     0        ;;TERMINATOR
3165 016204 000010          .BLKB     8         ;;RESERVE 8 BYTES FOR TTY INPUT
3166 016214 052536 005015 000 STTYIN:  .ASCIZ  "/U/<15><12>  ;;CONTROL "U"
3167 016221 136 006507 000012 SCNTLU:  .ASCIZ  "/G/<15><12>  ;;CONTROL "G"
3168 016226 005015 053523 020122 SMSWR:  .ASCIZ  <15><12>/SWR = /
3169 016234 020075          .WORD     0
3170 016237 040 047040 053505 SMNEW:  .ASCIZ  / NEW = /
3171 016244 036440 000040
3172          .SBTTL   BINARY TO OCTAL (ASCII) AND TYPE
3173
3174          ;*****
3175          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3176          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3177          ;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3178          ;*CALL:
3179          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3180          ;*      TYPOS      ;;CALL FOR TYPEOUT
3181          ;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3182          ;*      .BYTE   M              ;;M=1 OR 0
3183          ;*                                  ;;1=TYPE LEADING ZEROS
3184          ;*                                  ;;0=SUPPRESS LEADING ZEROS
3185          ;*
3186          ;*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3187          ;*STYPOS OR STYPOC
3188          ;*CALL:
3189          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3190          ;*      TYPON      ;;CALL FOR TYPEOUT
3191          ;*
3192          ;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3193          ;*CALL:
3194          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3195          ;*      TYPOC      ;;CALL FOR TYPEOUT
3196          ;*
3197          016250 017646 000000  $TYPOS:  MOV      0(SP),-(SP)  ;;PICKUP THE MODE
3198          016254 116667 000001 000211  MOVVB   1(SP),SOFILL  ;;LOAD ZERO FILL SWITCH
3199          016262 112667 000207  MOVVB   (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
3200          016266 062716 000002  ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
3201          016272 000406  BR       STYPON
3202          016274 112767 000001 000171 STYPOC: MOV      #1,SOFILL  ;;SET THE ZERO FILL SWITCH
3203          016302 112767 000006 000165 MOV      #6,SOMODE+1  ;;SET FOR SIX(6) DIGITS
3204          016310 112767 000005 000154 STYPON: MOV      #5,SOCNT  ;;SET THE ITERATION COUNT

```

```

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

```

```

3249          .SBTTL TRAP DECODER
3250
3251          ;*****
3252          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3253          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3254          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3255          ;*GO TO THAT ROUTINE.
3256
3257 016476 010046          STRAP:  MOV    R0,-(SP)          ;;SAVE R0
3258 016500 016600 000002          MOV    2(SP),R0          ;;GET TRAP ADDRESS
3259 016504 005740          TST    -(R0)            ;;BACKUP BY 2
3260 016506 111000          MOV    (R0),R0          ;;GET RIGHT BYTE OF TRAP
3261 016510 006300          ASL    R0                ;;POSITION FOR INDEXING
3262 016512 016000 016532          MOV    $TRPAD(R0),R0    ;;INDEX TO TABLE
3263 016516 000200          RTS    R0                ;;GO TO ROUTINE
3264
3265
3266          ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3267
3268 016520 011646          STRAP2: MOV   (SP),-(SP)    ;;MOVE THE PC DOWN
3269 016522 016666 000004 000002          MOV   4(SP),2(SP)        ;;MOVE THE PSW DOWN
3270 016530 000002          RTI                      ;;RESTORE THE PSW
3271
3272          .SBTTL TRAP TABLE
3273
3274          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3275          ;*BY THE "TRAP" INSTRUCTION.
3276
3277          ;          ROUTINE
3278          ;          -----
3279 016532 016520          STRPAD: .WORD   STRAP2
3280 016534 015152          STYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3281 016536 016274          STYPOC ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3282 016540 016250          STYPOS ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3283 016542 016310          STYPON ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3284
3285 016544 015544          $GTSWR ;;CALL=GTSWR    TRAP+5(104405)  GET SOFT-SWR SETTING
3286
3287 016546 015474          $CKSWR ;;CALL=CKSWR    TRAP+6(104406)  TEST FOR CHANGE IN SOFT-SWR
3288 016550 015756          $RDCHR ;;CALL=RDCHR    TRAP+7(104407)  TTY TYPEIN CHARACTER ROUTINE
3289 016552 016076          $RDLIN ;;CALL=RDLIN    TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3290 016554 015372          $RDOCT ;;CALL=RDOCT    TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
3291 016556 013046          WAIT_ON_READY ;;CALL=WAITREADY TRAP+12(104412) WAIT ON THE READY BIT TO
3292 016560 013152          WAIT_FOR_XFER ;;CALL=WAITXFER TRAP+13(104413) WAIT ON XFER REQ.

