

# TA11

MOTION TEST  
MD-11-DZTAD-C

EP-DZTAD-C-DL-A  
COPYRIGHT © 1976  
FICHE 1 OF 1

NOV 1976  
**digital**  
MADE IN U.S.A.

## IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAD-C-D  
PRODUCT NAME: TALL MOTION TEST.  
DATE REVISED: 21 JUNE 76  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: JIM LACEY

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

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

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

COPYRIGHT (C) 1973, 1976 BY DIGITAL EQUIPMEN CORPORATION

## 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 TESTS THAT CHECK THE TUBO DRIVE FOR PROPER OPERATION.

## 2. REQUIREMENTS

## 2.1 EQUIPMENT

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

## 2.2 STORAGE

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.

## 2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZTAA  
MAINDEC-11-DZTAB  
MAINDEC-11-DZTAC

## 3. LOADING PROCEDURE

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

## 4. STARTING PROCEDURE

## 4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

## 4.2 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS  
204 SELECT DRIVE(S) BEFORE STARTING TEST  
210 SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST  
214 SETUP FOR MANUAL LOOPING  
220 WRITE FILE GAP FROM BOT TO EOT  
224 WRITE 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



## 4.3 PROGRAM &amp; OPERATOR ACTION

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

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

## 4.3.1 DRIVE SELECTION

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

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

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

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

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

## 4.3.1.1 DRIVE SELECTION EXAMPLES

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

## 4.3.2 ADDRESS SELECTION

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

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

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
2. VECTOR ADDRESS
3. PRIORITY LEVEL

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR 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?

## 4.3.3 SUBTEST LOOPING

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

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

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

## 4.3.3.1 SUBTEST LOOPING EXAMPLES

```
:OPERATOR TYPES "↑C"
TEST PC? 2242
LOOP PC?
```

```
:OPERATOR TYPES "2242""CARRIAGE RETURN "(CR)"
:OPERATOR TYPES "CR" WHICH
:IMPLIES "LOOP PC"="TEST PC"
```

```
:OPERATOR TYPES "↑C"
TEST PC? 3000
LOOP PC? 3020
```

```
:OPERATOR TYPES "3000""CR"
:OPERATOR TYPES "3020""CR"
:PROGRAM USES THIS AS THE
:FIRST ADDRESS OF THE SCOPE
:LOOP
```

```
:OPERATOR TYPES "↑C"
TEST PC?
```

```
:OPERATOR TYPES "CR"
:PROGRAM CONTINUES
:FROM THIS POINT
```

## 5. OPERATING PROCEDURE

SEQ 0007

## 5.1 OPERATIONAL SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

## CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G (<↑G>); THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE ''NEW=''' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
  - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
  - B) IF A CONTROL U (<↑U>) IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

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

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



## 5.2 SUBROUTINE ABSTRACTS

## 5.2.1 SCOPE

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

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

## 5.2.2 TRAPCATCHER

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

## 5.2.3 ERROR

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

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

## 5.2.4 TRAP

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

## 5.2.4.1 TYPE (\$TYPE)

TYPE AN ASCIZ STRING ON THE TTY

## 5.2.4.2 RDCHR (\$RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

## 5.2.4.3 RDLIN (\$RDLIN)

READ AN ASCII STRING FROM THE TTY

## 5.2.4.4 SETLOOP (T.SETLOOP)

SETUP TO LOOP AS PER THE TTY

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

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

5.2.5.1 WFG (T.WFG) WRITE A FILE GAP

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

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

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

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

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

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

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

5.2.5.9 SELDRV (T.SELDRV) SELECT A DRIVE

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



## 5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A JSR

## 5.2.6.1 DO.CMD

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

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

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

## 5.2.6.2 WAITFLAG

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

THE EXITS FOLLOW THE CALL AND ARE:

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

## 5.2.6.3 FLAGS

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

THE RETURNS FOLLOW THE CALL AND ARE:

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

## 5.2.6.3 DO.CRC

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

## 5.2.6.4 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

## 5.2.6.5 GETDRV

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

## 5.2.6.6 ASKADR

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

## 5.2.6.7 TYPERR

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

## 5.2.6.8 EXAM

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

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

## 5.2.7.1 WFGSUB

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

## 5.2.7.2 WRTSUB

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

## 5.2.7.3 RDSUB

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

## 5.2.7.4 WGPBLK

WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO 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="

## 5.2.7.5 RGBLK

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

## 5.2.7.6 SFFGSB

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

## 5.2.7.7 BSFGSB

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

## 6. ERRORS

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

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

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

## 7. RESTRICTIONS

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

## 8. MISCELLANEOUS

## 8.1 EXECUTION TIME

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

## 8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

## 8.3 PASS COUNT

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

## 8.4 ITERATIONS

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

## 8.5 SPECIAL REGISTERS

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

## 9. PROGRAM DESCRIPTION

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

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

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

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200

```

GENERAL INFORMATION
OPERATIONAL SWITCH SETTINGS
BASIC DEFINITIONS
STARTING ADDRESSES
TRAP CATCHER
STARTING ADDRESS(ES)
COMMON TAGS
ERROR POINTER TABLE
START OF TEST
INITIALIZE THE COMMON TAGS
TYPE PROGRAM NAME
GET VALUE FOR SOFTWARE SWITCH REGISTER
T1 REWIND WRITE REWIND READ
T2 TEST BSBG AFTER WRITE AND READ AFTER BSBG
T3 TEST BSFG AFTER WRITE AND READ AFTER BSFG
T4 TEST BSBG AFTER READ
T5 TEST BSFG AFTER READ
T6 TEST SFBG AFTER REWIND AND BSFG AFTER SFBG
T7 TEST WRITE AFTER READ
T10 TEST WRITE AFTER WRITE AND READ AFTER READ
T11 TEST BSBG AFTER BSBG
T12 TEST READ AFTER SFBG
T13 TEST SFBG AFTER BSBG
T14 TEST SFBG AFTER BSFG
T15 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
T16 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
T17 TEST WFG AFTER WRITE AND BSFG AFTER WFG
T20 TEST BSBG AFTER WFG
T21 TEST WRITE AFTER WFG
T22 TEST WFG AFTER READ AND BSFG AFTER WRITE
T23 TEST SFFG AFTER REWIND AND READ AFTER SFFG
T24 TEST BSBG AFTER SFFG
T25 TEST SFBG AFTER SFFG
T26 TEST SFBG AFTER SFBG
T27 TEST BSFG AFTER SFBG AND BSFG AFTER BSFG
T30 TEST BSBG THRU A FILE GAP
T31 TEST BSFG AFTER BSBG
T32 TEST BSBG AFTER BSFG
T33 TEST BSFG AFTER SFFG AND SFFG AFTER READ
T34 TEST SFFG AFTER SFFG
T35 TEST SFFG AFTER SFBG
T36 TEST WFG AFTER SFBG
T37 TEST WRITE AFTER SFBG
*****THE FOLLOWING TESTS TRY TO GENERATE NOISE***
T40 3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
T41 TWO BLOCK FILES---MULTI SPACING FUNCTIONS
T42 MULTI SPACING & TEST SFBG AFTER READ
T43 TWO BLOCK FILES--MULTI SPACING & TEST SFFG AFTER BSBG
T44 6 ONE BLOCK FILES---MULTI SPACING
END OF PASS ROUTINE
ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR TYPEOUT ROUTINE
ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY

```



110051  
110050  
110049  
110048  
110047  
110046  
110045  
110044  
110043  
110042  
110041  
110040  
110039  
110038  
110037  
110036  
110035  
110034  
110033  
110032  
110031  
110030  
110029  
110028  
110027  
110026  
110025  
110024  
110023  
110022  
110021  
110020  
110019  
110018  
110017  
110016  
110015  
110014  
110013  
110012  
110011  
110010  
110009  
110008  
110007  
110006  
110005  
110004  
110003  
110002  
110001  
110000

ROUTINE TO CHANGE AN ASCII STRING TO OCTAL  
ROUTINE TO GET "TEST" AND "SCOPE LOOP" ADDRESSES FROM THE TTY  
ROUTINE TO SELECT THE DRIVE TO BE TESTED  
ROUTINE TO COMPARE THE READ AND WRITE BUFFERS  
ROUTINE TO "WRITE-FILE-GAP"  
ROUTINE TO "WRITE" A RECORD ON THE TAPE  
ROUTINE TO "READ" A RECORD  
ROUTINE TO "BACK-SPACE-FILE-GAP"  
ROUTINE TO "BACK-SPACE-BLOCK-GAP"  
ROUTINE TO "SPACE-FORWARD-FILE-GAP"  
ROUTINE TO "SPACE-FORWARD-BLOCK-GAP"  
ROUTINE TO REWIND THE TAPE  
ROUTINE TO WAIT FOR A FLAG  
ROUTINE TO EXAMINE THE FLAGS  
ROUTINE TO CALCULATE THE CRC  
ROUTINE TO PERFORM THE REQUESTED FUNCTION  
ROUTINE TO SERVICE INTERRUPTS  
ROUTINE TO SERVICE ILLGAL INTERRUPTS FROM THE TAI1  
\*\*\*\*\* MANUAL ADJUSTMENT ROUTINES \*\*\*\*\*  
WRITE FILE GAPS FROM "BOT" TO "EOT"  
WRITE CONTINUOUS BLOCKS OF DATA  
READ AND COMPARE CONTINUOUS BLOCKS OF DATA  
WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT  
READ AND COMPARE A BLOCK OF DATA AND A FILE GAP  
SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"  
BACK SPACE FILE GAP  
SETUP READ AND WRITE LINKS FOR SUBROUTINES  
TYPE ROUTINE  
TTY INPUT ROUTINE  
READ AN OCTAL NUMBER FROM THE TTY  
BINARY TO OCTAL (ASCII) AND TYPE  
TRAP DECODER  
TRAP TABLE  
POWER DOWN AND UP ROUTINES  
DATA TABLE POINTERS FOR ERRORS  
ASCII MESSAGES  
READ AND WRITE BUFFERS AND LINKS



F02

TALL MOTION TEST  
DZTADC.NEW 24-MAY-76 00:00

MAINDEC-11-DZTAD-C

MACY11 27(1005) 23-SEP-76 10:33 PAGE 2  
GENERAL INFORMATION

SEQ 0018

57

00

GO BIT

0

WRITE ONLY!

58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113

.SBTTL OPERATIONAL SWITCH SETTINGS

```

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

```

.SBTTL BASIC DEFINITIONS

```

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

```

.\*MISCELLANEOUS DEFINITIONS

```

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

```

.\*GENERAL PURPOSE REGISTER DEFINITIONS

```

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

```

.\*PRIORITY LEVEL DEFINITIONS

```

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

```

114 000300  
115 000340

PR6= 300  
PR7= 340

::PRIORITY LEVEL 6  
::PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

118 100000  
119 040000  
120 020000  
121 010000  
122 004000  
123 002000  
124 001000  
125 000400  
126 000200  
127 000100  
128 000040  
129 000020  
130 000010  
131 000004  
132 000002  
133 000001

SW15= 100000  
SW14= 40000  
SW13= 20000  
SW12= 10000  
SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1

.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

146 100000  
147 040000  
148 020000  
149 010000  
150 004000  
151 002000  
152 001000  
153 000400  
154 000200  
155 000100  
156 000040  
157 000020  
158 000010  
159 000004  
160 000002  
161 000001

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1

.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2

169

170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223

000004  
000010  
000014  
000014  
000014  
000020  
000024  
000030  
000034  
000060  
000064  
000240  
  
000000  
000002  
000004  
000006  
000010  
000012  
000014  
000016  
  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002  
000001  
000016  
  
000004  
000005

```

.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

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

;*****TAIL FUNCTIONS*****
XWFG= 0             ;;WRITE FILE GAP FUNCTION
XWRITE= 2           ;;WRITE FUNCTION
XREAD= 4            ;;READ FUNCTION
XBSFG= 6            ;;BACK SPACE FILE GAP FUNCTION
XBSBG= 10           ;;BACK SPACE BLOCK GAP FUNCTION
XSFFG= 12           ;;SPACE FWD FILE GAP FUNCTION
XSFBG= 14           ;;SPACE FWD BLOCK GAP FUNCTION
XRWND= 16           ;;REWIND FUNCTION

;*****TAIL BIT ASSIGNMENT*****
ERROR= BIT15
CRCERR= BIT14
LEADER= BIT13
WRTLOCK=BIT12
FGAP= BIT11
TIMERR= BIT10
OFFLINE=BIT09
UNIT= BIT08
TR.REQ= BIT07
INT.EN= BIT06
READY= BIT05
ILBS= BIT04
FUNC2= BIT03
FUNC1= BIT02
FUNCO= BIT01
GO= BIT00
FUNCTION=          FUNC2+FUNC1+FUNCO

;////////////////////////////////////
;////////////////////////////////////
;SPECIAL REGISTERS
TACS= %4           ;;R4 IS USED AS A POINTER TO THE TACS REGISTER
TADB= %5           ;;R5 IS USED AS A POINTER TO THE TADB REGISTER.
;////////////////////////////////////

```

224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246

000000  
  
000174 000174  
000176 000000  
  
000200 000137 001450  
000204 000137 001502  
000210 000137 001510  
000214 000137 002052  
000220 000137 012374  
000224 000137 012450  
000230 000137 012532  
000234 000137 012620  
000240 000137 012712  
000244 000137 013010  
000250 000137 013064

```
.SBTTL TRAP CATCHER
.=0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
.=174
DISPREG: .WORD 0      ;; SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP      @#BEGIN1     ;; JUMP TO STARTING ADDRESS OF PROGRAM
JMP      @#BEGIN2     ;SELECT DRIVE(S) BEFORE STARTING TEST
JMP      @#BEGIN3     ;SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
JMP      @#BEGINX    ;SETUP FOR MANUAL LOOPING
JMP      @#WFGSUB     ;WRITE FILE GAP FROM BOT TO EOT
JMP      @#WRTSUB     ;WRITE CONTINUOUS BLOCKS OF DATA
JMP      @#RDSUB      ;READ CONTINUOUS BLOCKS OF DATA
JMP      @#WGPBLK    ;WRITE FILE GAP AND A BLOCK OF DATA
JMP      @#RGPBLK    ;READ BLOCK OF DATA AND INTO A FILE GAP
JMP      @#SFFGSB    ;SPACE FWD FILE GAP FROM BOT TO EOT
JMP      @#BSFGSB    ;BACK SPACE FILE GAPS
;*****
```

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  
295  
296  
297  
298  
299  
300  
301  
302

.SBTTL COMMON TAGS

\*\*\*\*\*  
\*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
\*USED IN THE PROGRAM.

001100

.=1100

\$CMTAG:	.WORD	0	:: START OF COMMON TAGS
\$PASS:	.WORD	0	:: CONTAINS PASS COUNT
\$TSTNM:	.BYTE	0	:: CONTAINS THE TEST NUMBER
\$ERFLG:	.BYTE	0	:: CONTAINS ERROR FLAG
\$ICNT:	.WORD	0	:: CONTAINS SUBTEST ITERATION COUNT
\$LPADR:	.WORD	0	:: CONTAINS SCOPE LOOP ADDRESS
\$LPERR:	.WORD	0	:: CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL:	.WORD	0	:: CONTAINS TOTAL ERRORS DETECTED
\$ITEMB:	.BYTE	0	:: CONTAINS ITEM CONTROL BYTE
\$ERMAX:	.BYTE	1	:: CONTAINS MAX. ERRORS PER TEST
\$ERRPC:	.WORD	0	:: CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR:	.WORD	0	:: CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR:	.WORD	0	:: CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT:	.WORD	0	:: CONTAINS 'GOOD' DATA
\$BDDAT:	.WORD	0	:: CONTAINS 'BAD' DATA
	.WORD	0	:: RESERVED--NOT TO BE USED
\$AUTOB:	.BYTE	0	:: AUTOMATIC MODE INDICATOR
\$INTAG:	.BYTE	0	:: INTERRUPT MODE INDICATOR
	.WORD	0	
\$SWR:	.WORD	DSWR	:: ADDRESS OF SWITCH REGISTER
\$DISPLAY:	.WORD	DDISP	:: ADDRESS OF DISPLAY REGISTER
\$TKS:	177560		:: TTY KBD STATUS
\$TKB:	177562		:: TTY KBD BUFFER
\$TPS:	177564		:: TTY PRINTER STATUS REG. ADDRESS
\$TPB:	177566		:: TTY PRINTER BUFFER REG. ADDRESS
\$NULL:	.BYTE	0	:: CONTAINS NULL CHARACTER FOR FILLS
\$FILLS:	.BYTE	2	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC:	.BYTE	12	:: INSERT FILL CHARS. AFTER A "LINE FEED"
\$TPFLG:	.BYTE	0	:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
\$REGAD:	.WORD	0	:: CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
\$REGO:	.WORD	0	:: CONTAINS ((\$REGAD)+0)
\$REG1:	.WORD	0	:: CONTAINS ((\$REGAD)+2)
\$TMPO:	.WORD	0	:: USER DEFINED
\$TIMES:	0		:: MAX. NUMBER OF ITERATIONS
\$ESCAPE:	0		:: ESCAPE ON ERROR ADDRESS
\$BELL:	.ASCIZ	<207><377><377>	:: CODE FOR BELL
\$QUES:	.ASCII	/?/	:: QUESTION MARK
\$CRLF:	.ASCII	<15>	:: CARRIAGE RETURN
\$LF:	.ASCIZ	<12>	:: LINE FEED
*****			
\$AVPC:	.WORD	0	:: STORAGE FOR THE PC
\$BYTNUM:	.WORD	0	:: THE NUMBER OF BYTES "READ" OR "WRITTEN"
\$RCRC0:	.WORD	0	:: CRC CALCULATED FOR "WRITE"
\$RCRC1:	.WORD	0	:: CRC CALCULATED FOR "READ"
\$RCRC2:	.WORD	0	:: CRC FROM THE TA11 FOR "READ"
\$MXCNT:	1		:: MAX. NUMBER OF ITERATIONS

000377



TA11 MOTION TEST  
DZTADC.NEW

24-MAY-76 00:00

MAINDEC-11-DZTAD-C

COMMON TAGS

MACY11 27(1006)

23-SEP-76

10:33 PAGE 8

SEQ 0024

303	001220	177500	
304	001222	177501	
305	001224	177502	
306	001226	177503	
307	001230	000260	000262
308	001234	000300	
309	001236	000000	
310	001240	000000	000000
311	001244	001240	
312	001246	000000	
313	001250	177777	000000
314	001254	000000	
315	001256	001260	
316	001260	000000	
317	001262	000000	
318	001264	000000	000000

TACSL: 177500  
TACSH: 177501  
TADBL: 177502  
TADBH: 177503  
TAVEC: 260,262  
TAPRIO: 300  
DRIVE: 0  
DRVKEY: 0,0  
DRVPT: DRVKEY  
ASKKEY: 0  
CURDRV: -1,0  
WAITKEY: 0  
RWDFLG: RWDA  
RWDA: 0  
RWDB: 0  
STALL: 0,0

;LOW BYTE ADDRESS OF TACS  
;HIGH BYTE ADDRESS OF TACS  
;LOW BYTE ADDRESS OF TADB  
;HIGH BYTE ADDRESS OF TADB  
;TA11 VECTOR ADDRESS  
;TA11 BR LEVEL 6  
;NEXT DRIVE TO TEST  
;DRIVE SELECT KEY:

;CURRENT DRIVE BEING TESTED  
;WAIT ON INTERRUPT KEY  
;REWIND FLAG POINTER  
;DRIVE "A" REWIND FLAG  
;DRIVE "B" REWIND FLAG  
;STALL TIME

319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374

.SBTTL ERROR POINTER TABLE

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

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

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

;ITEM 1  
EM1 ;IMPROPER FLAG SETTING  
DH1 ;TEST ERROR  
;PC PC TACS TADB  
DT1 ;SAVPC \$ERRPC \$REGO \$REG1  
0

;ITEM 2  
EM2 ;IMPROPER FLAG OCCURRED  
DH1 ;TEST ERROR  
;PC PC TACS TADB  
DT1 ;SAVPC \$ERRPC \$REGO \$REG1  
0

;ITEM 3  
EM3 ;MISSED A FLAG  
DH1 ;TEST ERROR  
;PC PC TACS TADB  
DT1 ;SAVPC \$ERRPC \$REGO \$REG1  
0

;ITEM 4  
EM4 ;INTERRUPT FAILED  
DH1 ;TEST ERROR  
;PC PC TACS TADB  
DT1 ;SAVPC \$ERRPC \$REGO \$REG1  
0

;ITEM 5  
EM5 ;PREMATURE READY OCCURRED  
DH1 ;TEST ERROR  
;PC PC TACS TADB  
DT1 ;SAVPC \$ERRPC \$REGO \$REG1  
0

;ITEM 6  
EM6 ;DIDN'T STOP IN A FILE GAP  
DH1 ;TEST ERROR  
;PC PC TACS TADB

001270

001270 015550  
001272 016112  
001274 015054  
001276 000000

001300 015576  
001302 016112  
001304 015054  
001306 000000

001310 015625  
001312 016112  
001314 015054  
001316 000000

001320 015643  
001322 016112  
001324 015054  
001326 000000

001330 015664  
001332 016112  
001334 015054  
001336 000000

001340 015715  
001342 016112

375	001344	015054	DT1	;	SAVPC	\$ERRPC	\$REGO	\$REG1			
376	001346	000000	0								
377				;	ITEM	7					
378			EM7	;	DIDN'T	STOP	ON	CLEAR	LEADER		
379	001350	015747	DH1	;	TEST	ERROR					
380	001352	016112		;	PC	PC	TACS	TADB			
381				;	SAVPC	\$ERRPC	\$REGO	\$REG1			
382	001354	015054	DT1								
383	001356	000000	0								
384				;	ITEM	10					
385			EM10	;	BAD	DATA	READ				
386	001360	016003	DH2	;	TEST	ERROR		EXPT'D	RCV'D	BYTE	
387	001362	016172		;	PC	PC	TACS	DATA	DATA	NUMBER	
388				;	SAVPC	\$ERRPC	\$REGO	\$GDDAT	\$BDDAT	BYTNUM	
389	001364	015066	DT2								
390	001366	000000	0								
391				;	ITEM	11					
392			EM11	;	ILLEGAL	BUFFER					
393	001370	016021	DH5	;	TEST	ERROR					
394	001372	016611		;	PC	PC	TACS				
395				;	SAVPC	\$ERRPC	\$REGO				
396	001374	015136	DT5								
397	001376	000000	0								
398				;	ITEM	12					
399			EM12	;	CRC	ERROR					
400	001400	016045	DH3	;	TEST	ERROR		WRITE	READ	TU60	
401	001402	016333		;	PC	PC	TACS	CRC	CRC	CRC	
402				;	SAVPC	\$ERRPC	\$REGO	RCRC0	RCRC1	RCRC2	
403	001404	015104	DT3								
404	001406	000000	0								
405				;	ITEM	13					
406			EM13	;	SHORT	RECORD					
407	001410	016057	DH4	;	TEST	ERROR			BYTES		
408	001412	016471		;	PC	PC	TACS	TADB	LEFT		
409				;	SAVPC	\$ERRPC	\$REGO	\$REG1	\$TMP0		
410	001414	015122	DT4								
411	001416	000000	0								
412				;	ITEM	14					
413			EM14	;	BAD	INTERRUPT					
414	001420	016074	DH5	;	TEST	ERROR					
415	001422	016611		;	PC	PC	TACS				
416				;	SAVPC	\$ERRPC	\$REGO				
417	001424	015136	DT5								
418	001426	000000	0								
419											
420	001430			ITEMS2:							
421					;	ITEMS	201-202				
422	001430	016674	EM201	;	TA11	FAILED	TO	RESPOND			
423	001432	016746	DH201	;	PC	TACS					
424	001434	016662	DT201	;	\$ERRPC	TACS					
425	001436	000000	0	;	BOTH	NUMBERS	ARE	TYPED	AS	OCTAL	NUMBERS
426											
427	001440	016723	EM202	;	NO	DRIVES	AVAILABLE				
428	001442	016763	DH202	;	PC						
429	001444	016670	DT202	;	\$ERRPC						
430	001446	000000	0	;							

