

# TA11

DATA RELIABILITY  
MD-11-DZTAE-B

EP-DZTAE-B-DL-A  
COPYRIGHT © 1976  
FICHE 1 OF 1

NOV 1976  
**digital**  
MADE IN USA

.REM %

TA11 DATA RELIABILITY  
DZTAE B.P11  
MAINDEC-11-DZTAE-B  
MACY11 27(732) 25-SEP-76 11:17 PAGE 2

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAE-B-D  
PRODUCT NAME: TA11 DATA RELIABILITY  
DATE CREATED: 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
  - 6.1 ERROR TYPES
  - 6.2 DATA ERRORS
  - 6.3 ERROR RECOVERY
- 7. RESTRICTIONS
- 8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 END OF TEST
  - 8.4 DRIVE COMPATIBILITY
  - 8.5 DATA FORMAT
  - 8.6 TEST TYPEOUT
- 9. PROGRAM DESCRIPTION
  - 9.1 FORMAT PASS
  - 9.2 READ ONLY PASS
  - 9.3 WRITE ONLY PASS

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

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

1. ABSTRACT

THIS PROGRAM COLLECTS STATISTICAL INFORMATION PERTAINING TO THE DATA RELIABILITY OF THE TA11/TU60 WHEN RUN FOR EXTENDED PERIODS OF TIME. IT USES A NUMBER OF DIFFERENT PARAMETERS CONTROLLING THE DATA PATTERNS, THE NUMBER OF BYTES PER BLOCK (RECORD) AND THE NUMBER OF BLOCKS PER FILE.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TA11 CASSETTE

2.2 STORAGE

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.

2.3 PRELIMINARY PROGRAMS

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

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

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

4.3 PROGRAM & OPERATOR ACTION  
-----

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A WRITE ENABLED CASSETTE IN BOTH DRIVES
3. LOAD ADDRESS 200.
4. SET SWITCHES (SEE SECTION 4.1)
5. PRESS START.

\*\*\* 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.

STARTING THE PROGRAM AT 204 OR 210 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	:DRIVES A AND B SELECTED
DRIVE(S)? AB	:DRIVES A AND B SELECTED
DRIVE(S)? B,A	:DRIVES B AND A SELECTED
DRIVE(S)? B	:DRIVE B SELECTED
DRIVE(S)? SB?	:DRIVE S IS IMPOSSIBLE
DRIVE(S)?	:ASK FOR DRIVE

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

4.3.2 ADDRESS SELECTION

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

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

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

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONDES WITH A "Y" OR A "CARRIAGE RETURN" FOR "YES" ANYTHING ELSE IS A "NO".

4.3.2.1 ADDRESS SELECTION EXAMPLES

DRIVES(S) A  
TACS? 177500  
VECTOR? 260  
PRIORITY? 6  
TACS=177500 TADB=177502 VECTOR=000260 PRIORITY=000300  
OK?

DRIVES(S) A,B  
TACS? 470  
VECTOR?  
PRIORITY?  
TACS=177470 TADB=177472 VECTOR=000260 PRIORITY=000300  
OK?

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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

CONTROL:

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

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

WITH SW<15:10>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST.  
THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR  
SW<14>=1...LOOP ON TEST  
SW<13>=1...INHIBIT ERROR TYPEOUTS  
SW<10>=1...RING BELL ON ERROR  
SW<09>=1...HALT AFTER NEXT "END-OF-TEST" TYPEOUT  
SW<08>=1...AT NEXT "END-OF-TAPE" (EOT) GO TO "END-OF-TEST"  
SW<07>=1...PERFORM PASS AS PER SWR<1:0>  
SWR<1:0>=00=FORMAT  
SWR<1:0>=01=READ ONLY  
SWR<1:0>=10=WRITE ONLY  
SWR<1:0>=11=READ ONLY

220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273

274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318

## 5.2 SUBROUTINE ABSTRACTS

-----

### 5.2.1 SCOPE

-----

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED AT AN OPTIMUM POSITION IN THE INSTRUCTION SECTION OF THE "FORMAT", "READONLY" AND "WRITEONLY" CODE.

IF SWR<14>=1 THE PROGRAM WILL LOOP THROUGH A SPECIFIC SEQUENCE DEPENDING ON THE TYPE OF PASS BEING PERFORMED.  
\*\*\* THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

#### 5.2.1.1 FORMAT PASS SCOPE LOOP

1. SETUP FOR A WRITE
2. WRITE
3. BACKSPACE BLOCK GAP
4. SETUP FOR A READ
5. READ
6. REPEAT STEPS 1-5 UNTIL "EOT"

#### 5.2.1.2 READONLY PASS SCOPE LOOP

1. SETUP FOR READ
2. READ
3. CHECK FOR SYNC & DATA ERROR
4. BACK SPACE BLOCK GAP
5. REPEAT STEPS 2-4 INDEFINITELY

#### 5.2.1.3 WRITEONLY PASS SCOPE LOOP

1. SETUP FOR WRITE
2. WRITE
3. REPEAT STEP 2 UNTIL "EOT"

### 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.



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

5.2.3 ERROR

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

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

7

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  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A 'JSR'.  
-----

5.2.6.1 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.2 ASKDRV

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

5.2.6.3 ASKADR

THIS ROUTINE IS USED TO INPUT THE ADDRESSES OF THE "TACS"  
AND THE VECTOR TO USE.

5.2.6.4 TYPERR

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

5.2.6.5 CSRERR

THIS ROUTINE IS USED WHEN AN ERROR  
IS DETECTED. IT WILL EXAMINE THE  
"CONTROL AND STATUS" REGISTER TO  
DETERMINE THE TYPE OF ERROR AND  
TAKE THE APPROPRIATE ACTION.

5.2.6.6 SETUPW

THIS ROUTINE IS USED TO SETUP THE  
PARAMETER BLOCK AND THE WRITE  
BUFFER BEFORE STARTING A "WRITE" FUNCTION

5.2.6.7 FILL

USE TO FILL THE WRITE BUFFER  
WITH A DATA PATTERN.

5.2.6.8 SETUPR

THIS ROUTINE IS USED TO SETUP THE  
PARAMETER BLOCK BEFORE DOING A  
"READ" FUNCTION.

5.2.6.9 SYNCK

THIS ROUTINE IS CALLED AFTER  
PERFORMING A "READ".  
IT CHECKS THE FIRST FOUR BYTES  
OF THE DATA TO INSURE THAT THEY  
CONTAIN THE PROPER FILE AND  
BLOCK NUMBERS.

402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457

## 5.2.6.10 DATCMP

THIS ROUTINE IS USED TO CHECK THE  
DATA IN THE READ BUFFER TO INSURE  
IT IS CORRECT.

## 5.2.6.11 CNTSFT

THIS ROUTINE IS USED TO  
COUNT SOFT DATA ERRORS

## 5.2.6.12 CNTHRD

THIS ROUTINE IS USED TO  
COUNT HARD DATA ERRORS

## 5.2.6.13 SAVRGI OR SAVREG

ROUTINE TO SAVE ALL THE REGISTERS

## 5.2.6.14 CASSETTE PRIMITIVE

THIS IS THE CASSETTE DRIVER

## 5.2.6.15 CASINT

CASSETTE INTERRUPT HANDLER

## 5.2.6.16 DBCD

CHANGES A DOUBLE LENGTH BINARY  
NUMBER TO A DECIMAL ASCIZ STRING

## 5.2.6.17 SBCD

CHANGES A SINGLE LENGTH BINARY  
NUMBER TO A DECIMAL ASCIZ STRING.

## 5.2.6.18 SUPRES

TYPES A DECIMAL ASCII STRING  
SUPPRESSING LEADING ZEROS.

## 5.2.6.19 TYPF

USED TO TYPE THE ASCIZ STRING  
IMMEDIATELY FOLLOWING THE CALL.

## 5.2.6.20 TPDRV

TYPES THE DRIVE TO BE TESTED

## 5.2.6.21 EXAM

L01

TA11 DATA RELIABILITY  
DZTAEB.P11

MAINDEC-11-DZTAE-B

MACY11 27(732) 25-SEP-76 11:17 PAGE 12

458  
459  
460  
461  
462  
463

THIS ROUTINE EXAMINES THE SELECTED DRIVES  
TO INSURE THEY ARE AVAILABLE FOR TESTING.

5.2.6.22 \$B2OCT

TYPES AN OCTAL NUMBER

464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519

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

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

6.1 ERROR TYPES

-----  
THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO THREE (3) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

6.1.1 PRETEST ERROR

THESE ERRORS WILL BE DETECTED BEFORE TRYING TO TEST THE DATA RELIABILITY OF THE TA11/TU60.

6.1.2 NON-FATAL ERROR

THESE ERRORS WILL BE DUE TO "CRC" OR "DATA" FAILURES WHICH WILL BE REPORTED AS THEY OCCUR. AFTER REPORTING THE ERROR THE PROGRAM WILL CONTINUE TESTING.

6.1.3 FATAL ERROR

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND OF ERROR THAT CAUSES THE PROGRAM TO LOSE TRACK OF THE TAPE POSITION, OR THE MAXIMUM NUMBER OF DATA ERRORS HAVE OCCURRED.

THIS ERROR WILL BE REPORTED WHEN IT OCCURS, THEN THE PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

6.2 DATA ERRORS

-----  
THERE ARE TWO TYPES OF DATA ERRORS THAT CAN OCCUR WHICH ARE DEFINED AND EXPLAINED AS FOLLOWS:

6.2.1 SOFT ERROR

A SOFT ERROR IS BY DEFINITION ANY "CRC" OR "READ DATA" ERROR THAT OCCURS WHILE READING A BLOCK OF DATA. A SOFT ERROR WILL INVOKE A REREAD OF THE BLOCK.

520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557

6.2.2 HARD ERROR

A HARD ERROR IS DEFINED AS ANY  
"CRC" OR "READ DATA" ERROR THAT OCCURS  
ON THE INITIAL READ OF A BLOCK  
OF DATA AND CAN NOT BE READ  
CORRECTLY WITHIN THREE (3) RETRYS.

6.3 ERROR RECOVERY  
-----

6.3.1 PRETEST ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THEN DEPENDING  
ON HOW THE PROGRAM WAS STARTED IT WILL ASK FOR THE DRIVES AND  
ADDRESSES FOR TESTING OR RETURN TO MONITOR.

6.3.2 NON-FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED AND  
THE PROGRAM WILL CONTINUE IN TEST.

6.3.3 FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THE  
PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

7. RESTRICTIONS  
-----

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

8. MISCELLANEOUS  
-----

8.1 EXECUTION TIME  
-----

TESTING THE TA11/TU60 TO SPECIFICATION TAKES APPROXIMATELY  
2 HOURS 30 MINUTES WITH EACH DRIVE TAKING 75 MINUTES.

8.2 STACK POINTER  
-----

STACK IS INITIALLY SET TO 1100.

61.0  
61.1  
61.2  
61.3  
61.4  
61.5  
61.6  
61.7  
61.8  
61.9  
62.0  
62.1  
62.2  
62.3  
62.4  
62.5  
62.6  
62.7  
62.8  
62.9  
63.0  
63.1  
63.2  
63.3  
63.4  
63.5  
63.6  
63.7  
63.8  
63.9  
64.0  
64.1  
64.2  
64.3  
64.4  
64.5  
64.6  
64.7  
64.8  
64.9  
65.0  
65.1  
65.2  
65.3  
65.4  
65.5  
65.6  
65.7  
65.8  
65.9  
66.0  
66.1  
66.2  
66.3  
66.4  
66.5  
66.6  
66.7  
66.8  
66.9  
67.0  
67.1  
67.2  
67.3  
67.4  
67.5  
67.6  
67.7  
67.8  
67.9  
68.0  
68.1  
68.2  
68.3  
68.4  
68.5  
68.6  
68.7  
68.8  
68.9  
69.0  
69.1  
69.2  
69.3  
69.4  
69.5  
69.6  
69.7  
69.8  
69.9  
70.0  
70.1  
70.2  
70.3  
70.4  
70.5  
70.6  
70.7  
70.8  
70.9  
71.0  
71.1  
71.2  
71.3  
71.4  
71.5  
71.6  
71.7  
71.8  
71.9  
72.0  
72.1  
72.2  
72.3  
72.4  
72.5  
72.6  
72.7  
72.8  
72.9  
73.0  
73.1  
73.2  
73.3  
73.4  
73.5  
73.6  
73.7  
73.8  
73.9  
74.0  
74.1  
74.2  
74.3  
74.4  
74.5  
74.6  
74.7  
74.8  
74.9  
75.0  
75.1  
75.2  
75.3  
75.4  
75.5  
75.6  
75.7  
75.8  
75.9  
76.0  
76.1  
76.2  
76.3  
76.4  
76.5  
76.6  
76.7  
76.8  
76.9  
77.0  
77.1  
77.2  
77.3  
77.4  
77.5  
77.6  
77.7  
77.8  
77.9  
78.0  
78.1  
78.2  
78.3  
78.4  
78.5  
78.6  
78.7  
78.8  
78.9  
79.0  
79.1  
79.2  
79.3  
79.4  
79.5  
79.6  
79.7  
79.8  
79.9  
80.0  
80.1  
80.2  
80.3  
80.4  
80.5  
80.6  
80.7  
80.8  
80.9  
81.0  
81.1  
81.2  
81.3  
81.4  
81.5  
81.6  
81.7  
81.8  
81.9  
82.0  
82.1  
82.2  
82.3  
82.4  
82.5  
82.6  
82.7  
82.8  
82.9  
83.0  
83.1  
83.2  
83.3  
83.4  
83.5  
83.6  
83.7  
83.8  
83.9  
84.0  
84.1  
84.2  
84.3  
84.4  
84.5  
84.6  
84.7  
84.8  
84.9  
85.0  
85.1  
85.2  
85.3  
85.4  
85.5  
85.6  
85.7  
85.8  
85.9  
86.0  
86.1  
86.2  
86.3  
86.4  
86.5  
86.6  
86.7  
86.8  
86.9  
87.0  
87.1  
87.2  
87.3  
87.4  
87.5  
87.6  
87.7  
87.8  
87.9  
88.0  
88.1  
88.2  
88.3  
88.4  
88.5  
88.6  
88.7  
88.8  
88.9  
89.0  
89.1  
89.2  
89.3  
89.4  
89.5  
89.6  
89.7  
89.8  
89.9  
90.0  
90.1  
90.2  
90.3  
90.4  
90.5  
90.6  
90.7  
90.8  
90.9  
91.0  
91.1  
91.2  
91.3  
91.4  
91.5  
91.6  
91.7  
91.8  
91.9  
92.0  
92.1  
92.2  
92.3  
92.4  
92.5  
92.6  
92.7  
92.8  
92.9  
93.0  
93.1  
93.2  
93.3  
93.4  
93.5  
93.6  
93.7  
93.8  
93.9  
94.0  
94.1  
94.2  
94.3  
94.4  
94.5  
94.6  
94.7  
94.8  
94.9  
95.0  
95.1  
95.2  
95.3  
95.4  
95.5  
95.6  
95.7  
95.8  
95.9  
96.0  
96.1  
96.2  
96.3  
96.4  
96.5  
96.6  
96.7  
96.8  
96.9  
97.0  
97.1  
97.2  
97.3  
97.4  
97.5  
97.6  
97.7  
97.8  
97.9  
98.0  
98.1  
98.2  
98.3  
98.4  
98.5  
98.6  
98.7  
98.8  
98.9  
99.0  
99.1  
99.2  
99.3  
99.4  
99.5  
99.6  
99.7  
99.8  
99.9

8.3 END OF TEST  
-----

WITH ALL SWITCHES ON A "0" THE END OF TEST TYPEOUT WILL OCCUR WHEN THE PROGRAM COMPLETES 18 TAPE PASSES FROM "BOT" TO "EOT" ON THE DRIVE UNDER TEST OR A FATAL ERROR OCCURS.

8.3.1 EXAMPLE OF AN END-OF TEST TYPEOUT

\*\*\* END-OF-TEST \*\*\*  
SOFT ERRORS=0  
HARD ERRORS=0  
BYTES READ=1471488  
BYTES WRITTEN=369000  
TAPE PASSES=18  
FILES/PASS=12  
BLOCKS/FILE=16

8.4 DRIVE COMPATIBILITY  
-----

THE COMPATIBILITY BETWEEN DRIVES CAN BE TESTED BY DOING A "FORMAT" PASS ON ONE DRIVE AND THEN READING IT ON ANOTHER DRIVE.

8.4.1 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #1  
-----

THIS EXAMPLE FORMATS ON DRIVE A AND READS FROM DRIVE B

8.4.1.1 "FORMAT" DRIVE "A"

1. PLACE A WRITE ENABLED TAPE IN DRIVE "A"
2. INSURE DRIVE "B" IS EMPTY
3. LOAD ADDRESS 200
4. SET SW09 AND SW08 TO "1" ALL OTHERS TO "0"
5. PRESS START
6. PROGRAM WILL PERFORM A "FORMAT" PASS ON DRIVE "A", TYPE "END-OF-TEST" STATISTICS AND HALT

8.4.1.2 "READ" DRIVE "B"

1. REMOVE THE TAPE FROM DRIVE "A" AND PLACE IT IN DRIVE "B"
2. LOAD ADDRESS 200
3. SET SW09, SW08, SW07 AND SW06 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS ON DRIVE "B", TYPE "END-OF-TEST" STATISTICS AND HALT.

618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673

9.4.2 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #2

-----  
THIS EXAMPLE READS KNOWN GOOD TAPE(S)

8.4.2.1 "READ"

1. PLACE THE KNOWN GOOD TAPE(S) IN THE DRIVE(S) TO BE TESTED. (NOTE: IT MIGHT BE WISE TO HAVE THE TAPE(S) WRITE LOCKED.)
2. LOAD ADDRESS 200
3. SET SW08, SW07 AND SW00 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS AND TYPE "END-OF-TEST" STATISTICS ON DRIVE(S) TO BE TESTED.