```

```

3293          .SBTTL POWER DOWN AND UP ROUTINES
3294
3295          ;*****
3296          ;POWER DOWN ROUTINE
3297 016562 012737 016726 000024          SPWRDN: MOV    $SILLUP,@PWRVEC ;;SET FOR FAST UP
3298 016570 012737 000340 000026          MOV    $340,@PWRVEC+2 ;;PRIO:7
3299 016576 010046          MOV    R0,-(SP)          ;;PUSH R0 ON STACK
3300 016600 010146          MOV    R1,-(SP)          ;;PUSH R1 ON STACK
3301 016602 010246          MOV    R2,-(SP)          ;;PUSH R2 ON STACK
3302 016604 010346          MOV    R3,-(SP)          ;;PUSH R3 ON STACK
3303 016606 010446          MOV    R4,-(SP)          ;;PUSH R4 ON STACK
3304 016610 010546          MOV    R5,-(SP)          ;;PUSH R5 ON STACK
3305 016612 017746 162322          MOV    @SWR,-(SP)        ;;PUSH @SWR ON STACK
3306 016616 010667 000110          MOV    SP,$SAVR6         ;;SAVE SP
3307 016622 012737 016634 000024          MOV    $SPWRUP,@PWRVEC  ;;SET UP VECTOR
3308 016630 000000          HALT
3309 016632 000776          BR     #-2                ;;HANG UP
3310
3311          ;*****
3312          ;POWER UP ROUTINE
3313 016634 012737 016726 000024          $PWRUP: MOV    $SILLUP,@PWRVEC ;;SET FOR FAST DOWN
3314 016642 016706 000064          MOV    $SAVR6,SP        ;;GET SP
3315 016646 005067 000060          CLR    $SAVR6           ;;WAIT LOOP FOR THE TTY
3316 016652 005267 000054          1$:  INC    $SAVR6         ;;WAIT FOR THE INC
3317 016656 001375          BNE    1$                ;;OF WORD
3318 016660 012677 162254          MOV    (SP)+,@SWR        ;;POP STACK INTO @SWR
3319 016664 012605          MOV    (SP)+,R5          ;;POP STACK INTO R5
3320 016666 012604          MOV    (SP)+,R4          ;;POP STACK INTO R4
3321 016670 012603          MOV    (SP)+,R3          ;;POP STACK INTO R3
3322 016672 012602          MOV    (SP)+,R2          ;;POP STACK INTO R2
3323 016674 012601          MOV    (SP)+,R1          ;;POP STACK INTO R1
3324 016676 012600          MOV    (SP)+,R0          ;;POP STACK INTO R0
3325 016700 012737 016562 000024          MOV    $SPWRDN,@PWRVEC  ;;SET UP THE POWER DOWN VECTOR
3326 016706 012737 000340 000026          MOV    $340,@PWRVEC+2  ;;PRIO:7
3327 016714 104401          TYPE          ;;REPORT THE POWER FAILURE
3328 016716 016734          $PWRMG: .WORD   $POWER    ;;POWER FAIL MESSAGE POINTER
3329 016720 012716          MOV    (PC)+,(SP)        ;;RESTART AT PWRST
3330 016722 002206          $PWRAD: .WORD   PWRST     ;;RESTART ADDRESS
3331 016724 000002          RTI
3332 016726 000000          $SILLUP: HALT           ;;THE POWER UP SEQUENCE WAS STARTED
3333 016730 000776          BR     #-2                ;; BEFORE THE POWER DOWN WAS COMPLETE
3334 016732 000000          $SAVR6: 0                ;;PUT THE SP HERE
3335 016734 005015 047520 042527          $POWER: .ASCIZ <15><12>"POWER"
3336 016742 000122
3337          .EVEN

```


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

3394	017402	020040	020040	051102	
3395	017410	046040	053105	046105	
3396	017416	000			
3397		017420			.EVEN
3398	017420	021116	001162	000000	DT1: ,WORD \$ERRPC,\$REG0,0
3399	017426	001116	001162	001202	DT2: ,WORD \$ERRPC,\$REG0,\$AVPC,0
3400	017434	000000			
3401	017436	001116	001162	001124	DT5: ,WORD \$ERRPC,\$REG0,\$GGDAT,\$BDDAT,0
3402	017444	001126	000000		
3403	017450	001116	001162	001124	DT6: ,WORD \$ERRPC,\$REG0,\$GGDAT,0