400000  
400001  
400002  
400003  
400004  
400005  
400006  
400007  
400008  
400009  
400010  
400011  
400012  
400013  
400014  
400015  
400016  
400017  
400018  
400019  
400020  
400021  
400022  
400023  
400024  
400025  
400026  
400027  
400028  
400029  
400030  
400031  
400032  
400033  
400034  
400035  
400036  
400037  
400038  
400039  
400040  
400041  
400042  
400043  
400044  
400045  
400046  
400047  
400048  
400049  
400050  
400051  
400052  
400053  
400054  
400055  
400056  
400057  
400058  
400059  
400060  
400061  
400062  
400063  
400064  
400065  
400066  
400067  
400068  
400069  
400070  
400071  
400072  
400073  
400074  
400075  
400076  
400077  
400078  
400079  
400080  
400081  
400082  
400083  
400084  
400085  
400086  
400087  
400088  
400089  
400090  
400091  
400092  
400093  
400094  
400095  
400096  
400097  
400098  
400099  
400100

////////////////////////////////////  
////////////////////////////////////  
:\*\*\*\*\*

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

:\*\*\*\*\*

001450 005005  
001452 012737 041101 001240  
001454 122737 000005 000041  
001456 001015  
001470 022737 000260 001230  
001476 001011  
001500 000403  
001502 012705 000001  
001506 000405  
001510 012705 000002  
001514 000402  
001516 012705 000003  
001522

BEGIN1: CLR R5 ;NORMAL START  
MOV #AB,3#DRVKEY  
CMPB #5,3#41 ;CASSETTE DDP?  
BNE BGNCMN ;GO BEGIN COMMON CODE IF NO  
CMP #260,3#TAVEC ;STANDARD VECTOR?  
BNE BGNCMN ;GO BEGIN COMMON CODE IF NO  
BR BEGIN3 ;GET DRIVES AND ADDRESSES  
BEGIN2: MOV #1,R5 ;ASK FOR DRIVES FLAG  
BR BGNCMN ;BEGIN COMMON CODE  
BEGIN3: MOV #2,R5 ;ASK FOR DRIVES AND ADDRESSES  
BR BGNCMN  
BEGIN4: MOV #3,R5  
BGNCMN:

.SBTTL INITIALIZE THE COMMON TAGS  
:CLEAR THE COMMON TAGS (\$CMTAG) AREA  
MOV #CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED  
CLR (R6)+ ;:CLEAR MEMORY LOCATION  
CMP #SWR,R6 ;:DONE?  
BNE -6 ;:LOOP BACK IF NO  
MOV #STACK,SP ;:SETUP THE STACK POINTER  
:INITIALIZE A FEW VECTORS  
MOV #SCOPE,3#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE  
MOV #340,3#IOTVEC+2 ;:LEVEL 7  
MOV #ERROR,3#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE  
MOV #340,3#EMTVEC+2 ;:LEVEL 7  
MOV #TRAP,3#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS  
MOV #340,3#TRAPVEC+2 ;:LEVEL 7  
MOV #SPWRDN,3#PWRVEC ;:POWER FAILURE VECTOR  
MOV #340,3#PWRVEC+2 ;:LEVEL 7  
MOV SENDCT,3#EOPCT ;:SETUP END-OF-PROGRAM COUNTER  
CLR STIMES ;:INITIALIZE NUMBER OF ITERATIONS  
CLR \$ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS  
MOVB #1,\$ERMAX ;:ALLOW ONE ERROR PER TEST  
MOV #,\$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE  
MOV #,\$SLPERR ;:SETUP THE ERROR LOOP ADDRESS  
:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS  
:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.  
MOV 3#ERRVEC, -(SP) ;:SAVE ERROR VECTOR  
MOV #64,\$ERRVEC ;:SET UP ERROR VECTOR  
MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER  
MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER  
CMP #-1,\$SWR ;:TRY TO REFERENCE HARDWARE SWR  
BNE 66\$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED  
;AND THE HARDWARE SWR IS NOT = -1  
BR 65\$ ;:BRANCH IF NO TIMEOUT

```

497 001722 012716 001730 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
498 001726 000002 RTI
499 001730 012767 000176 177202 65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
500 001736 012767 000174 177176 MOV #DISPREG, DISPLAY
501 001744 012637 000004 66$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR

.SBTTL TYPE PROGRAM NAME
;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001750 005227 177777 INC #-1 ;;FIRST TIME?
001754 001036 BNE HERE ;;BRANCH IF NO
001756 022737 006442 000042 CMP #SENDAD, @#42 ;;ACT-11?
001764 001432 BEQ HERE ;;BRANCH IF YES
001766 104401 002024 TYPE MSGID ;;TYPE ASCIZ STRING

.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
001772 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
001776 001006 BNE 67$ ;;BRANCH IF YES
002000 026727 177134 000176 CMP SWR, #SWREG ;;SOFTWARE SWITCH REG SELECTED?
002006 001005 BNE 68$ ;;BRANCH IF NO
002010 104405 GTSWR ;;GET SOFT-SWR SETTINGS
002012 000403 BR 68$
002014 112767 000001 177112 67$: MOVB #1, SAUTOB ;;SET AUTO-MODE INDICATOR
002022 68$: BR HERE ;;GET OVER THE ASCIZ
002022 000413 .MSGID: .ASCIZ <CRLF>/MAINDEC-11-DZTAD-C/<CRLF>
002052 HERE:

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000008  
000009  
000010  
000011  
000012  
000013  
000014  
000015  
000016  
000017  
000018  
000019  
000020  
000021  
000022  
000023  
000024  
000025  
000026  
000027  
000028  
000029  
000030  
000031  
000032  
000033  
000034  
000035  
000036  
000037  
000038  
000039  
000040  
000041  
000042  
000043  
000044  
000045  
000046  
000047  
000048  
000049  
000050  
000051  
000052  
000053  
000054  
000055  
000056  
000057  
000058  
000059  
000060  
000061  
000062  
000063  
000064  
000065  
000066  
000067  
000068  
000069  
000070  
000071  
000072  
000073  
000074  
000075  
000076  
000077  
000078  
000079  
000080  
000081  
000082  
000083  
000084  
000085  
000086  
000087  
000088  
000089  
000090  
000091  
000092  
000093  
000094  
000095  
000096  
000097  
000098  
000099  
000100

\*\*\*\*\*  
\*\*\*\*\*

: THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE

R5=3 MANUAL OPERATIONS  
R5=2 ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)  
R5=1 ASK FOR DRIVE(S)  
R5=0 DON'T ASK FOR ANYTHING

\*\*\*\*\*

002052 010504  
002054 005305  
002056 002406  
002060 004737 007432  
002064 005305  
002066 002402  
002070 004737 007542

BEGINX: MOV R5,R4 ; COPY R5  
DEC R5 ; ASK FOR DRIVES?  
BLT CHKADR ; BR IF NO  
JSR PC, @#ASKDRV ; GO GET DRIVES TO BE TESTED  
DEC R5 ; ASK FOR ADDRESSES?  
BLT CHKADR ; BR IF NO  
JSR PC, @#ASKADR ; GO GET TAIL ADDRESSES

\*\*\*\*\*  
\*\*\*\*\*

: CHECK THAT "TACS" WILL RESPOND TO ADDRESSING

I. TIMEOUT OCCURRED  
A. TYPE ERROR MESSAGE  
B. EXAMINE R4  
1. R4>0 GOTO BEGINX  
2. R4=0 EXAMINE (42)  
A. (42)=0 GOTO BEGINX  
B. (42)>0 GOTO SENDAD

II. TIMEOUT DIDN'T OCCUR  
A. CONTINUE

\*\*\*\*\*

002074 012737 002112 000004  
002102 005000  
002104 005777 177110  
002110 000402  
002112 005200 1\$:  
002114 022626  
002116 012737 000006 000004 2\$:  
002124 005700  
002126 001412  
002130 104201  
002132 012705 000002  
002136 005704  
002140 001344  
002142 013700 000042  
002146 001741  
002150 000137 006442  
002154 3\$:

CHKADR: MOV #1\$, @#ERRVEC ; IN CASE OF TIMEOUTS  
CLR R0 ; USE AS A SWITCH  
TST @TACSL ; SEE IF TAIL RESPONDS  
BR 2\$ ; BR IF NO TIMEOUT  
INC R0 ; COME HERE ON TIMEOUT  
CMP (SP)+, (SP)+ ; CLEANUP THE STACK  
MOV #ERRVEC+2, @#ERRVEC ; RESTORE TIMEOUT VECTOR  
TST R0 ; DID A TIMEOUT OCCUR?  
BEQ 3\$ ; BR IF NO  
ERROR 201 ; TAIL FAILED TO RESPOND  
MOV #2, R5 ; DRIVES & ADDRESSES  
TST R4 ; OPERATOR INPUTS?  
BNE BEGINX ; BR IF YES  
MOV @#42, R0 ; GET MONITOR RETURN ADDRESS  
BEQ BEGINX ; BR IF NO MONITOR  
JMP @#SENDAD ; GO TO END

564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619

\*\*\*\*\*  
\*\*\*\*\*

:MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED

- I. DESIRED DRIVES CAN NOT BE TESTED
  - A. TYPE ERROR MESSAGE
  - B. EXAMINE R4
    - 1. R4>0 GOTO BEGINX
    - 2. R4=0 EXAMINE (42)
      - A. (42)=0 GOTO BEGINX
      - B. (42)>0 GOTO \$ENDAD
- II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
  - A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
  - B. CONTINUE IN PROGRAM
- III. DESIRED DRIVE(S) CAN BE TESTED
  - A. CONTINUE IN PROGRAM

\*\*\*\*\*

002154	012700	001240	
002160	004737	006474	
002164	000410		
002166	116010	000001	
002172	001412		
002174	004737	006474	
002200	000407		
002202	005010		
002204	000405		
002206	005200		
002210	004737	006474	
002214	000401		
002216	105010		
002220	012700	001240	
002224	010037	001244	
002230	121060	000001	
002234	001002		
002236	105060	000001	
002242	005710		
002244	001401		
002246	000412		
002250	104202		
002252	012705	000002	
002256	005704		
002260	001274		
002262	013700	000042	
002266	001671		
002270	000137	006442	
002274	020427	000003	
002300	001002		
002302	016704	175471	
002306	010437	001246	
002312	000405		
002314	104401	002024	
002320	012737	001240	001244

```

CHKDRV: MOV      #DRVKEY, R0      ; PICKUP ADDRESS OF ASCII DRIVE KEY
        JSR      PC, @#EXAM      ; GO EXAMINE FIRST DRIVE
        BR       1$              ; OK TO TEST---GO CHECK NEXT
        MOVB    1(R0), (R0)      ; REPLACE 1ST WITH 2ND
        BEQ     2$              ; BR IF NO 2ND DRIVE SELECTED
        JSR      PC, @#EXAM      ; GO EXAMINE DRIVE
        BR       2$              ; OK TO TEST
        CLR     (R0)             ; CLEAR DRIVE CODES
        BR       2$
1$:     INC     R0                ; POINT TO 2ND
        JSR      PC, @#EXAM      ; GO EXAMINE DRIVE
        BR       2$              ; OK TO TEST
        CLRB   (R0)             ; CLEAR 2ND
        BR       2$              ; RESET ADDRESS POINTERS
2$:     MOV     #DRVKEY, R0
        MOV     R0, @#DRVPNT
        CMPB   (R0), 1(R0)
        BNE    3$              ; 1ST = 2ND?
        CLRB   1(R0)           ; BR IF NO
        TST   (R0)             ; YES---CLEAR 2ND
        BEQ     5$              ; ANY DRIVES?
        BR     MANUAL           ; BR IF NO
3$:     TST   (R0)
        BEQ     5$
5$:     ERROR  202              ; NO DRIVES AVAILABLE
        MOV     #2, R5          ; DRIVES & ADDRESS
        TST   R4                ; OPERATOR INPUTS?
        BNE   BEGINX           ; BR IF YES
        MOV     @#42, R0        ; GET MONITOR RETURN ADDRESS
        BEQ   BEGINX           ; NO MONITOR
        JMP    @#$ENDAD        ; GO TO END
MANUAL: CMP    R4, #3
        BNE   OK
        MOV   -1, R4
OK:     MOV   R4, @#ASKKEY
        BR   START
PWRST: TYPE  #MSGID            ; POWER FAIL RESTART
        MOV   #DRVKEY, @#DRVPNT

```

# F03

TAII MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 15  
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0031

```

620 002326 012777 012366 176674 START: MOV #BADINT, @TAVEC ;SETUP TAIL TRAP VECTOR
621 002334 013777 001234 176670 MOV @TAPRIO, @TAVEC+2
622 002342 012737 000340 177776 MOV #340, @#PS ;LOCKOUT ALL I/O INT
623 002350 013704 001220 MOV @#TACSL, TACS ;SETUP TACS
624 002354 013705 001224 MOV @#TADBL, TADB ;SETUP TADB
625 002360 005037 001102 CLR @#STSTNM ;ZERO THE TEST NUMBER
626 002364 005037 006624 CLR @#LOOPKEY ;CLEAR THE LOOP UNDER TTY CONTROL KEY
627 002370 012737 000001 001216 MOV #1, @#SMXCNT ;SET FOR ONE ITERATION PER TEST
628 002376 013701 001244 MOV @#DRVPT, R1 ;GET DRIVE POINTER
629 002402 010167 176630 MOV R1, DRIVE ;SET DRIVE TO POINTER
630
631 ;////////////////////////////////////
632 ;TYPE THE DRIVE(S) THAT WILL BE TESTED
633 MOVB (R1), @#CURDRV ;GET CURRENT DRIVE FOR TYPE-OUT
634 TYPE ,MTSTDRV ;TYPE "TESTING DRIVE"
635 TYPE ,CURDRV ;TYPE THE CURRENT DRIVE
636 002406 111137 001250 BIT #SW07, @SWR ;LOCK ON ONE DRIVE?
637 002412 104401 015271 BNE 2$ ;BR IF YES
638 002416 104401 001250 TSTB @#DRVKEY+1 ;TWO DRIVES AVAILABLE?
639 002422 032777 000200 176510 BEQ 2$ ;BR IF NO
640 002430 001021 INC @#SMXCNT ;SET FOR TWO ITERATIONS PER TEST (ONE PER DRIVE)
641 002432 105737 001241 INC R1 ;INCREMENT THE POINTER
642 002436 001416 MOVB (R1), @#CURDRV ;SET SECOND DRIVE FOR TYPE-OUT
643 002440 005237 001216 BNE 1$ ;GO TYPE IT
644 002444 005201 MOV #DRVKEY, R1 ;RESET POINTER
645 002446 111137 001250 MOVB (R1), @#CURDRV ;NOW GET THE SECOND DRIVE
646 002452 001004 TYPE ,MANDRV ;TYPE "AND DRIVE"
647 002454 012701 001240 TYPE ,CURDRV ;TYPE DRIVE
648 002460 111137 001250 1$: MOV R1, @#DRVPT ;SAVE POINTER FOR NEXT TRIP THRU
649 002464 104401 015312 2$: MOV #5$, @#SLPADR ;SETUP THE SCOPE LOOP ADDRESS
650 002470 104401 001250 TST @#SPASS ;ONCE ONLY
651 002474 010137 001244 BNE TST1
652 ;////////////////////////////////////
653 ;ON THE FIRST PASS OF THE PROGRAM DETERMINE THE AMOUNT OF TIME TO
654 ;WAIT ON A FLAG ("READY"/"TRANSFER REQUEST") AND THE STALL TIME TO USE
655 ;WHEN STALLING AFTER A FUNCTION IS COMPLETED
656 SELDRV ;SELECT DRIVE
657 002514 104422 MOV #FATAL, @#$ESCAPE ;SETUP ESCAPE ON ERROR
658 002516 012737 002716 001172 CLR @RWDFLG ;SET FOR NO REWIND ERRORS
659 002524 005077 176526 MOV #1CBIT15, @#MAXCNT ;SETUP IMPOSSIBLE COUNT
660 002530 012737 077777 012022 MOV @XRWD, @TACS ;INSURE NO ERROR DUE TO "CLEAR LEADER"
661 002536 112714 000016 REWIND ;GO TO BOT
662 002542 104421 WFG ;NOW TIME A WFG
663 002544 104412 SUB @#HICNT, @#MAXCNT ;GET THE TIME IT TOOK TO WFG
664 002546 163737 012020 012022 INC @#MAXCNT ;MAKE IT BIGGER
665 002554 005237 012022 ROL @#MAXCNT
666 002560 006137 012022 ROL @#MAXCNT
667 002564 006137 012022 CLR R0
668 002570 005000 CLR R1
669 002572 005001 MOV #4$, @TAVEC ;SETUP TAIL VECTOR
670 002574 012777 002632 176426 MOV @XWFG!INT.EN!GO, @TACS ;START A "WFG"
671 002602 112714 000101 CLR @#PS ;ALLOW INTERRUPTS
672 002606 005037 177776 3$: SUB #1, R0 ;START TIMING HOW LONG
673 002612 162700 000001 SBC R1 ; IT TAKES TO WRITE A FILE GAP
674 002616 005601 BNE 3$
675 002620 001374 MOV #340, @#PS ;LOCK OUT INTERRUPTS
676 002622 012737 000340 177776 BR FATAL ;IT TOOK TO LONG
677 002630 000432

```



```

676 002632 006201          4$:  ASR      R1          ;DIVIDE THE TIME BY 2
677 002634 006000          ROR      RO
678 002636 010037 001264    MOV      RO,@#STALL ;AND SAVE IT AS THE STALL TIME
679 002642 010137 001266    MOV      R1,@#STALL+2
680 002646 012777 012366 176354  MOV      #BADINT,@TAVEC ;SET TRAP CATCHER
681 002654 105737 001241    TSTB    @#DRVKEY+1 ;TWO DRIVES?
682 002660 001407          BEQ      5$          ;BR IF NO
683 002662 104422          SELDRV
684 002664 005077 176366    CLR      @RWDFLG    ;SELECT THE OTHER DRIVE
685 002670 112714 000016    MOVB    #XRWND,@TACS ;SET REWIND FLAG
686 002674 104421          REWIND ;AVOID "CLEAR LEADER" ERROR
687 002676 104412          WFG      ;GO TO BOT
688 002700 005737 001246    5$:  TST      @#ASKKEY  ;GO TO OXIDE
689 002704 002034          BGE      TST1      ;GO START TESTING IF NO MANUAL
690 002706 005000          CLR      RO        ;OPERATIONS REQUESTED
691 002710 006137 012022    ROL      @#MAXCNT
692 002714 000000          HALT     ;INCREASE THE WAIT TIME
693                                     ;GIVE CONTROL TO THE OPERATOR
694 002716                                     FATAL:
695 002716 104401 002724    TYPE    ,65$      ;;TYPE ASCIZ STRING
696 002722 000414          BR      64$      ;;GET OVER THE ASCIZ
697                                     ;;65$: .ASCIZ <15><12>*FATAL ERROR ON DRIVE *
698                                     64$:
699 002754 104401 001250    TYPE    ,CURDRV  ;CURRENT DRIVE
700 002760 013700 000042    MOV      @#42,RO ;CHECK FOR A MONITOR
701 002764 001402          BEQ      1$      ;BR IF NONE
702 002766 000137 006442    JMP      @#SENDAD ;LEAVE
703 002772 000000          1$:  HALT     ;ERROR OCCURRED BEFORE TESTING STARTED
704 002774 000776          BR      1$      ;HANGUP
    
```

H03

TA11 MOTION TEST  
DZTADC.NEW

24-MAY-76

MAINDEC-11-DZTAD-C  
00:00 T1

MACY11 27(1006) 23-SEP-76 10:33 PAGE 17  
REWIND,WRITE,REWIND,READ

SEQ 0033

```

705
706
707
708 002776 000004
709 003000 012767 003032 176164
710 003006 104422
711 003010 104421
712 003012 104413 017022
713 003016 104421
714 003020 104414 016766
715 003024 104423
716 003026 016766
717 003030 017022
718
719
720
721 003032 000004
722 003034 012767 003066 176130
723 003042 104422
724 003044 104421
725 003046 104413 017022
726 003052 104416
727 003054 104414 016766
728 003060 104423
729 003062 016766
730 003064 017022
731
732
733
734 003066 000004
735 003070 012767 003122 176074
736 003076 104422
737 003100 104421
738 003102 104413 017022
739 003106 104415
740 003110 104414 016766
741 003114 104423
742 003116 016766
743 003120 017022
744
745
746
747 003122 000004
748 003124 012767 003172 176040
749 003132 104422
750 003134 104421
751 003136 104413 017022
752 003142 104421
753 003144 104414 016766
754 003150 104423
755 003152 016766
756 003154 017022
757 003156 104416
758 003160 104414 016766
759 003164 104423
760 003166 016766

```