8.5 DATA FORMAT

-----  
THE DATA FORMAT USED IN THIS PROGRAM WILL RESULT IN APPROXIMATELY ELEVEN (11) FILES OF 8192 BYTES TO BE WRITTEN ON TAPE.

8.5.1 FILE STRUCTURE

EACH FILE WILL CONSIST OF SIXTEEN (16) BLOCKS OF DATA, WITH EACH BLOCK CONTAINING AN UNIQUE DATA PATTERN.

8.5.2 BLOCK STRUCTURE

EACH BLOCK WILL HAVE AN "ID" CODE AS THE FIRST FOUR (4) BYTES OF DATA. THIS "ID" WILL BE THE FILE NUMBER AND BLOCK NUMBER AND THEIR COMPLEMENTS. THE DATA FOLLOWING THE "ID" IS A PATTERN THAT REPEATS ITSELF EVERY EIGHT (8) BYTES FOR THE LENGTH OF THE BLOCK.

8.5.3 BLOCK SIZE AND PATTERN

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
-----	-----	-----
0	1024	314 063 314 063 146 231 146 231
1	512	314 231 063 146



D02

TA11 DATA RELIABILITY  
DZTAEB.P11

MAINDEC-11-DZTAE-B

MACY11 27(732)

25-SEP-76

11:17

PAGE 17

674  
675  
676  
677  
678

314  
231  
063  
146

8.5.3 (CONT.)

	BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
	-----	-----	-----
679			
680			
681			
682			
683			
684			
685			
686	2	1024	001
687			002
688			004
689			010
690			020
691			040
692			100
693			200
694			
695	3	256	177
696			277
697			337
698			357
699			367
700			373
701			375
702			376
703			
704	4	1024	252
705			125
706			252
707			125
708			252
709			125
710			252
711			125
712			
713	5	128	000
714			111
715			222
716			333
717			044
718			155
719			266
720			377
721			
722	6	1024	000
723			044
724			111
725			155
726			222
727			266
728			333
729			377
730			

8.5.3 (CONT.)

	<u>BLOCK NUMBER</u>	<u>BLOCK SIZE</u>	<u>BLOCK PATTERN</u>
731			
732			
733			
734			
735			
736			
737			
738	7	64	000
739			222
740			044
741			266
742			111
743			333
744			155
745			377
746			
747	8	1024	001
748			003
749			007
750			017
751			037
752			077
753			177
754			377
755			
756	9	32	376
757			374
758			370
759			360
760			340
761			300
762			200
763			000
764			
765	10	128	001
766			376
767			002
768			375
769			004
770			373
771			010
772			367
773			
774	11	256	020
775			357
776			040
777			337
778			100
779			277
780			200
781			177
782			

783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835

8.5.3 (CONT.)

<u>BLOCK NUMBER</u>	<u>BLOCK SIZE</u>	<u>BLOCK PATTERN</u>
12	512	000 000 000 000 000 000 000
13	1024	377 377 377 377 377 377 377
14	32	000 377 000 377 000 377 000
15	128	017 360 207 170 303 074 341 036

8.6 TEST TYPEOUT

THE FOLLOWING EXAMPLE SHOWS A TYPICAL TYPEOUT WHERE BOTH DRIVES WERE TESTED AND NO ERRORS OCCURRED.

836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891

MAINDEC-11-DZTAE-A

DRIVE A AND DRIVE B WILL BE TESTED

\*\*\* FORMAT \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* WRITE \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* FORMAT \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* WRITE \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A  
\*\*\* READ \*\*\* DRIVE A

\*\*\* END-OF-TEST \*\*\*  
SOFT ERRORS=0  
HARD ERRORS=0  
BYTES READ=1476608  
BYTES WRITTEN=370209  
TAPE PASSES=18  
FILES/PASS=12  
BLOCKS/FILE=16

\*\*\* FORMAT \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* WRITE \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* FORMAT \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* WRITE \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B  
\*\*\* READ \*\*\* DRIVE B

\*\*\* END-OF-TEST \*\*\*  
SOFT ERRORS=0  
HARD ERRORS=0  
BYTES READ=1504096  
BYTES WRITTEN=376868

892  
893  
894

TAPE PASSES=18  
FILES/PASS=12  
BLOCKS/FILE=16

895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
9509. PROGRAM DESCRIPTION  
-----

THIS PROGRAM IS DESIGNED AROUND THREE PRIMARY ROUTINES THAT WILL TRANSFER DATA TO AND/OR FROM THE TA11/TU60 GOING FROM "BOT" TO "EOT" OF THE TAPE. EACH OF THESE ROUTINES MAKE USE OF COMMON SUBROUTINES TO MANIPULATE TAPE MOTION, KEEP TRACK OF TAPE POSITION, SETUP DATA BUFFERS AND CHECK, COUNT AND REPORT ERRORS. THESE ROUTINES ARE DEFINED AND EXPLAINED BELOW.

9.1 FORMAT PASS  
-----

THIS IS A WRITE, BACKSPACE AND READ ROUTINE STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PREFORMED:

1. WRITE A BLOCK OF DATA
2. BACKSPACE A BLOCK GAP
3. READ THE BLOCK
4. CHECK FOR SYNC ERROR
5. CHECK FOR DATA ERROR
6. REPEAT STEPS 1-5 SIXTEEN TIMES
7. WRITE A FILE GAP
8. REPEAT STEPS 1-7 UNTIL "EOT"

9.2 WRITEONLY PASS  
-----

THIS IS A WRITE ONLY ROUTINE. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PREFORMED:

1. WRITE SIXTEEN BLOCKS OF DATA
2. WRITE A FILE GAP
3. REPEAT STEPS 1 & 2 TO "EOT"

9.3 READONLY PASS  
-----

THIS IS A READ ONLY ROUTINE AND REQUIRES THAT A "FORMAT" OR "WRITEONLY" PASS HAS ALREADY BEEN PREFORMED. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PREFORMED:

1. READ A BLOCK OF DATA
2. CHECK FOR SYNC ERROR
3. CHECK FOR DATA ERROR
4. REPEAT STEPS 1-3 SIXTEEN (16) TIMES
5. SPACE FORWARD FILE GAP
6. REPEAT STEPS 1-5 UNTIL THE LAST BLOCK OF THE LAST FILE HAS BEEN READ.

%

.TITLE TA11 DATA RELIABILITY MAINDEC-11-DZTAE-B  
; \*COPYRIGHT (C) 1973, 1976  
; \*DIGITAL EQUIPMENT CORP.  
; \*MAYNARD, MASS. 01754

951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001

```

;*
;*PROGRAM BY JIM LACEY
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-CD),MAR 21, 1976.
;*
;*****
;*****
;*****
.REM!

```

GENERAL INFORMATION ABOUT THE TA11/TU60 CASSETTE

ADDRESS MNEMONIC DESCRIPTION

-----

```

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

```

TACS REGISTER DESCRIPTION

-----

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



```

1002 .SBTTL OPERATIONAL SWITCH SETTINGS
1003 : *
1004 : * SWITCH USE
1005 : * -----
1006 : * 15 HALT ON ERROR
1007 : * 14 LOOP ON TEST
1008 : * 13 INHIBIT ERROR TYPEOUTS
1009 : * 10 DING BELL ON ERROR
1010 : * 9 HALT AFTER NEXT "END-OF-TEST" TYPE OUT
1011 : * 8 AT NEXT "EOT" GOTO "END-OF-TEST"
1012 : * 7 PERFORM AS PER SWR<1:0>
1013 : * 00=FORMAT
1014 : * 01=READONLY
1015 : * 10=WRITEONLY
1016 : * 11=READONLY
1017 :
1018 :
1019 :
1020 .SBTTL BASIC DEFINITIONS
1021 :
1022 : *INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1023 001100 STACK= 1100
1024 .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
1025 .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
1026 :
1027 : *MISCELLANEOUS DEFINITIONS
1028 000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
1029 000012 LF= 12 ;;CODE FOR LINE FEED
1030 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
1031 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
1032 177776 PS= 177776 ;;PROCESSOR STATUS WORD
1033 .EQUIV PS,PSW
1034 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
1035 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
1036 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
1037 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
1038 :
1039 : *GENERAL PURPOSE REGISTER DEFINITIONS
1040 000000 R0= %0 ;;GENERAL REGISTER
1041 000001 R1= %1 ;;GENERAL REGISTER
1042 000002 R2= %2 ;;GENERAL REGISTER
1043 000003 R3= %3 ;;GENERAL REGISTER
1044 000004 R4= %4 ;;GENERAL REGISTER
1045 000005 R5= %5 ;;GENERAL REGISTER
1046 000006 R6= %6 ;;GENERAL REGISTER
1047 000007 R7= %7 ;;GENERAL REGISTER
1048 .EQUIV R6,SP ;;STACK POINTER
1049 .EQUIV R7,PC ;;PROGRAM COUNTER
1050 :
1051 : *PRIORITY LEVEL DEFINITIONS
1052 000000 PR0= 0 ;;PRIORITY LEVEL 0
1053 000040 PR1= 40 ;;PRIORITY LEVEL 1
1054 000100 PR2= 100 ;;PRIORITY LEVEL 2
1055 000140 PR3= 140 ;;PRIORITY LEVEL 3
1056 000200 PR4= 200 ;;PRIORITY LEVEL 4
1057 000240 PR5= 240 ;;PRIORITY LEVEL 5

```

1058 000300 PR6= 300 ::PRIORITY LEVEL 6  
1059 000340 PR7= 340 ::PRIORITY LEVEL 7

1061 :\*"SWITCH REGISTER" SWITCH DEFINITIONS

1062	100000	SW15=	100000
1063	040000	SW14=	40000
1064	020000	SW13=	20000
1065	010000	SW12=	10000
1066	004000	SW11=	4000
1067	002000	SW10=	2000
1068	001000	SW09=	1000
1069	000400	SW08=	400
1070	000200	SW07=	200
1071	000100	SW06=	100
1072	000040	SW05=	40
1073	000020	SW04=	20
1074	000010	SW03=	10
1075	000004	SW02=	4
1076	000002	SW01=	2
1077	000001	SW00=	1
1078		.EQUIV	SW09,SW9
1079		.EQUIV	SW08,SW8
1080		.EQUIV	SW07,SW7
1081		.EQUIV	SW06,SW6
1082		.EQUIV	SW05,SW5
1083		.EQUIV	SW04,SW4
1084		.EQUIV	SW03,SW3
1085		.EQUIV	SW02,SW2
1086		.EQUIV	SW01,SW1
1087		.EQUIV	SW00,SW0

1089 :\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

1090	100000	BIT15=	100000
1091	040000	BIT14=	40000
1092	020000	BIT13=	20000
1093	010000	BIT12=	10000
1094	004000	BIT11=	4000
1095	002000	BIT10=	2000
1096	001000	BIT09=	1000
1097	000400	BIT08=	400
1098	000200	BIT07=	200
1099	000100	BIT06=	100
1100	000040	BIT05=	40
1101	000020	BIT04=	20
1102	000010	BIT03=	10
1103	000004	BIT02=	4
1104	000002	BIT01=	2
1105	000001	BIT00=	1
1106		.EQUIV	BIT09,BIT9
1107		.EQUIV	BIT08,BIT8
1108		.EQUIV	BIT07,BIT7
1109		.EQUIV	BIT06,BIT6
1110		.EQUIV	BIT05,BIT5
1111		.EQUIV	BIT04,BIT4
1112		.EQUIV	BIT03,BIT3
1113		.EQUIV	BIT02,BIT2

```

1114      .EQUIV BIT01,BIT1
1115      .EQUIV BIT00,BIT0
1116
1117      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1118      000004  ERRVEC= 4          ;: TIME OUT AND OTHER ERRORS
1119      000010  RESVEC= 10         ;: RESERVED AND ILLEGAL INSTRUCTIONS
1120      000014  TBITVEC=14        ;: "T" BIT
1121      000014  TRTVEC= 14         ;: TRACE TRAP
1122      000014  BPTVEC= 14        ;: BREAKPOINT TRAP (BPT)
1123      000020  IOTVEC= 20         ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
1124      000024  PWRVEC= 24         ;: POWER FAIL
1125      000030  EMTVEC= 30        ;: EMULATOR TRAP (EMT) **ERROR**
1126      000034  TRAPVEC=34        ;: "TRAP" TRAP
1127      000060  TKVEC= 60          ;: TTY KEYBOARD VECTOR
1128      000064  TPVEC= 64          ;: TTY PRINTER VECTOR
1129      000240  PIRQVEC=240       ;: PROGRAM INTERRUPT REQUEST VECTOR
1130      ;:*****
1131
1132      ;*****TA11 FUNCTIONS*****
1133      000000  XWFG= 0             ;: WRITE FILE GAP FUNCTION
1134      000002  XWRITE= 2          ;: WRITE FUNCTION
1135      000004  XREAD= 4           ;: READ FUNCTION
1136      000006  XBSFG= 6          ;: BACK SPACE FILE GAP FUNCTION
1137      000010  XBSBG= 10         ;: BACK SPACE BLOCK GAP FUNCTION
1138      000012  XSFFG= 12         ;: SPACE FWD FILE GAP FUNCTION
1139      000014  XSFBG= 14         ;: SPACE FWD BLOCK GAP FUNCTION
1140      000016  XRWND= 16         ;: REWIND FUNCTION
1141
1142      ;*****TA11 BIT ASSIGNMENT*****
1143      100000  ERROR= BIT15
1144      040000  CRCERR= BIT14
1145      020000  LEADER= BIT13
1146      010000  WRTLOCK=BIT12
1147      004000  FGAP= BIT11
1148      002000  TIMERR= BIT10
1149      001000  OFFLINE=BIT09
1150      000400  UNIT= BIT08
1151      000200  TR.REQ= BIT07
1152      000100  INT.EN= BIT06
1153      000040  READY= BIT05
1154      000020  ILBS= BIT04
1155      000010  FUNC2= BIT03
1156      000004  FUNC1= BIT02
1157      000002  FUNC0= BIT01
1158      000001  GO= BIT00
1159      000016  FUNCTION=          FUNC2+FUNC1+FUNC0
1160

```

```

1161
1162
1163      000000
1164
1165
1166
1167      000174
1168      000174      000000
1169      000176      000000
1170
1171      000200      000137      001752
1172      000204      000137      002004
1173      000210      000137      002012
1174
1175