3404	017456	000000			
3405	017460	001116	001206	000000	DT201: ,WORD \$ERRPC,TACSL,0
3406					
3407	017466	001116	000000		DT202: ,WORD \$ERRPC,0
3408					
3409	017472	040524	030461	043040	EM201: ,ASCIZ "TA11 FAILED TO RESPOND"
3410	017500	044501	042514	020104	
3411	017506	047524	051040	051505	
3412	017514	047520	042116	000	
3413	017521	116	020117	051104	EM202: ,ASCIZ "NO DRIVE AVAILABLE"
3414	017526	053111	020105	053101	
3415	017534	044501	040514	046102	
3416	017542	000105			
3417	017544	041520	020040	020040	DH201: ,ASCIZ /PC TACS/
3418	017552	020040	040524	051503	
3419	017560	000			
3420	017561	120	000103		DH202: ,ASCIZ /PC/
3421		000001			.END

ASKADR	013364	563	2512*																	
ASKDRV	013254	560	2480*	2503																
ASKKEY	001232	396*	651*	681																
BEGINX	001740	557*	593	595	644	646														
BEGIN1	001336	231	476*																	
BEGIN2	001370	232	483*																	
BEGIN3	001376	233	482	485*																
BEGIN4	001404	234	487*																	
BGNCMN	001410	479	481	484	486	488*														
BIT0	= 000001	166*																		
BIT00	= 000001	156*	166	208	2588	2916	2918													
BIT01	= 000002	155*	165	207	2597	2918														
BIT02	= 000004	154*	164	206																
BIT03	= 000010	153*	163	205																
BIT04	= 000020	152*	162	204																
BIT05	= 000040	151*	161	203																
BIT06	= 000100	150*	160	202																
BIT07	= 000200	149*	159	201	2916	2918														
BIT08	= 000400	148*	158	200	2298															
BIT09	= 001000	147*	157	199	2306	2370														
BIT1	= 000002	165*																		
BIT10	= 002000	146*	198	2354																
BIT11	= 004000	145*	197	2313																
BIT12	= 010000	144*	196																	
BIT13	= 020000	143*	195	2361																
BIT14	= 040000	142*	194	2284	2597															
BIT15	= 100000	141*	193	662																
BIT2	= 000004	164*																		
BIT3	= 000010	163*																		
BIT4	= 000020	162*																		
BIT5	= 000040	161*																		
BIT6	= 000100	160*																		
BIT7	= 000200	159*																		
BIT8	= 000400	158*																		
BIT9	= 001000	157*																		
BLKLLM	014654	2020*	2834*	2841	2858															
BPTVEC	= 000014	173*																		
BBSG	= 000010	186*	868	1505	1589	1620	1680	1778	1783	1814	1859	1902	1932	1967						
BSFG	= 000006	2024	2163	2186																
BSFGSB	014512	185*	864	1447	1562	1676	1731	1739	1928	2797										
CHKADR	001762	241	2794*	2795	2802															
CHKDRV	002046	559	562	581*																
CKSWR	= 104406	620*																		
CR	= 000015	2283	2346	2369	3287*															
CR	= 000015	81*	2985	2995																
CR	= 000015	194*	1873	1916	1992	2052														
CRCERR	= 040000	2028*	2038	2046	2585	2606*	2613*													
CRC,WD	013774	82*	545	2956	2995	3338	3355													
CRLF	= 000200	397*	674*	675																
CURDRV	001234	88*	365	516																
DDISP	= 177570	418	3379*																	
DH1	017256	424	430	436	3382*															
DH2	017273	456	3417*																	
DH201	017544	461	3420*																	
DH202	017561	442	3387*																	
DH5	017330	448	3392*																	
DH6	017366																			

DISPLA	001142	365*	516*	524*	2327*	2353*														
DISPRE	000174	228*	524																	