```

*****
*TEST 1      REWIND,WRITE,REWIND,READ
*****
TST1:  SCOPE
      MOV      #TST2,$ESCAPE  ;;ESCAPE TO TEST 2 ON ERROR
      SELDRV   ;SELECT DRIVE FOR TESTING
      REWIND   ;GO TO "CLEAR LEADER"
      WRITE    ,WLNK1        ;WRITE ON TAPE
      REWIND   ;GO TO BEGINNING OF TAPE
      READ     ,RLNK1        ;READ
      BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
      RLNK1   ;LINK TO THE READ BUFFER
      WLNK1   ;LINK TO THE WRITE BUFFER
*****
*TEST 2      TEST BSBG AFTER WRITE AND READ AFTER BSBG
*****
TST2:  SCOPE
      MOV      #TST3,$ESCAPE  ;;ESCAPE TO TEST 3 ON ERROR
      SELDRV   ;SELECT DRIVE FOR TESTING
      REWIND   ;GO TO "BOT"
      WRITE    ,WLNK1        ;WRITE ONE BLOCK
      BSBG     ;BACK OVER THE BLOCK
      READ     ,RLNK1        ;NOW READ
      BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
      RLNK1   ;LINK TO THE READ BUFFER
      WLNK1   ;LINK TO THE WRITE BUFFER
*****
*TEST 3      TEST BSFG AFTER WRITE AND READ AFTER BSFG
*****
TST3:  SCOPE
      MOV      #TST4,$ESCAPE  ;;ESCAPE TO TEST 4 ON ERROR
      SELDRV   ;SELECT DRIVE FOR TESTING
      REWIND   ;GO TO "BOT"
      WRITE    ,WLNK1        ;WRITE ONE BLOCK
      BSFG     ;BACK OVER THE BLOCK
      READ     ,RLNK1        ;NOW READ
      BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
      RLNK1   ;LINK TO THE READ BUFFER
      WLNK1   ;LINK TO THE WRITE BUFFER
*****
*TEST 4      TEST BSBG AFTER READ
*****
TST4:  SCOPE
      MOV      #TST5,$ESCAPE  ;;ESCAPE TO TEST 5 ON ERROR
      SELDRV   ;SELECT DRIVE FOR TESTING
      REWIND   ;GO TO "BOT"
      WRITE    ,WLNK1        ;WRITE ONE BLOCK
      REWIND   ;GO TO "BOT"
      READ     ,RLNK1        ;READ
      BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
      RLNK1   ;LINK TO THE READ BUFFER
      WLNK1   ;LINK TO THE WRITE BUFFER
      BSBG     ;BACK UP
      READ     ,RLNK1        ;READ
      BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
      RLNK1   ;LINK TO THE READ BUFFER

```

```

761 003170 017022          WLNK1          ;LINK TO THE WRITE BUFFER
762          ;*****
763          ;*TEST 5      TEST BSBG AFTER READ
764          ;*****
765 003172 000004          TST5:  SCOPE
766 003174 012767 003242 175770  MOV      #TST6,$ESCAPE  ;;ESCAPE TO TEST 6 ON ERROR
767 003202 104422          SELDRV   ;SELECT DRIVE FOR TESTING
768 003204 104421          REWIND   ;GO TO "BOT"
769 003206 104413 017022  WRITE    ,WLNK1      ;WRITE A BLOCK
770 003212 104421          REWIND   ;GO TO "BOT"
771 003214 104414 016766  READ     ,RLNK1      ;READ A BLOCK
772 003220 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
773 003222 016766          RLNK1    ;LINK TO THE READ BUFFER
774 003224 017022          WLNK1    ;LINK TO THE WRITE BUFFER
775 003226 104415          BSFG     ;BACK OVER THE DATA
776 003230 104414 016766  READ     ,RLNK1      ;NOW READ THE DATA
777 003234 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
778 003236 016766          RLNK1    ;LINK TO THE READ BUFFER
779 003240 017022          WLNK1    ;LINK TO THE WRITE BUFFER
780          ;*****
781          ;*TEST 6      TEST SFBG AFTER REWIND AND BSFG AFTER SFBG
782          ;*****
783 003242 000004          TST6:  SCOPE
784 003244 012767 003302 175720  MOV      #TST7,$ESCAPE  ;;ESCAPE TO TEST 7 ON ERROR
785 003252 104422          SELDRV   ;SELECT DRIVE FOR TESTING
786 003254 104421          REWIND   ;GO TO "BOT"
787 003256 104413 017022  WRITE    ,WLNK1      ;WRITE ONE BLOCK OF DATA
788 003262 104421          REWIND   ;GO TO "BOT"
789 003264 104420          SFBG     ;SPACE OVER THE DATA
790 003266 104415          BSFG     ;BACK OVER THE DATA
791 003270 104414 016766  READ     ,RLNK1      ;READ THE BLOCK
792 003274 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
793 003276 016766          RLNK1    ;LINK TO THE READ BUFFER
794 003300 017022          WLNK1    ;LINK TO THE WRITE BUFFER
795          ;*****
796          ;*TEST 7      TEST WRITE AFTER READ
797          ;*****
798 003302 000004          TST7:  SCOPE
799 003304 012767 003356 175660  MOV      #TST10,$ESCAPE ;;ESCAPE TO TEST 10 ON ERROR
800 003312 104422          SELDRV   ;SELECT DRIVE FOR TESTING
801 003314 104421          REWIND   ;GO TO "CLEAR LEADER"
802 003316 104413 017022  WRITE    ,WLNK1      ;WRITE ONE BLOCK
803 003322 104416          BSBG     ;BACK OVER THE DATA BLOCK
804 003324 104414 016766  READ     ,RLNK1      ;AND READ IT
805 003330 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
806 003332 016766          RLNK1    ;LINK TO THE READ BUFFER
807 003334 017022          WLNK1    ;LINK TO THE WRITE BUFFER
808 003336 104413 017026  WRITE    ,WLNK2      ;WRITE A SECOND BLOCK
809 003342 104416          BSBG     ;BACK OVER IT
810 003344 104414 016772  READ     ,RLNK2      ;AND READ IT
811 003350 104423          BLKCMP   ;COMPARE THE READ AND WRITE BUFFERS
812 003352 016772          RLNK2    ;LINK TO THE READ BUFFER
813 003354 017026          WLNK2    ;LINK TO THE WRITE BUFFER
814          ;*****
815          ;*TEST 10     TEST WRITE AFTER WRITE AND READ AFTER READ
816          ;*****

```

TAII MOTION TEST  
DZTADC.NEW 24-MAY-76

MAINDEC-11-DZTAD-C  
00:00 T10

MACY11 27(1006) 23-SEP-76 10:33 PAGE 19  
TEST WRITE AFTER WRITE AND READ AFTER READ

SEQ 0035

```

817 003356 000004 TST10: SCOPE
818 003360 012767 003422 175604 MOV #TST11,$ESCAPE ;;ESCAPE TO TEST 11 ON ERROR
819 003366 104422 SELDRV ;SELECT DRIVE FOR TESTING
820 003370 104421 REWIND ;GO TO "CLEAR LEADER"
821 003372 104413 017022 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
822 003376 104413 017026 WRITE ,WLNK2
823 003402 104421 REWIND ;GO TO BOT
824 003404 104414 016766 READ ,RLNK1 ;READ BOTH BLOCKS
825 003410 104414 016772 READ ,RLNK2
826 003414 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
827 003416 016772 RLNK2 ;LINK TO THE READ BUFFER
828 003420 017026 WLNK2 ;LINK TO THE WRITE BUFFER
829 *****
830 ;*TEST 11 TEST BSBG AFTER BSBG
831 *****
832 003422 000004 TST11: SCOPE
833 003424 012767 003464 175540 MOV #TST12,$ESCAPE ;;ESCAPE TO TEST 12 ON ERROR
834 003432 104422 SELDRV ;SELECT DRIVE FOR TESTING
835 003434 104421 REWIND ;GO TO "CLEAR LEADER"
836 003436 104413 017022 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
837 003442 104413 017026 WRITE ,WLNK2
838 003446 104416 BSBG ;BACK OVER THE 2ND BLOCK
839 003450 104416 BSBG ;BACK OVER THE 1ST BLOCK
840 003452 104414 016766 READ ,RLNK1 ;READ THE 1ST BLOCK
841 003456 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
842 003460 016766 RLNK1 ;LINK TO THE READ BUFFER
843 003462 017022 WLNK1 ;LINK TO THE WRITE BUFFER
844 *****
845 ;*TEST 12 TEST READ AFTER SFBG
846 *****
847 003464 000004 TST12: SCOPE
848 003466 012767 003526 175476 MOV #TST13,$ESCAPE ;;ESCAPE TO TEST 13 ON ERROR
849 003474 104422 SELDRV ;SELECT DRIVE FOR TESTING
850 003476 104421 REWIND ;GO TO "CLEAR LEADER"
851 003500 104413 017022 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
852 003504 104413 017026 WRITE ,WLNK2
853 003510 104421 REWIND ;BOT
854 003512 104420 SFBG ;SPACE OVER THE 1ST BLOCK
855 003514 104414 016772 READ ,RLNK2 ;READ THE 2ND BLOCK
856 003520 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
857 003522 016772 RLNK2 ;LINK TO THE READ BUFFER
858 003524 017026 WLNK2 ;LINK TO THE WRITE BUFFER
859 *****
860 ;*TEST 13 TEST SFBG AFTER BSBG
861 *****
862 003526 000004 TST13: SCOPE
863 003530 012767 003572 175434 MOV #TST14,$ESCAPE ;;ESCAPE TO TEST 14 ON ERROR
864 003536 104422 SELDRV ;SELECT DRIVE FOR TESTING
865 003540 104421 REWIND ;GO TO "CLEAR LEADER"
866 003542 104413 017022 WRITE ,WLNK1 ;WRITE TWO(2) BLOCKS
867 003546 104413 017026 WRITE ,WLNK2
868 003552 104416 BSBG ;BACK SPACE OVER BOTH BLOCKS
869 003554 104416 BSBG
870 003556 104420 SFBG ;SPACE FWD OVER 1ST BLOCK
871 003560 104414 016772 READ ,RLNK2 ;READ 2ND BLOCK
872 003564 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS

```

K03

TA11 MOTION TEST  
DZTADC.NEW

24-MAY-76

MAINDEC-11-DZTAD-C  
00:00 T13

MACY11 27(1006) 23-SEP-76 10:33 PAGE 20  
TEST SFBG AFTER BSBG

SEQ 0036

```

873 003566 016772          RLNK2          ;LINK TO THE READ BUFFER
874 003570 017026          WLNK2          ;LINK TO THE WRITE BUFFER
875                                     ;*****
876                                     ;*TEST 14      TEST SFBG AFTER BSBG
877                                     ;*****
878 003572 000004          TST14: SCOPE
879 003574 012767 003634 175370      MOV          #TST15,$ESCAPE ;;ESCAPE TO TEST 15 ON ERROR
880 003602 104422          SELDRV       ;SELECT DRIVE FOR TESTING
881 003604 104421          REWIND       ;GO TO "CLEAR LEADER"
882 003606 104413 017022          WRITE       ,WLNK1      ;WRITE TWO BLOCKS
883 003612 104413 017026          WRITE       ,WLNK2
884 003616 104415          BSBG
885 003620 104420          SFBG        ;SPACE REV FILE
886 003622 104414 016772          READ        ,RLNK2     ;SPACE FWD BLOCK
887 003626 104423          BLKCMP      ;READ THE 2ND BLOCK
888 003630 016772          RLNK2       ;COMPARE THE READ AND WRITE BUFFERS
889 003632 017026          WLNK2       ;LINK TO THE READ BUFFER
890                                     ;LINK TO THE WRITE BUFFER
891                                     ;*****
892                                     ;*TEST 15-    REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER WRITE
893                                     ;*****
893 003634 000004          TST15: SCOPE
894 003636 012767 004026 175326      MOV          #TST16,$ESCAPE ;;ESCAPE TO TEST 16 ON ERROR
895 003644 104422          SELDRV       ;SELECT DRIVE FOR TESTING
896 003646 104421          REWIND       ;GO TO "CLEAR LEADER"
897 003650 104413 017022          WRITE       ,WLNK1      ;WRITE 2 BLOCKS
898 003654 104413 017026          WRITE       ,WLNK2
899 003660 104416          BSBG        ;BACK OVER BOTH OF THEM
900 003662 104416          BSBG
901 003664 104414 016766          READ        ,RLNK1     ;READ THE FIRST ONE
902 003670 104423          BLKCMP      ;COMPARE THE READ AND WRITE BUFFERS
903 003672 016766          RLNK1       ;LINK TO THE READ BUFFER
904 003674 017022          WLNK1       ;LINK TO THE WRITE BUFFER
905 003676 104414 016772          READ        ,RLNK2     ;NOW READ THE 2ND ONE
906 003702 104423          BLKCMP      ;COMPARE THE READ AND WRITE BUFFERS
907 003704 016772          RLNK2       ;LINK TO THE READ BUFFER
908 003706 017026          WLNK2       ;LINK TO THE WRITE BUFFER
909 003710 104416          BSBG        ;BACK OVER THE 2ND ONE
910 003712 104413 017032          WRITE       ,WLNK3     ;WRITE OVER THE 2ND BLOCK
911 003716 104416          BSBG        ;NOW BACK UP AND
912 003720 104414 016776          READ        ,RLNK3     ;READ IT
913 003724 104423          BLKCMP      ;COMPARE THE READ AND WRITE BUFFERS
914 003726 016776          RLNK3       ;LINK TO THE READ BUFFER
915 003730 017032          WLNK3       ;LINK TO THE WRITE BUFFER
916 003732 104416          BSBG        ;BACK OVER IT AGAIN
917 003734 104413 017036          WRITE       ,WLNK4     ;AND WRITE OVER IT AGAIN
918 003740 104416          BSBG        ;BACK UP AND
919 003742 104414 017002          READ        ,RLNK4     ;READ THE BLOCK
920 003746 104423          BLKCMP      ;COMPARE THE READ AND WRITE BUFFERS
921 003750 017002          RLNK4       ;LINK TO THE READ BUFFER
922 003752 017036          WLNK4       ;LINK TO THE WRITE BUFFER
923 003754 104416          BSBG        ;BACK OVER THE BLOCK AGAIN
924 003756 104413 017026          WRITE       ,WLNK2     ;AND WRITE OVER IT AGAIN
925 003762 104416          BSBG        ;NOW BACK UP
926 003764 104414 016772          READ        ,RLNK2     ;AND READ IT
927 003770 104423          BLKCMP      ;COMPARE THE READ AND WRITE BUFFERS
928 003772 016772          RLNK2       ;LINK TO THE READ BUFFER

```

```

929 003774 017026      WLNK2                ;LINK TO THE WRITE BUFFER
930 003776 104416      BSBG                 ;BACK OVER BOTH BLOCKS
931 004000 104416      BSBG
932 004002 104414 016766 READ      ,RLNK1      ;AND READ THE 1ST ONE
933 004006 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
934 004010 016766      RLNK1              ;LINK TO THE READ BUFFER
935 004012 017022      WLNK1              ;LINK TO THE WRITE BUFFER
936 004014 104414 016772 READ      ,RLNK2      ;AND THE 2ND ONE
937 004020 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
938 004022 016772      RLNK2              ;LINK TO THE READ BUFFER
939 004024 017026      WLNK2              ;LINK TO THE WRITE BUFFER
940
941 *****
942 *TEST 16 REWRITE THE LAST BLOCK 3 TIMES AFTER DOING A WRITE AFTER READ
943 *****
944 004026 000004      TST16: SCOPE
945 004030 012767 004204 175134 MOV      #TST17,$ESCAPE ;;ESCAPE TO TEST 17 ON ERROR
946 004036 104422      SELDRV             ;SELECT DRIVE FOR TESTING
947 004040 104421      REWIND            ;GO TO "CLEAR LEADER"
948 004042 104413 017022 WRITE     ,WLNK1      ;WRITE A BLOCK OF DATA
949 004046 104416      BSBG              ;BACK OVER IT
950 004050 104414 016766 READ      ,RLNK1      ;AND READ IT
951 004054 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
952 004056 016766      RLNK1              ;LINK TO THE READ BUFFER
953 004060 017022      WLNK1              ;LINK TO THE WRITE BUFFER
954 004062 104413 017026 WRITE     ,WLNK2      ;WRITE 2ND BLOCK
955 004066 104416      BSBG              ;BACK OVER IT
956 004070 104414 016772 READ      ,RLNK2      ;AND READ IT
957 004074 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
958 004076 016772      RLNK2              ;LINK TO THE READ BUFFER
959 004100 017026      WLNK2              ;LINK TO THE WRITE BUFFER
960 004102 104416      BSBG              ;BACK OVER IT AND
961 004104 104413 017036 WRITE     ,WLNK4      ;WRITE OVER 2ND BLOCK
962 004110 104416      BSBG              ;BACK UP AND
963 004112 104414 017002 READ      ,RLNK4      ;READ IT
964 004116 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
965 004120 017002      RLNK4              ;LINK TO THE READ BUFFER
966 004122 017036      WLNK4              ;LINK TO THE WRITE BUFFER
967 004124 104416      BSBG              ;BACK OVER IT AGAIN
968 004126 104413 017042 WRITE     ,WLNK5      ;REWRITE 2ND BLOCK
969 004132 104416      BSBG
970 004134 104414 017006 READ      ,RLNK5      ;READ 2ND BLOCK
971 004140 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
972 004142 017006      RLNK5              ;LINK TO THE READ BUFFER
973 004144 017042      WLNK5              ;LINK TO THE WRITE BUFFER
974 004146 104416      BSBG              ;BACK OVER IT AGAIN
975 004150 104413 017032 WRITE     ,WLNK3      ;WRITE ANOTHER BLOCK OVER IT
976 004154 104416      BSBG              ;BACK OVER 2 BLOCKS
977 004156 104416      BSBG
978 004160 104414 016766 READ      ,RLNK1      ;READ THE 1ST BLOCK
979 004164 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
980 004166 016766      RLNK1              ;LINK TO THE READ BUFFER
981 004170 017022      WLNK1              ;LINK TO THE WRITE BUFFER
982 004172 104414 016776 READ      ,RLNK3      ;READ 2ND BLOCK
983 004176 104423      BLKCMP              ;COMPARE THE READ AND WRITE BUFFERS
984 004200 016776      RLNK3              ;LINK TO THE READ BUFFER
984 004202 017032      WLNK3              ;LINK TO THE WRITE BUFFER

```

M03

TAII MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00 T17

MACY11 27(1006) 23-SEP-76 10:33 PAGE 22  
TEST WFG AFTER WRITE AND BSFG AFTER WFG

SEQ 0038

```

985
986
987
988 004204 000004
989 004206 012767 004242 174756
990 004214 104422
991 004216 104421
992 004220 104413 017022
993 004224 104412
994 004226 104415
995 004230 104414 016766
996 004234 104423
997 004236 016766
998 004240 017022
999
1000
1001
1002 004242 000004
1003 004244 012767 004300 174720
1004 004252 104422
1005 004254 104421
1006 004256 104413 017022
1007 004262 104412
1008 004264 104416
1009 004266 104414 016766
1010 004272 104423
1011 004274 016766
1012 004276 017022
1013
1014
1015
1016 004300 000004
1017 004302 012767 004356 174662
1018 004310 104422
1019 004312 104421
1020 004314 104413 017022
1021 004320 104421
1022 004322 104414 016766
1023 004326 104423
1024 004330 016766
1025 004332 017022
1026 004334 104412
1027 004336 104413 017026
1028 004342 104416
1029 004344 104414 016772
1030 004350 104423
1031 004352 016772
1032 004354 017026
1033
1034
1035
1036 004356 000004
1037 004360 012767 004434 174604
1038 004366 104422
1039 004370 104421
1040 004372 104413 017022

```