```

.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
      ;*****

```

.SBTTL COMMON TAGS

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

1176			
1177			
1178			
1179			
1180			
1181			
1182		001100	
1183	001100		
1184	001100	000000	
1185	001102	000	
1186	001103	000	
1187	001104	000000	
1188	001106	000000	
1189	001110	000000	
1190	001112	000000	
1191	001114	000	
1192	001115	001	
1193	001116	000000	
1194	001120	000000	
1195	001122	000000	
1196	001124	000000	
1197	001126	000000	
1198	001130	000000	
1199	001132	000000	
1200	001134	000	
1201	001135	000	
1202	001136	000000	
1203	001140	177570	
1204	001142	177570	
1205	001144	177560	
1206	001146	177562	
1207	001150	177564	
1208	001152	177566	
1209	001154	000	
1210	001155	002	
1211	001156	012	
1212	001157	000	
1213	001160	000000	
1214			
1215	001162	000000	
1216	001164	000000	
1217	001166	000000	
1218	001170	000000	
1219	001172	000000	
1220	001174	000000	
1221	001176	000000	
1222	001200	000000	
1223	001202	077	
1224	001203	015	
1225	001204	000012	
1226			
1227	001206	000000	
1228	001210	000000	
1229	001212	000000	
1230	001214	000000	000000
1231	001220	000000	000000

	.=1100	
SCMTAG:	.WORD	0
SPASS:	.WORD	0
STSTNM:	.BYTE	00
SERFLG:	.BYTE	00
SICNT:	.WORD	00
SLPADR:	.WORD	00
SLPERR:	.WORD	00
SERTTL:	.WORD	00
SITEMB:	.BYTE	00
SERMAX:	.BYTE	01
SERRPC:	.WORD	00
SGDADR:	.WORD	00
SGDADR:	.WORD	00
SGDDAT:	.WORD	00
SBDDAT:	.WORD	00
	.WORD	00
	.WORD	00
SAUTOB:	.BYTE	00
SINTAG:	.BYTE	00
	.WORD	00
SWR:	.WORD	DSWR
DISPLAY:	.WORD	DDISP
STKS:	177560	
STKB:	177562	
STPS:	177564	
STPB:	177566	
SNULL:	.BYTE	0
SFILLS:	.BYTE	2
SFILLC:	.BYTE	12
STPFLG:	.BYTE	0
SREGAD:	.WORD	0
SREG0:	.WORD	0
SREG1:	.WORD	00
SREG2:	.WORD	00
SREG3:	.WORD	00
STMP0:	.WORD	00
STMP1:	.WORD	00
STMP2:	.WORD	00
STMP3:	.WORD	00
SQUES:	.ASCII	/?/
SCRLF:	.ASCII	<15>
SLF:	.ASCII	<12>
SOFTNM:	.WORD	0
HARDNM:	.WORD	0
EOTS:	.WORD	0
RBYTTL:	.WORD	0,0
WBYTTL:	.WORD	0,0

START OF COMMON TAGS
CONTAINS PASS COUNT
CONTAINS THE TEST NUMBER
CONTAINS ERROR FLAG
CONTAINS SUBTEST ITERATION COUNT
CONTAINS SCOPE LOOP ADDRESS
CONTAINS SCOPE RETURN FOR ERRORS
CONTAINS TOTAL ERRORS DETECTED
CONTAINS ITEM CONTROL BYTE
CONTAINS MAX. ERRORS PER TEST
CONTAINS PC OF LAST ERROR INSTRUCTION
CONTAINS ADDRESS OF 'GOOD' DATA
CONTAINS ADDRESS OF 'BAD' DATA
CONTAINS 'GOOD' DATA
CONTAINS 'BAD' DATA
RESERVED--NOT TO BE USED
AUTOMATIC MODE INDICATOR
INTERRUPT MODE INDICATOR
ADDRESS OF SWITCH REGISTER
ADDRESS OF DISPLAY REGISTER
TTY KBD STATUS
TTY KBD BUFFER
TTY PRINTER STATUS REG. ADDRESS
TTY PRINTER BUFFER REG. ADDRESS
CONTAINS NULL CHARACTER FOR FILLS
CONTAINS # OF FILLER CHARACTERS REQUIRED
INSERT FILL CHARS. AFTER A "LINE FEED"
"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
CONTAINS THE ADDRESS FROM WHICH (\$REG0) WAS OBTAINED
CONTAINS ((\$REGAD)+0)
CONTAINS ((\$REGAD)+2)
CONTAINS ((\$REGAD)+4)
CONTAINS ((\$REGAD)+6)
USER DEFINED
USER DEFINED
USER DEFINED
USER DEFINED
QUESTION MARK
CARRIAGE RETURN
LINE FEED
*****
NUMBER OF SOFT ERRORS
NUMBER OF HARD ERRORS
NUMBER OF TAPE PASSES
NUMBER OF BYTES READ
NUMBER OF BYTES WRITTEN

1232	001224	000000		RDTRYS: .WORD	0	: KEEPS COUNT OF REREADS
1233	001226	000000		WRTRYS: .WORD	0	: KEEPS COUNT OF REWRITES
1234	001230	000000	000000	BYTNUM: .WORD	0,0	: THE NUMBER OF BYTES "READ" OR "WRITTEN
1235						: DURING AN OPERATION
1236	001234	000017		LASTBK: .WORD	15.	: WILL CONTAIN THE # OF THE LAST BLOCK
1237						: AFTER A "FORMAT" OR "WRITE" PASS
1238	001236	000011		LASTFL: .WORD	9.	: WILL CONTAIN THE # OF THE LAST FILE
1239						: AFTER A "FORMAT" OR "WRITE" PASS
1240						
1241						
1242	001240	177500		TACSL: 177500		: ADDRESS OF TACS
1243	001242	177502		TADBL: 177502		: ADDRESS OF TADB
1244	001244	000260	000262	TAVEC: 260,262		: TAI1 VECTOR ADDRESS
1245	001250	000300		TAPRIO: 300		: TAI1 BR LEVEL 6
1246	001252	000000	000000	DRVKEY: 0,0		
1247	001256	001252		DRVPNT: DRVKEY		
1248	001260	000003		MAXRDS: .WORD	3	: MAX REREADS BEFORE CALLING IT A HARD ERROR
1249	001262	000003		MAXERR: .WORD	3	: MAX HARD ERRORS ALLOWED
1250	001264	000022		MAXEOT: .WORD	18.	: NUMBER OF TAPE PASSES BEFORE END-OF-TEST
1251	001266	000000		PASCNT: .WORD	0	: COUNT # OF TAPE PASSES
1252	001270	001274	000000	PSCNTL: .WORD	FORPAS,0	: CONTROLS THE TYPE OF PASS
1253	001274	000001		FORPAS: .WORD	1	: 1 FORMAT PASS
1254	001276	000003		RD1PAS: .WORD	3	: 3 READONLY PASSES
1255	001300	000001		WRTPAS: .WORD	1	: 1 WRITEONLY PASS
1256	001302	000004		RD2PAS: .WORD	4	: 4 READONLY PASSES
1257	001304	000		FILE: .BYTE	0	: FILE NUMBER
1258	001305	377				: 1'S COMPLEMENT ON FILE NUMBER
1259	001306	000		BLOCK: .BYTE	0	: BLOCK NUMBER
1260	001307	377				: 1'S COMPLEMENT OF BLOCK NUMBER
1261	001310	000020		FILESZ: .WORD	16.	: NUMBER OF BLOCKS PER FILE

1262  
 1263  
 1264  
 1265  
 1266  
 1267 001312 002000  
 1268 001314 001000  
 1269 001316 002000  
 1270 001320 000400  
 1271 001322 002000  
 1272 001324 000200  
 1273 001326 002000  
 1274 001330 000100  
 1275 001332 002000  
 1276 001334 000040  
 1277 001336 000200  
 1278 001340 000400  
 1279 001342 001000  
 1280 001344 002000  
 1281 001346 000040  
 1282 001350 000200  
 1283  
 1284  
 1285  
 1286  
 1287  
 1288 001352 001412  
 1289 001354 001422  
 1290 001356 001432  
 1291 001360 001442  
 1292 001362 001452  
 1293 001364 001462  
 1294 001366 001472  
 1295 001370 001502  
 1296 001372 001512  
 1297 001374 001522  
 1298 001376 001532  
 1299 001400 001542  
 1300 001402 001552  
 1301 001404 001562  
 1302 001406 001572  
 1303 001410 001602  
 1304

;///

.SBTTL BLOCK SIZE TABLE

BLKSZ:	.WORD	1024.	;BLOCK	0
	.WORD	512.	;BLOCK	1
	.WORD	1024.	;BLOCK	2
	.WORD	256.	;BLOCK	3
	.WORD	1024.	;BLOCK	4
	.WORD	128.	;BLOCK	5
	.WORD	1024.	;BLOCK	6
	.WORD	64.	;BLOCK	7
	.WORD	1024.	;BLOCK	8
	.WORD	32.	;BLOCK	9
	.WORD	128.	;BLOCK	10
	.WORD	256.	;BLOCK	11
	.WORD	512.	;BLOCK	12
	.WORD	1024.	;BLOCK	13
	.WORD	32.	;BLOCK	14
	.WORD	128.	;BLOCK	15

;///

.SBTTL TABLE OF POINTERS TO THE DIFFERENT PATTERNS

PATS:	PAT0
	PAT1
	PAT2
	PAT3
	PAT4
	PAT5
	PAT6
	PAT7
	PAT8
	PAT9
	PAT10
	PAT11
	PAT12
	PAT13
	PAT14
	PAT15

				;////////////////////////////////////	
				.SBTTL	DATA PATTERNS
1305					
1306					
1307					
1308					
1309					
1310	001412	314	063	314	PAT0: .BYTE 314,063,314,063,146,231,146,231
1311	001415	063	146	231	
1312	001420	146	231		
1313	001422	314	231	063	PAT1: .BYTE 314,231,063,146,314,231,063,146
1314	001425	146	314	231	
1315	001430	063	146		
1316	001432	001	002	004	PAT2: .BYTE 001,002,004,010,020,040,100,200
1317	001435	010	020	040	
1318	001440	100	200		
1319	001442	177	277	337	PAT3: .BYTE 177,277,337,357,367,373,375,376
1320	001445	357	367	373	
1321	001450	375	376		
1322	001452	252	125	252	PAT4: .BYTE 252,125,252,125,252,125,252,125
1323	001455	125	252	125	
1324	001460	252	125		
1325	001462	000	111	222	PAT5: .BYTE 000,111,222,333,044,155,266,377
1326	001465	333	044	155	
1327	001470	266	377		
1328	001472	000	044	111	PAT6: .BYTE 000,044,111,155,222,266,333,377
1329	001475	155	222	266	
1330	001500	333	377		
1331	001502	000	222	044	PAT7: .BYTE 000,222,044,266,111,333,155,377
1332	001505	266	111	333	
1333	001510	155	377		
1334	001512	001	003	007	PAT8: .BYTE 001,003,007,017,037,077,177,377
1335	001515	017	037	077	
1336	001520	177	377		
1337	001522	376	374	370	PAT9: .BYTE 376,374,370,360,340,300,200,000
1338	001525	360	340	300	
1339	001530	200	000		
1340	001532	001	376	002	PAT10: .BYTE 001,376,002,375,004,373,010,367
1341	001535	375	004	373	
1342	001540	010	367		
1343	001542	020	357	040	PAT11: .BYTE 020,357,040,337,100,277,200,177
1344	001545	337	100	277	
1345	001550	200	177		
1346	001552	000	000	000	PAT12: .BYTE 000,000,000,000,000,000,000,000
1347	001555	000	000	000	
1348	001560	000	000		
1349	001562	377	377	377	PAT13: .BYTE 377,377,377,377,377,377,377,377
1350	001565	377	377	377	
1351	001570	377	377		
1352	001572	000	377	000	PAT14: .BYTE 000,377,000,377,000,000,377,377
1353	001575	377	000	000	
1354	001600	377	377		
1355	001602	017	360	207	PAT15: .BYTE 017,360,207,170,303,074,341,036
1356	001605	170	303	074	
1357	001610	341	036		
1358					



```

1359
1360
1361
1362
1363
1364 001612 000
1365 001613 000
1366 001614 001612
1367 001616 013440
1368 001620 000000
1369

```

;/;;

.SBTTL PARAMETER BLOCK USED WITH ALL FUNCTIONS

```

PARMBK: .BYTE 0 ;USED FOR STATUS/ERROR
         .BYTE 0 ;DRIVE # (DRIVE A=0, B=1)
         .WORD PARMBK ;POINTS TO STATUS/ERROR BYTE
         .WORD BUFFER ;FIRST ADDRESS OF DATA BUFFER
         .WORD 0 ;USED FOR BYTE COUNT.

```

1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425

.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 SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;\*NOTE1: IF SITEMB 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:

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

ITEMSO: ;ITEMS 001-002

;NOTE: ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS  
; UNLESS OTHERWISE NOTED

;ITEM 1  
EM1 ;DATA ERROR  
DH1 ;PC FILE BLOCK BYTE GDDAT BDDAT GDADR BDADR  
DT1 ;\$ERRPC \$REGO \$REG2 BYTNUM \$GDDAT \$BDDAT \$GDADR \$BDADR  
DF1 ;FILE,BLOCK AND BYTE ARE TYPED IN DECIMAL

;ITEM 2  
EM2 ;SYNC ERROR  
DH2 ; EXPT'D EXPT'D RCV'D RCV'D  
;PC FILE BLOCK FILE BLOCK  
DT2 ;\$ERRPC \$REGO \$REG2 \$TMPD \$TMP2  
DF2 ;ALL NUMBERS EXCEPT \$ERRPC ARE TYPED IN DECIMAL

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

ITEMS1: ;ITEMS 101-107

EM101 ;DRIVE IS OFF-LINE  
DH101 ;PC FILE BLOCK FUNCTION  
DT101 ;\$ERRPC \$REGO \$REG2 \$TMPD  
0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
; \$TMPD WILL BE TYPED AS A FUNCTION NAME

EM102 ;DRIVE IS WRITE-LOCK  
DH101 ;PC FILE BLOCK FUNCTION  
DT101 ;\$ERRPC \$REGO \$REG2 \$TMPD  
0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
; \$TMPD WILL BE TYPED AS A FUNCTION NAME

001622

001622

001622 012220  
001624 012511  
001626 011744  
001630 012026

001632 012233  
001634 012607

001636 011766  
001640 012036

001642

001642 012246  
001644 012724  
001646 012002  
001650 000000

001652 012270  
001654 012724  
001656 012002  
001660 000000

1426 001662 012314  
 1427 001664 012724  
 1428 001666 012002  
 1429 001670 000000  
 1430  
 1431  
 1432 001672 012337  
 1433 001674 012724  
 1434 001676 012002  
 1435 001700 000000  
 1436  
 1437  
 1438 001702 012356  
 1439 001704 012724  
 1440 001706 012002  
 1441 001710 000000  
 1442  
 1443  
 1444 001712 012373  
 1445 001714 012724  
 1446 001716 012002  
 1447 001720 000000  
 1448  
 1449  
 1450 001722 012415  
 1451 001724 012724  
 1452 001726 012002  
 1453 001730 000000  
 1454  
 1455  
 1456  
 1457  
 1458  
 1459  
 1460 001732  
 1461  
 1462 001732 012437  
 1463 001734 012765  
 1464 001736 012014  
 1465 001740 000000  
 1466  
 1467 001742 012466  
 1468 001744 013002  
 1469 001746 012022  
 1470 001750 000000  
 1471

EM103 ;CLEAR LEADER ERROR  
 DH101 ;PC FILE BLOCK FUNCTION  
 DT101 ;SERRPC \$REG0 \$REG2 \$TMPO  
 0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
 ;\$TMPO WILL BE TYPED AS A FUNCTION NAME

EM104 ;FILE GAP ERROR  
 DH101 ;PC FILE BLOCK FUNCTION  
 DT101 ;SERRPC \$REG0 \$REG2 \$TMPO  
 0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
 ;\$TMPO WILL BE TYPED AS A FUNCTION NAME

EM105 ;TIMING ERROR  
 DH101 ;PC FILE BLOCK FUNCTION  
 DT101 ;SERRPC \$REG0 \$REG2 \$TMPO  
 0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
 ;\$TMPO WILL BE TYPED AS A FUNCTION NAME

EM106 ;BLOCK CHECK ERROR  
 DH101 ;PC FILE BLOCK FUNCTION  
 DT101 ;SERRPC \$REG0 \$REG2 \$TMPO  
 0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
 ;\$TMPO WILL BE TYPED AS A FUNCTION NAME

EM107 ;UNKNOWN INTERRUPT  
 DH101 ;PC FILE BLOCK FUNCTION  
 DT101 ;SERRPC \$REG0 \$REG2 \$TMPO  
 0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
 ;\$TMPO WILL BE TYPED AS A FUNCTION NAME

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

ITEMS2: ;ITEMS 201-202

EM201 ;TA11 FAILED TO RESPOND  
 DH201 ;PC TACS  
 DT201 ;SERRPC TACS  
 0 ;BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS

EM202 ;NO DRIVES AVAILABLE  
 DH202 ;PC  
 DT202 ;SERRPC  
 0 ;

```

1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482 001752 005005
1483 001754 012737 041101 001252
1484 001762 122737 000005 000041
1485 001770 001012
1486 001772 022737 000260 001244
1487 002000 001006
1488 002002 000403
1489 002004 012705 000001
1490 002010 000402
1491 002012 012705 000002
1492 002016
1493
1494
1495 002016 012706 001100
1496 002022 005026
1497 002024 022706 001140
1498 002030 001374
1499 002032 012706 001100
1500
1501 002036 012737 004750 000020
1502 002044 012737 000340 000022
1503 002052 012737 005050 000030
1504 002060 012737 000340 000032
1505 002066 012737 011512 000034
1506 002074 012737 000340 000036
1507 002102 012737 011562 000024
1508 002110 012737 000340 000026
1509 002116 016767 002314 002304
1510 002124 012767 002124 176754
1511
1512
1513 002132 013746 000004
1514 002136 012737 002172 000004
1515 002144 012767 177570 176766
1516 002152 012767 177570 176762
1517 002160 022777 177777 176752
1518 002166 001012
1519
1520 002170 000403
1521 002172 012716 002200
1522 002176 000002
1523 002200 012767 000176 176732
1524 002206 012767 000174 176726
1525 002214 012637 000004
1526
1527

```

```

;////////////////////////////////////
;////////////////////////////////////
;*****
;BEGIN1 IS FOR NORMAL START
;BEGIN2 IS FOR DRIVE SELECTION
;BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
;*****
BEGIN1: CLR R5 ;NORMAL START
        MOV #AB, @DRVKEY
        CMPB #5, @41 ;CASSETTE DDP?
        BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
        CMP #260, @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
BGNCMN:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR, R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK, SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV #SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340, @IOTVEC+2 ;;LEVEL 7
MOV #ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340, @EMTVEC+2 ;;LEVEL 7
MOV #TRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340, @TRAPVEC+2 ;;LEVEL 7
MOV #SPWRDN, @PWRVEC ;;POWER FAILURE VECTOR
MOV #340, @PWRVEC+2 ;;LEVEL 7
MOV SENDCT, SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
MOV #, SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
MOV #64$, @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
64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
RTI
65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG, DISPLAY
66$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
.SBTTL TYPE PROGRAM NAME

```