DO,CRC	013654	2029	2578*																	
DRVKEY	001224	394*	395	477*	620	633	654	679	2075	2128	2481*	2486	2500	2504						
DRVPNT	001230	2890																		
DSWR	= 177570	395*	634*	654*	665	680*	2884	2891*												
DT1	017420	87*	364																	
DT2	017426	419	3398*																	
DT201	017460	425	431	437	3399*															
DT202	017466	457	3405*																	
DT5	017436	462	3407*																	
DT6	017450	443	3401*																	
EMTVEC	= 000030	449	3403*																	
EM1	017066	176*	499*	500*																
EM2	017105	417	3356*																	
EM201	017472	423	3359*																	
EM202	017521	455	3409*																	
EM3	017133	460	3413*																	
EM4	017174	429	3363*																	
EM5	017217	435	3369*																	
EM6	017234	441	3373*																	
EM6	017234	447	3376*																	
ERROR	= 100000	193*	1538	1567	1736	1741	1780	1873	2052											
ERRVEC	= 000004	169*	513	514*	525*	581*	587*	2289	2290*	2292*	2295*									
EXAM	015046	621	625	630	2903*															
EXIT	013354	2499	2504*																	
FGAP	= 004000	197*	1449	1483	1538	1564	1598	1629	1660	1728	1733	1775	1780							
FUNCTI	= 000016	209*																		
FUNCT0	= 000002	207*	209																	
FUNCT1	= 000004	206*	209																	
FUNCT2	= 000010	205*	209																	
GNS	= ***** U	227	544	672	3280	3281	3282	3283	3285	3287	3288	3289	3290	3291						
GO	= 000001	3292																		
GO	= 000001	208*	303	306	325	328	693	710	712	731	733	771	773	782						
GO	= 000001	784	823	825	827	1292	1295	1303	1335	1366	1368	1371	1437	1439						
GO	= 000001	1441	1447	1467	1469	1471	1477	1479	1481	1495	1497	1499	1505	1522						
GO	= 000001	1524	1526	1532	1534	1536	1554	1556	1562	1579	1581	1587	1589	1591						
GO	= 000001	1593	1610	1612	1618	1620	1622	1624	1642	1644	1646	1654	1656	1658						
GO	= 000001	1710	1712	1714	1722	1724	1726	1731	1739	1757	1759	1761	1769	1771						
GO	= 000001	1773	1778	1783	1804	1806	1814	1816	1839	1841	1844	1859	1861	1889						
GO	= 000001	1891	1893	1902	1904	1957	1959	1967	1969	2004	2006	2009	2024	2026						
GO	= 000001	2086	2136	2138	2142	2152	2163	2168	2186	2193	2205	2230	2236	2269						
GO	= 000001	2668	2691	2722	2725	2750	2756	2778	2781	2797	2842	2860								



ENGINEERING CHANGE ORDER

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

ECO NO. MD-11-DZTAB-C1/61
SHEET 1 OF 2
DATE RECEIVED 18 Aug-76
ISSUE DATE
FINAL RELEASE

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

UNIT TO BE CHANGED
MD-11-DZTAB

CORRECTION /DEPO
Install patch to diagnostic
Conditional

PRODUCT AFFECTED
TA11

TYPE OF ECO
[] HARDWARE
[X] SOFTWARE
[] PURCHASE SPEC.

BREAK-IN/EFFECTIVITY
Immediately

FIELD SERVICE AFFECTED
[] YES [] NO
[F] D, P, R DISTR.
[] L.O.U. CODE

QUICK-CHECK

WHERE USED

TEST

Table with 4 columns: DOCUMENT/PART NO., OLD REV, NEW REV, DESCRIPTION OF CHANGE/DISPOSITION OF MATERIAL. Contains rows for document changes and location changes.

APPROVAL SIGNATURE (TYPE NAME and SIGN)
PROJECT ENGR. O. Choate
ENG. MGR. M. Horovitz
TEL. EXT. 4088 COST CENTER NO. 301
DISC. PROJ. NO. V20-07846 DATE 8/18/76

REVIEW SIGNATURES (SEE INSTRUCTIONS FOR APPLIC.)
FIELD SERVICE
DIAGNOSTIC ENGR.

COORD. NO. REF. NO.

digital

CONTINUATION

ECO NO. DZTAB-C1/01

ENGINEERING
CHANGE ORDER

SHEET

SHEET 2 OF 2

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