```

*****
*TEST 17 TEST WFG AFTER WRITE AND BSFG AFTER WFG
*****
TST17: SCOPE
MOV #TST20,$ESCAPE ;;ESCAPE TO TEST 20 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK
WFG ;FOLLOWED BY A FILE GAP
BSFG ;BACK OVER THE DATA BLOCK
READ ,RLNK1 ;READ THE DATA
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 20 TEST BSBG AFTER WFG
*****
TST20: SCOPE
MOV #TST21,$ESCAPE ;;ESCAPE TO TEST 21 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO BOT
WRITE ,WLNK1 ;WRITE ONE BLOCK OF DATA
WFG ;WRITE A FILE GAP
BSBG ;BACK OVER THE DATA
READ ,RLNK1 ;READ ONE BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
*****
*TEST 21 TEST WRITE AFTER WFG
*****
TST21: SCOPE
MOV #TST22,$ESCAPE ;;ESCAPE TO TEST 22 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO BOT
WRITE ,WLNK1 ;WRITE ONE BOCK
REWIND ;GO TO BOT
READ ,RLNK1 ;READ ONE BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER
WFG ;WRITE A FILE GAP
WRITE ,WLNK2 ;WRITE ANOTHER BLOCK
BSBG ;BACK OVER THE LAST BLOCK
READ ,RLNK2 ;READ THE SECOND BLOCK
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK2 ;LINK TO THE READ BUFFER
WLNK2 ;LINK TO THE WRITE BUFFER
*****
*TEST 22 TEST WFG AFTER READ AND BSFG AFTER WRITE
*****
TST22: SCOPE
MOV #TST23,$ESCAPE ;;ESCAPE TO TEST 23 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE ONE BLOCK

```

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



11097	004600	104421		REWIND		:GO TO "BOT"
11098	004602	104417		SFFG		:SPACE OVER 1ST FILE
11099	004604	104420		SFBG		:SPACE OVER BLOCK
11100	004606	104416		BSBG		:BACK OVER THE BLOCK
11101	004610	104414	016772	READ	,RLNK2	:READ THE LAST BLOCK
11102	004614	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
11103	004616	016772		RLNK2		:LINK TO THE READ BUFFER
11104	004620	017026		WLNK2		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :TEST 26 TEST SFBG AFTER SFBG  
 \*\*\*\*\*

11105	004622	000004		†ST26: SCOPE		
11106	004624	012767	004702 174340	MOV	#TST27,\$ESCAPE	::ESCAPE TO TEST 27 ON ERROR
11107	004632	104422		SELDRV		:SELECT DRIVE FOR TESTING
11108	004634	104421		REWIND		:GO TO "BOT"
11109	004636	104413	017022	WRITE	,WLNK1	:WRITE A BLOCK OF DATA
11110	004642	104412		WFG		:WRITE A FILE GAP
11111	004644	104413	017026	WRITE	,WLNK2	:WRITE 3 BLOCKS OF DATA
11112	004650	104413	017032	WRITE	,WLNK3	
11113	004654	104413	017036	WRITE	,WLNK4	
11114	004660	104421		REWIND		:GO TO "BOT"
11115	004662	104417		SFFG		:SPACE OVER THE 1ST FILE
11116	004664	104420		SFBG		:SPACE OVER TWO BLOCK
11117	004666	104420		SFBG		
11118	004670	104414	017002	READ	,RLNK4	:READ LAST BLOCK OF DATA
11119	004674	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
11120	004676	017002		RLNK4		:LINK TO THE READ BUFFER
11121	004700	017036		WLNK4		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :TEST 27 TEST BSFG AFTER SFBG AND BSFG AFTER BSFG  
 \*\*\*\*\*

11122	004702	000004		†ST27: SCOPE		
11123	004704	012767	004754 174260	MOV	#TST30,\$ESCAPE	::ESCAPE TO TEST 30 ON ERROR
11124	004712	104422		SELDRV		:SELECT DRIVE FOR TESTING
11125	004714	104421		REWIND		:GO TO "BOT"
11126	004716	104413	017022	WRITE	,WLNK1	:WRITE ONE BLOCK OF DATA
11127	004722	104412		WFG		:WRITE A FILE GAP
11128	004724	104413	017026	WRITE	,WLNK2	:WRITE A BLOCK OF DATA
11129	004730	104421		REWIND		:GO TO "BOT"
11130	004732	104417		SFFG		:SPACE OVER 1ST FILE
11131	004734	104420		SFBG		:SPACE OVER 1ST BLOCK OF 2ND FILE
11132	004736	104415		BSFG		:BACK TO BEGINNING OF 2ND FILE
11133	004740	104415		BSFG		:BACK TO BEGINNING OF 1ST FILE
11134	004742	104414	016766	READ	,RLNK1	:READ 1ST BLOCK OF 1ST FILE
11135	004746	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
11136	004750	016766		RLNK1		:LINK TO THE READ BUFFER
11137	004752	017022		WLNK1		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :TEST 30 TEST BSFG THRU A FILE GAP  
 \*\*\*\*\*

11138	004754	000004		†ST30: SCOPE		
11139	004756	012767	005032 174206	MOV	#TST31,\$ESCAPE	::ESCAPE TO TEST 31 ON ERROR
11140	004764	104422		SELDRV		:SELECT DRIVE FOR TESTING
11141	004766	104421		REWIND		:GO TO "BOT"
11142	004770	104413	017022	WRITE	,WLNK1	:WRITE ONE BLOCK OF DATA
11143	004774	104412		WFG		:WRITE A FILE GAP

1153	004776	104413	017026	WRITE	,WLNK2	:WRITE TWO BLOCKS OF DATA
1154	005002	104413	017032	WRITE	,WLNK3	
1155	005006	104421		REWIND		:GO TO "BOT"
1156	005010	104417		SFFG		:SPACE OVER 1ST FILE
1157	005012	104420		SFBG		:SPACE OVER 1ST BLOCK OF 2ND FILE
1158	005014	104416		BSBG		:BACK OVER 1 BLOCK
1159	005016	104416		BSBG		:BACK OVER 1 BLOCK
1160	005020	104414	016766	READ	,RLNK1	:READ
1161	005024	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
1162	005026	016766		RLNK1		:LINK TO THE READ BUFFER
1163	005030	017022		WLNK1		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :\*TEST 31 TEST BSBG AFTER BSBG  
 \*\*\*\*\*

1167	005032	000004		TST31: SCOPE		
1168	005034	012767	005104 174130	MOV	#TST32,\$ESCAPE	::ESCAPE TO TEST 32 ON ERROR
1169	005042	104422		SELDRV		:SELECT DRIVE FOR TESTING
1170	005044	104421		REWIND		:GO TO "BOT"
1171	005046	104413	017022	WRITE	,WLNK1	:WRITE ONE BLOCK OF DATA
1172	005052	104412		WFG		:WRITE A FILE GAP
1173	005054	104413	017026	WRITE	,WLNK2	:WRITE A BLOCK OF DATA
1174	005060	104421		REWIND		:GO TO "BOT"
1175	005062	104417		SFFG		:SPACE OVER FILE
1176	005064	104420		SFBG		:SPACE OVER 1ST BLOCK OF 2ND FILE
1177	005066	104416		BSBG		:BACK OVER BLOCK
1178	005070	104415		BSFG		:BACK OVER FILE
1179	005072	104414	016766	READ	,RLNK1	:READ 1ST BLOCK ON TAPE
1180	005076	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
1181	005100	016766		RLNK1		:LINK TO THE READ BUFFER
1182	005102	017022		WLNK1		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :\*TEST 32 TEST BSBG AFTER BSFG  
 \*\*\*\*\*

1186	005104	000004		TST32: SCOPE		
1187	005106	012767	005150 174056	MOV	#TST33,\$ESCAPE	::ESCAPE TO TEST 33 ON ERROR
1188	005114	104422		SELDRV		:SELECT DRIVE FOR TESTING
1189	005116	104421		REWIND		:GO TO "CLEAR LEADER"
1190	005120	104413	017022	WRITE	,WLNK1	:WRITE BLOCK
1191	005124	104412		WFG		:WRITE A FILE GAP
1192	005126	104413	017026	WRITE	,WLNK2	:FOLLOW WITH ANOTHER BLOCK
1193	005132	104415		BSFG		:BACK UP 1 FILE
1194	005134	104416		BSBG		:BACK UP 1 BLOCK
1195	005136	104414	016766	READ	,RLNK1	:READ 1ST BLOCK ON TAPE
1196	005142	104423		BLKCMP		:COMPARE THE READ AND WRITE BUFFERS
1197	005144	016766		RLNK1		:LINK TO THE READ BUFFER
1198	005146	017022		WLNK1		:LINK TO THE WRITE BUFFER

\*\*\*\*\*  
 :\*TEST 33 TEST BSFG AFTER SFFG AND SFFG AFTER READ  
 \*\*\*\*\*

1202	005150	000004		TST33: SCOPE		
1203	005152	012767	005232 174012	MOV	#TST34,\$ESCAPE	::ESCAPE TO TEST 34 ON ERROR
1204	005160	104422		SELDRV		:SELECT DRIVE FOR TESTING
1205	005162	104421		REWIND		:GO TO "BOT"
1206	005164	104413	017022	WRITE	,WLNK1	:WRITE A BLOCK OF DATA
1207	005170	104412		WFG		:WRITE A FILE GAP
1208	005172	104413	017026	WRITE	,WLNK2	:WRITE A BLOCK OF DATA

```

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

```

```

1265 005374 104422 SELDRV ;SELECT DRIVE FOR TESTING
1266 005376 104421 REWIND
1267 005400 104413 017022 WRITE ,WLNK1 ;WRITE 2 BLOCKS
1268 005404 104413 017026 WRITE ,WLNK2
1269 005410 104421 REWIND ;GO TO "BOT"
1270 005412 104420 SFBG ;GET OVER THE 1ST BLOCK
1271 005414 104412 WFG ;WRITE A GAP OVER 2ND BLOCK
1272 005416 104413 017032 WRITE ,WLNK3 ;WRITE A BLOCK
1273 005422 104415 BSFG ;BACK OVER THE BLOCK
1274 005424 104414 016776 READ ,RLNK3 ;AND READ IT
1275 005430 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1276 005432 016776 RLNK3 ;LINK TO THE READ BUFFER
1277 005434 017032 WLNK3 ;LINK TO THE WRITE BUFFER
1278
1279 ::*****
1280 *TEST 37 TEST WRITE AFTER SFBG
1281 005436 000004 †TST37: SCOPE
1282 005440 012767 005510 173524 MOV #TST40,$ESCAPE ;;ESCAPE TO TEST 40 ON ERROR
1283 005446 104422 SELDRV ;SELECT DRIVE FOR TESTING
1284 005450 104421 REWIND ;GO TO BOT
1285 005452 104413 017022 WRITE ,WLNK1 ;WRITE 2 BLOCK
1286 005456 104413 017026 WRITE ,WLNK2
1287 005462 104421 REWIND ;GO TO BOT
1288 005464 104420 SFBG ;SPACE OVER 1ST BLOCK
1289 005466 104413 017032 WRITE ,WLNK3 ;WRITE OVER 2ND BLOCK
1290 005472 104415 BSFG ;GO TO BEGINNING OF FILE
1291 005474 104420 SFBG ;SPACE OVER 1ST BLOCK
1292 005476 104414 016776 READ ,RLNK3 ;READ THE NEW 2ND BLOCK
1293 005502 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1294 005504 016776 RLNK3 ;LINK TO THE READ BUFFER
1295 005506 017032 WLNK3 ;LINK TO THE WRITE BUFFER
1296
1297 ::*****
1298 *TEST 40 3 ONE BLOCK FILES---MULTI SPACING & TEST SFFG AFTER BSFG
1299 005510 000004 †TST40: SCOPE
1300 005512 012767 005600 173452 MOV #TST41,$ESCAPE ;;ESCAPE TO TEST 41 ON ERROR
1301 005520 104422 SELDRV ;SELECT DRIVE FOR TESTING
1302 005522 104421 REWIND ;GO TO "BOT"
1303 005524 104413 017022 WRITE ,WLNK1 ;WRITE A BLOCK
1304 005530 104412 WFG ;FILE GAP
1305 005532 104413 017026 WRITE ,WLNK2 ;BLOCK
1306 005536 104412 WFG ;FILE GAP
1307 005540 104413 017032 WRITE ,WLNK3 ;BLOCK
1308 005544 104412 WFG ;FILE GAP
1309 005546 104421 REWIND ;GO TO "BOT"
1310 005550 104417 SFFG ;SPACE OVER 1ST FILE
1311 005552 104417 SFFG ;SPACE OVER 2ND FILE
1312 005554 104417 SFFG ;SPACE OVER 3RD FILE
1313 005556 104415 BSFG ;BACK OVER 3RD FILE
1314 005560 104415 BSFG ;BACK OVER 2ND FILE
1315 005562 104415 BSFG ;BACK OVER 1ST FILE
1316 005564 104417 SFFG ;GO TO BEGINNING OF 2ND FILE
1317 005566 104414 016772 READ ,RLNK2 ;READ THE BLOCK
1318 005572 104423 BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
1319 005574 016772 RLNK2 ;LINK TO THE READ BUFFER
1320 005576 017026 WLNK2 ;LINK TO THE WRITE BUFFER

```

F04

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00 T41

MACY11 27(1006) 23-SEP-76 10:33 PAGE 28  
TWO BLOCK FILES---MULTI SPACING FUNCTIONS

SEQ 0044

```

1321
1322
1323
1324 005600 000004
1325 005602 012767 005706 173362
1326 005610 104422
1327 005612 104421
1328 005614 104413 017022
1329 005620 104413 017026
1330 005624 104412
1331 005626 104413 017032
1332 005632 104413 017036
1333 005636 104412
1334 005640 104413 017042
1335 005644 104413 017046
1336 005650 104412
1337 005652 104421
1338 005654 104417
1339 005656 104417
1340 005660 104414 017006
1341 005664 104423
1342 005666 017006
1343 005670 017042
1344 005672 104415
1345 005674 104414 017006
1346 005700 104423
1347 005702 017006
1348 005704 017042
1349
1350
1351
1352 005706 000004
1353 005710 012767 006056 173254
1354 005716 104422
1355 005720 104421
1356 005722 104413 017022
1357 005726 104413 017026
1358 005732 104413 017032
1359 005736 104412
1360 005740 104413 017036
1361 005744 104413 017042
1362 005750 104412
1363 005752 104421
1364 005754 104417
1365 005756 104417
1366 005760 104415
1367 005762 104414 017002
1368 005766 104423
1369 005770 017002
1370 005772 017036
1371 005774 104415
1372 005776 104415
1373 006000 104414 016766
1374 006004 104423
1375 006006 016766
1376 006010 017022

*****
*TEST 41 TWO BLOCK FILES---MULTI SPACING FUNCTIONS
*****
TST41: SCOPE
MOV #TST42,$ESCAPE ;;ESCAPE TO TEST 42 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE A 2 BLOCK FILE
WRITE ,WLNK2
WFG
WRITE ,WLNK3 ;WRITE A 2 BLOCK FILE
WRITE ,WLNK4
WFG
WRITE ,WLNK5 ;WRITE A 2 BLOCK FILE
WRITE ,WLNK6
WFG
REWIND ;GO TO "BOT"
SFFG ;SPACE OVER 1ST FILE
SFFG ;SPACE OVER 2ND FILE
READ ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK5 ;LINK TO THE READ BUFFER
WLNK5 ;LINK TO THE WRITE BUFFER
BSFG ;BACK TO BEGINNING OF 3RD FILE
READ ,RLNK5 ;READ 1ST BLOCK OF 3RD FILE
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK5 ;LINK TO THE READ BUFFER
WLNK5 ;LINK TO THE WRITE BUFFER
*****
*TEST 42 MULTI SPACING & TEST SFBG AFTER READ
*****
TST42: SCOPE
MOV #TST42,$ESCAPE ;;ESCAPE TO TEST 43 ON ERROR
SELDRV ;SELECT DRIVE FOR TESTING
REWIND ;GO TO "BOT"
WRITE ,WLNK1 ;WRITE A 3 BLOCK FILE
WRITE ,WLNK2
WRITE ,WLNK3
WFG
WRITE ,WLNK4 ;WRITE A 2 BLOCK FILE
WRITE ,WLNK5
WFG
REWIND ;GO TO "BOT"
SFFG ;SPACE OVER THE 1ST FILE
SFFG ;SPACE OVER THE 2ND FILE
BSFG ;BACK TO BEGINNING OF 2 FILE
READ ,RLNK4 ;READ 1ST BLOCK OF 2ND FILE
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK4 ;LINK TO THE READ BUFFER
WLNK4 ;LINK TO THE WRITE BUFFER
BSFG ;BACK TO BEGINNING OF 2ND FILE
BSFG ;BACK OVER 1ST FILE
READ ,RLNK1 ;READ 1ND BLOCK OF 1ST FILE
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNK1 ;LINK TO THE READ BUFFER
WLNK1 ;LINK TO THE WRITE BUFFER

```

```

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

```

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

```

1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486 006366
1487 006366 000004
1488 006370 005067 172506
1489 006374 005067 172570
1490 006400 005267 172474
1491 006404 042767 100000 172466
1492 006412 005327
1493 006414 000001
1494 006416 003015
1495 006420 012737
1496 006422 000001
1497 006424 006414
1498 006426 104401 006461
1499 006432 013700 000042
1500 006436 001405
1501 006440 000005
1502 006442 004710
1503 006444 000240
1504 006446 000240
1505 006450 000240
1506 006452
1507 006452 000137
1508 006454 002326
1509 006456 377 377 000
1510 006461 015 042412 042116
1511 006466 050040 051501 000123

```

```

.SBTTL END OF PASS ROUTINE
:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE "END PASS"
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO START
:*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
:*SENDMG CAN BE CHANGED TO 7.

$EOP:
SCOPE
CLR $STNM ;; ZERO THE TEST NUMBER
CLR $TIMES ;; ZERO THE NUMBER OF ITERATIONS
INC $PASS ;; INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;; LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,2(PC)+ ;; RESTORE COUNTER

$ENDCT: .WORD 1
TYPE $SENDMG ;; TYPE "END PASS"
MOV 2#42,R0 ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
RESET ;; CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11

$DOAGN: JMP 2(PC)+ ;; RETURN

$RTNAD: .WORD START
$ENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12>/END PASS/

```



```

1512 ;*****
1513 ;
1514 ;SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
1515 ;
1516 ;CALL:
1517 ;             MOV      #DRVKEY,RO
1518 ;             JSR      PC,@#EXAM          ;R1 IS DESTROYED
1519 ;             NORMAL RETURN
1520 ;             ERROR RETURN
1521 ;
1522 006474 013701 001220 EXAM: MOV      @#TACSL,R1          ;PICKUP THE "CONTROL & STATUS" REG. ADR.
1523 006500 005011          CLR      (R1)          ;DRIVE="A" FUNCTION="WFG"
1524 006502 122710 000101 CMPB    #'A,(R0)        ;EXAMINE DRIVE "A"?
1525 006506 001402          BEQ      1$          ;BR IF YES
1526 006510 052711 000400 1$: BIS      #UNIT,(R1)      ;SELECT DRIVE "B"
1527 006514 032711 000040 BIT      #READY,(R1)    ;WAIT ON READY
1528 006520 001775          BEQ      1$
1529 006522 005711          TST      (R1)          ;ANY ERROR?
1530 006524 100024          BPL      4$          ;BR IF NO
1531 006526 032711 001000 BIT      #OFFLINE,(R1)  ;ERROR DUE TO "OFF LINE"?
1532 006532 001017          BNE      3$          ;BR IF YES
1533 006534 032711 010000 BIT      #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
1534 006540 001411          BEQ      2$          ;BR IF NO
1535 006542 122777 000201 172370 CMPB    #BIT07!BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
1536 006550 001412          BEQ      4$          ;BR IF YES
1537 006552 122777 000203 172360 CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
1538 006560 001406          BEQ      4$          ;BR IF YES
1539 006562 000403          BR       3$          ;TAKE THE ERROR EXIT
1540 006564 032711 020000 2$: BIT      #LEADER,(R1)  ;ERROR DUE TO "CLEAR LEADER"?
1541 006570 001002          BNE      4$          ;BR IF YES
1542 006572 062716 000002 3$: ADD      #2,(SP)        ;TAKE ERROR RETURN
1543 006576 000207          RTS      PC          ;RETURN

```

```

1544 .SBTTL SCOPE HANDLER ROUTINE
1545
1546 *****
1547 *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1548 *AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1549 *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1550 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1551 *SW14=1 LOOP ON TEST
1552 *SW11=1 INHIBIT ITERATIONS
1553 *SW09=1 LOOP ON ERROR
1554 *CALL
1555 * SCOPE ;:SCOPE=IOT
1556
1557 $SCOPE:
1558 006600 104406 CKSNR ;:TEST FOR CHANGE IN SOFT-SWR
1559 006602 105777 172336 TSTB @STKS ;:TTY FLAG UP?
1560 006606 001405 BEQ 100$ ;:BR IF NO
1561 006610 104407 RDCHR ;:GO GET ONE CHARACTER FROM TTY
1562 006612 122627 000003 CMPB (SP)+,#3 ;:IS IT A CONTROL/C ?
1563 006616 001001 BNE 100$ ;:BR IF NO
1564 006620 104424 SETLOOP ;:GO GET "TEST" AND "SCOPE LOOP" ADDRESSES
1565 006622 005727 100$: TST (PC)+ ;:LOOP ON TEST?
1566 006624 000000 LOOPKEY: 0
1567 006626 001105 BNE $OVER ;:BR IF YES
1568 006630 032777 040000 172302 1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
1569 006636 001101 BNE $OVER ;:YES IF SW14=1
1570 *****START OF CODE FOR THE XOR TESTER*****
1571 006640 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
1572 ;:THIS INSTRUCTION TO A "NOP" (NOP=240)
1573 006642 013746 000004 MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
1574 006646 012737 006666 000004 MOV #5,@#ERRVEC ;:SET FOR TIMEOUT
1575 006654 005737 177060 TST @#177060 ;:TIME OUT ON XOR?
1576 006660 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
1577 006664 000453 BR $SVLAD ;:GO TO THE NEXT TEST
1578 006666 022626 5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
1579 006670 012637 000004 MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
1580 006674 000413 BR 7$ ;:LOOP ON THE PRESENT TEST
1581 006676 6$: ;:*****END OF CODE FOR THE XOR TESTER*****
1582 006676 105767 172201 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
1583 006702 001421 BEQ 3$ ;:BR IF NO
1584 006704 126767 172205 172171 CMPB $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
1585 006712 101015 BHI 3$ ;:BR IF NO
1586 006714 032777 001000 172216 BIT #BIT09,@SWR ;:LOOP ON ERROR?
1587 006722 001404 BEQ 4$ ;:BR IF NO
1588 006724 016767 172160 172154 7$: MOV $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
1589 006732 000443 BR $OVER
1590 006734 105067 172143 4$: CLRB $ERFLG ;:ZERO THE ERROR FLAG
1591 006740 005067 172224 CLR $TIMES ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
1592 006744 000415 BR 1$ ;:ESCAPE TO THE NEXT TEST
1593 006746 032777 004000 172164 3$: BIT #BIT11,@SWR ;:INHIBIT ITERATIONS?
1594 006754 001011 BNE 1$ ;:BR IF YES
1595 006756 005767 172116 TST $PASS ;:IF FIRST PASS OF PROGRAM
1596 006762 001406 BEQ 1$ ;: INHIBIT ITERATIONS
1597 006764 005267 172114 INC $ICNT ;:INCREMENT ITERATION COUNT
1598 006770 026767 172174 172106 CMP $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
1599 006776 002021 BGE $OVER ;:BR IF MORE ITERATION REQUIRED

```

1600	007000	012767	000001	172076	1S:	MOV	#1,\$ICNT	::REINITIALIZE THE ITERATION COUNTER
1601	007006	016767	172204	172154		MOV	\$MXCNT,\$TIMES	::SET NUMBER OF ITERATIONS TO DO
1602	007014	105267	172062		\$SVLAD:	INCB	\$TSTNM	::COUNT TEST NUMBERS
1603	007020	011667	172062			MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS
1604	007024	011667	172060			MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS
1605	007030	005067	172136			CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
1606	007034	112767	000001	172053		MOVB	#1,\$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1607	007042	016777	172034	172072	\$OVER:	MOV	\$TSTNM,@DISPLAY	::DISPLAY TEST NUMBER
1608	007050	016716	172032			MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS
1609	007054	000002				RTI		::FIXES PS

1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665

007056  
007056 104406  
007060 011437 001162  
007064 011537 001164  
007070 105714  
007072 100003  
007074 112714 000020  
007100 000406  
007102 032714 000040  
007106 001003  
007110 000005  
007112 013714 001162  
007116 022626  
007120 005767 172122  
007124 002002  
007126 011667 172040  
007132 016616 177774  
007136 105267 171741  
007142 001775  
007144 016777 171732 171770  
007152 032777 002000 171760  
007160 001402  
007162 104401 001174  
007166 005267 171720  
007172 011667 171720  
007176 162767 000002 171712  
007204 117767 171706 171702  
007212 032777 020000 171720  
007220 001004  
007222 004767 000060  
007226 104401 001201  
007232  
007232 005777 171702  
007236 100002  
007240 000000  
007242 104406  
007244 032777 001000 171666  
007252 001402  
007254 016716 171630  
007260 005767 171706  
007264 001402  
007266 016716 171700

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

$ERROR: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
MOV $TACS,$#$REG0 ;;SAVE THE TAI1 STATUS
MOV $TADB,$#$REG1 ;;SAVE THE DATA BUFFER
TSTB $TACS ;;IF "XFER REQ" IS UP KNOCK IT DOWN
BPL 100$
MOVB #ILBS,$TACS
BR 101$
100$: BIT #READY,$TACS ;;IS READY SET?
BNE 101$ ;;BR IF YES
RESET ;;REGAIN CONTROL
MOV $#$REG0,$TACS ;;RESTORE THE TACS
101$: CMP (SP)+,(SP)+ ;;POP THE STACK
TST ASKKEY ;;MANUAL LOOPING?
BGE 102$ ;;BR IF NO
MOV (SP),$ESCAPE ;;SETUP FOR MANUAL ESCAPE
102$: MOV -4(SP),(SP) ;;GET PC+2 OF ERROR
7$: INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,TYPERR ;;GO TO USEF ERROR ROUTINE
TYPE $CRLF

20$: TST $SWR ;;HALT ON ERROR
2$: BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
3$: BIT #BIT09,$SWR ;;LOOP ON ERROR SWITCH SET?
BEQ 4$ ;;BR IF NO
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
BEQ 5$ ;;BR IF NONE
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
```