```

1528                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1529 002220 005227 177777                 INC      #-1                ;;FIRST TIME?
1530 002224 001036                       BNE      67$                ;;BRANCH IF NO
1531 002226 022737 004716 000042        CMP      #SENDAD,2#42      ;;ACT-11?
1532 002234 001432                       BEQ      67$                ;;BRANCH IF YES
1533 002236 104400 002274                 TYPE     68$                ;;TYPE ASCIZ STRING
1534                                     .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1535 002242 005737 000042                 TST      2#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
1536 002246 001006                       BNE      69$                ;;BRANCH IF YES
1537 002250 026727 176664 000176        CMP      SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
1538 002256 001005                       BNE      70$                ;;BRANCH IF NO
1539 002260 104404                       GTSWR
1540 002262 000403                       BR       70$                ;;GET SOFT-SWR SETTINGS
1541 002264 112767 000001 176642 69$:   MOV      #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
1542 002272                                     70$:
1543 002272 000413                       BR       67$                ;;GET OVER THE ASCIZ
1544                                     ;;68$: .ASCIZ <CRLF>/MAINDEC-11-DZTAE-B/<CRLF>
1545 002322                                     67$:
1546                                     ;;*****
1547                                     ;;*****
1548
1549                                     ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
1550
1551                                     R5=2   ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
1552                                     R5=1   ASK FOR DRIVE(S)
1553                                     R5=0   DON'T ASK FOR ANYTHING
1554
1555                                     ;;*****
1556 002322 010504   BEGINX: MOV      R5,R4                ;COPY R5
1557 002324 005305   DEC      R5                ;ASK FOR DRIVES?
1558 002326 002406   BLT     CHKADR            ;BR IF NO
1559 002330 004737 007610   JSR     PC,2#ASKDRV       ;GO GET DRIVES TO BE TESTED
1560 002334 005305   DEC      R5                ;ASK FOR ADDRESSES?
1561 002336 002402   BLT     CHKADR            ;BR IF NO
1562 002340 004737 007720   JSR     PC,2#ASKADR       ;GO GET TA11 ADDRESSES
1563                                     ;;*****
1564                                     ;;*****
1565
1566                                     ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
1567
1568                                     I.   TIMEOUT OCCURRED
1569                                     A.   TYPE ERROR MESSAGE
1570                                     B.   EXAMINE R4
1571                                     1.   R4>0 GOTO BEGINX
1572                                     2.   R4=0 EXAMINE (42)
1573                                     A.   (42)=0 GOTO BEGINX
1574                                     B.   (42)>0 GOTO SENDAD
1575
1576                                     II.  TIMEOUT DIDN'T OCCUR
1577                                     A.   CONTINUE
1578
1579                                     ;;*****
1580 002344 012737 002362 000004   CHKADR: MOV      #1$,2#ERRVEC ;IN CASE OF TIMEOUTS
1581 002352 005000   CLR      R0                ;USE AS A SWITCH
1582 002354 005777 176660   TST     2TACSL            ;SEE IF TA11 RESPONDS
1583 002360 000402   BR      2$                ;BR IF NO TIMEOUT

```



```

1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621 002436 012700 001252
1622 002442 004737 010164
1623 002446 000410
1624 002450 116010 000001
1625 002454 001412
1626 002456 004737 010164
1627 002462 000407
1628 002464 005010
1629 002466 000405
1630 002470 005200
1631 002472 004737 010164
1632 002476 000401
1633 002500 105010
1634 002502 012700 001252
1635 002506 010037 001256
1636 002512 121060 000001
1637 002516 001002
1638 002520 105060 000001
1639 002524 005710
1640 002526 001424
1641 002530 112037 001613
1642 002534 104400 001203
1643 002540 106237 001613
1644 002544 004737 007554
1645 002550 111037 001613
1646 002554 001406
1647 002556 104400 013174
1648 002562 106237 001613
1649 002566 004737 007554
1650 002572 104400 013201
1651 002576 000412
1652 002600 104202
1653 002602 012705 000002

```

```

*****
*****
;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 $ENDAD
;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
;A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
;B. TYPE THE DRIVE TO BE TESTED
;C. CONTINUE IN PROGRAM
;III. DESIRED DRIVE(S) CAN BE TESTED
;A. TYPE THE DRIVE(S) TO BE TESTED
;B. CONTINUE IN PROGRAM
*****
CHKDRV: MOV    #DRVKEY,RO      ;PICKUP ADDRESS OF ASCII DRIVE KEY
        JSR    PC,@#EXAM     ;GO EXAMINE FIRST DRIVE
        BR     1$            ;OK TO TEST---GO CHECK NEXT
        MOVB   1(RO),(RO)    ;REPLACE 1ST WITH 2ND
        BEQ    2$            ;BR IF NO 2ND DRIVE SELECTED
        JSR    PC,@#EXAM     ;GO EXAMINE DRIVE
        BR     2$            ;OK TO TEST
        CLR    (RO)         ;CLEAR DRIVE CODES
        BR     2$
1$: INC    RO                ;POINT TO 2ND
        JSR    PC,@#EXAM     ;GO EXAMINE DRIVE
        BR     2$            ;OK TO TEST
        CLRB   (RO)         ;CLEAR 2ND
        MOV    #DRVKEY,RO    ;RESET ADDRESS POINTERS
        MOV    RO,@#DRVPTNT
        CMPB   (RO),1(RO)    ;1ST = 2ND?
        BNE   3$            ;BR IF NO
        CLRB   1(RO)        ;YES---CLEAR 2ND
        TST    (RO)         ;ANY DRIVES?
        BEQ    5$            ;BR IF NO
        MOVB   (RO)+,@#PARMBK+1 ;SETUP TO TYPE THIS DRIVE
        TYPE   $CRLF        ;TYPE "CR" & "LF"
        ASRB   @#PARMBK+1    ;ADJUST FOR TYPING
        JSR    PC,@#TPDRV    ;GO TYPE DRIVE
        MOVB   (RO),@#PARMBK+1 ;GET 2ND
        BEQ    4$            ;BR IF NONE
        TYPE   MSG9         ;"AND"
        ASRB   @#PARMBK+1    ;ADJUST FOR TYPING
        JSR    PC,@#TPDRV    ;GO TYPE THE 2ND DRIVE
        TYPE   MSG10        ;"WILL BE TESTED"<CRLF>
        BR     $START
4$: TYPE   MSG10
5$: ERROR  202              ;NO DRIVES AVAILABLE
        MOV    #2,R5        ;DRIVES & ADDRESS

```

1654 002606 005704  
1655 002610 001244  
1656 002612 013700 000042  
1657 002616 001641  
1658 002620 000137 004716

TST R4  
BNE BEGINX  
MOV @#42,R0  
BEQ BEGINX  
JMP @#\$ENDAD

: OPERATOR INPUTS?  
: BR IF YES  
: GET MONITOR RETURN ADDRESS  
: NO MONITOR  
: GO TO END



```

1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672 002624 012700 001114
1673 002630 012701 000005
1674 002634 004737 002760
1675 002640 012700 001206
1676 002644 012701 000014
1677 002650 004737 002760
1678 002654 013701 001256
1679 002660 112100
1680 002662 006200
1681 002664 042700 177776
1682 002670 110037 001613
1683 002674 105711
1684 002676 001002
1685 002700 012701 001252
1686 002704 010137 001256
1687 002710 012737 000011 001236
1688 002716 012737 000017 001234
1689 002724 013737 001260 001224
1690 002732 013737 001262 001226
1691 002740 012737 001274 001270
1692 002746 005037 001272
1693 002752 005037 001266
1694 002756 000404
1695 002760 005020
1696 002762 005301
1697 002764 003375
1698 002766 000207

```

```

////////////////////////////////////
////////////////////////////////////
*****
*****
1. CLEAR THE VARIABLE STORAGE AREA
2. SETUP FOR THE DRIVE THAT IS TO BE TESTED
3. SETUP "LASTFL" AND "LASTBK" INCASE OPERATOR SELECTS "READONLY" PASS
4. SETUP THE MAX. NUMBER OF RETRYS FOR READS AND WRITES
5. SETUP TO START WITH A "FORMAT" PASS
6. CLEAR THE PASS COUNTER
*****
*****
START:  MOV    #SITEMB,RO           ;CLEAR VARIABLE STORAGE AREA
        MOV    #SBDDAT-SITEMB/2,R1
        JSR    PC,@#CLEAR           ;CLEAR AREA
        MOV    #SOFTNM,RO
        MOV    #LASTFL-SOFTNM/2,R1
        JSR    PC,@#CLEAR
        MOV    @#DRVPNT,R1         ;GET DRIVE POINTER
        MOVSB  (R1)+,RO           ;SETUP THE DRIVE
        ASR    RO
        BIC    #1CBIT00,RO
        MOVSB  RO,@#PARMBK+1
        TSTB   (R1)
        BNE    2$
        MOV    #DRVKEY,R1
        MOV    R1,@#DRVPNT
2$:     MOV    #9,@#LASTFL
        MOV    #15,@#LASTBK
        MOV    @#MAXRDS,@#RDTRYS
        MOV    @#MAXERR,@#WRTRYS
        MOV    #FORPAS,@#PSCNTL
        CLR    @#PSCNTL+2
        CLR    @#PASCNT           ;CLEAR THE PASS COUNT
        BR     LOOPER            ;GET AROUND CLEAR
CLEAR:  CLR    (RO)+
        DEC    R1
        BGT   CLEAR
        RTS   PC

```



1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802

////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////

.SBTTL "FORMAT" ROUTINE

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

THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE

- 1. REWIND TO BOT
- 2. WRITE FILE GAP
- 3. WRITE BLOCK OF DATA
- 4. BACK BLOCK GAP
- 5. READ BLOCK OF DATA
- 6. CHECK FOR SYNC ERROR
- 7. CHECK FOR DATA ERROR
  - A. DATA ERROR OCCURRED
    - 1.) (RDTRYS)<(MAXRDS) GOTO 4.
    - 2.) (RDTRYS)=(MAXRDS)
      - A.) (WRTRYS)<(MAXERR)
        - 1.) BACK BLOCK GAP
        - 2.) GOTO 3.
      - B.) (WRTRYS)=(MAXERR) GOTO SEOP
  - B. NO DATA ERROR --- GOTO 8.
- 8. UPDATE THE BLOCK NUMBER
- 9. END OF FILE?
  - A. NO --- GOTO 3.
  - B. YES
    - 1.) UPDATE FILE NUMBER
    - 2.) RESET BLOCK NUMBER TO 0
    - 3.) GOTO 2.

\*\*\*\*\*

1784	003126	012737	003126	001106
1785	003134	104400	013222	
1786	003140	004737	007554	
1787	003144	104400	001203	
1788	003150	012737		
1789	003152	000	377	
1790	003154	001306		
1791	003156	012737		
1792	003160	000	377	
1793	003162	001304		
1794				
1795				
1796				
1797	003164	012700	001612	
1798	003170	004737	006746	
1799	003174	105737	001612	
1800	003200	001775		
1801	003202	100022		
1802	003204	004037	005526	

```

FORMAT: MOV    #FORMAT,2#SLPADR      ;SETUP SCOPE LOOP ADDRESS
        TYPE    MSG11                ;"*** FORMAT ***"
        JSR     PC,2#TPDRV            ;GO TYPE THE DRIVE
        TYPE    $CRLF
        MOV     (PC)+,2(PC)+
        .BYTE   0,377                ;INITIALIZE THE BLOCK NUMBER
        .WORD   BLOCK                ;SET BYTE 0 TO 0 AND BYTE 1
        MOV     (PC)+,2(PC)+         ;TO THE 1'S COMP. OF BYTE 0
        .BYTE   0,377                ;INITIALIZE THE FILE NUMBER
        .WORD   FILE                 ;SET BYTE 0 TO 0 AND BYTE 1
        .WORD   FILE                 ;TO THE 1'S COMP. OF BYTE 0

////////////////////////////////////
"REWIND" TO "BOT"
////////////////////////////////////
18:     MOV     #PARMBK,R0            ;R0=1ST ADDRESS OF PARAMETER BLOCK
        JSR     PC,2#REWIND          ;GO TO REWIND
        TSTB   2#PARMBK             ;WAIT ON FLAG
        BEQ    1$
        BPL    4$
        JSR     R0,2#CSRERR          ;BR IF NO ERROR
        ;GO TO CSR ERROR CHECK
  
```



```

1859
1860
1861
1862 003420 004737 006346
1863 003424 000430
1864 003426 104001
1865 003430 004037 006540
1866 003434 000730
1867 003436 004037 006576
1868 003442 000402
1869 003444 000137 004406
1870
1871
1872
1873 003450 005337 001226
1874 003454 002414
1875 003456 012700 001612
1876 003462 004737 006732
1877 003466 105737 001612
1878 003472 001775
1879 003474 100265
1880 003476 004037 005526
1881 003502 000137 004406
1882 003506 013737 001260 001224
1883 003514 013737 001262 001226
1884
1885
1886
1887 003522 062737
1888 003524 001 377
1889 003526 001306
1890 003530 123737 001310 001306
1891 003536 003244
1892 003540 062737
1893 003542 001 377
1894 003544 001304
1895 003546 012737
1896 003550 000 377
1897 003552 001306
1898 003554 000617

: ////////////////////////////////////////////////////////////////////
: CHECK FOR DATA ERROR IF SOFT REREAD
: ////////////////////////////////////////////////////////////////////
12$: JSR PC, @DATCMP ; CHECK THE DATA
BR 15$ ; RETURN HERE IF DATA IS GOOD
ERROR 1 ; DATA ERROR
JSR RD, @CNTSFT ; COUNT SOFT ERROR
BR 7$ ; GO REREAD
JSR RD, @CNTHRD ; COUNT HARD ERROR
BR 13$ ; REWRITE
JMP @SEOP ; TO MANY HARD ERRORS

: ////////////////////////////////////////////////////////////////////
: IF HARD ERROR REWRITE
: ////////////////////////////////////////////////////////////////////
13$: DEC @WRTRYS ; TRY TO REWRITE THIS BLOCK?
BLT 15$ ; BR IF NO
MOV #PARMBK, RD ; RD=1ST ADDRESS OF PARAMETER BLOCK
JSR PC, @BSBG ; GO TO BSBG
14$: TSTB @PARMBK ; WAIT ON FLAG
BEQ 14$
BPL 4$ ; GO START A WRITE IF NO ERROR
JSR RD, @CSRERR ; GO TO CSR ERROR CHECK
JMP @SEOP
15$: MOV @MAXRDS, @RDTRYS ; RESET THE REREAD COUNT
MOV @MAXERR, @WRTRYS ; RESET THE REWRITE COUNT

: ////////////////////////////////////////////////////////////////////
: UPDATE BLOCK # & FILE #
: ////////////////////////////////////////////////////////////////////
ADD (PC)+, @PC+ ; INCREMENT THE BLOCK NUMBER
.BYTE 1, -1
.WORD BLOCK
CMPB @FILESZ, @BLOCK ; TIME FOR A FILE GAP?
BGT 4$ ; BR IF NO--GO START A WRITE
ADD (PC)+, @PC+ ; INCREMENT THE FILE NUMBER
.BYTE 1, -1
.WORD FILE
MOV (PC)+, @PC+ ; INITIALIZE THE BLOCK NUMBER
.BYTE 0, 377 ; SET BYTE 0 TO 0 AND BYTE 1
.WORD BLOCK ; TO THE 1'S COMP. OF BYTE 0
BR 2$ ; GO WRITE FILE GAP

```

1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954

////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////  
////////////////////////////////////

.SBTTL "READ ONLY" ROUTINE

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

: THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE

1. REWIND TO BOT
2. READ BLOCK OF DATA
3. CHECK FOR SYNC ERROR
4. CHECK FOR DATA ERROR
  - A. DATA ERROR OCCURRED
    - 1.) (RDTRYS)<(MAXRDS)
      - A.) BACK BLOCK GAP
      - B.) GOTO 2.
    - 2.) (RDTRYS)=(MAXRDS)
      - A.) (WRTRYS)<(MAXERR) GOTO 5.
      - B.) (WRTRYS)=(MAXERR) GOTO SEOP
  - B. NO DATA ERROR --- GOTO 5.
5. UPDATE THE BLOCK NUMBER
6. END OF FILE?
  - A. NO --- GOTO 2.
  - B. YES
    - 1.) UPDATE FILE NUMBER
    - 2.) RESET BLOCK NUMBER TO 0
    - 3.) SPACE FORWARD FILE GAP
    - 4.) GOTO 2.

\*\*\*\*\*

READONLY:

```

MOV #READONLY, @#SLPADR ;SETUP SCOPE LOOP ADDRESS
TYPE MSG12 ;"*** READ ***"
JSR PC, @#TPDRV ;GO TYPE DRIVE TO BE TESTED
TYPE $CRLF
MOV (PC)+, @ (PC)+ ;INITIALIZE THE BLOCK NUMBER
.BYTE 0, 377 ;SET BYTE 0 TO 0 AND BYTE 1
.WORD BLOCK ;TO THE 1'S COMP. OF BYTE 0
MOV (PC)+, @ (PC)+ ;INITIALIZE THE FILE NUMBER
.BYTE 0, 377 ;SET BYTE 0 TO 0 AND BYTE 1
.WORD FILE ;TO THE 1'S COMP. OF BYTE 0

```

////////////////////////////////////  
"REWIND" TO "BOT"

```

MOV #PARMBK, RO ;RO=1ST ADDRESS OF PARAMETER BLOCK
JSR PC, @#REWIND ;GO TO REWIND
1$: TSTB @#PARMBK ;WAIT ON FLAG
BEQ 1$
BPL 2$ ;BR IF NO ERROR
JSR RO, @#CSRERR ;GO TO CSR ERROR CHECK
JMP @#$EOP ;GO TO END-OF-PROGRAM

```

////////////////////////////////////

1933	003556			
1934	003556	012737	003556	001106
1935	003564	104400	013241	
1936	003570	004737	007554	
1937	003574	104400	001203	
1938	003600	012737		
1939	003602	000	377	
1940	003604	001306		
1941	003606	012737		
1942	003610	000	377	
1943	003612	001304		
1947	003614	012700	001612	
1948	003620	004737	006746	
1949	003624	105737	001612	
1950	003630	001775		
1951	003632	100004		
1952	003634	004037	005526	
1953	003640	000137	004406	