N04

TA11 MOTION TEST  
DZTADC.NEW

24-MAY-76 00:00

MAINDEC-11-DZTAD-C

MACY11 27(1006)  
ERROR HANDLER ROUTINE

23-SEP-76 10:33 PAGE 36

SEQ 0052

1666	007272			5\$:			
1667	007272	022737	006442	000042	CMP	#\$ENDAD, 3#42	::ACT-11 AUTO-ACCEPT?
1668	007300	001001			BNE	6\$	::BRANCH IF NO
1669	007302	000000			HALT		::YES
1670	007304			6\$:			
1671	007304	000002			RTI		::RETURN

```

1672 *****
1673 THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
1674 TYPERR: TYPE ,SCLF ;TYPE A CARRIAGE RETURN & LINE FEED
1675 MFUNC: 0 ;TYPE THE FUNCTION BEING PERFORMED
1676 ;MESSAGE POINTER GOES HERE
1677 007306 104401 001201 ;
1678 007312 104401 ;
1679 007314 000000 ;
1680 007316 104401 001201 ;
1681 007322 162737 000002 001204 ;ADJUST THE MAIN FLOW PC
1682 007330 010045 ;SAVE RO
1683 007332 113700 001114 ;PICKUP THE ITEM INDEX
1684 007336 005300 ;ADJUST THE INDEX
1685 007340 006300 ;SO IT WILL WORK FOR
1686 007342 006300 ;THE ERROR TABLE
1687 007344 006300 ;
1688 007346 062700 001270 ;FORM THE TABLE POINTER
1689 007352 012067 000002 ;PICKUP "ERROR MESSAGE" POINTER
1690 007356 104401 ;TYPE "ERROR MESSAGE"
1691 007360 000000 1S: ;"ERROR MESSAGE POINTER" GOES HERE
1692 007362 012067 000004 ;PICKUP "DATA HEADER" POINTER
1693 007366 001402 ;IF "0" DON'T TYPE
1694 007370 104401 ;TYPE "DATA HEADER"
1695 007372 000000 2S: ;"DATA HEADER" POINTER GOES HERE
1696 007374 012000 3S: ;PICKUP "DATA POINTER"
1697 007376 001004 ;IF THERE IS DATA TO TYPE GO DO IT
1698 007400 012600 4S: ;RESTORE RO
1699 007402 104401 001201 ;TYPE A CARRAGE RETURN&LINE FEED
1700 007406 000207 ;RETURN
1701 007410 5S: ;
1702 007410 013046 ;MOV 3(RO)+, -(SP) ;:SAVE 3(RO)+ FOR TYPEOUT
1703 007412 104402 ;TYPE DATA
1704 007414 005710 ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1705 007416 001770 ;TERMINATOR?
1706 007420 104401 007426 ;BR IF YES
1707 007424 000771 ;TYPE 2 SPACES
1708 007426 020040 000 ;LOOP
1709 007432 007432 5S: ;ASCII STRING OF 2 SPACES
1710 ;EVEN

```

```

1708      ::*****
1709
1710      .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
1711
1712      :CALL
1713      :              JSR      PC, @ASKDRV
1714      :              RETURN
1715      :              ;NOTE: R0 AND R1 ARE DESTROYED
1716      ASKDRV: TYPE    MSGDRV          ;<CRLF>"DRIVE(S)"
1717      CLR          DRVKEY
1718      RDLIN
1719      MOV          (SP)+, R0          ;GO GET A DRIVE
1720      TSTB        @R0                ;SETUP TO CHECK FOR VALID DRIVE(S)
1721      BEQ         NOTLGL             ;WAS A DRIVE SELECTED?
1722      MOV          @DRVKEY, R1       ;BR IF NO
1723      LOOP:  CMPB   #'A, @R0         ;WAS DRIVE "A" SELECTED?
1724      BNE         NOTA              ;BR IF NO
1725      MOVB        (R0)+, (R1)+      ;SET KEY FOR DRIVE "A"
1726      BR         NEXT
1727      NOTA:  CMPB   #'B, @R0         ;WAS DRIVE "B" SELECTED?
1728      BNE         NOTB              ;BR IF NO
1729      MOVB        (R0)+, (R1)+      ;SET KEY FOR DRIVE "B"
1730      BR         NEXT
1731      NOTB:  CMPB   #54, @R0         ;WAS A COMMA TYPED?
1732      BNE         NOTLGL           ;BR IF NO
1733      TSTB        (R0)+              ;DUMP THE COMMA
1734      NEXT:  TSTB   @R0              ;TERMINATOR?
1735      BEQ         EXIT              ;BR IF YES
1736      CMP         @DRVKEY+2, R1      ;TWO DRIVES SELECTED?
1737      BHI         LOOP              ;BR IF NO
1738      NOTLGL: TYPE   $QUES          ;ILLEGAL INPUT DETECTED
1739      BR         ASKDRV             ;GO TRY AGAIN
1740      EXIT:  TST   DRVKEY            ;ANY DRIVE SELECTED?
1741      BEQ         NOTLGL           ;BR IF NO
1742      RTS          PC
1743
1744      ::*****
1745
1746      :CALL
1747      :              JSR      PC, @ASKADR
1748      ASKADR: MOV    R0, -(SP)        ;SAVE R0
1749      1$:  TYPE    ,MSGASK           ;"TACS?"
1750      RDOCT
1751      MOV    (SP)+, R0              ;GET VALUE
1752      BEQ    3$                    ;PICK UP THE OCTAL NUMBER
1753      CMP    R0, #160000           ;IF "0" USE OLD VALUES
1754      BLO    1$                    ;MAKE SURE IT IS A BUS ADDRESS
1755      MOV    R0, @TACSL            ;SAVE TOE TACS
1756      ADD    #2, R0                 ;STEP TO TADB ADDRESS
1757      MOV    R0, @TADBL           ;AND SAVE IT
1758      MOV    @TACSL, @TACSH       ;SET UP TACS UPPER
1759      INC    @TACSH               ;BYTE POINTER
1760
1761      MOV    @TADBL, @TADBH       ;SET UP TADB UPPER
1762      INC    @TADBH               ;BYTE POINTER
1763      3$:  TYPE    ,MSGVEC         ;"VECTOR?"

```

```

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

```

\*\*\*\*\*  
THIS ROUTINE WILL CHANGE AN .ASCII STRING TO AN OCTAL NUMBER

REGISTER R1 MUST BE INITIALIZED BEFORE ENTERING

```

CALL: JSR R3,2#A20CT ;CALL THE ROUTINE
;ADDRESS OF THE FIRST CHARACTER IN THE STRING
;STRING MUST BE TERMINATED BY A BYTE OF ALL 0'S
;RETURN HERE IF ILLEGAL CHARACTER
ERROR RETURN

```

1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819



TAIL MOTION TEST  
DZTADC.NEW 24-MAY-76 00:00

MAINDEC-11-DZTAD-C

MACY11 27(1006) 23-SEP-76 10:33 PAGE 40  
ROUTINE TO CHANGE AN .ASCII STRING TO OCTAL

SEQ 0056

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

# F05

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 41  
ROUTINE TO GET "TEST" AND "SCOPE LOOP" ADDRESSES FROM THE TTY

SEQ 0057

```

1844                                     ;*****
1845                                     ;THIS ROUTINE WILL INPUT THE ADDRESSES OF "TEST" AND "SCOPE LOOP" FROM THE TTY
1846
1847 010116 T.SETLOOP:
1848 010116 104401 015326 TYPE      .MTEST
1849 010122 104410 RDLIN
1850 010124 012667 000022 MOV      (SP)+,11$
1851 010130 105777 000016 TSTB    @11$
1852 010134 001442 BEQ      3$
1853 010136 022626 CMP      (SP)+,(SP)+
1854 010140 010167 000110 MOV      R1,SAVR1
1855 010144 005001 CLR      R1
1856 010146 004337 010032 JSR      R3,@#A20CT
1857 010152 000000 11$:      0
1858 010154 000760 BR       T.SETLOOP
1859 010156 005721 TST      (R1)+
1860 010160 010116 MOV      R1,(SP)
1861 010162 104401 015341 5$:      TYPE      .MLOOP
1862 010166 104410 RDLIN
1863 010170 012667 000014 MOV      (SP)+,12$
1864 010174 105777 000010 TSTB    @12$
1865 010200 001410 BEQ      1$
1866 010202 005001 CLR      R1
1867 010204 004337 010032 JSR      R3,@#A20CT
1868 010210 000000 12$:      0
1869 010212 000763 BR       5$
1870 010214 010137 001106 MOV      R1,@#$LPADR
1871 010220 000402 BR       2$
1872 010222 011637 001106 1$:      MOV      (SP),@#$LPADR
1873 010226 011637 001110 2$:      MOV      (SP),@#$LPERR
1874 010232 012737 177777 006624 MOV      #-1,@#LOOPKEY
1875 010240 000402 BR       4$
1876 010242 005037 006624 3$:      CLR      @#LOOPKEY
1877 010246 016701 000002 4$:      MOV      SAVR1,R1
1878 010252 000002 RTI
1879 010254 000000 SAVR1:  0

```

```

;TYPE "TEST PC?"
;GET AN ASCII STRING
;GET FIRST ADDRESS OF INPUT BUFFER
;FIRST CHAR. A "CR" ?
;BR IF YES---GO CLEAR THE LOOP KEY
;POP POP
;SAVE R1
;SET PARTIAL TO ZERO
;GO CHANGE ASCII TO OCTAL
;POINTER TO STRING
;ERROR RETURN--GO TRY AGAIN
;JUST INCREMENTING
;PUT "TEST PC"; ON THE STACK
;TYPE "LOOP PC ?"
;GET A STRING OF ASCII
;GET FIRST ADDRESS OF INPUT BUFFER
;JUST A "CR" ?
;BR IF YES
;CLEAR PARTIAL
;CHANGE TO OCTAL

;SAVE SCOPE LOOP ADDRESS
;GO SET LOOP KEY
;SAVE SCOPE LOOP ADDRESS
;ERROR LOOP SAME AS THE TEST ADDRESS
;SET THE LOOP KEY

;CLEAR THE LOOP KEY
;RESTORE R1
;BYE BYE
;HOLD R1 HERE

```

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

010256  
010256 032777 000200 170654  
010264 001041  
010266 010146  
010270 016701 170742  
010274 111137 001250  
010300 121127 000101  
010304 001406  
010306 052714 000400  
010312 012737 001262 001256  
010320 000405  
010322 042714 000400 1\$:  
010326 012737 001260 001256 2\$:  
010334 005201  
010336 105711  
010340 001002  
010342 012701 001240  
010346 010167 170664 3\$:  
010352 032777 001000 170560  
010360 001002  
010362 105037 001103  
010366 012601 4\$:  
010370 000002 5\$:

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

T.SELDRV:  
BIT #SW07, @SWR ; LOCK ON CURRENT DRIVE?  
BNE 5\$ ; BR IF YES  
MOV R1, -(SP) ; SAVE R1 ON THE STACK  
MOV DRIVE, R1 ; GET THE DRIVE  
MOVB (R1), @#CURDRV ; SAVE CURRENT DRIVE  
CMPB (R1), #'A ; IS IT DRIVE "A"?  
BEQ 1\$ ; BR IF YES  
SIS #UNIT, @TACS ; SELECT DRIVE "B"  
MOV #RWDB, @#RWDFLG ; SET REWIND POINTER TO "B"  
BR 2\$  
BIC #UNIT, @TACS ; SELECT DRIVE "A"  
MOV #RWDA, @#RWDFLG ; SET REWIND POINTER TO "A"  
INC R1 ; INCREMENT TO THE NEXT DRIVE  
TSTB (R1) ; OUT OF DRIVES TO TEST?  
BNE 3\$ ; BR IF NO  
MOV #DRVKEY, R1 ; RESET TO THE FIRST DRIVE  
MOV R1, DRIVE ; SAVE FOR NEXT TIME  
BIT #SW09, @SWR ; IS SWITCH 9=1?  
BNE 4\$ ; BR IF YES  
CLRB @#SERFLG ; CLEAR THE ERROR INDICATOR  
MOV (SP)+, R1 ; RESTORE R1  
RTI

# H05

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 43  
ROUTINE TO COMPARE THE READ AND WRITE BUFFERS

SEQ 0059

```

1907
1908
1909 010372
1910 010372 011600
1911 010374 010037 001204
1912 010400 012737 015531 007314
1913 010406 012003
1914 010410 012301
1915 010412 011303
1916 010414 013002
1917 010416 005067 000034
1918 010422 010016
1919 010424 005037 001206
1920 010430 032777 000010 170502
1921 010436 001401
1922 010440 000002
1923 010442 005303 1$:
1924 010444 002001
1925 010446 000002
1926 010450 005237 001206 4$:
1927 010454 122162 000000 2$:
1928 010460 001006
1929 010462 005267 177770
1930 010466 042767 177774 177762
1931 010474 000762
1932 010476 114137 001126 3$:
1933 010502 066702 177750
1934 010506 111237 001124
1935 010512 104010
1936
1937
1938 010514 011637 001204
1939 010520 012737 015512 007314
1940 010526 004037 012234 1$:
1941 010532 000101
1942 010534 005037 177776
1943 010540 004037 011726
1944 010544 104003
1945 010546 104004
1946 010550 004037 012024
1947 010554 104002
1948 010556 104002
1949 010560 000002
1950 010562 104002

```

```

*****
: THIS ROUTINE WILL COMPARE THE READ AND WRITE BUFFERS
: .BLKCMP:
MOV      (SP),R0          ;GET THE PC OF THE CALL
MOV      R0,@#SAVPC      ;SAVE THE PC OF THE CALL
MOV      #MBUFCK,@#MFUNC ;POINTER FOR .ASCII MESSAGE
MOV      (R0)+,R3        ;PICKUP THE READ LINK ADDRESS
MOV      (R3)+,R1        ;PICKUP THE READ BUFFER ADDRESS
MOV      (R3),R3         ;PICKUP THE READ BUFFER SIZE
MOV      @ (R0)+,R2      ;PICKUP THE WRITE BUFFER ADDRESS
CLR      2$+2           ;ZERO THE OFFSET
MOV      R0,(SP)        ;SET THE RETURN
CLR      @#BYTNUM        ;ZERO THE BYTE NUMBER
BIT      #SW03,@SWR     ;COMPARE DATA BUFFERS?
BEQ      1$             ;BR IF YES
RTI      1$             ;NO---GO BACK
1$:      DEC      R3     ;CHECK FOR THE TERMINATION
BGE      4$             ;BR IF MORE BYTES TO COMPARE
RTI      1$             ;RETURN
4$:      INC      @#BYTNUM ;COUNT THIS BYTE
2$:      CMPB    (R1)+,0(R2) ;COMPARE READ AND WRITE BUFFER
BNE      3$             ;BR IF NOT EQUAL
INC      2$+2           ;INCREMENT WRITE BUFFER INDEX
BIC      #1C3,2$+2     ;DON'T LET IT GET BIGGER THAN 3
BR       1$
3$:      MOVB   -(R1),@#SBDDAT ;GET THE RECEIVED BYTE
ADD      2$+2,R2       ;FORM ADDRESS
MOVB    (R2),@#SGDDAT  ;GET THE EXPECTED BYTE
ERROR   10             ;RECEIVED AND EXPECTED DATA NOT EQUAL
*****
: THIS ROUTINE WILL WRITE A FILE GAP ON THE TAPE
: .WFG:
MOV      (SP),@#SAVPC   ;SAVE THE PC OF THE CALL
MOV      #MXWFG,@#MFUNC ;SET MESSAGE POINTER FOR "WFG"
1$:      JSR    R0,@#DO.CMD ;GO START A "WFG"
XWFG!INT.EN!GO
CLR      @#PS           ;ALLOW INTERRUPTS
JSR     R0,@#WAITFLAG  ;GO WAIT FOR A FLAG
ERROR   3              ;NO FLAGS OCCURRED
ERROR   4              ;NO INTERRUPT
JSR     R0,@#FLAGS     ;NORMAL RETURN---FIND OUT WHAT FLAG
ERROR   2              ;RETURN HERE IF "XFER REQ" AND "READY" =0
ERROR   2              ;RETURN HERE IF "XFER REQ" =1
RTI     1$             ;"RETURN HERE IF "READY"=1 AND "ERROR" =0
ERROR   2              ;"READY"=1 AND "ERROR"=1

```

```

1951
1952
1953 010564 011637 001204
1954 010570 012737 015363 007314
1955 010576 017602 000000
1956 010602 062716 000002
1957 010606 012201
1958 010610 005037 012232
1959 010614 005067 000066
1960 010620 012202
1961 010622 003001
1962 010624 104011
1963 010626 004037 012234 2$:
1964 010632 000103 XWRITE!INT.EN!GO
1965 010634 005037 177776 3$:
1966 010640 004037 011726 JSR RO,#WAITFLAG
1967 010644 104003 ERROR 3 ;ALLOW INTERRUPTS
1968 010646 104004 ERROR 4 ;WAIT FOR A FLAG
1969 010650 005037 001254 CLR #WAITKEY ;NO FLAGS OCCURRED
1970 010654 012737 000340 177776 MOV #340,#PS ;INTERRUPT FAILED
1971 010662 004037 012024 JSR RO,#FLAGS ;CLEAR THE WAIT FOR INTERRUPT KEY
1972 010666 104002 ERROR 2 ;LOCK OUT THE WORLD
1973 010670 000402 BR 4$ ;WAIT FLAG OCCURRED?
1974 010672 104005 ERROR 5 ;"READY"=0 & "XFER REQ" =0
1975 010674 104002 ERROR 2 ;"XFER REQ"=1
1976 010676 005302 4$: DEC R2 ;"READY"=1 & "ERROR"=0
1977 010700 002417 BLT 5$ ;"READY"=1 & "ERROR"=1
1978 010702 005000 CLR RO ;NEED TO WRITE A BYTE ON TAPE
1979 010704 156100 000000 6$: BISB 0(R1),RO ;BR IF NO
1980 010710 110015 MOVB RO,#ATADB ;PICKUP BYTE FOR TRANSFER
1981 010712 004737 012112 JSR PC,#DO.CRC ;TRANSFER A BYTE TO THE TAIL
1982 010716 005267 177764 INC 6$+2 ;CALCULATE CRC FOR THIS BYTE
1983 010722 042767 177774 177756 BIC #1C3,6$+2 ;INCREASE THE INDEX
1984 010730 012777 012350 170272 MOV #SERV,#TAVEC ;KEEP IT LESS THAN 4
1985 010736 000736 BR 3$ ;SETUP TO SERVICE THE INTERRUPT
1986 010740 013761 012232 000004 5$: MOV #CRC.WD,4(R1) ;LOOP
1987 010746 004037 012234 JSR RO,#DO.CMD ;SAVE THE CRC WORD
1988 010752 000122 XWRITE!INT.EN!ILBS ;START A "ILBS"
1989 010754 005037 177776 CLR #PS ;ALLOW INTERRUPTS
1990 010760 004037 011726 JSR RO,#WAITFLAG ;WAIT FOR A FLAG
1991 010764 104003 ERROR 3 ;NO FLAG
1992 010766 104004 ERROR 4 ;NO INTERRUPT
1993 010770 004037 012024 JSR RO,#FLAGS ;WHAT FLAG IS SET?
1994 010774 104002 ERROR 2 ;NONE ("XFER REQ"=0 & "READY"=0)
1995 010776 104002 ERROR 2 ;"XFER REQ"
1996 011000 000002 RTI ;"READY"
1997 011002 104002 ERROR 2 ;"READY" & "ERROR"

```

```

1998
1999
2000 011004 011637 001204
2001 011010 012737 015371 007314
2002 011016 017602 000000
2003 011022 062716 000002
2004 011026 010200
2005 011030 016000 000034
2006 011034 016067 000004 170146
2007 011042 012201
2008 011044 012202
2009 011046 003001
2010 011050 104011
2011 011052 005037 012232 2$:
2012 011056 004037 012234
2013 011062 000105
2014 011064 005037 177776 3$:
2015 011070 105011
2016 011072 004037 011726
2017 011076 104003
2018 011100 104004
2019 011102 005037 001254
2020 011106 012737 000340 177776
2021 011114 004037 012024
2022 011120 104002
2023 011122 000402
2024 011124 104005
2025 011126 000413
2026 011130 005302 4$:
2027 011132 002431
2028 011134 005000
2029 011136 151500
2030 011140 110021
2031 011142 004737 012112
2032 011146 012777 012350 170054
2033 011154 000743
2034 011156 010237 001166 8$:
2035 011162 001414
2036 011164 013746 001172
2037 011170 005037 001172
2038 011174 016646 000004
2039 011200 016646 000004
2040 011204 104013
2041 011206 012637 001172
2042 011212 000002
2043 011214 104002 9$:
2044 011216 013737 012232 001212 5$:
2045 011224 004037 012234
2046 011230 000124
2047 011232 111537 001214
2048 011236 005037 177776
2049 011242 004037 011726
2050 011246 104003
2051 011250 104004
2052 011252 111567 167737
2053 011256 004037 012024

```

```

*****
THIS ROUTINE WILL READ A RECORD FROM THE TAPE
READ: MOV (SP), @#SAVPC ;SAVE THE PC OF THE CALL
MOV @#MXREAD, @#MFUNC ;SET POINTER OF .ASCIZ MESSAGE
MOV @ (SP), R2 ;GET THE READ LINK POINTER
ADD #2, (SP) ;ADJUST FOR RETURN
MOV R2, R0 ;PICKUP AND SAVE THE "CRC" THAT
MOV WLNK1-RLNK1(R0), R0 ; WAS CALCULATED ON THE WRITE
MOV 4(R0), RCRC0 ; DATA FOR THIS BLOCK
MOV (R2)+, R1 ;GET ADDRESS OF BUFFER
MOV (R2)+, R2 ;GET NUMBER OF BYTES
BGT 2$ ;MAKE SURE NUMBER IS LEGAL
ERROR 11 ;ILLEGAL NUMBER OF BYTES
CLR @#CRC.WD ;CLEAR CRC WORD
JSR R0, @#DO.CMD ;GO START A "READ"
XREAD! INT. EN! GO
3$: CLR @#PS ;ALLOW INTERRUPTS
CLRB (R1) ;SET TERMINATOR IF LAST BYTE
JSR R0, @#WAITFLAG ;WAIT FOR A FLAG
ERROR 3 ;NONE OCCURRED
ERROR 4 ;NO INTERRUPT
CLR @#WAITKEY ;CLEAR THE WAIT ON INTERRUPT KEY
MOV #340, @#PS ;LOCK OUT ALL INTERRUPTS
JSR R0, @#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;?UNKNOWN FLAG
BR 4$ ;"XFER REQ"
ERROR 5 ;"READY"
BR 8$ ;"READY" WITH "ERROR"
4$: DEC R2 ;READ A BYTE?
BLT 5$ ;BR IF NO
CLR R0
BISB @TADB, R0 ;READ ONE BYTE
MOVB R0, (R1)+ ;AND PUT IT IN THE BUFFER
JSR PC, @#DO.CRC ;CALCULATE CRC FOR THIS BYTE
MOV #SERV, @TAVEC ;SETUP TO SERVICE THE INTERRUPT
BR 3$ ;LOOP
8$: MOV R2, @#STMPO ;GET NUMBER OF BYTES NOT READ
BEQ 9$ ;SKIP OUT IF 0
MOV @#$ESCAPE, -(SP) ;SAVE THE ESCAPE ADDRESS
CLR @#$ESCAPE ;NO ESCAPE DESIRED
MOV 4(SP), -(SP) ;COPY THE STACK
MOV 4(SP), -(SP)
ERROR 13 ;BLOCK READ SHORT
MOV (SP)+, @#$ESCAPE ;RESTORE ESCAPE ADDRESS
RTI ;LEAVE
9$: ERROR 2 ;ERROR OCCURRED DURING READ
5$: MOV @#CRC.WD, @#RCRC1 ;SAVE THE CALCULATED CRC
JSR R0, @#DO.CMD ;DO AN ILBS
XREAD! INT. EN! ILBS
MOV @TADB, @#RCRC2 ;SAVE 1ST BYTE OF CRC FROM TA11
CLR @#PS ;ALLOW INTERRUPTS
JSR R0, @#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NONE OCCURRED
ERROR 4 ;INTERRUPT FAILED
MOVB @TADB, RCRC2+1 ;SAVE 2ND BYTE OF CRC FROM TA11
JSR R0, @#FLAGS ;CHECK FLAGS

```



```

2065
2066
2067 011314 011637 001204
2068 011320 012737 015443 007314
2069 011326 004037 012234
2070 011332 000107
2071 011334 005037 177776
2072 011340 004037 011726
2073 011344 104003
2074 011346 104004
2075 011350 004037 012024
2076 011354 104002
2077 011356 104002
2078 011360 000401
2079 011362 104002
2080 011364 032714 004000
2081 011370 001401
2082 011372 000002
2083 011374 104006
2084
2085
2086
2087
2088 011376 011637 001204
2089 011402 012737 015467 007314
2090 011410 004037 012234
2091 011414 000111
2092 011416 005037 177776
2093 011422 004037 011726
2094 011426 104003
2095 011430 104004
2096 011432 004037 012024
2097 011436 104002
2098 011440 104002
2099 011442 000002
2100 011444 104002

;*****
;THIS ROUTINE WILL BACK SPACE ONE FILE GAP
↑.BSFG: MOV (SP),Q#SAVPC ;SAVE PC OF CALL
MOV #MXBSFG,Q#MFUNC ;SAVE POINTER OF .ASCIZ NAME
1$: JSR RO,Q#DO.CMD ;START A "BSFG"
XBSFG!INT.EN!GO
CLR Q#PS ;ALLOW INTERRUPTS
JSR RO,Q#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NO FLAG
ERROR 4 ;FLAG DIDN'T INTERRUPT
JSR RO,Q#FLAGS ;FIND OUT WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
ERROR 2 ;"XFER REQ"
BR 2$ ;"READY"
ERROR 2 ;"READY" & "ERROR"
2$: BIT #FGAP,Q#TACS ;IN A FILE GAP?
BEQ 3$ ;BR IN NO
RTI ;GO BACK
3$: ERROR 6 ;"BSFG" DIDN'T STOP IN A FILE GAP

;*****
;THIS ROUTINE WILL BACK SPACE ONE BLOCK GAP
↑.BSBG: MOV (SP),Q#SAVPC ;SAVE THE PC OF THE CALL
MOV #MXBSBG,Q#MFUNC ;MESSAGE POINTER
1$: JSR RO,Q#DO.CMD ;START A "BSBG"
XBSBG!INT.EN!GO
CLR Q#PS ;ALLOW INTERRUPTS
JSR RO,Q#WAITFLAG ;WAIT FOR A FLAG
ERROR 3 ;NO FLAG OCCURRED
ERROR 4 ;FLAG DIDN'T INTERRUPT
JSR RO,Q#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
ERROR 2 ;"XFER REQ"
RTI ;"READY"-- GO BACK TO USER
ERROR 2 ;"READY" & "ERROR"

```



M05

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 48  
ROUTINE TO "SPACE-FORWARD-FILE-GAP"

SEQ 0064

```

2101      ;:*****
2102      ;THIS ROUTINE WILL SPACE FORWARD FOR ONE FILE GAP
2103      T.SFFG: MOV      (SP),A#SAVPC      ;SAVE PC OF CALL
2104      MOV      #MXSFFG,A#MFUNC      ;POINTER TO .ASCIZ MESSAGE
2105      1$: JSR      RD,A#DO.CMD      ;START A "SFFG"
2106      XSFFG:INT. EN!GO
2107      CLR      A#PS      ;ALLOW INTERRUPTS
2108      JSR      RD,A#WAITFLAG      ;WAIT FOR A FLAG
2109      ERROR    3      ;FLAG FAILED TO OCCUR
2110      ERROR    4      ;FLAG DIDN'T INTERRUPT
2111      JSR      RD,A#FLAGS      ;WHAT FLAG OCCURRED
2112      ERROR    2      ;UNKNOWN
2113      ERROR    2      ;"XFER REQ"
2114      BR      2$      ;"READY"
2115      ERROR    2      ;"READY" AND "ERROR"
2116      2$: BIT      #FGAP,A#TACS      ;DID "SFFG" STOP IN A "FILE GAP"?
2117      BEQ      3$      ;BR IF NO
2118      RTI      ;RETURN TO CALLER
2119      3$: ERROR    6      ;"SFFG" DIDN'T STOP IN A FILE GAP
2120
2121      ;:*****
2122      ;THIS ROUTINE WILL SPACE FORWARD ONE BLOCK GAP
2123      T.SFBG: MOV      (SP),A#SAVPC      ;SAVE PC OF CALL
2124      MOV      #MXSFBG,A#MFUNC      ;POINTER TO .ASCIZ MESSAGE
2125      1$: JSR      RD,A#DO.CMD      ;START A "SFBG"
2126      XSFBG:INT. EN!GO
2127      CLR      A#PS      ;ALLOW INTERRUPTS
2128      JSR      RD,A#WAITFLAG      ;WAIT FOR A FLAG
2129      ERROR    3      ;NO FLAG
2130      ERROR    4      ;NO INTERRUPT
2131      JSR      RD,A#FLAGS      ;FIND OUT WHAT FLAG
2132      ERROR    2      ;UNKNOWN
2133      ERROR    2      ;"XFER REQ"
2134      RTI      ;"READY"
2135      TST      A#ASKKEY      ;RUNNING UNDER MANUAL LOOPING?
2136      BGE      2$      ;BR IF NO
2137      RTI      ;RETURN
2138      2$: ERROR    2      ;"READY" AND "ERROR"
2139

```

```

2140
2141
2142 011610 011637 001204
2143 011614 012737 015354 007314
2144 011622 004037 012024
2145 011626 104001
2146 011630 104001
2147 011632 000411
2148 011634 005777 167416
2149 011640 001406
2150 011642 032714 020000
2151 011646 001403
2152 011650 005077 167402
2153 011654 104001
2154 011656 004037 012234 1$:
2155 011662 000117
2156 011664 005037 177776
2157 011670 004037 011726
2158 011674 104003
2159 011676 104004
2160 011700 004037 012024
2161 011704 104002
2162 011706 104002
2163 011710 000401
2164 011712 104002
2165 011714 032714 020000 2$:
2166 011720 001401
2167 011722 000002
2168 011724 104007 3$:

```

```

*****
↑ THIS ROUTINE WILL REWIND THE TAPE
↑.RWND: MOV (SP),@#SAVPC ;SAVE PC OF CALL
MOV #MXRWND,@#MFUNC ;POINTER TO .ASCIZ MESSAGE
JSR RO,@#FLAGS ;CHECK THE FLAGS
ERROR 1 ;NO FLAGS
ERROR 1 ;"XFER REQ"
BR 1$ ;NORMAL RETURN
TST @RWDFLG ;WAS LAST FUNCTION A REWIND?
BEQ 1$ ;BR IF YES
BIT #LEADER,@TACS ;ON CLEAR LEADER?
BEQ 1$ ;BR IF NO
CLR @RWDFLG ;ALLOW THIS ERROR ONLY ONE TIME
ERROR 1 ;ON CLEAR LEADER BUT SHOULDN'T BE
1$: JSR RO,@#DO.CMD ;GO START A "REWIND"
XRWND!INT.EN!GO
CLR @#PS ;ALLOW INTERRUPTS
JSR RO,@#WAITFLAG ;WAIT ON FLAG
ERROR 3 ;NO FLAG OCCURRED
ERROR 4 ;NO INTERRUPT
JSR RO,@#FLAGS ;WHAT FLAG OCCURRED
ERROR 2 ;UNKNOWN
ERROR 2 ;"XFER REQ"
BR 2$ ;"READY"
ERROR 2 ;"READY" & "ERROR"
2$: BIT #LEADER,@TACS ;ON "CLEAR LEADER"?
BEQ 3$ ;BR IF NO
RTI ;YES---RETURN
3$: ERROR 7 ;"REWIND" FAILED TO GO TO "CLEAR LEADER"

```

\*\*\*\*\*  
THIS ROUTINE IS USED TO WAIT ON A FLAG

CALL: JSR RO, @WAITFLAG  
RETURN FOR NO FLAGS  
RETURN FOR NO INTERRUPT  
NORMAL RETURN

```

WAITFLAG: CLR LOCNT ;ZERO THE LOW COUNTER
          MOV MAXCNT, HICNT ;SET THE HIGH COUNTER TO MAX.COUNT
          BIT #SW05, @SWR ;RUNNING WITH INTERRUPTS?
          BEQ 1$ ;BR IF YES
          MOV #-1, WAITKEY ;SET THE INTERRUPT OCCURRED FLAG
          BIT #TR:REQ!READY, @TACS ;TEST FOR FLAGS
          BNE 2$ ;BR IF A FLAG IS UP
          INC LOCNT ;COUNT THE LOW COUNTER
          BNE 1$ ;LOOP IF NOT ZERO
          DEC HICNT ;COUNT THE HIGH COUNTER
          BPL 1$ ;LOOP IF NOT NEG.
          BR 3$ ;LEAVE
          TST (RO)+ ;INCREMENT THE RETURN ADDRESS
          TST WAITKEY ;LOOK AT THE WAIT ON INT. KEY
          BEQ 3$ ;BR IF NOT SET
          TST (RO)+ ;INCREMENT THE RETURN ADDRESS
          RTS ;RETURN TO USER
          LOCNT: 00
          HICNT: 00
          MAXCNT: 0

```

```

011726 005067 000064
011726 016767 000064 000060
011732 016767 000064 167172
011740 032777 000040
011746 001403
011750 012767 177777 167276
011756 032714 000240 1$:
011762 001007
011764 005267 000026
011770 001372
011772 005367 000022
011776 100367
012000 000405
012002 005720 2$:
012004 005767 167244
012010 001401
012012 005720
012014 000200 3$:
012016 000000
012020 000000
012022 000000

```

012024  
012030  
012032  
012034  
012036  
012040  
012042  
012050  
012052  
012054  
012056  
012062  
012066  
012072  
012074  
012076  
012100  
012102  
012104  
012106  
012110

03271 000240  
001427  
005720  
105714  
100424  
005720  
032777 000100 167070  
001414  
010046  
010146  
013700 001264  
013701 001266  
062700 000001  
001374  
012601  
012600  
005714  
100001  
005720  
000200

```

*****
THIS ROUTINE IS USED TO DETERMINE WHAT FLAG IS SET
CALL:
      JSR      RO, @#FLAGS
      RETURN   FOR UNKNOWN FLAG
      RETURN   FOR "TRANSFER REQUEST"
      RETURN   FOR "READY"
      RETURN   FOR "READY" & "ERROR"

FLAGS: BIT      #TR.REQ!READY, @TACS      ;CHECK FOR "XFER REQ" OR "READY"
      BEQ      LVFLAG                      ;BR IF NO FLAGS
      TST      (RO)+                       ;INCREMENT RETURN ADDRESS
      TSTB     @TACS                       ;CHECK FOR "XFER REQ"
      BMI      LVFLAG                      ;BR IF SET
      TST      (RO)+                       ;INCREMENT RETURN ADDRESS
      BIT      #SW06, @SWR                 ;STALL?
      BEQ      3$                          ;BR IF NO
      MOV      RO, -(SP)                   ;;PUSH RO ON STACK
      MOV      R1, -(SP)                   ;;PUSH R1 ON STACK
      MOV      @#STALL, RO                  ;GET STALL TIME
      MOV      @#STALL+2, R1
1$:   ADD      #1, RO                       ;STALL
      ADC      R1
      BNE      1$                          ;LOOP
      MOV      (SP)+, R1                   ;;POP STACK INTO R1
      MOV      (SP)+, RO                   ;;POP STACK INTO RO
3$:   TST      @TACS                        ;IS "ERROR" SET
      BPL      LVFLAG                      ;BR IN NO "ERROR"
      TST      (RO)+                       ;INCREMENT RETURN ADDRESS
LVFLAG: RTS      RO

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000008  
000009  
000010  
000011  
000012  
000013  
000014  
000015  
000016  
000017  
000018  
000019  
000020  
000021  
000022  
000023  
000024  
000025  
000026  
000027  
000028  
000029  
000030  
000031  
000032  
000033  
000034  
000035  
000036  
000037  
000038  
000039  
000040  
000041  
000042  
000043  
000044  
000045  
000046  
000047  
000048  
000049  
000050  
000051  
000052  
000053  
000054  
000055  
000056  
000057  
000058  
000059  
000060  
000061  
000062  
000063  
000064  
000065  
000066  
000067  
000068  
000069  
000070  
000071  
000072

\*\*\*\*\*  
THIS ROUTINE WILL CALCULATE THE CRC

CALL: JSR PC, @DO.CRC ;RO=1 BYTE OF DATA

DO.CRC:

```

MOV R0, -(SP)      ;; PUSH R0 ON STACK
MOV R1, -(SP)      ;; PUSH R1 ON STACK
MOV R2, -(SP)      ;; PUSH R2 ON STACK
MOV R3, -(SP)      ;; PUSH R3 ON STACK
MOV R4, -(SP)      ;; PUSH R4 ON STACK
MOV #8, R3          ;MAKE EIGHT ITERATIONS
MOV CRC.WD, R3     ;PICKUP THE CRC WORD
1$: CLR R1
MOV R3, R2         ;GET THE PARTIAL CRC WORD
BIC #1<BIT00, R2  ;STRIP AWAY EVERYTHING BUT BIT00
ROR R0             ;PULL OFF THIS BIT
ROL R1             ;AND SETUP TO XOR IT
MOV R1, R4         ;FORM THE XOR OF "R1" AND "R2"
BIC R2, R4
BIC R1, R2
BIS R4, R2         ;RESULTS TO "R2"
ROR R2
BCC #2$
MOV #BIT14!BIT01, R1
MOV R1, R4         ;FORM THE XOR OF "R1" AND "R3"
BIC R3, R4
BIC R1, R3
BIS R4, R3         ;RESULTS TO "R3"
ROR R3
2$: DEC (PC)+
3$: O
BGT 1$            ;BR IF MORE BITS TO DO
MOV R3, CRC.WD
MOV (SP)+, R4     ;; POP STACK INTO R4
MOV (SP)+, R3     ;; POP STACK INTO R3
MOV (SP)+, R2     ;; POP STACK INTO R2
MOV (SP)+, R1     ;; POP STACK INTO R1
MOV (SP)+, R0     ;; POP STACK INTO R0
RTS PC
CRC.WD: C        ;CRC WORD

```

000010 000054

000074

1\$:

177776

040002

2\$:

3\$:

000014

E06

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 53  
ROUTINE TO PERFORM THE REQUESTED FUNCTION

SEQ 0069

2273  
2274  
2275  
2276  
2277  
2278  
2279  
2280  
2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288  
2289  
2290  
2291  
2292  
2293  
2294  
2295  
2296  
2297  
2298  
2299  
2300  
2301  
2302  
2303  
2304  
2305  
2306  
2307  
2308  
2309  
2310  
2311  
2312  
2313  
2314  
2315  
2316

012234 010146  
012236 012737 000340 177776  
012244 012001  
012246 042701 177640  
012252 032777 000040 166660  
012260 001405  
012262 042701 000100  
012266 012777 012366 166734  
012274 005067 166754  
012300 032701 000100  
012304 001403  
012306 012777 012350 166714  
012314 110114  
012316 032701 000001  
012322 001403  
012324 032714 000040  
012330 001375  
012332 005101  
012334 042701 177761  
012340 010177 166712  
012344 012601  
012346 000200

\*\*\*\*\*  
; THIS ROUTINE WILL LOAD THE LOW BYTE OF TACS WITH THE DESIRED FUNCTION

```
CALL:
      JSR      RO, @#DO, CMD
      DESIRED  FUNCTIN
      RETURN

DO.CMD: MOV      R1, -(SP)           ;SAVE R1
        MOV      #340, @#PS        ;LOCK OUT ALL INERRUPTS
        MOV      (RO)+, R1         ;GET THE COMMAND
        BIC      #1<INT.EN!ILBS!FUNCTION!GO>, R1 ;STRIP AWAY JUNK
        BIT      #SW05, @SWR       ;WITH OR WITHOUT INTERRUPTS
        BEQ      4$                ;BR IF WITH
        BIC      #INT.EN, R1       ;CLEAR INTERRUPT ENABLE IF SET
        MOV      #BADINT, @TAVEC   ;SETUP TO CATCH BAD INTERRUPT
        CLR      WAITKEY           ;ZERO THE WAIT ON INTERRUPT KEY
        BIT      #INT.EN, R1       ;RUNNING WITH INTERRUPTS?
        BEQ      3$                ;BR IF NO
        MOV      #SERV, @TAVEC     ;SETUP TO SERVICE THE INTERRUPT
        MOV      R1, @TACS         ;LOAD THE COMMAND
        BIT      #GO, R1           ;IS GO BIT ON A "1"?
        BEQ      2$                ;BR IF GO =0
        BIT      #READY, @TACS     ;WAIT FOR "READY" TO CLEAR
        BNE      1$                ;
        COM      R1                ;SET OR CLEAR REWIND FLAG
        BIC      #1<FUNCTION>, R1  ;EXTRACT FUNCTION BITS
        MOV      R1, @RWDFLG       ;AND SAVE AS THE REWIND FLAG
        MOV      (SP)+, R1        ;GET R1 BACK
        RTS      RO                ;RETURN TO USER
```

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

012350 012737 177777 001254  
012356 012777 012366 166644  
012364 000002