```

1955      ;          SETUP FOR "READ"
1956      ;          ////////////////////////////////////////////////////
1957 003644 004737 006210 2$: JSR    PC,@#SETUPR      ;SETUP FOR "READ"
1958 003650 000417          BR      5$          ;GO START A READ
1959      ;          ////////////////////////////////////////////////////
1960      ;          "BSBG"--IF SCOPE OR SOFT DATA ERROR
1961      ;          ////////////////////////////////////////////////////
1962 003652 012737 003710 001106 3$: MOV    #5$,@#SLPADR      ;SETUP THE SCOPE LOOP ADDRESS
1963 003660 012700 001612          MOV    #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
1964 003664 004737 006732          JSR    PC,@#BSBG          ;GO TO BSBG
1965 003670 105737 001612 4$: TSTB   @#PARMBK          ;WAIT ON FLAG
1966 003674 001775          BEQ    4$
1967 003676 100004          BPL    5$          ;GO TRY TO REREAD
1968 003700 004037 005526          JSR    RO,@#CSRERR      ;GO CHECK CSR ERROR
1969 003704 000137 004406          JMP    @#SEOP          ;GO TO END-OF-PROGRAM
1970      ;          ////////////////////////////////////////////////////
1971      ;          "READ" BLOCK OF DATA
1972      ;          ////////////////////////////////////////////////////
1973 003710 012737 003652 001106 5$: MOV    #3$,@#SLPADR      ;SET THE LOOP ADDRESS
1974 003716 012700 001612          MOV    #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
1975 003722 004737 006722          JSR    PC,@#READ          ;GO TO READ
1976 003726 105737 001612 6$: TSTB   @#PARMBK          ;WAIT ON FLAG
1977 003732 001775          BEQ    6$
1978 003734 100004          BPL    7$          ;BR IF NO ERROR
1979 003736 004037 005526          JSR    RO,@#CSRERR      ;GO TO CSR ERROR CHECK
1980 003742 000137 004406          JMP    @#SEOP          ;GO TO END-OF-PROGRAM
1981      ;          ////////////////////////////////////////////////////
1982      ;          CHECK FOR SYNC ERROR
1983      ;          ////////////////////////////////////////////////////
1984 003746 004737 006244 7$: JSR    PC,@#SYNCK      ;GO CHECK FOR SYNC ERROR
1985 003752 000403          BR      8$          ;RETURN HERE IF NO ERROR
1986 003754 104002          ERROR  2          ;SYNC ERROR
1987 003756 000137 004406          JMP    @#SEOP          ;GO TO END-OF-PROGRAM
1988      ;          ////////////////////////////////////////////////////
1989      ;          CHECK FOR DATA ERROR
1990      ;          ////////////////////////////////////////////////////
1991 003762 004737 006346 8$: JSR    PC,@#DATCMP      ;GO CHECK THE DATA
1992 003766 000411          BR      9$          ;RETURN HERE IF NO ERROR
1993 003770 104001          ERROR  1          ;DATA ERROR
1994 003772 004037 006540          JSR    RO,@#CNTSFT      ;COUNT SOFT ERROR
1995 003776 000725          BR      3$          ;TRY TO REREAD
1996 004000 004037 006576          JSR    RO,@#CNTHRD      ;COUNT HARD ERROR
1997 004004 000402          BR      9$          ;MOVE TO NEXT BLOCK
1998 004006 000137 004406          JMP    @#SEOP          ;TO MANY HARD ERRORS
1999 004012 000004 9$: SCOPE
2000      ;          ////////////////////////////////////////////////////
2001      ;          IF LAST FILE OF LAST BLOCK GOTO CKEOTS
2002      ;          ////////////////////////////////////////////////////
2003 004014 013737 001260 001224 MOV    @#MAXRDS,@#RDTRYS ;RESET THE REREAD TRYS
2004 004022 123737 001236 001304 CMPB   @#LASTFL,@#FILE   ;LAST FILE
2005 004030 003007          BGT    10$          ;BR IF NO
2006 004032 123737 001234 001306 CMPB   @#LASTBK,@#BLOCK ;LAST BLOCK
2007 004040 003003          BGT    10$          ;BR IF NO
2008 004042 005237 001212          INC    @#EOTS          ;COUNT END-OF-TAPE
2009 004046 000544          BR      CKEOTS        ;CHECK FOR END-OF-PASS
2010      ;          ////////////////////////////////////////////////////

```

```

2011                                     ; UPDATE BLOCK # & FILE #
2012                                     ; ////////////////////////////////////////////////////
2013 004050                             10$: ADD      (PC)+,2(PC)+          ; INCREMENT THE BLOCK NUMBER
2014 004050 062737                       .BYTE 1,-1
2015 004052 001 377                       .WORD BLOCK
2016 004054 001306                       CMPB    2#FILESZ,2#BLOCK          ; TIME FOR A FILE GAP?
2017 004056 123737 001310 001306         BGT     2$                      ; BR IF NO
2018 004064 003267                       ADD      (PC)+,2(PC)+          ; INCREMENT THE FILE NUMBER
2019 004066 062737                       .BYTE 1,-1
2020 004070 001 377                       .WORD FILE
2021 004072 001304                       MOV      (PC)+,2(PC)+          ; INITIALIZE THE BLOCK NUMBER
2022 004074 012737                       .BYTE 0,377                   ; SET BYTE 0 TO 0 AND BYTE 1
2023 004076 000 377                       .WORD BLOCK                   ; TO THE 1'S COMP. OF BYTE 0
2024 004100 001306                       ; ////////////////////////////////////////////////////
2025                                     ; "SPACE-FORWARD-FILE-GAP"
2026                                     ; ////////////////////////////////////////////////////
2027                                     ;
2028 004102 012700 001612                 MOV      #PARMBK,RO            ; RO=1ST ADDRESS OF PARAMETER BLOCK
2029 004106 004737 006736                 JSR     PC,2#SFFG             ; GO TO SFFG
2030 004112 105737 001612                 11$: TSTB 2#PARMBK           ; WAIT ON FLAG
2031 004116 001775                       BEQ     11$
2032 004120 100251                       BPL     2$                      ; GO READ
2033 004122 004037 005526                 JSR     RO,2#CSRERR           ; GO CHECK CSR ERROR
2034 004126 000137 004406                 JMP     2#$EOP                ; GO TO END-OF-PROGRAM
2035

```



2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091

```

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

```

.SBTTL "WRITE ONLY" ROUTINE

```

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

```

THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE

1. REWIND TO BOT
2. WRITE FILE GAP
3. WRITE BLOCK OF DATA
4. UPDATE THE BLOCK NUMBER
5. END OF FILE?
  - A. NO --- GOTO 3.
  - B. YES
    - 1.) UPDATE FILE NUMBER
    - 2.) RESET BLOCK NUMBER TO 0
    - 3.) GOTO 2.

```

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

```

```

004132
004132 012737 004132 001106
004140 104400 013260
004144 004737 007554
004150 104400 001203
004154 012737
004156 000 377
004160 001306
004162 012737
004164 000 377
004166 001304
004170 012700 001612
004174 004737 006746
004200 105737 001612
004204 001775
004206 100022
004210 004037 005526
004214 000137 004406
004220
004220 012700 001612
004224 004737 006712
004230 105737 001612
004234 001775
004236 100006
004240 004037 005526

```

```

WRITEONLY:
MOV #WRITEONLY, @#SLPADR ;SETUP SCOPE LOOP ADDRESS
TYPE MSG13 ;"*** WRITE ***"
JSR PC, @#TPDRV ;TYPE DRIVE TO BE TESTED
TYPE $CRLF
MOV (PC)+, @ (PC)+ ;INITIALIZE THE BLOCK NUMBER
.BYTE 0, 377 ;SET BYTE 0 TO 0 AND BYTE 1
.WORD BLOCK ;TO THE 1'S COMP. OF BYTE 0
MOV (PC)+, @ (PC)+ ;INITIALIZE THE FILE NUMBER
.BYTE 0, 377 ;SET BYTE 0 TO 0 AND BYTE 1
.WORD FILE ;TO THE 1'S COMP. OF BYTE 0

;////////////////////////////////////
; "REWIND" TO "BOT"
;////////////////////////////////////
MOV #PARMBK, RO ;RO=1ST ADDRESS OF PARAMETER BLOCK
JSR PC, @#REWIND ;GO TO REWIND
1$: TSTB @#PARMBK ;WAIT ON FLAG
BEQ 1$
BPL 4$ ;BR IF NO ERROR
JSR RO, @#CSRERR ;GO TO CSR ERROR CHECK
JMP @#$EOP ;GO TO END-OF-PROGRAM

;////////////////////////////////////
; "WRITE-FILE-GAP"
;////////////////////////////////////
2$: MOV #PARMBK, RO ;RO=1ST ADDRESS OF PARAMETER BLOCK
JSR PC, @#WFG ;GO TO WFG
3$: TSTB @#PARMBK ;WAIT ON FLAG
BEQ 3$
BPL 4$ ;BR IF NO ERROR
JSR RO, @#CSRERR ;GO TO CSR ERROR CHECK

```

```

2092 004244 000137 004406      JMP      @#SEOP                ;GO TO END-OF-PROGRAM
2093 004250 000137 004360      JMP      @#CKEOTS
2094                               ;//////////////////////////
2095                               ;"WRITE" A BLOCK OF DATA
2096                               ;//////////////////////////
2097 004254 004737 006062      4$: JSR      PC,@#SETUPW        ;GO SETUP FOR "WRITE"
2098 004260 012737 004266 001106  MOV      #5$,@#SLPADR        ;SETUP SCOPE LOOP ADDRESS
2099 004266
2100 004266 012700 001612      5$: MOV      #PARMBK,RO        ;RO=1ST ADDRESS OF PARAMETER BLOCK
2101 004272 004737 006716      JSR      PC,@#WRITE          ;GO TO WRITE
2102 004276 105737 001612      6$: TSTB   @#PARMBK          ;WAIT ON FLAG
2103 004302 001775      BEQ      6$
2104 004304 100006      BPL      7$                  ;BR IF NO ERROR
2105 004306 004037 005526      JSR      RO,@#CSRERR         ;GO TO CSR ERROR CHECK
2106 004312 000137 004406      JMP      @#SEOP                ;GO TO END-OF-PROGRAM
2107 004316 000137 004360      JMP      @#CKEOTS
2108 004322 000004      7$: SCOPE
2109                               ;//////////////////////////
2110                               ;UPDATE BLOCK # & FILE #
2111                               ;//////////////////////////
2112 004324 062737      ADD      (PC)+,@(PC)+        ;INCREMENT THE BLOCK NUMBER
2113 004326      .BYTE 1,-1
2114 004330 001306      .WORD  BLOCK
2115 004332 123737 001310 001306  CMPB   @#FILESZ,@#BLOCK      ;TIME FOR A FILE GAP?
2116 004340 003345      BGT      4$                  ;BR IF NO
2117 004342 062737      ADD      (PC)+,@(PC)+        ;INCREMENT THE FILE NUMBER
2118 004344      .BYTE 1,-1
2119 004346 001304      .WORD  FILE
2120 004350 012737      MOV      (PC)+,@(PC)+        ;INITIALIZE THE BLOCK NUMBER
2121 004352      .BYTE 0,377              ;SET BYTE 0 TO 0 AND BYTE 1
2122 004354 001306      .WORD  BLOCK                ;TO THE 1'S COMP. OF BYTE 0
2123 004356 000720      BR      2$                  ;START A FILE GAP

```



.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
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)+,a(PC)+ ; RESTORE COUNTER
```

\$ENDCT:

```
.WORD 1
```

\$GET42:

```
MOV $SENDMG ; TYPE "END PASS"
BEQ a#42,RO ; GET MONITOR ADDRESS
SCOPE $DOAGN ; BRANCH IF NO MONITOR
```

```
////////////////////////////////////
; TYPE END-OF-TEST STATISTICS
;////////////////////////////////////
```

```
MOV #PARMBK,RO ; RO=PARAMETER BLOCK ADDRESS
JSR PC,a#REWIND ; START A REWIND
TYPE ,MSG1 ; <CRLF>"*** END-OF-TEST ***"
```

```
TYPE MSG2 ; <CRLF>"SOFT ERRORS="
MOV a#SOFTNM,-(SP) ; PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,a#SSB2D ; CHANGE IT TO DECIMAL ASCIZ
JSR PC,a#SSUPRS ; TYPE WITHOUT LEADING ZEROS
```

```
TYPE MSG3 ; <CRLF>"HARD ERRORS="
MOV a#HARDNM,-(SP) ; PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,a#SSB2D ; CHANGE IT TO DECIMAL ASCIZ
JSR PC,a#SSUPRS ; TYPE WITHOUT LEADING ZEROS
```

```
TYPE MSG4 ; <CRLF>"BYTES READ="
MOV #RBYTTL,-(SP) ; GET ADDRESS OF DOUBLE PRECISION BINARY #
JSR PC,a#SDB2D ; CHANGE IT TO DECIMAL ASCIZ
JSR PC,a#SSUPRS ; TYPE IT WITHOUT LEADING ZERO
```

```
TYPE MSG5 ; <CRLF>"BYTES WRITTEN="
MOV #WBYTTL,-(SP) ; GET ADDRESS OF DOUBLE PRECISION BINARY #
JSR PC,a#SDB2D ; CHANGE IT TO DECIMAL ASCIZ
JSR PC,a#SSUPRS ; TYPE IT WITHOUT LEADING ZERO
```

```
TYPE MSG6 ; <CRLF>"TAPE PASSES="
MOV a#EOTS,-(SP) ; PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,a#SSB2D ; CHANGE IT TO DECIMAL ASCIZ
JSR PC,a#SSUPRS ; TYPE WITHOUT LEADING ZEROS
```

2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161 004406  
2162 004406 000004  
2163 004410 005067 174466  
2164 004414 005267 174460  
2165 004420 042767 100000 174452  
2166 004426 005327  
2167 004430 000001  
2168 004432 003135  
2169 004434 012737  
2170 004436 000001  
2171 004440 004430  
2172 004442 104400 004735  
2173 004446 013700 000042  
2174 004452 001525  
2175 004454 000004  
2176  
2177  
2178  
2179 004456 012700 001612  
2180 004462 004737 006746  
2181 004466 104400 013005  
2182  
2183 004472 104400 013032  
2184 004476 013746 001206  
2185 004502 004737 007356  
2186 004506 004737 007412  
2187  
2188 004512 104400 013050  
2189 004516 013746 001210  
2190 004522 004737 007356  
2191 004526 004737 007412  
2192  
2193 004532 104400 013066  
2194 004536 012746 001214  
2195 004542 004737 007162  
2196 004546 004737 007412  
2197  
2198 004552 104400 013103  
2199 004556 012746 001220  
2200 004562 004737 007162  
2201 004566 004737 007412  
2202  
2203 004572 104400 013123  
2204 004576 013746 001212  
2205 004602 004737 007356  
2206 004606 004737 007412

```

2207
2208 004612 104400 013141 TYPE MSG7 ;<CRLF>"FILES/PASS="
2209 004616 013700 001236 MOV @#LASTFL,RO ;PICKUP THE LAST FILE NUMBER
2210 004622 105200 INCB RO ;ADD 1 TO MAKE IT # OF FILES
2211 004624 010046 MOV RO,-(SP) ;PUT IT ON THE STACK
2212 004626 004737 007356 JSR PC,@#$$SB2D ;CHANGE IT TO DECIMAL ASCIZ
2213 004632 004737 007412 JSR PC,@#$$SUPRS ;TYPE IT WITHOUT LEADING ZEROS
2214
2215 004636 104400 013156 TYPE MSG8 ;<CRLF>"BLOCKS/FILE="
2216 004642 013746 001310 MOV @#FILESZ,-(SP) ;PICKUP SINGLE PRECISION BINARY NUMBER
2217 004646 004737 007356 JSR PC,@#$$SB2D ;CHANGE IT TO DECIMAL ASCIZ
2218 004652 004737 007412 JSR PC,@#$$SUPRS ;TYPE WITHOUT LEADING ZEROS
2219
2220 004656 104400 001204 TYPE ,SLF
2221 ; ////////////////////////////////////////////////////////////////////
2222 ; CHECK HALT AT END-OF-TEST SWITCH
2223 ; ////////////////////////////////////////////////////////////////////
2224 004662 032777 001000 174250 BIT #SW09,@SWR ;HALT AT END-OF-TEST?
2225 004670 001406 BEQ 100$ ;BR IF NO
2226 004672 000000 HALT ;YES
2227
2228 004674 022767 000176 174236 CMP #SWREG,SWR ;USING S/W SWITCH REG?
2229 004702 001001 BNE 20$ ;NO- GET OUT
2230 004704 104404 GTSWR ;GET VALUE
2231 004706 20$: ;CONTINUE
2232- 004706 100$:
2233 004706 013700 000042 MOV @#42,RO ;INSURE RO CONTAINS THE MONITORS
2234 004712 001405 BEQ $DOAGN ;RETURN ADDRESS
2235 004714 000005 RESET ;CLEAR THE WORLD
2236 004716 004710 SENDAD: JSR PC,(RO) ;GO TO MONITOR
2237 004720 000240 NOP ;SAVE ROOM
2238 004722 000240 NOP ;FOR
2239 004724 000240 NOP ;ACT11
2240 004726 $DOAGN:
2241 004726 000137 JMP @PC)+ ;RETURN
2242 004730 $RTNAD: .WORD START
2243 004732 377 $ENULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
2244 004735 015 042412 042116 SENDMG: .ASCIZ <15><12>/END PASS/
2245 004742 050040 051501 000123

```

.SBTTL SCOPE HANDLER ROUTINE

```

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

```

```

SSCOPE:
        CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
1$:     BIT           #BIT14,$SWR      ;;LOOP ON PRESENT TEST?
        BNE          $OVER           ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR           b$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                           ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
                                           ;;SAVE THE CONTENTS OF THE ERROR VECTOR
        MOV          @ERRVEC, -(SP)   ;;SET FOR TIMEOUT
        MOV          @SS,@ERRVEC     ;;TIME OUT ON XOR?
        TST         @177060         ;;RESTORE THE ERROR VECTOR
        MOV          (SP)+,@ERRVEC   ;;GO TO THE NEXT TEST
        BR          $SVLAD          ;;CLEAR THE STACK AFTER A TIME OUT
5$:     CMP          (SP)+,(SP)+     ;;RESTORE THE ERROR VECTOR
        MOV          (SP)+,@ERRVEC   ;;LOOP ON THE PRESENT TEST
        BR          $OVER
b$:     *****END OF CODE FOR THE XOR TESTER*****
$SVLAD: INCB         $TSTNM          ;;COUNT TEST NUMBERS
        MOV          (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
        CLRB        $ERFLG         ;;ZERO THE ERROR FLAG
$OVER:  MOV          $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
        MOV          $LPADR,(SP)    ;;FUDGE RETURN ADDRESS
        RTI                    ;;FIXES PS