```
SERV:  MOV      #-1, @#WAITKEY     ;SET THE WAIT KEY
        MOV      #BADINT, @TAVEC   ;SET TO CATCH ILLEGAL INTERRUPTS
        RTI
```

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

012366 011637 001204  
012372 104014

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

F06

TA11 MOTION TEST  
DZTADC.NEW

MAINDEC-11-DZTAD-C  
24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 54

\*\*\*\*\* MANUAL ADJUSTMENT ROUTINES \*\*\*\*\*

SEQ 0070

2317  
2318  
2319  
2320

\*\*\*\*\*  
\*\*\*\*\*  
:THE FOLLOWING ROUTINES CAN BE USED TO MAKE ADJUSTMENTS TO THE TUBO  
:NOTE: \*\*\* BEFORE USING ANY OF THE ROUTINES LOAD AND START AT 214 \*\*\*

2321  
2322  
2323  
2324  
2325  
2326  
2327  
2328  
2329  
2330  
2331  
2332  
2333  
2334  
2335  
2336  
2337  
2338  
2339  
2340  
2341  
2342  
2343  
2344  
2345  
2346  
2347  
2348  
2349  
2350  
2351  
2352  
2353  
2354  
2355  
2356  
2357  
2358  
2359  
2360  
2361

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

\*\*\*\*\*

\*\*\*\*\*

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

\*\*\*\*\*

012374 012706 001100  
012400 013704 001220  
012404 013705 001224  
012410 000005  
012412 012737 012374 001110  
012420 104422  
012422 112714 000017  
012426 032714 000040  
012432 001775  
012434 104412  
012436 032714 020000  
012442 001774  
012444 000000  
012446 000752

WFGSUB: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY  
MOV @#TACSL, TACS ;SETUP THE TAI1 STATUS AND  
MOV @#TADBL, TADB ;DATA BUFFER REGISTERS  
RESET ;RESET THE WORLD  
MOV #WFGSUB, @#\$LPERR ;SETUP THE LOOP ON ERROR ADDRESS  
SELDRV ;SELECT THE DRIVE  
100\$: MOVB #XRWD!GO, @TACS ;SEND TAPE TO "BOT"  
BIT #READY, @TACS ;WAIT ON READY  
BEQ 100\$  
1\$: WFG ;WRITE A FILE GAP  
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?  
BEQ 1\$ ;BR IF NO  
HALT ;STOP IF YES  
BR WFGSUB ;LOOP ON CONT.

\*\*\*\*\*

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



# H06

TA11 MOTION TEST  
DZTADC.NEW

24-MAY-76 00:00  
MAINDEC-11-DZTAD-C

MACY11 27(1006) 23-SEP-76 10:33 PAGE 56  
WRITE CONTINUOUS BLOCKS OF DATA

SEQ 0072

```
2362 ; THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
2363 ;*****
2364 012450 004737 013116 WRTSUB: JSR PC, @#SETBUF ; GET BLOCK SIZE AND PATTERN
2365 012454 012706 001100 WLOOP: MOV #STACK, SP ; KEEP THE STACK OUT OF THE WAY
2366 012460 013704 001220 MOV @#TACSL, TACS ; SETUP THE TA11 STATUS AND
2367 012464 013705 001224 MOV @#TADBL, TADB ; DATA BUFFER REGISTERS
2368 012470 000005 RESET ; RESET THE WORLD
2369 012472 012737 012454 001110 MOV #WLOOP, @#$LPERR ; SETUP THE LOOP ON ERROR ADDRESS
2370 012500 104422 SELDRV ; SELECT THE DRIVE
2371 012502 112714 000017 MOV #XRWD!GO, @TACS ; SEND TAPE TO "BOT"
2372 012506 032714 000040 100$: BIT #READY, @TACS ; WAIT ON READY
2373 012512 001775 BEQ 100$
2374 012514 104413 017052 1$: WRITE ,WLNKX ; WRITE A BLOCK
2375 012520 032714 020000 BIT #LEADER, @TACS ; AT "CLEAR LEADER"?
2376 012524 001773 BEQ 1$ ; BR IF NO
2377 012526 000000 HALT ; STOP IF "EOT"
2378 012530 000751 BR WLOOP ; LOOP IF CONT.
2379
2380 ;*****
2381 ; READ AND COMPARE CONTINUOUS BLOCKS OF DATA
2382 ; START AT 230
2383 ; THIS ROUTINE CAN BE USED TO ADJUST THE "SIGNAL MONO"
2384 ; AND THE "THRESHOLD POT".
2385 ;*****
2386 012532 012706 001100 RDSUB: MOV #STACK, SP ; KEEP THE STACK OUT OF THE WAY
2387 012536 013704 001220 MOV @#TACSL, TACS ; SETUP THE TA11 STATUS AND
2388 012542 013705 001224 MOV @#TADBL, TADB ; DATA BUFFER REGISTERS
2389 012546 000005 RESET ; RESET THE WORLD
2390 012550 012737 012532 001110 MOV #RDSUB, @#$LPERR ; SETUP THE LOOP ON ERROR ADDRESS
2391 012556 104422 SELDRV ; SELECT THE DRIVE
2392 012560 112714 000017 MOV #XRWD!GO, @TACS ; SEND TAPE TO "BOT"
2393 012564 032714 000040 100$: BIT #READY, @TACS ; WAIT ON READY
2394 012570 001775 BEQ 100$
2395 012572 104414 017016 1$: READ ,RLNKX ; READ A BLOCK
2396 012576 012706 001100 MOV #STACK, SP ; INIT. IN CASE OF ERROR
2397 012602 032714 020000 BIT #LEADER, @TACS ; AT "CLEAR LEADER"?
2398 012606 001351 BNE RDSUB ; BR IF YES---LOOP
2399 012610 104423 BLKCMP ; COMPARE THE READ AND WRITE BUFFERS
2400 012612 017016 RLINKX ; LINK TO THE READ BUFFER
2401 012614 017052 WLINKX ; LINK TO THE WRITE BUFFER
2402 012616 000765 BR 1$
2403
2404 ;*****
2405 ; WRITE A FILE GAP AND A BLOCK OF DATA FROM BOT TO EOT
2406 ; START AT 234
2407 ; THE PROGRAM WILL HALT THREE(3) TIMES
2408 ; AFTER EACH HALT SET THE SWR AND PRESS CONTINUE
2409 ; HALT 1 --- SWR<7:0> = NUMBER OF BYTES PER BLOCK
2410 ; HALT 2 --- SWR<7:0> = PATTERN DESIRED
2411 ; HALT 3 --- SWR<15:0> = OPERATIONAL SWITCH SETTINGS
2412 ; THIS ROUTINE CAN BE USED TO ADJUST THE "WRITE GAP MONO"
2413 ; AND THE "GAP TIME MONO".
2414
2415
```

```

2416
2417
2418 012620 004737 013116
2419 012624 012706 001100
2420 012630 013704 001220
2421 012634 013705 001224
2422 012640 000005
2423 012642 012737 012624 001110
2424 012650 104422
2425 012652 112714 000017
2426 012656 032714 000040
2427 012662 001775
2428 012664 104412
2429 012666 032714 020000
2430 012672 001005
2431 012674 104413 017052
2432 012700 032714 020000
2433 012704 001767
2434 012706 000000
2435 012710 000745
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445 012712 012706 001100
2446 012716 013704 001220
2447 012722 013705 001224
2448 012726 000005
2449 012730 012737 012712 001110
2450 012736 104422
2451 012740 112714 000017
2452 012744 032714 000040
2453 012750 001775
2454 012752 104414 017016
2455 012756 012706 001100
2456 012762 032714 020000
2457 012766 001351
2458 012770 104423
2459 012772 017016
2460 012774 017052
2461 012776 104420
2462 013000 032714 020000
2463 013004 001342
2464 013006 000761
2465
2466
2467
2468
2469
2470
2471

```

```

; THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS
;*****
WGPBLK: JSR PC, @SETBUF ;GET BLOCK SIZE AND PATTERN
WGBLOP: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY
MOV @TACSL, TACS ;SETUP THE TAII STATUS AND
MOV @TADBL, TADB ;DATA BUFFER REGISTERS
RESET ;RESET THE WORLD
MOV #WGBLOP, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
SELDRV ;SELECT THE DRIVE
MOVB #XRWIND!GO, @TACS ;SEND TAPE TO "BOT"
100$: BIT #READY, @TACS ;WAIT ON READY
BEQ 100$
1$: WFG ;WRITE A FILE GAP
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
BNE 2$ ;BR IF YES
WRITE ,WLNKX ;WRITE A BLOCK
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
BEQ 1$ ;BR IF NO
2$: HALT ;STOP AT "EOT"
BR WGBLOP ;START OVER ON CONT.

```

```

;*****
; READ AND COMPARE A BLOCK OF DATA AND A FILE GAP
; START AT 240
; THIS ROUTINE IS USED AFTER "WRITE A BLOCK AND A FILE GAP" ROUTINE
; IT CAN BE USED TO ADJUST THE "SIGNAL MONO", THE THRESHOLD POT"
; AND THE "TAPE BLANK MONO".
;*****
RGPBLK: MOV #STACK, SP ;KEEP THE STACK OUT OF THE WAY
MOV @TACSL, TACS ;SETUP THE TAII STATUS AND
MOV @TADBL, TADB ;DATA BUFFER REGISTERS
RESET ;RESET THE WORLD
MOV #RGPBLK, @SLPERR ;SETUP THE LOOP ON ERROR ADDRESS
SELDRV ;SELECT THE DRIVE
MOVB #XRWIND!GO, @TACS ;SEND TAPE TO "BOT"
100$: BIT #READY, @TACS ;WAIT ON READY
BEQ 100$
1$: READ ,RLNKX ;READ A BLOCK OF DATA
MOV #STACK, SP ;KEEP OUT OF THE WAY
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
BNE RGPBLK ;BR IF YES
BLKCMP ;COMPARE THE READ AND WRITE BUFFERS
RLNKX ;LINK TO THE READ BUFFER
WLNKX ;LINK TO THE WRITE BUFFER
SFBG ;GET INTO FILE GAP
BIT #LEADER, @TACS ;AT "CLEAR LEADER"?
BNE RGPBLK ;BR IF YES
BR 1$ ;LOOP

```

```

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

```

# J06

TAII MOTION TEST  
DZTADC.NEW

24-MAY-76 00:00

MAINDEC-11-DZTAD-C

MACY11 27(1006) 23-SEP-76 10:33 PAGE 58  
SPACE FORWARD FILE GAP FROM "BOT" TO "EOT"

SEQ 0074

```
2472                                     ;WRITE A FILE GAP AND A BLOCK OF DATA FOR HIGH SPEED SPACE FORWARD
2473                                     ;(SIGNAL MONO CAN BE CHECKED).
2474                                     ;*****
2475 013010 012706 001100 $FFGSB: MOV #STACK,SP ;KEEP THE STACK OUT OF THE WAY
2476 013014 013704 001220 MOV @#TACSL,TACS ;SETUP THE TAI1 STATUS AND
2477 013020 013705 001224 MOV @#TADBL,TADB ;DATA BUFFER REGISTERS
2478 013024 000005 RESET ;RESET THE WORLD
2479 013026 012737 013010 001110 MOV #SFFGSB,@#$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
2480 013034 104422 SELDRV ;SELECT THE DRIVE
2481 013036 112714 000017 MOVB #XRWND!GO,@TACS ;SEND TAPE TO "BOT"
2482 013042 032714 000040 100$: BIT #READY,@TACS ;WAIT ON READY
2483 013046 001775 BEQ 100$
2484 013050 104417 1$: SFFG ;SPACE INTO A FILE GAP
2485 013052 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2486 013056 001774 BEQ 1$ ;BF IF NO
2487 013060 000000 HALT ;STOP AT "EOT"
2488 013062 000752 BR SFFGSB ;LOOP ON CONT.
2489
2490
2491 ;*****
2492 ; BACK SPACE FILE GAP
2493 ; START AT 250
2494 ; THIS ROUTINE CAN BE USED TO ADJUST OR CHECK THE "SIGNAL MONO".
2495 ;*****
2496 013064 000005 BSFGSB: RESET ;RESET THE WORLD
2497 013066 012737 013064 001110 MOV #BSFGSB,@#$LPERR ;LOOP ON ERROR ADDRESS
2498 013074 005067 166072 CLR $ESCAPE ;DON'T ESCAPE ON ERROR
2499 013100 104422 SELDRV ;SELECT THE DRIVE
2500 013102 104415 1$: BSFG ;BACK SPACE A FILE GAP
2501 013104 032714 020000 BIT #LEADER,@TACS ;AT "CLEAR LEADER"?
2502 013110 001774 BEQ 1$ ;BR IF NO
2503 013112 000000 HALT ;STOP AT BOT
2504 013114 000753 BR BSFGSB ;START OVER ON CONT.
2505
2506
2507 ;*****
2508 ; SETUP READ AND WRITE LINKS FOR SUBROUTINES
2509 013116 005000 SETBUF: CLR RO
2510 013120 000000 HALT ;OPERATOR PUTS BYTE COUNT IN THE SWR
```

# K06

TA11 MOTION TEST      MAINDEC-11-DZTAD-C  
 DZTADC.NEW      24-MAY-76 00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 59  
 SETUP READ AND WRITE LINKS FOR SUBROUTINES

SEQ 0075

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



M06

TA11 MOTION TEST  
DZTADC.NEW 24-MAY-76 00:00

MAINDEC-11-DZTAD-C

MACY11 27(1006) 23-SEP-76 10:33 PAGE 61  
SETUP READ AND WRITE LINKS FOR SUBROUTINES

SEQ 0077

2535 013224 022767 000176 165706  
2536 013232 001001  
2537 013234 104405  
2538 013236 22\$:  
2539 013236 000207

CMP #SWREG,SWR  
BNE 22\$  
GTSWR  
RTS PC

:USING S/W SWITCH REG?  
:NO- GET OUT  
:GET VALUE  
:CONTINUE  
:RETURN

2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592  
2593  
2594  
2595

.SBTTL TYPE ROUTINE

\*\*\*\*\*  
\*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.  
\*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.  
\*NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.  
\*NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.  
\*NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.  
\*

\*CALL:  
\*1) USING A TRAP INSTRUCTION  
\* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING  
\*OR  
\* TYPE  
\* MESADR  
\*

013240	105767	165713	\$TYPE:	TSTB	\$STPFLG	:: IS THERE A TERMINAL?
013244	100002			BPL	1\$	:: BR IF YES
013246	000000			HALT		:: HALT HERE IF NO TERMINAL
013250	000407			BR	3\$	:: LEAVE
013252	010046		1\$:	MOV	RO,-(SP)	:: SAVE RO
013254	017600	000002		MOV	22(SP),RO	:: GET ADDRESS OF ASCIZ STRING
013260	112046		2\$:	MOVB	(RO)+,-(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK
013262	001005			BNE	4\$	:: BR IF IT ISN'T THE TERMINATOR
013264	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
013266	012600		60\$:	MOV	(SP)+,RO	:: RESTORE RO
013270	062716	000002	3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
013274	000002			RTI		:: RETURN
013276	122716	000011	4\$:	CMPB	#HT,(SP)	:: BRANCH IF <HT>
013302	001430			BEQ	8\$	
013304	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
013310	001006			BNE	5\$	
013312	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
013314	104401			TYPE		:: TYPE A CR AND LF
013316	001201			\$CRLF		
013320	105067	000130		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
013324	000755			BR	2\$	:: GET NEXT CHARACTER
013326	004767	000056	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
013332	126726	165620	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
013336	001350			BNE	2\$	:: IF NO GO GET NEXT CHAR.
013340	016746	165610		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
013344	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
013350	002770			BLT	6\$	:: BR IF NO--GO POP THE NULL OFF OF STACK
013352	004767	000032		JSR	PC,\$TYPEC	:: GO TYPE A NULL
013356	105367	000072		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT
013362	000770			BR	7\$	:: LOOP

;HORIZONTAL TAB PROCESSOR

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

```

013406 000724 BR 25 :: GET NEXT CHARACTER
013410 105777 165534 $TYPEC: TSTB 3$TSPS :: WAIT UNTIL PRINTER IS READY
013414 100375 BPL $TYPEC
013416 116677 000002 165526 MOVB 2(SP), 3$TPB :: LOAD CHAR TO BE TYPED INTO DATA REG.
013418 122766 000015 000002 CMPB #CR, 2(SP) :: IS CHARACTER A CARRIAGE RETURN?
013420 001003 BNE 1$ :: BRANCH IF NO
013422 105067 000014 CLRB $CHARCNT :: YES--CLEAR CHARACTER COUNT
013424 000406 BR $TYPEX :: EXIT
013426 122766 000012 000002 1$: CMPB #LF, 2(SP) :: IS CHARACTER A LINE FEED?
013428 001403 BEQ $TYPEX :: BRANCH IF YES
013430 105227 INCB (PC)+ :: COUNT THE CHARACTER
013432 000000 $CHARCNT: .WORD 0 :: CHARACTER COUNT STORAGE
013434 000207 $TYPEX: RTS PC

```

.SBTTL TTY INPUT ROUTINE

::\*\*\*\*\*  
:ENABL LSB

::\*\*\*\*\*  
:\*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.  
:\*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL  
:\*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL  
:\*WHEN OPERATING IN TTY FLAG MODE.

```

013460 022767 000176 165452 $KSWR: CMP #SWREG, SWR :: IS THE SOFT-SWR SELECTED?
013462 001074 BNE 15$ :: BRANCH IF NO
013464 105777 165450 TSTB 3$TKS :: CHAR THERE?
013466 100071 BPL 15$ :: IF NO, DON'T WAIT AROUND
013468 117746 165444 MOVB 3$TKB, -(SP) :: SAVE THE CHAR
013470 042716 177600 BIC #C177, (SP) :: STRIP-OFF THE ASCII
013472 022726 000007 CMP #7, (SP)+ :: IS IT A CONTROL G?
013474 001062 BNE 15$ :: NO, RETURN TO USER
013476 126727 165414 000001 CMPB $AUTOB, #1 :: ARE WE RUNNING IN AUTO-MODE?
013478 001456 BEQ 15$ :: BRANCH IF YES

```

```

013524 104401 014205 $GTSWR: TYPE , $CNTLG :: ECHO THE CONTROL-G (!G)
013530 104401 014212 TYPE , $MSWR :: TYPE CURRENT CONTENTS
013534 016746 164436 MOV SWREG, -(SP) :: SAVE SWREG FOR TYPEOUT
013540 104402 TYPOC :: GO TYPE--OCTAL ASCII(ALL DIGITS)
013542 104401 014223 TYPE , $MNEW :: PROMPT FOR NEW SWR
013546 005046 19$: CLR -(SP) :: CLEAR COUNTER
013550 005046 7$: CLR -(SP) :: THE NEW SWR
013552 105777 165366 TSTB 3$TKS :: CHAR THERE?
013556 100375 BPL 7$ :: IF NOT TRY AGAIN

```

```

013560 117746 165362 MOVB 3$TKB, -(SP) :: PICK UP CHAR
013564 042716 177600 BIC #C177, (SP) :: MAKE IT 7-BIT ASCII

```

```

013570 021627 000025 9$: CMP (SP), #25 :: IS IT A CONTROL-U?
013574 001005 BNE 10$ :: BRANCH IF NOT
013576 104401 014200 TYPE , $CNTLU :: YES, ECHO CONTROL-U (!U)
013602 062706 000006 20$: ADD #6, SP :: IGNORE PREVIOUS INPUT
013606 000757 BR 19$ :: LET'S TRY IT AGAIN

```



2652	013610	021627	000015	10\$:	CMP	(SP),#15	:: IS IT A <CR>?
2653	013614	001022			BNE	16\$	:: BRANCH IF NO
2654	013616	005766	000004		TST	4(SP)	:: YES, IS IT THE FIRST CHAR?
2655	013622	001403			BEQ	11\$	:: BRANCH IF YES
2656	013624	016677	000002	165306	MOV	2(SP),@SWR	:: SAVE NEW SWR
2657	013632	062706	000006	11\$:	ADD	#6,SP	:: CLEAR UP STACK
2658	013636	104401	001201	14\$:	TYPE	\$CRLF	:: ECHO <CR> AND <LF>
2659	013642	126727	165267	000001	CMPB	\$INTAG,#1	:: RE-ENABLE TTY KBD INTERRUPTS?
2660	013650	001003			BNE	15\$	:: BRANCH IF NOT
2661	013652	012777	000100	165264	MOV	#100,@\$TKS	:: RE-ENABLE TTY KBD INTERRUPTS
2662	013660	000002		15\$:	RTI		:: RETURN
2663	013662	004767	177522	16\$:	JSR	PC,\$TYPEC	:: ECHO CHAR
2664	013666	021627	000060		CMP	(SP),#60	:: CHAR < 0?
2665	013672	002420			BLT	18\$	:: BRANCH IF YES
2666	013674	021627	000067		CMP	(SP),#67	:: CHAR > 7?
2667	013700	003015			BGT	18\$	:: BRANCH IF YES
2668	013702	042726	000060		BIC	#60,(SP)+	:: STRIP-OFF ASCII
2669	013706	005766	000002		TST	2(SP)	:: IS THIS THE FIRST CHAR
2670	013712	001403			BEQ	17\$	:: BRANCH IF YES
2671	013714	006316			ASL	(SP)	:: NO, SHIFT PRESENT
2672	013716	006316			ASL	(SP)	:: CHAR OVER TO MAKE
2673	013716	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
2674	013720	006316			ASL	(SP)	::
2675	013722	005266	000002	17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
2676	013726	056616	177776		BIS	-2(SP),(SP)	:: SET IN NEW CHAR
2677	013732	000707			BR	7\$	:: GET THE NEXT ONE
2678	013734	104401	001200	18\$:	TYPE	\$QUES	:: TYPE ?<CR><LF>
2679	013740	000720			BR	20\$	:: SIMULATE CONTROL-U
2680				.DSABL	LSB		

::\*\*\*\*\*

::\*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

::\*CALL:

::\* RDCHR  
::\* RETURN HERE

::\* INPUT A SINGLE CHARACTER FROM THE TTY  
::\* CHARACTER IS ON THE STACK  
::\* WITH PARITY BIT STRIPPED OFF

2691	013742	011646		\$RDCHR:	MOV	(SP),-(SP)	:: PUSH DOWN THE PC	
2692	013744	016666	000004	000002	MOV	4(SP),2(SP)	:: SAVE THE PS	
2693	013752	105777	165166	1\$:	TSTB	@\$TKS	:: WAIT FOR	
2694	013756	100375			BPL	1\$	:: A CHARACTER	
2695	013760	117766	165162	000004	MOVB	@\$TKB,4(SP)	:: READ THE TTY	
2696	013766	042766	177600	000004	BIC	#1C(177),4(SP)	:: GET RID OF JUNK IF ANY	
2697	013774	026627	000004	000023	CMP	4(SP),#23	:: IS IT A CONTROL-S?	
2698	014002	001013			BNE	3\$	:: BRANCH IF NO	
2699	014004	105777	165134	2\$:	TSTB	@\$TKS	:: WAIT FOR A CHARACTER	
2700	014010	100375			BPL	2\$	:: LOOP UNTIL ITS THERE	
2701	014012	117746	165130		MOVB	@\$TKB,-(SP)	:: GET CHARACTER	
2702	014016	042716	177600		BIC	#1C177,(SP)	:: MAKE IT 7-BIT ASCII	
2703	014022	022627	000021		CMP	(SP)+,#21	:: IS IT A CONTROL-Q?	
2704	014026	001366			BNE	2\$	:: IF NOT DISCARD IT	
2705	014030	000750			BR	1\$	:: YES, RESUME	
2706	014032	026627	000004	000140	3\$:	CMP	4(SP),#140	:: IS IT UPPER CASE?
2707	014040	002407			BLT	4\$	:: BRANCH IF YES	