```

```

2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257 004750
2258 004750 104405
2259 004752 032777 040000 174160
2260 004760 001025
2261
2262 004762 000416
2263
2264 004764 013746 000004
2265 004770 012737 005010 000004
2266 004776 005737 177060
2267 005002 012637 000004
2268 005006 000404
2269 005010 022626
2270 005012 012637 000004
2271 005016 000406
2272 005020
2273 005020 105267 174056
2274 005024 011667 174056
2275 005030 105067 174047
2276 005034 016777 174042 174100
2277 005042 016716 174040
2278 005046 000002

```

```

2279 .SBTTL ERROR HANDLER ROUTINE
2280
2281 ::*****
2282 *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2283 *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2284 *AND GO TO TYPERR ON ERROR
2285 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2286 *SW15=1 HALT ON ERROR
2287 *SW13=1 INHIBIT ERROR TYPEOUTS
2288 *SW10=1 BELL ON ERROR
2289 *CALL
2290 *
2291 * ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2292
2293 $ERROR:
2294 005050 104405 174025 7$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2295 005052 105267 174025 INCB $ERFLG ;;SET THE ERROR FLAG
2296 005056 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
2297 005060 016777 174016 174054 MOV $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
2298 005066 032777 002000 174044 BIT #BIT10,$SWR ;;BELL ON ERROR?
2299 005074 001402 BEQ 1$ ;;NO - SKIP
2300 005076 104400 005174 TYPE $BELL ;;RING BELL
2301 005102 005267 174004 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
2302 005106 011667 174004 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
2303 005112 162767 000002 173776 SUB #2,$ERRPC
2304 005120 117767 173772 173766 MOVB $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
2305 005126 032777 020000 174004 BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
2306 005134 001004 BNE 20$ ;;SKIP TYPEOUTS
2307 005136 004767 000036 JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
2308 005142 104400 001203 TYPE ,$CRLF
2309 005146 005777 173766 20$: TST $SWR ;;HALT ON ERROR
2310 005152 100002 BPL 3$ ;;SKIP IF CONTINUE
2311 005154 000000 HALT ;;HALT ON ERROR!
2312 005156 104405 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2313 005160
2314 005160 022737 004716 000042 3$: CMP #SENDAD,$#42 ;;ACT-11 AUTO-ACCEPT?
2315 005166 001001 BNE 6$ ;;BRANCH IF NO
2316 005170 000000 HALT ;;YES
2317 005172
2318 005172 000002 6$: RTI ;;RETURN
2319 005174 177607 000377 $BELL: .ASCIZ <207><377><377> ;;ASCII CODE FOR BELL

```

```

2320          ::*****
2321          ::THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
2322 005200 104400 001203  TYPERR: TYPE      $CRLF          ;TYPE "CR" & "LF"
2323 005204 010046          MOV      R0,-(SP)        ;SAVE R0
2324 005206 010146          MOV      R1,-(SP)        ;SAVE R1
2325 005210 005046          CLR      -(SP)          ;PICKUP THE ITEM BYTE
2326 005212 113716 001114  MOVB    @#SITEMB,(SP)
2327 005216 005316          DEC      (SP)          ;ADJUST THE INDEX SO IT
2328 005220 006316          ASL      (SP)          ;WILL WORK FOR THE
2329 005222 006316          ASL      (SP)          ;ERROR TABLE
2330 005224 006316          ASL      (SP)
2331 005226 116601 000001  MOVB    1(SP),R1        ;GET THE ERROR TYPE
2332 005232 112600          MOVB    (SP)+,R0       ;GET THE ERROR NUMBER
2333 005234 066100 005242  ADD     1$(R1),R0       ;FORM THE TABLE POINTER
2334 005240 000403          BR      2$
2335 005242 001622          1$: ITEMS0
2336 005244 001642          ITEMS1
2337 005246 001732          ITEMS2
2338
2339 005250 012067 000002  2$: MOV     (R0)+,3$     ;PICKUP "ERROR MESSAGE" POINTER
2340 005254 104400          TYPE    ;TYPE "ERROR MESSAGE"
2341 005256 000000          3$: 0
2342 005260 104400 001203  TYPE    $CRLF          ;TYPE "CR" & "LF"
2343 005264 012067 000002  MOV     (R0)+,4$     ;PICKUP "DATA HEADER" POINTER
2344 005270 104400          TYPE    ;TYPE "DATA HEADER"
2345 005272 000000          4$: 0
2346 005274 104400 001203  TYPE    $CRLF          ;TYPE "CR" & "LF"
2347 005300 060107          ADD     R1,PC          ;GO TYPE THE DATA
2348 005302 000402          BR      ERROR0
2349 005304 000430          BR      ERROR1
2350 005306 000473          BR      ERROR2
2351
2352          ::////////////////////
2353 005310 010246  ERROR0: MOV     R2,-(SP)        ;SAVE R2
2354 005312 012001          MOV     (R0)+,R1     ;PICKUP THE "DATA POINTER"
2355 005314 012002          MOV     (R0)+,R2     ;PICKUP "FORMAT" POINTER
2356 005316 000402          BR      2$
2357 005320 104400 013277  1$: TYPE    MSG14        ;" "
2358 005324 012100          2$: MOV     (R1)+,R0     ;GET ADDRESS OF DATA WORD
2359 005326 001473          BEQ     EREXT1       ;GO TO EXIT IF 0
2360 005330 105722          TSTB   (R2)+         ;TYPE DECIMAL OR OCTAL?
2361 005332 001003          BNE    3$           ;BR IF DECIMAL
2362 005334 011046          MOV     (R0),-(SP)   ;;SAVE (R0) FOR TYPEOUT
2363 005336 104401          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2364 005340 000767          BR      1$           ;GO GET NEXT DATA WORD
2365 005342 010046          3$: MOV     R0,-(SP)
2366 005344 004737 007162  JSR     PC,@#SDB2D    ;CHANGE NUMBER TO ASCIZ
2367 005350 062716 000004  ADD     #4,(SP)       ;TYPE 6 DIGITS
2368 005354 012667 000002  MOV     (SP)+,4$
2369 005360 104400          TYPE    ;CALL TYPE ASCII MESSAGE ROUTINE
2370 005362 000000          .WORD  0           ;ADDRESS OF ASCII STRING
2371 005364 000755          BR      1$           ;GO GET NEXT DATA WORD

```



```

2372
2373 005366 011001
2374 005370 013146
2375
2376 005372 104401
2377 005374 012767 000003 000006
2378
2379 005402 104400 013277
2380 005406 005327
2381 005410 000000
2382 005412 001412
2383 005414 012146
2384 005416 004737 007162
2385 005422 062716 000004
2386 005426 012667 000002
2387 005432 104400
2388 005434 000000
2389 005436 000761
2390 005440 013101
2391 005442 016167 005456 000002
2392 005450 104400
2393 005452 000000
2394 005454 000421
2395
2396
2397 005456 012043
2398 005460 012062
2399 005462 012070
2400 005464 012075
2401 005466 012121
2402 005470 012144
2403 005472 012167
2404 005474 012211
2405
2406
2407 005476 011000
2408 005500 005710
2409 005502 001406
2410 005504 013046
2411 005506 104401
2412 005510 104400 013277
2413 005514 000771
2414
2415
2416 005516 012602
2417 005520 012601
2418 005522 012600
2419 005524 000207
2420
2421

```

```

////////////////////////////////////
ERROR1: MOV (R0),R1 ;PICKUP THE "DATA POINTER"
MOV @ (R1)+,-(SP) ;;SAVE @ (R1)+ FOR TYPEOUT
;;SERRPC
;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPOC ;PRINT 2 SPACES 3 TIMES
MOV #3,2$ ;AND 2 DECIMAL #'S 2 TIMES
1$: TYPE MSG14
DEC (PC)+ ;TYPE ANOTHER # ?
2$: .WORD 0
BEQ 4$ ;BR IF NO
MOV (R1)+,-(SP) ;YES---PICKUP ADDRESS OF #
JSR PC,@#DB2D ;GO CHANGE TO DECIMAL ASCIZ
ADD #4,(SP) ;ONLY TYPE THE LAST 6 DIGITS
MOV (SP)+,3$ ;SAVE ADDRESS OF ASCIZ STRING
TYPE ;CALL THE TYPE ROUTINE
3$: .WORD 0 ;POINTER GOES HERE
BR 1$ ;LOOP
4$: MOV @ (R1)+,R1 ;PICKUP FUNCTION INDEX
MOV BADFUN(R1),5$ ;GET THE FUNCTION MESSAGE
TYPE ;AND TYPE IT
5$: .WORD 0 ;MESSAGE POINTER GOES HERE
BR EREXT2 ;GO TO EXIT
;THIS TABLE CONTAINS THE POINTERS TO THE DIFFERENT ASCIZ MESSAGES
;FOR THE FUNCTION BEING PERFORMED WHEN THE ERROR OCCURRED.
BADFUN: MXWFG ;WRITE-FILE-GAP
MXWRIT ;WRITE
MXREAD ;READ
MXBSFG ;BACK-SPACE-FILE-GAP
MXBSBG ;BACK-SPACE-BLK-GAP
MXSFFG ;SPACE-FWD-FILE-GAP
MXSFBG ;SPACE-FWD-BLK-GAP
MXRWND ;REWIND
////////////////////////////////////
ERROR2: MOV (R0),R0 ;PICKUP THE DATA POINTER
1$: TST (R0) ;ALL DATA TYPED?
BEQ EREXT2 ;BR IF YES
MOV @ (R0)+,-(SP) ;;SAVE @ (R0)+ FOR TYPEOUT
;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPOC ;PRINT 2 SPACES 3 TIMES
MOV #3,2$ ;AND 2 DECIMAL #'S 2 TIMES
BR 1$ ;LOOP
////////////////////////////////////
EREXT1: MOV (SP)+,R2 ;RESTORE R2
EREXT2: MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
RTS PC ;RETURN

```

```

2422      ;////////////////////////////////////
2423      ;////////////////////////////////////
2424      .SBTTL      DETERMINE CONTROL AND STATUS ERROR
2425
2426 005526 113737 001304 001162 CSRERR: MOVB  @#FILE,@#$REG0      ;GET THE FILE NUMBER
2427 005534 113737 001306 001166      MOVB  @#BLOCK,@#$REG2      ;GET THE BLOCK NUMBER
2428 005542 013746 007160      MOV   @#FUNLOC,-(SP)      ;GET THE LAST FUNCTION
2429 005546 042716 177761      BIC   @#CFUNCTION,(SP)   ;CLEAR AWAY THE JUNK
2430 005552 011637 001172      MOV   (SP),@#STMP0      ;SAVE THE FUNCTION CODE
2431 005556 113766 001612 000001      MOVB  @#PARMBK,1(SP)     ;COMBINE THE ST/ER WITH
2432 005564 012601      MOV   (SP)+,R1          ;THE FUNCTION
2433 005566 005046      CLR   -(SP)             ;FIND OUT WHAT CAUSED THE ERROR
2434 005570 032701 001000      BIT   #OFFLINE,R1       ;OFF LINE?
2435 005574 001025      BNE   6$                ;BR IF YES
2436 005576 032701 010000      BIT   #WRTLOCK,R1      ;WRITE LOCK?
2437 005602 001021      BNE   5$                ;BR IF YES
2438 005604 032701 020000      BIT   #LEADER,R1       ;BOT/EOT?
2439 005610 001015      BNE   4$                ;BR IF YES
2440 005612 032701 004000      BIT   #FGAP,R1         ;FILE GAP?
2441 005616 001011      BNE   3$                ;BR IF YES
2442 005620 032701 002000      BIT   #TIMERR,R1       ;TIMING?
2443 005624 001005      BNE   2$                ;BR IF YES
2444 005626 032701 040000      BIT   #CRCERR,R1      ;BLOCK CHECK?
2445 005632 001001      BNE   1$                ;BR IF YES
2446 005634 005216      INC   (SP)              ; 6 = UNKNOWN
2447 005636 005216      1$: INC   (SP)           ; 5 = CRCERR
2448 005640 005216      2$: INC   (SP)           ; 4 = TIMERR
2449 005642 005216      3$: INC   (SP)           ; 3 = FGAP
2450 005644 005216      4$: INC   (SP)           ; 2 = LEADER
2451 005646 005216      5$: INC   (SP)           ; 1 = WRTLOCK
2452 005650      6$: INC   (SP)
2453 005650 106316      ASLB  (SP)              ;POSITION INDEX
2454 005652 062607      ADD   (SP)+,PC          ;BR TO THE ROUTINE
2455 005654 000406      BR   OL.ERR
2456 005656 000407      BR   WL.ERR
2457 005660 000410      BR   CL.ERR
2458 005662 000462      BR   FG.ERR
2459 005664 000463      BR   TM.ERR
2460 005666 000464      BR   BC.ERR
2461 005670 000472      BR   X.ERR
2462
2463 005672 104101      OL.ERR: ERROR 101      ;DRIVE IS OFFLINE
2464 005674 000200      RTS   RO
2465
2466 005676 104102      WL.ERR: ERROR 102      ;DRIVE IS WRITE LOCK
2467 005700 000200      RTS   RO
2468

```



```

2510      ;:*****
2511      ;:*****
2512      .SBTTL          ROUTINE TO SETUP FOR A WRITE OPERATION
2513
2514      ;THIS ROUTINE WILL SETUP THE BYTE COUNT, DETERMINE THE PATTERN TO
2515      ;BE WRITTEN ON TAPE, ADD THE SIZE OF THIS BLOCK TO THE TOTAL
2516      ;NUMBER OF BYTES TRANSFERRED AND CALL THE ROUTINE TO FILL THE
2517      ;WRITE BUFFER.
2518
2519      ;CALL
2520      ;      JSR      PC, @#SETUPW
2521
2522
2523      006062 013700 001306      SETUPW: MOV      @#BLOCK, R0      ;GET THE BLOCK NUMBER
2524      006066 042700 177760      BIC      #15., R0      ;KEEP MAX PATTERN
2525      006072 006300      ASL      R0      ;*2
2526      006074 066037 001312 001220      ADD      BLKSZ(R0), @#WBYTTL      ;ADD THE BLOCK SIZE TO THE TOTAL
2527      006102 005537 001222      ADC      @#WBYTTL+2      ;NUMBER OF BYTES WRITTEN
2528      006106 016037 001312 001620      MOV      BLKSZ(R0), @#PARMBK+6      ;SET THE BYTE COUNT
2529      006114 016000 001352      MOV      PATS(R0), R0      ;GET PATTERN POINTER
2530      006120 004737 006126      JSR      PC, @#FILL      ;GO FILL WRITE BUFFER
2531      006124 000207      RTS      PC      ;RETURN
2532
2533      ;:*****
2534      .SBTTL          ROUTINE TO FILL THE BUFFER BEFORE A WRITE
2535
2536      ;CALL
2537      ;      MOV      #PATADR, R0      ;1ST ADDRESS OF 8 BYTE PATADR
2538      ;      JSR      PC, @#FILL
2539      ;THE BUFFER IS FILLED AS FOLLOWS:
2540      ;BYTE# DATA
2541      ;0001 FILE NUMBER
2542      ;0002 FILE NUMBER COMPLEMENTED
2543      ;0003 BLOCK NUMBER
2544      ;0004 BLOCK NUMBER COMPLEMENTED
2545      ;0005 DATA PATTERN
2546      ;0006 DATA PATTERN
2547      ;* * *
2548      ;* * *
2549      ;* * *
2550      ;1027. DATA PATTERN
2551      ;1028. DATA PATTERN
2552
2553
2554      006126      FILL:
2555      006126 010046      MOV      R0, -(SP)      ;:PUSH R0 ON STACK
2556      006130 010146      MOV      R1, -(SP)      ;:PUSH R1 ON STACK
2557      006132 010246      MOV      R2, -(SP)      ;:PUSH R2 ON STACK
2558      006134 010346      MOV      R3, -(SP)      ;:PUSH R3 ON STACK
2559      006136 012701 000200      MOV      #1024./8., R1      ;FILL 1024. BYTES WITH PATTERN
2560      006142 012702 013440      MOV      #BUFFER, R2      ;1ST ADDRESS OF DATA BUFFER
2561      006146 013722 001304      MOV      @#FILE, (R2)+      ;FILE NUMBER AND ITS COMPLEMENT
2562      006152 013722 001306      MOV      @#BLOCK, (R2)+      ;BLOCK NUMBER AND ITS COMPLEMENT
2563      006156 010003      MOV      R0, R3      ;SAVE FIRST ADDRESS OF PATTERN
2564      006160 010300      1$: MOV      R3, R0      ;RESET PATTERN POINTER
2565      006162 012022      MOV      (R0)+, (R2)+      ;MOVE THE PATTERN