```

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

```

# E07

TAII MOTION TEST  
DZTADC.NEW 24-MAY-76

MAINDEC-11-DZTAD-C  
00:00

MACY1: 27(1006) 23-SEP-76 10:33 PAGE 66  
READ AN OCTAL NUMBER FROM THE TTY

SEQ 0082

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

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 15-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS

```

```

*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT

```

2812	014336	017646	000000		\$TYPOS: MOV	2(SP),-(SP)	;; PICKUP THE MODE
2813	014342	116667	000001	000211	MOV	1(SP), \$OFILL	;; LOAD ZERO FILL SWITCH
2814	014350	112667	000207		MOV	(SP)+, \$OMODE+1	;; NUMBER OF DIGITS TO TYPE
2815	014354	062716	000002		ADD	#2, (SP)	;; ADJUST RETURN ADDRESS
2816	014360	000406			BR	\$TYPON	
2817	014362	112767	000001	000171	\$TYPOC: MOV	#1, \$OFILL	;; SET THE ZERO FILL SWITCH
2818	014370	112767	000006	000165	MOV	#6, \$OMODE+1	;; SET FOR SIX(6) DIGITS
2819	014376	112767	000005	000154	\$TYPON: MOV	#5, \$OCNT	;; SET THE ITERATION COUNT
2820	014404	010346			MOV	R3, -(SP)	;; SAVE R3
2821	014406	010446			MOV	R4, -(SP)	;; SAVE R4
2822	014410	010546			MOV	R5, -(SP)	;; SAVE R5
2823	014412	116704	000145		MOV	\$OMODE+1, R4	;; GET THE NUMBER OF DIGITS TO TYPE
2824	014416	005404			NEG	R4	
2825	014420	062704	000006		ADD	#6, R4	;; SUBTRACT IT FOR MAX. ALLOWED
2826	014424	110467	000132		MOV	R4, \$OMODE	;; SAVE IT FOR USE
2827	014430	116704	000125		MOV	\$OFILL, R4	;; GET THE ZERO FILL SWITCH
2828	014434	016605	000012		MOV	12(SP), R5	;; PICKUP THE INPUT NUMBER
2829	014440	005003			CLR	R3	;; CLEAR THE OUTPUT WORD
2830	014442	006105		1\$:	ROL	R5	;; ROTATE MSB INTO "C"
2831	014444	000404			BR	3\$	;; GO DO MSB
2832	014446	006105		2\$:	ROL	R5	;; FORM THIS DIGIT
2833	014450	006105			ROL	R5	
2834	014452	006105			ROL	R5	
2835	014454	010503			MOV	R5, R3	
2836	014456	006103		3\$:	ROL	R3	;; GET LSB OF THIS DIGIT
2837	014460	105367	000076		DECB	\$OMODE	;; TYPE THIS DIGIT?
2838	014464	100016			BPL	7\$	;; BR IF NO
2839	014466	042703	177770		BIC	#177770, R3	;; GET RID OF JUNK
2840	014472	001002			BNE	4\$	;; TEST FOR 0
2841	014474	005704			TST	R4	;; SUPPRESS THIS 0?
2842	014476	001403			BEQ	5\$	;; BR IF YES

2843	014500	005204		4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
2844	014502	052703	000060		BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
2845	014506	052703	000040	5\$:	BIS	#',R3	:: MAKE ASCII IF NOT ALREADY
2846	014512	110367	000040		MOVB	R3,8\$	:: SAVE FOR TYPING
2847	014516	104401	014556		TYPE	8\$	:: GO TYPE THIS DIGIT
2848	014522	105367	000032	7\$:	DECB	\$OCNT	:: COUNT BY 1
2849	014526	003347			BGT	2\$	:: BR IF MORE TO DO
2850	014530	002402			BLT	6\$	:: BR IF DONE
2851	014532	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
2852	014534	000744			BR	2\$	:: GO DO THE LAST DIGIT
2853	014536	012605		6\$:	MOV	(SP)+,R5	:: RESTORE R5
2854	014540	012604			MOV	(SP)+,R4	:: RESTORE R4
2855	014542	012603			MOV	(SP)+,R3	:: RESTORE R3
2856	014544	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
2857	014552	012616			MOV	(SP)+,(SP)	
2858	014554	000002			RTI		:: RETURN
2859	014556	000		8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
2860	014557	000			.BYTE	00	:: TERMINATOR FOR TYPE ROUTINE
2861	014560	000		\$OCNT:	.BYTE	00	:: OCTAL DIGIT COUNTER
2862	014561	000		\$OFILL:	.BYTE	00	:: ZERO FILL SWITCH
2863	014562	000000		\$CMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901  
2902  
2903  
2904  
2905  
2906  
2907  
2908  
2909  
2910  
2911  
2912  
2913  
2914  
2915  
2916

.SBTTL TRAP DECODER

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

\$TRAP: MOV RO, -(SP) ;:SAVE RO  
MOV 2(SP),RO ;:GET TRAP ADDRESS  
TST -(RO) ;:BACKUP BY 2  
MOVB (RO),RO ;:GET RIGHT BYTE OF TRAP  
ASL RO ;:POSITION FOR INDEXING  
MOV \$TRPAD(RO),RO ;:INDEX TO TABLE  
RTS RO ;:GO TO ROUTINE

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

\$TRAP2: MOV (SP), -(SP) ;:MOVE THE PC DOWN  
MOV 4(SP), 2(SP) ;:MOVE THE PSW DOWN  
RTI ;:RESTORE THE PSW

.SBTTL TRAP TABLE

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

ROUTINE	STARTING ADDRESS	CALL	ROUTINE NAME
\$TRPAD	014620	CALL=TYPE	TTY TYPEOUT ROUTINE
\$TRPAD	014622	CALL=TYPOC	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TRPAD	014624	CALL=TYPOS	TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TRPAD	014626	CALL=TYPON	TYPE OCTAL NUMBER (AS PER LAST CALL)
\$GTSWR	014632	CALL=GTSWR	GET SOFT-SWR SETTING
\$CKSWR	014634	CALL=CKSWR	TEST FOR CHANGE IN SOFT-SWR
\$RDCHR	014636	CALL=RDCHR	TTY TYPEIN CHARACTER ROUTINE
\$RDLIN	014640	CALL=RDLIN	TTY TYPEIN STRING ROUTINE
\$RDOCT	014642	CALL=RDOCT	READ AN OCTAL NUMBER FROM TTY
T.WFG	014644	CALL=WFG	ROUTINE TO WRITE A FILE GAP
T.WRIT	014646	CALL=WRITE	ROUTINE TO WRITE A BLOCK OF DATA
T.READ	014650	CALL=READ	ROUTINE TO READ A BLOCK OF DATA
T.BSFG	014652	CALL=BSFG	ROUTINE TO BACK SPACE A FILE GAP
T.BSBG	014654	CALL=BSBG	ROUTINE TO BACK SPACE A BLOCK GAP
T.SFFG	014656	CALL=SFFG	ROUTINE TO SPACE FWD A FILE GAP
T.SFBG	014660	CALL=SFBG	ROUTINE TO SPACE FWD A BLOCK GAP
T.RWND	014662	CALL=REWIND	ROUTINE TO REWIND THE TAPE
T.SELDRV	014664	CALL=SELDRV	ROUTINE TO SELECT THE NEXT DRIVE
T.BLKCMP	014666	CALL=BLKCMP	COMPARE READ AND WRITE BUFFERS
T.SETLOOP	014670	CALL=SETLOOP	SETUP TO LOOP AS PER THE TTY

.SBTTL POWER DOWN AND UP ROUTINES

2917  
2918  
2919  
2920  
2921  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937  
2938  
2939  
2940  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951  
2952  
2953  
2954  
2955  
2956  
2957  
2958  
2959  
2960  
2961

014672 012737 015036 000024  
014700 012737 000340 000026  
014706 010046  
014710 010146  
014712 010246  
014714 010346  
014716 010446  
014720 010546  
014722 017746 164212  
014726 010667 000110  
014732 012737 014744 000024  
014740 000000  
014742 000776  
  
014744 012737 015036 000024  
014752 016706 000064  
014756 005067 000060  
014762 005267 000054  
014766 001375  
014770 012677 164144  
014774 012605  
014776 012604  
015000 012603  
015002 012602  
015004 012601  
015006 012600  
015010 012737 014672 000024  
015016 012737 000340 000026  
015024 104401  
015026 015044  
015030 012716  
015032 002314  
015034 000002  
015036 000000  
015040 000776  
015042 000000  
015044 005015 047520 042527  
015052 000122

\*\*\*\*\*

:POWER DOWN ROUTINE

\$PWRDN: MOV \$SILLUP, @#PWRVEC ;; SET FOR FAST UP  
MOV #340, @#PWRVEC+2 ;; PRIO:7  
MOV RO, -(SP) ;; PUSH RO ON STACK  
MOV R1, -(SP) ;; PUSH R1 ON STACK  
MOV R2, -(SP) ;; PUSH R2 ON STACK  
MOV R3, -(SP) ;; PUSH R3 ON STACK  
MOV R4, -(SP) ;; PUSH R4 ON STACK  
MOV R5, -(SP) ;; PUSH R5 ON STACK  
MOV @SWR, -(SP) ;; PUSH @SWR ON STACK  
MOV SP, \$SAVR6 ;; SAVE SP  
MOV #PWRUP, @#PWRVEC ;; SET UP VECTOR  
HALT  
BR -2 ;; HANG UP

\*\*\*\*\*

:POWER UP ROUTINE

\$PWRUP: MOV \$SILLUP, @#PWRVEC ;; SET FOR FAST DOWN  
MOV \$SAVR6, SP ;; GET SP  
CLR \$SAVR6 ;; WAIT LOOP FOR THE TTY  
1\$: INC \$SAVR6 ;; WAIT FOR THE INC  
BNE 1\$ ;; OF WORD  
MOV (SP)+, @SWR ;; POP STACK INTO @SWR  
MOV (SP)+, R5 ;; POP STACK INTO R5  
MOV (SP)+, R4 ;; POP STACK INTO R4  
MOV (SP)+, R3 ;; POP STACK INTO R3  
MOV (SP)+, R2 ;; POP STACK INTO R2  
MOV (SP)+, R1 ;; POP STACK INTO R1  
MOV (SP)+, R0 ;; POP STACK INTO R0  
MOV #PWRDN, @#PWRVEC ;; SET UP THE POWER DOWN VECTOR  
MOV #340, @#PWRVEC+2 ;; PRIO:7  
TYPE \$POWER ;; REPORT THE POWER FAILURE  
\$PWRMG: .WORD \$POWER ;; POWER FAIL MESSAGE POINTER  
MOV (PC)+, (SP) ;; RESTART AT PWRST  
\$PWRAD: .WORD PWRST ;; RESTART ADDRESS  
RTI  
\$SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED  
BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE  
\$SAVR6: 0 ;; PUT THE SP HERE  
\$POWER: .ASCIZ <15><12>"POWER"  
  
.EVEN

```

2962      ;*****
2963
2964      ;DATA POINTERS
2965
2966      015054 001204 001116 001162 DT1:  .WORD  SAVPC,$ERRPC,$REGO,$REG1,0
2967      015062 001164 000000
2968
2969      015066 001204 001116 001162 DT2:  .WORD  SAVPC,$ERRPC,$REGO,$GDDAT,$BDDAT,BYTNUM,0
2970      015074 001124 001126 001206
2971      015102 000000
2972
2973      015104 001204 001116 001162 DT3:  .WORD  SAVPC,$ERRPC,$REGO,RCRCO,RCRC1,RCRC2,0
2974      015112 001210 001212 001214
2975      015120 000000
2976
2977      015122 001204 001116 001162 DT4:  .WORD  SAVPC,$ERRPC,$REGO,$REG1,$TMPO,0
2978      015130 001164 001166 000000
2979
2980      015136 001204 001116 001162 DT5:  .WORD  SAVPC,$ERRPC,$REGO,0
2981      015144 000000
2982

```



```

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

```

TA11 MOTION TEST  
DZTADC.NEW 24-MAY-76MAINDEC-11-DZTAD-C  
00:00MACY11 27(1006) 23-SEP-76 10:33 PAGE 73  
ASCII MESSAGES

SEQ 0089

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

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

DH2: .ASCII <15><12>  
 .ASCII /TEST ERROR EXPT'D RCV'D BYTE/<15><12>

.ASCIZ /PC PC TACS DATA DATA NUMBER/<15><12>

DH3: .ASCII <15><12>  
 .ASCII /TEST ERROR WRITE READ TU60/<15><12>

.ASCIZ /PC PC TACS CRC CRC CRC/<15><12>

DH4: .ASCII <15><12>  
 .ASCII /TEST ERROR BYTES/<15><12>

.ASCIZ /PC PC TACS TABS LEFT/<15><12>

DH5: .ASCII <15><12>  
 .ASCII /TEST ERROR/<15><12>

.ASCIZ /PC PC TACS/<15><12>

N07

TA11 MOTION TEST  
DZTADC.NEW 24-MAY-76

MAINDEC-11-DZTAD-C  
00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 75  
ASCII MESSAGES

SEQ 0091

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

0171  
0172  
0173  
0174  
0175  
0176  
0177  
0178  
0179  
0180  
0181  
0182  
0183  
0184  
0185  
0186  
0187  
0188  
0189  
0190  
0191  
0192  
0193  
0194  
0195  
0196  
0197  
0198  
0199  
0200  
0201  
0202  
0203  
0204  
0205  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215  
0216  
0217  
0218  
0219  
0220  
0221  
0222  
0223  
0224  
0225  
0226

016766 017130 000200  
016776 017232 000200  
016777 017234 000010  
017000 017736 000040  
017000 020140 000020  
017010 020342 000100  
017016 017130 000000  
017020 017056 000200  
017026 017064 000200  
017030 017072 000010  
017036 017100 000040  
017040 017106 000020  
017046 017114 000100  
017052 017122 000000

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

RLNK1: RBUF1,128.  
RLNK2: RBUF2,128.  
RLNK3: RBUF3,8.  
RLNK4: RBUF4,32.  
RLNK5: RBUF5,16.  
RLNK6: RBUF6,64.  
RLNKX: RBUF1,0  
WLNK1: WBUF1,128.  
WLNK2: WBUF2,128.  
WLNK3: WBUF3,8.  
WLNK4: WBUF4,32.  
WLNK5: WBUF5,16.  
WLNK6: WBUF6,64.  
WLNKX: WBUF6,0

:BUFFERS

WBUF1: .BYTE ↑B11001100  
          .BYTE ↑B00110011  
          .BYTE ↑B11001100  
          .BYTE ↑B00110011  
WCRC1: .WORD 0 ;CRC STORAGE FOR WBUF1  
  
WBUF2: .BYTE ↑B11111111  
          .BYTE ↑B00000000  
          .BYTE ↑B11111111  
          .BYTE ↑B00000000  
WCRC2: .WORD 0 ;CRC STORAGE FOR WBUF2  
  
WBUF3: .BYTE ↑B10011001  
          .BYTE ↑B01100110  
          .BYTE ↑B10011001  
          .BYTE ↑B01100110  
WCRC3: .WORD 0 ;CRC STORAGE FOR WBUF3  
  
WBUF4: .BYTE ↑B11110000  
          .BYTE ↑B00001111  
          .BYTE ↑B11110000  
          .BYTE ↑B00001111  
WCRC4: .WORD 0 ;CRC STORAGE FOR WBUF4  
  
WBUF5: .BYTE ↑B10101010  
          .BYTE ↑B01010101  
          .BYTE ↑B10101010  
          .BYTE ↑B01010101  
WCRC5: .WORD 0 ;CRC STORAGE FOR WBUF5

017056 000002  
017056 314  
017057 063  
017060 314  
017061 063  
017062 000000  
  
017064 000002  
017064 377  
017065 000  
017066 377  
017067 000  
017070 000000  
  
017072 000002  
017072 231  
017073 146  
017074 231  
017075 146  
017076 000000  
  
017100 000002  
017100 360  
017101 017  
017102 360  
017103 017  
017104 000000  
  
017106 000002  
017106 252  
017107 125  
017110 252  
017111 125  
017112 000000

```

017114 000001
017115 377
017116 377
017117 000
017118 000
017120 000000
017122 000004
017123 000
017124 000
017125 000
017126 000000
017130 000202
017332 000202
017534 000202

```

```

WBUF6:
  .BYTE 1811111111
  .BYTE 1811111111
  .BYTE 1800000000
  .BYTE 1800000000
WCR6:  .WORD 0 ;CRC STORAGE FOR WBUF6
WBUFx:
  .BYTE 0
  .BYTE 0000
  .BYTE 0000
  .BYTE 0000
WCRcx:  .WORD 0
RBUF1:  .BLKB 130.
RBUF2:  .BLKB 130.
RBUF3:  .BLKB 130.

```

D08

TAII MOTION TEST  
DZTADC.NEW 24-MAY-76

MAINDEC-11-DZTAD-C  
00:00

MACY11 27(1006) 23-SEP-76 10:33 PAGE 78  
READ AND WRITE BUFFERS AND LINKS

SEQ 0094

017736	000202
020140	000202
020342	000202
020544	
000001	

RBUF4:	.BLKB	130.
RBUF5:	.BLKB	130.
RBUF6:	.BLKB	130.
LASTADDRESS=		
	.END	





# FO8

CRC.WD	012232	1958*	1986	2011*	2044	2244	2265*	2272#							
CRLF =	000200	97#	511	2571	2610	2987	3004								
CURDRV	001250	313#	632*	634	641*	544*	646	699	1889*						
DDISP =	177570	93#	275	482											
DH1	016112	338	345	352	359	366	373	380	3085#						
DH2	016172	387	3095#												
DH201	016746	423	3166#												
DH202	016763	428	3169#												
DH3	016333	401	3113#												
DH4	016471	408	3130#												
DH5	016611	394	415	3145#											
DISPLA	001142	275#	482*	490*	1607*	1643*									
DISPRE	000174	231#	490												
DO.CMD	012234	1940	1963	1987	2012	2045	2069	2090	2105	2126	2154	2281#			
DO.CRC	012112	1981	2031	2237#											
DRIVE	001236	309#	629*	1888	1901*										
DRVKEY	001240	310#	311	443*	585	598	619	637	643	681	1717*	1722	1736	1740	
		1900													
DRVPNT	001244	311#	599*	619*	628	647*									
DSWR =	177570	92#	274	481											
DT1	015054	340	347	354	361	368	375	382	2966#						
DT2	015066	389	2969#												
DT201	016662	424	3154#												
DT202	016670	429	3156#												
DT3	015104	403	2973#												
DT4	015122	410	2977#												
DT5	015136	396	417	2980#											
EMTVEC=	000030	181#	465*	466*											
EM1	015550	337	3041#												
EM10	016003	386	3070#												
EM11	016021	393	3073#												
EM12	016045	400	3077#												
EM13	016057	407	3079#												
EM14	016074	414	3082#												
EM2	015576	344	3045#												
EM201	016674	422	3158#												
EM202	016723	427	3162#												
EM3	015625	351	3049#												
EM4	015643	358	3052#												
EM5	015664	365	3055#												
EM6	015715	372	3060#												
EM7	015747	379	3065#												
ERROR =	100000	199#													
ERRVEC=	000004	174#	479	480*	491*	547*	553*	1573	1574*	1576*	1579*				
EXAM	006474	586	590	595	1522#										
EXIT	007532	1735	1740#												
FATAL	002716	656	675	694#											
FGAP =	004000	203#	2057	2080	2116										
FLAGS	012024	1946	1971	1993	2021	2053	2075	2096	2111	2132	2144	2160	2211#		
FUNCTI=	000016	215#	2284	2299											
FUNCO =	000002	213#	215												
FUNC1 =	000004	212#	215												
FUNC2 =	000010	211#	215												
GNS =	***** U	230	510	697	2895	2896	2897	2898	2900	2902	2903	2904	2905	2906	
		2907	2908	2909	2910	2911	2912	2913	2914	2915	2916				
GO =	000001	214#	669	1941	1964	2013	2070	2091	2106	2127	2155	2284	2294	2343	

















\$TRAP	014564	467	2872#																	
\$TRAP2	014606	2883#	2894																	
\$TRP =	000025	2887#	2896#	2897#	2898#	2899#	2900	2901#	2902	2903#	2904#	2905#	2906#	2907#						
		2908#	2909#	2910#	2911#	2912#	2913#	2914#	2915#	2916#	2917#									
\$TRPAD	014620	2877	2894#																	
\$TSTNM	001102	256#	625*	1488*	1549	1602*	1607	1610	1643	1672										
\$TTYIN	014170	2720	2721	2738	2742#															
\$TYPBN=	***** U	2899																		
\$TYPDS=	***** U	2899																		
\$TYPE	013240	2557#	2887	2895																
\$TYPEC	013410	2578	2585	2592	2597#	2598	2664													
\$TYPEX	013456	2603	2605	2608#																
\$TYPOC	014362	2817#	2896																	
\$TYPON	014376	2816	2819#	2898																
\$TYPOS	014336	2812#	2897																	
\$XTSTR	006640	1571#																		
\$SGET4=	000000	1501#																		
\$OFILL	014561	2913*	2817*	2827	2862#															
\$40CAT=	***** U	1568	1653																	
.	= 020544	226#	230#	253#	295	460	475	476	511#	1509	1512	1610	1672	1707#						
		2610	2613	2742#	2743	2749	2933	2957	3153#	3243#	3244#	3245#	3246#	3247#						
		3248#	3250																	

. ABS. 020544 000

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

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