```

```

2566 006164 012022      MOV      (R0)+,(R2)+      ; INTO THE BUFFER AREA.
2567 006166 012022      MOV      (R0)+,(R2)+
2568 006170 012022      MOV      (R0)+,(R2)+
2569 006172 005301      DEC      R1
2570 006174 001371      BNE     1$
2571 006176 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
2572 006200 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
2573 006202 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
2574 006204 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
2575 006206 000207      RTS      PC
2576
2577      ;; *****
2578      ;; *****
2579      .SBTTL      ROUTINE TO SETUP FOR A READ
2580
2581      ; THIS ROUTINE WILL SETUP THE BYTE COUNT AND ADD THE SIZE OF THIS
2582      ; BLOCK TO THE TOTAL NUMBER OF BYTES READ.
2583
2584      ; CALL
2585      ;      JSR      PC,SETUPR
2586
2587
2588 006210 013700 001306      SETUPR: MOV      @#BLOCK,R0      ; GET THE BLOCK NUMBER
2589 006214 042700 177760      BIC      #1C15.,R0      ; KEEP MAX PATTERN
2590 006220 006300      ASL      R0      ; X2
2591 006222 066037 001312 001214      ADD      BLKSZ(R0),@#RBYTTL ; ADD THE BLOCK SIZE TO THE TOTAL
2592 006230 005537 001216      ADC      @#RBYTTL+2      ; NUMBER OF BYTES READ
2593 006234 016037 001312 001620      MOV      BLKSZ(R0),@#PARMBK+6 ; SET THE BYTE COUNT
2594 006242 000207      RTS      PC      ; RETURN
2595
2596      ;; *****
2597      .SBTTL      ROUTINE TO CHECK FOR SYNC PROBLEMS
2598
2599      ; THIS ROUTINE CHECKS THE FIRST FOUR BYTES OF THE DATA
2600      ; BUFFER TO INSURE THE PROPER BLOCK WAS READ
2601      ; BYTE 1 = FILE NUMBER
2602      ; BYTE 2 = FILE NUMBER COMPLEMENT
2603      ; BYTE 3 = BLOCK NUMBER
2604      ; BYTE 4 = BLOCK NUMBER COMPLEMENT
2605
2606      ; CALL
2607      ;      JSR      PC,@#SYNCK
2608      ;      NORMAL RETURN
2609      ;      ERROR RETURN
2610
2611
2612
2613 006244 012700 013440      SYNCK:  MOV      #BUFFER,R0      ; ADDRESS OF FIRST WORD
2614 006250 011001      MOV      (R0),R1      ; CHECK FILE # WAS READ OK
2615 006252 000301      SWAB    R1      ; BYTE 1 SHOULD BE THE
2616 006254 062001      ADD      (R0)+,R1      ; COMPLEMENT OF BYTE 0
2617 006256 005101      COM     R1
2618 006260 001031      BNE     2$      ; BR IF FILE NUMBER READ WRONG
2619 006262 011001      MOV      (R0),R1      ; CHECK BLOCK # WAS READ OK
2620 006264 000301      SWAB    R1      ; BYTE 3 SHOULD BE THE
2621 006266 062001      ADD      (R0)+,R1      ; COMPLEMENT OF BYTE 2

```



```

2678 006524 105737 001612      5$:  TSTB  @#PARMBK      ;DATA IS GOOD BUT WHAT ABOUT
2679                                     ;THE ERROR FLAG
2680 006530 100002                                     ;BR IF ERROR FLAG=0
2681 006532 062716 000004      6$:  BPL  6$      ;SETUP TO TAKE ERROR2 EXIT
2682 006536 000207      ADD  #4,(SP)
2683      RTS  PC      ;RETURN
2684                                     ;*****
2685                                     ;
2686      .SBTTL      COUNT SOFT DATA ERROR
2687
2688      ;CALL
2689      ;          JSR  RD,@#CNTSFT
2690      ;          RETURN FOR SOFT RETRY
2691      ;          RETURN FOR HARD
2692
2693 006540 005237 001206      CNTSFT: INC  @#SOFTNM      ;COUNT THIS SOFT ERROR
2694 006544 005337 001224      DEC  @#RDTRY5      ;REREAD THE RECORD?
2695 006550 002011      BGE  1$      ;BR IF YES
2696 006552 013737 001260 001224      MOV  @#MAXRDS,@#RDTRY5      ;RESET REREAD COUNT
2697 006560 163737 001260 001206      SUB  @#MAXRDS,@#SOFTNM      ;DON'T COUNT AS SOFT
2698 006566 005337 001206      DEC  @#SOFTNM
2699 006572 005720      TST  (RD)+      ;STEP OVER THE REREAD RETURN
2700 006574 000200      1$:  RTS  RD      ;RETURN
2701                                     ;*****
2702                                     ;
2703      .SBTTL      COUNT HARD ERROR
2704
2705      ;CAL
2706      ;          JSR  RD,@#CNTHRD
2707      ;          RETURN MORE ERRORS ALLOWED
2708      ;          RETURN MAX. ERRORS HAVE OCCURRED
2709
2710 006576 005237 001210 001262      CNTHRD: INC  @#HARDNM      ;COUNT THIS HARD ERROR
2711 006602 023737 001210      CMP  @#HARDNM,@#MAXERR      ;MAX. HARD ERRORS OCCURRED
2712 006610 002401      BLT  1$      ;BR IF NO
2713 006612 005720      TST  (RD)+      ;TAKE FATAL EXIT
2714 006614 000200      1$:  RTS  RD      ;RETURN

```

2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762  
2763

;;\*\*\*\*\*

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

;;\*\*\*\*\*

;\*SAVE R0-R5

;\*CALL:

;\* SAVREG

;\*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:

;\*

;\*TOP---(+16)

;\* +2---(+18)

;\* +4---R5

;\* +6---R4

;\* +8---R3

;\*+10---R2

;\*+12---R1

;\*+14---R0

\$SAVREG:

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	R5,-(SP)	;;	PUSH R5 ON STACK
MOV	22(SP),-(SP)	;;	SAVE PS OF MAIN FLOW
MOV	22(SP),-(SP)	;;	SAVE PC OF MAIN FLOW
MOV	22(SP),-(SP)	;;	SAVE PS OF CALL
MOV	22(SP),-(SP)	;;	SAVE PC OF CALL
RTI			

006616			
006616	010046		
006620	010146		
006622	010246		
006624	010346		
006626	010446		
006630	010546		
006632	016646	000022	
006636	016646	000022	
006642	016646	000022	
006646	016646	000022	
006652	000002		

;\*RESTORE R0-R5

;\*CALL:

;\* RESREG

\$RESREG:

MOV	(SP)+,22(SP)	;;	RESTORE PC OF CALL
MOV	(SP)+,22(SP)	;;	RESTORE PS OF CALL
MOV	(SP)+,22(SP)	;;	RESTORE PC OF MAIN FLOW
MOV	(SP)+,22(SP)	;;	RESTORE PS OF MAIN FLOW
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
RTI			

006654			
006654	012666	000022	
006660	012666	000022	
006664	012666	000022	
006670	012666	000022	
006674	012605		
006676	012604		
006700	012603		
006702	012602		
006704	012601		
006706	012600		
006710	000002		



```

2764 ;*****
2765
2766 .SBTTL          CASSETTE PRIMITIVES ROUTINE
2767
2768 ;CALLED VIA     JSR PC,(FUNCTION)
2769 ;WITH RD POINTING TO:
2770
2771 .BYTE          0
2772 .BYTE          DRIVE # (0=A,1=B)
2773 .WORD          POINTER TO STATUS/ERROR BYTE
2774 .WORD          BUFFER ADDRESS
2775 .WORD          BYTE COUNT
2776
2777 WFG:           MOV      PC,-(SP)          ; WRITE A FILE GAP
2778 FUNTAB:       BR       IOCOM
2779 WRITE:        MOV      PC,-(SP)          ; WRITE A BLOCK OF DATA
2780              BR       IOCOM
2781 READ:         MOV      PC,-(SP)          ; READ A BLOCK OF DATA
2782              BR       IOCOM
2783 BSFG:         MOV      PC,-(SP)          ; BACK SPACE A FILE GAP
2784              BR       IOCOM
2785 BSBG:         MOV      PC,-(SP)          ; BACK SPACE A BLOCK GAP
2786              BR       IOCOM
2787 SFFG:         MOV      PC,-(SP)          ; SPACE FORWARD A FILE GAP
2788              BR       IOCOM
2789 SFBG:         MOV      PC,-(SP)          ; SPACE FORWARD A BLOCK GAP
2790              BR       IOCOM
2791 REWIND:       MOV      PC,-(SP)          ; REWIND TO BEGINNING OF TAPE
2792 IOCOM:        SUB      #FUNTAB,(SP)      ; PC ON STACK IS USED TO DETERMINE
2793              ASR      (SP)              ; FUNCTION
2794              BIT      #40,@TACSL        ; READY BIT UP?
2795              BEQ      CHKRDY            ; WAIT ON IT
2796              MOV      (SP)+,FUNLOC      ; PUT FUNC. IN BITS 1-3 OF FUNLOC
2797              MOV      R1,-(SP)
2798              MOV      RD,-(SP)
2799              MOV      #CASPAR,R1        ;PT. TO INT. HANDLER PARAM. BLK.
2800              MOV      2(RD),(R1)        ;PTR. TO ST/ERR WD. INTO P.BLK.
2801              CLRB   @R1+                ;ZERO ST/ERR BYTE
2802              INC     RD
2803 CHANOK:       BISB   (RD)+,FUNLOC+1      ;COPY CHAN. NUM. TO FUNC. WORD
2804              TST    (RD)+
2805              MOV    (RD)+,2(R1)         ;BUFF. ADR.
2806              MOV    (RD),(R1)+         ;BYTE COUNT
2807              TST    (R1)+
2808              BISB   #101,(R1)          ;ADD INT. ENABLE AND GO BITS
2809              ;TO THE FUNCTION
2810 DOFUN:        MOV    (R1),@TACSL        ;START THE FUNCTION
2811              MOV    (SP)+,RD
2812              MOV    (SP)+,R1
2813              RTS     PC
    
```

```

2814                                     ;*****
2815                                     ;
2816                                     ;SBTTL          CASSETTE INTERRUPT HANDLER
2817                                     ;
2818                                     ;
2819                                     ;CASINT:
2820 007046 010146                       MOV     R1,-(SP)          ;;PUSH R1 ON STACK
2821 007050 010246                       MOV     R2,-(SP)          ;;PUSH R2 ON STACK
2822 007052 016702 172162                MOV     TACSL,R2
2823 007056 012701 007154                MOV     #CBCNT,R1      ;PARAM. BLOCK
2824 007062 105712                       TSTB   (R2)             ;TRANSFER REQUEST?
2825 007064 100411                       BMI    TREQ             ;YES
2826 007066 011204                       MOV     (R2),R4
2827 007070 000304                       SWAB   R4
2828 007072 052704 000001                BIS    #1,R4           ;SET THE DONE BIT
2829 007076 110451                       MOVB   R4,@-(R1)       ;LOAD ST/ERR BYTE
2830 007100 105012                       CLRB   (R2)            ;TURN OFF INTERS.
2831 007102 012602                       MOV     (SP)+,R2       ;;POP STACK INTO R2
2832 007104 012601                       MOV     (SP)+,R1       ;;POP STACK INTO R1
2833 007106 000002                       RTI
2834
2835 007110 005321                       ;TREQ:  DEC    (R1)+
2836 007112 100412                       BMI    RWILBS          ;WHEN COUNT NEG.  DONE
2837 007114 032712 000004                BIT    #4,(R2)        ;CHECK READ OR WRITE FUNC.
2838 007120 001003                       BNE    READR
2839 007122 113177 172114                MOVB   @R1+,@TADBL     ;WRITE BYTE
2840 007126 000402                       BR     RWDONE
2841 007130 117731 172106                READR: MOVB   @TADBL,@(R1)+ ;READ BYTE
2842 007134 005241                       RWDONE: INC    -(R1)   ;BUMP  ADDR.
2843 007136 000402                       BR     BACK           ;GET BACK
2844
2845 007140 052712 000020                ;RWILBS: BIS    #20,(R2) ;INIT. LAST BYTE SEQ.
2846 007144
2847 007144 012602                       BACK:  MOV     (SP)+,R2  ;;POP STACK INTO R2
2848 007146 012601                       MOV     (SP)+,R1       ;;POP STACK INTO R1
2849 007150 000002                       RTI
2850
2851                                     ;
2852                                     ; CASSETTE INTER. HANDLER PARAM. BLOCK
2853                                     ;
2854 007152 000000                CASPAR: .WORD 0          ;STATUS/ERROR POINTER
2855 007154 000000                CBCNT:  .WORD 0          ;WORKING BYTE COUNT
2856 007156 000000                .WORD 0                  ;WORKING BUFFER ADDRESS
2857 007160 000000                FUNLOC: .WORD 0          ;FUNC. WORD BUILT HERE
2858
2859

```

```

2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874 007162 104411
2875 007164 016602 000002
2876 007170 012700 007342
2877 007174 010066 000002
2878 007200 012201
2879 007202 012202
2880 007204 012767 000012 000046
2881 007212 012704 007272
2882 007216 012705 007274
2883 007222 005003
2884 007224 161401
2885 007226 005602
2886 007230 161502
2887 007232 002402
2888 007234 005203
2889 007236 000772
2890 007240 062401
2891 007242 005502
2892 007244 062402
2893 007246 022525
2894 007250 052703 000060
2895 007254 110320
2896 007256 005327
2897 007260 000000
2898 007262 001357
2899 007264 105020
2900 007266 104412
2901 007270 000207
2902 007272 145000
2903 007274 035632
2904 007276 160400
2905 007300 002765
2906 007302 113200
2907 007304 000230
2908 007306 041100
2909 007310 000017
2910 007312 103240
2911 007314 000001
2912 007316 023420
2913 007320 000000
2914 007322 001750
2915 007324 000000

;*****
;SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
;*****
;THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
;DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
;POSITIVE.
;CALL
;*
;* MOV #PNTR, -(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
;* JSR PC, @#$DB2D
;* RETURN ;; THE FIRST ADDRESS OF ASCII
;; IS ON THE STACK

$DB2D: SAVREG ;; SAVE REGISTERS
MOV 2(SP), R2 ;; PICKUP THE DATA POINTER
MOV #SDECVL, R0 ;; GET ADDRESS OF "SDECVL" STRING
MOV R0, 2(SP) ;; PUT ADDRESS OF ASCII STRING ON STACK
MOV (R2)+, R1 ;; PICKUP THE BINARY NUMBER
MOV (R2)+, R2
MOV #10, 4$ ;; SET UP TO DO 10 CONVERSIONS
MOV #STNPWR, R4 ;; ADDRESS OF TEN POWER
MOV #STNPWR+2, R5
1$: CLR R3 ;; CLEAR PARTIAL
2$: SUB (R4), R1 ;; SUBTRACT TEN POWER
SBC R2
SUB (R5), R2
BLT 3$ ;; BR IF TEN POWER TOO LARGE
INC R3 ;; ADD 1 TO PARTIAL
BR 2$ ;; LOOP
3$: ADD (R4)+, R1 ;; RESTORE SUBTRACTED VALUE
ADC R2
ADD (R4)+, R2
CMP (R5)+, (R5)+ ;; MOVE TO NEXT TEN POWER
BIS #0, R3 ;; CHANGE PARTIAL TO ASCII
MOV R3, (R0)+ ;; SAVE IT
DEC (PC)+ ;; DONE?
4$: .WORD 0
BNE 1$ ;; BR IF NO
CLRB (R0)+ ;; TERMINATOR
RESREG ;; RESTORE REGISTERS
RTS PC ;; RETURN
STNPWR: 145000 ;; 1.0E09
35632
160400 ;; 1.0E08
2765
113200 ;; 1.0E07
230
041100 ;; 1.0E06
17
103240 ;; 1.0E05
1
23420 ;; 1.0E04
0
1750 ;; 1.0E03
0

```

2916 007326 000144  
2917 007330 000000  
2918 007332 000012  
2919 007334 000000  
2920 007336 000001  
2921 007340 000000  
2922 007342 000014  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934 007356 016667 000002 000022  
2935 007364 012746 007406  
2936 007370 004737 007162  
2937 007374 062716 000005  
2938 007400 012666 000002  
2939 007404 000207  
2940 007406 000000 000000  
2941  
2942  
2943  
2944  
2945  
2946  
2947  
2948  
2949  
2950  
2951 007412 010046  
2952 007414 016600 000004  
2953 007420 105710  
2954 007422 001403  
2955 007424 122720 000060  
2956 007430 001773  
2957 007432 005300  
2958 007434 010067 000002  
2959 007440 104400  
2960 007442 000000  
2961 007444 012600  
2962 007446 012616  
2963 007450 000207  
2964  
2965  
2966  
2967  
2968  
2969  
2970  
2971

```
144 ;:1.0E02
0
12 ;:1.0E01
0
1 ;:1.0E00
0
$DECVL: .BLKB 12. ;:RESERVE STORAGE FOR ASCIZ STRING
.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
;:*****
;:THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
;:UNSIGNED DECIMAL ASCII NUMBER.
;:CALL
;: MOV NUMBER, -(SP) ;:PUT BINARY NUMBER ON THE STACK
;: JSR PC, @#$SB2D ;:CALL
;: RETURN ;:ADDRESS OF THE 1ST ASCII CHAR. IS ON THE STACK
$SB2D: MOV 2(SP), 1$ ;:SAVE BINARY NUMBER
MOV #1$, -(SP) ;:SET POINTER
JSR PC, @#$DB2D ;:CALL DOUBLE LENGTH CONVERT
ADD #5, (SP) ;:ONLY ALLOW FIVE CHARACTERS
MOV (SP)+, 2(SP) ;:PICKUP POINTER
RTS PC ;:RETURN
1$: .WORD 0,0
.SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
;:*****
;:THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE
;:LEADING NUMBERS.
;:CALL
;: MOV #NUMADR, -(SP) ;:FIRST ADDRESS OF ASCII STRING
;: JSR PC, @#$SUPRS
$SUPRS: MOV R0, -(SP) ;:SAVE R0
MOV 4(SP), R0 ;:PICKUP THE POINTER
1$: TSTB (R0) ;:TERMINATEOR?
BEQ 2$ ;:BR IF YES
CMPB #'0, (R0)+ ;:IS THIS AN ASCII "0" ?
BEQ 1$ ;:BR IF YES
2$: DEC R0 ;:BACKUP BY "1"
MOV R0, 3$ ;:SAVE FOR TYPING
TYPE ;:GO TYPE
3$: .WORD 0 ;:ASCII POINTER GOES HERE
MOV (SP)+, R0 ;:RESTORE R0
MOV (SP)+, (SP) ;:RESTORE THE STACK
RTS PC ;:RETURN
.SBTTL READ AN OCTAL NUMBER FROM THE TTY
;:*****
;:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;:CHANGE IT TO BINARY.
;:CALL:
;: RDOCT ;:READ AN OCTAL NUMBER
;: RETURN HERE ;:LOW ORDER BITS ARE ON TOP OF THE STACK
```

```

2972                                     ;*                                     ;;HIGH ORDER BITS ARE IN $HIOCT
2973
2974 007452 011646                                     $RDOCT: MOV      (SP),-(SP)           ;; PROVIDE SPACE FOR THE
2975 007454 016666 000004 000002                   MOV      4(SP),2(SP)           ;; INPUT NUMBER
2976 007462 010046                                     MOV      RO,-(SP)             ;; PUSH RO ON STACK
2977 007464 010146                                     MOV      R1,-(SP)            ;; PUSH R1 ON STACK
2978 007466 010246                                     MOV      R2,-(SP)            ;; PUSH R2 ON STACK
2979 007470 104407                                     1$: RDLIN                                     ;; READ AN ASCII LINE
2980 007472 012600                                     MOV      (SP)+,RO            ;; GET ADDRESS OF 1ST CHARACTER
2981 007474 005001                                     CLR      R1                   ;; CLEAR DATA WORD
2982 007476 005002                                     CLR      R2
2983 007500 112046                                     2$: MOVB      (RO)+,-(SP)       ;; PICKUP THIS CHARACTER
2984 007502 001412                                     BEQ      3$                   ;; IF ZERO GET OUT
2985 007504 006301                                     ASL      R1                   ;; *2
2986 007506 006102                                     ROL      R2
2987 007510 006301                                     ASL      R1                   ;; *4
2988 007512 006102                                     ROL      R2
2989 007514 006301                                     ASL      R1                   ;; *8
2990 007516 006102                                     ROL      R2
2991 007520 042716 177770                             BIC      #1C7,(SP)           ;; STRIP THE ASCII JUNK
2992 007524 062601                                     ADD      (SP)+,R1            ;; ADD IN THIS DIGIT
2993 007526 000764                                     BR       2$                   ;; LOOP
2994 007530 005726                                     3$: TST      (SP)+            ;; CLEAN TERMINATOR FROM STACK
2995 007532 010166 000012                             MOV      R1,12(SP)           ;; SAVE THE RESULT
2996 007536 010267 000010                             MOV      R2,$HIOCT
2997 007542 012602                                     MOV      (SP)+,R2           ;; POP STACK INTO R2
2998 007544 012601                                     MOV      (SP)+,R1           ;; POP STACK INTO R1
2999 007546 012600                                     MOV      (SP)+,RO           ;; POP STACK INTO RO
3000 007550 000002                                     RTI                          ;; RETURN
3001 007552 000000                                     $HIOCT: .WORD 0              ;; HIGH ORDER BITS GO HERE
    
```

```

3002 ;:*****
3003 .SBTTL ROUTINE TO TYPE DRIVE
3004
3005 ;CALL
3006 ; JSR PC,@TPDRV
3007 ;
3008
3009
3010 007554 104400 013302 TPDRV: TYPE MSG15 ;" DRIVE "
3011 007560 113746 001613 MOVB @#PARMBK+1,-(SP) ;PICKUP THE DRIVE NUMBER
3012 007564 042716 177776 BIC #1C1,(SP) ;STRIP ANY JUNK
3013 007570 062716 000101 ADD #'A,(SP) ;MAKE IT ASCIZ
3014 007574 012667 000006 MOV (SP)+,1$ ;SAVE IT FOR TYPEOUT
3015 007600 104400 TYPE ;TYPE IT
3016 007602 007606 1$
3017 007604 000207 RTS PC
3018 007606 000000 1$: .WORD 0

```

```

3019 ;:*****
3020
3021 .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
3022
3023 ;CALL
3024 ;              JSR      PC, @ASKDRV
3025 ;              RETURN
3026 ;NOTE: R0 AND R1 ARE DESTROYED
3027 007610 104400 013312 ASKDRV: TYPE      MSGDRV          ;<CRLF>"DRIVE(S)? "
3028 007614 005067 171432 CLR          DRVKEY
3029 007620 104407 RDLIN
3030 007622 012600 MOV          (SP)+, R0          ;GO GET A DRIVE
3031 007624 105710 TSTB        @R0                ;SETUP TO CHECK FOR VALID DRIVE(S)
3032 007626 001425 BEQ          NOTLGL           ;WAS A DRIVE SELECTED?
3033 007630 012701 001252 MOV          #DRVKEY, R1       ;BR IF NO
3034 007634 122710 000101 LOOP:  CMPB        #'A, @R0          ;WAS DRIVE "A" SELECTED?
3035 007640 001002 BNE          NOTA            ;BR IF NO
3036 007642 112021 MOVB        (R0)+, (R1)+       ;SET KEY FOR DRIVE "A"
3037 007644 000411 BR          NEXT
3038 007646 122710 000102 NOTA:  CMPB        #'B, @R0          ;WAS DRIVE "B" SELECTED?
3039 007652 001002 BNE          NOTB            ;BR IF NO
3040 007654 112021 MOVB        (R0)+, (R1)+       ;SET KEY FOR DRIVE "B"
3041 007656 000404 BR          NEXT
3042 007660 122710 000054 NOTB:  CMPB        #54, @R0        ;WAS A COMMA TYPED?
3043 007664 001006 BNE          NOTLGL           ;BR IF NO
3044 007666 105720 TSTB        (R0)+            ;DUMP THE COMMA
3045 007670 105710 NEXT:  TSTB        @R0                ;TERMINATOR?
3046 007672 001406 BEQ          EXIT            ;BR IF YES
3047 007674 022701 001254 CMP          #DRVKEY+2, R1     ;TWO DRIVES SELECTED?
3048 007700 101355 BHI          LOOP            ;BR IF NO
3049 007702 104400 001202 NOTLGL: TYPE      $QUES          ;ILLEGAL INPUT DETECTED
3050 007706 000740 BR          ASKDRV           ;GO TRY AGAIN
3051 007710 005767 171336 EXIT:  TST          DRVKEY       ;ANY DRIVE SELECTED?
3052 007714 001772 BEQ          NOTLGL           ;BR IF NO
3053 007716 000207 RTS          PC
3054

```

```

3055      ;:*****
3056      ;SBTTL ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
3057      ;CALL
3058      ;JSR   PC,@#ASKADR
3059
3060      ASKADR: MOV     RO,-(SP)           ;SAVE RO
3061      1$:   TYPE    ,MSGASK           ;"TACS?"
3062      RDOCT  ;GET VALUE
3063      MOV     (SP)+,RO              ;PICK UP THE OCTAL NUMBER
3064      BEQ    3$                    ;IF "0" USE OLD VALUES
3065      CMP    RO,#160000            ;MAKE SURE IT IS A BUS ADDRESS
3066      BLO   1$
3067      MOV    RO,@#TACSL            ;SAVE THE TACS
3068      ADD    #2,RO                 ;STEP TO TADB ADDRESS
3069      MOV    RO,@#TADBL            ;AND SAVE IT
3070      3$:   TYPE    ,MSGVEC           ;"VECTOR?"
3071      RDOCT
3072      MOV    (SP)+,RO
3073      BEQ    5$
3074      CMP    RO,#1000              ;MAKE SURE ADDRESS IS IN VECTOR AREA
3075      BHIS  3$
3076      MOV    RO,@#TAVEC            ;SAVE AS VECTOR ADDRESS
3077      ADD    #2,RO
3078      MOV    RO,@#TAVEC+2
3079      5$:   TYPE    ,MSGPRI           ;ASK FOR PRIORITY
3080      RDOCT
3081      MOV    (SP)+,RO
3082      BEQ    6$                    ;IF "0" USE OLD VALUE
3083      CMP    RO,#7                 ;MAKE SURE ITS VALID
3084      BHI   5$
3085      SWAB  RO                     ;PUT INTO HIGH BYTE
3086      ASR   RO                     ;AND SHIFT
3087      ASR   RO                     ;INTO PROPER
3088      ASR   RO                     ;POSITION
3089      BIC   #1C<340>,RO            ;SAVE ONLY PRIORITY BITS
3090      MOV    RO,@#TAPRIO            ;STORE IT AWAY
3091      6$:   TYPE    ,MTACS            ;TACS="
3092      MOV    TACSL,-(SP)           ;;SAVE TACSL FOR TYPEOUT
3093      TYPOC  ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3094      TYPE   ,MTADB                ;"TADB="
3095      MOV    TADBL,-(SP)           ;;SAVE TADBL FOR TYPEOUT
3096      TYPOC  ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3097      TYPE   ,MTAVEC              ;"VECTOR="
3098      MOV    TAVEC,-(SP)          ;;SAVE TAVEC FOR TYPEOUT
3099      TYPOC  ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3100      TYPE   ,MTAPRI              ;"PRIORITY="
3101      MOV    TAPRIO,-(SP)        ;;SAVE TAPRIO FOR TYPEOUT
3102      TYPOC  ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3103      TYPE   ,MSGOK               ;"OK?"
3104      RDCHR  ;GO READ ONE CHARACTER
3105      MOV    (SP)+,RO              ;GET IT
3106      CMP    #15,RO               ;IS IT "CR"?
3107      BEQ    7$                    ;BRANCH IF YES
3108      CMP    #'Y',RO              ;IS IT "Y"?
3109      BEQ    7$                    ;IT WAS
3110      TYPE   ,SQUES               ;TYPE "?"

```



H06

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-B MACY11 27(732) 25-SEP-76 11:17 PAGE 73  
DZTAEB.P11 ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY

3111 010152 000663  
3112 010154 104400  
3113 010160 012600  
3114 010162 000207

013432

7S:

BR 1S  
TYPE ,MYES  
MOV (SP)+,RO  
RTS PC

;AND LET HIM CORRECT THEM  
;TYPE OUT "YES"  
;RESTORE RO  
;AND RETURN

```

3115 ;*****
3116 ;
3117 ;.SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
3118 ;
3119 ;CALL:
3120 ;           MOV      #DRVKEY,RO
3121 ;           JSR      PC,@#EXAM          ;R1 IS DESTROYED
3122 ;           NORMAL RETURN
3123 ;           ERROR RETURN
3124 ;
3125 010164 013701 001240 EXAM: MOV      @#TACSL,R1          ;PICKUP THE "CONTROL & STATUS" REG. ADR.
3126 010170 005011      CLR      (R1)          ;DRIVE="A" FUNCTION="WFG"
3127 010172 122710 000101      CMPB   #'A,(R0)          ;EXAMINE DRIVE "A"?
3128 010176 001402      BEQ     1$          ;BR IF YES
3129 010200 052711 000400      BIS    #UNIT,(R1)          ;SELECT DRIVE "B"
3130 010204 032711 000040 1$: BIT    #READY,(R1)          ;WAIT ON READY
3131 010210 001775      BEQ     1$
3132 010212 005711      TST    (R1)          ;ANY ERROR?
3133 010214 100024      BPL    4$          ;BR IF NO
3134 010216 032711 001000      BIT    #OFFLINE,(R1)          ;ERROR DUE TO "OFF LINE"?
3135 010222 001017      BNE    3$          ;BR IF YES
3136 010224 032711 010000      BIT    #WRTLOCK,(R1)          ;ERROR DUE TO "WRITE LOCK"?
3137 010230 001411      BEQ     2$          ;BR IF NO
3138 010232 122777 000201 170700      CMPB   #BIT07!BIT00,@SWR          ;"READONLY" SELECTED? (RD1PAS)
3139 010240 001412      BEQ     4$          ;BR IF YES
3140 010242 122777 000203 170670      CMPB   #BIT07!BIT01!BIT00,@SWR          ;(RD2PAS)?
3141 010250 001406      BEQ     4$          ;BR IF YES
3142 010252 000403      BR     3$          ;TAKE THE ERROR EXIT
3143 010254 032711 020000 2$: BIT    #LEADER,(R1)          ;ERROR DUE TO "CLEAR LEADER"?
3144 010260 001002      BNE    4$          ;BR IF YES
3145 010262 062716 000002 3$: ADD    #2,(SP)          ;TAKE ERROR RETURN
3146 010266 000207 4$: RTS    PC          ;RETURN

```

.SBTTL TYPE ROUTINE

3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154  
3155  
3156  
3157  
3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166  
3167  
3168  
3169  
3170  
3171  
3172  
3173  
3174  
3175  
3176  
3177  
3178  
3179  
3180  
3181  
3182  
3183  
3184  
3185  
3186  
3187  
3188  
3189  
3190  
3191  
3192  
3193  
3194  
3195  
3196  
3197  
3198  
3199  
3200  
3201  
3202

```
*****
*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
*
```

```
$TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV RD,-(SP) ;; SAVE RD
MOV 22(SP),RD ;; GET ADDRESS OF ASCIZ STRING
2$: MOVB (RD)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,RD ;; RESTORE RD
3$: ADD #2,(SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE ;; TYPE A CR AND LF
$CRLF
CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
BR 2$ ;; GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;; IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;; GO TYPE A NULL
DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
BR 7$ ;; LOOP
```

;HORIZONTAL TAB PROCESSOR

```
8$: MOVB #'(SP) ;; REPLACE TAB WITH SPACE
9$: JSR PC,$TYPEC ;; TYPE A SPACE
BITB #7,$CHARCNT ;; BRANCH IF NOT AT
BNE 9$ ;; TAB STOP
TST (SP)+ ;; POP SPACE OFF STACK
```

```

3203 010436 000724          BR      2$          ;;GET NEXT CHARACTER
3204 010440 105777 170504 $TYPEC: TSTB  @STPS          ;;WAIT UNTIL PRINTER IS READY
3205 010444 100375          BPL      $TYPEC
3206 010446 116677 000002 170476 MOVB  2(SP),@STPB          ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3207 010454 122766 000015 000002 CMPB  #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
3208 010462 001003          BNE      1$          ;;BRANCH IF NO
3209 010464 105067 000014 CLR   $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
3210 010470 000406          BR      $TYPEX          ;;EXIT
3211 010472 122766 000012 000002 1$:  CMPB  #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
3212 010500 001402          BEQ      $TYPEX          ;;BRANCH IF YES
3213 010502 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
3214 010504 000000          $CHARCNT: WORD 0          ;;CHARACTER COUNT STORAGE
3215 010506 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.
3227 010510 022767 000176 170422 $CKSWR: CMP      #SWREG,SWR          ;;IS THE SOFT-SWR SELECTED?
3228 010516 001074          BNE      15$          ;;BRANCH IF NO
3229 010520 105777 170420 TSTB  @STKS          ;;CHAR THERE?
3230 010524 100071          BPL      15$          ;;IF NO, DON'T WAIT AROUND
3231 010526 117746 170414 MOVB  @STKB,-(SP)          ;;SAVE THE CHAR
3232 010532 042716 177600 BIC   #1C177,(SP)          ;;STRIP-OFF THE ASCII
3233 010536 022726 000007 CMP   #7,(SP)+          ;;IS IT A CONTROL G?
3234 010542 001062          BNE      15$          ;;NO, RETURN TO USER
3235 010544 126727 170364 000001 CMPB  $AUTOB,#1          ;;ARE WE RUNNING IN AUTO-MODE?
3236 010552 001456          BEQ      15$          ;;BRANCH IF YES
3237
3238 010554 104400 011235          TYPE   , $CNTLG          ;;ECHO THE CONTROL-G (↑G)
3239 010560 104400 011242          $GTSWR: TYPE   , $MSWR          ;;TYPE CURRENT CONTENTS
3240 010564 016746 167406 MOV   $WREG,-(SP)          ;;SAVE SWREG FOR TYPEOUT
3241 010570 104401          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3242 010572 104400 011253          TYPE   , $MNEW          ;;PROMPT FOR NEW SWR
3243 010576 005046          19$:  CLR   -(SP)          ;;CLEAR COUNTER
3244 010600 005046          CLR   -(SP)          ;;THE NEW SWR
3245 010602 105777 170336          7$:  TSTB  @STKS          ;;CHAR THERE?
3246 010606 100375          BPL      7$          ;;IF NOT TRY AGAIN
3247
3248 010610 117746 170332          MOVB  @STKB,-(SP)          ;;PICK UP CHAR
3249 010614 042716 177600 BIC   #1C177,(SP)          ;;MAKE IT 7-BIT ASCII
3250
3251
3252
3253 010620 021627 000025          9$:  CMP   (SP),#25          ;;IS IT A CONTROL-U?
3254 010624 001005          BNE      10$          ;;BRANCH IF NOT
3255 010626 104400 011230          TYPE   , $CNTLU          ;;YES, ECHO CONTROL-U (↑U)
3256 010632 062706 000006          20$:  ADD   #6,SP          ;;IGNORE PREVIOUS INPUT
3257 010636 000757          BR      19$          ;;LET'S TRY IT AGAIN
3258

```

3259											
3260	010640	021627	000015		10\$:	CMP	(SP), #15		:: IS IT A <CR>?		
3261	010644	001022				BNE	16\$		:: BRANCH IF NO		
3262	010646	005766	000004			TST	4(SP)		:: YES, IS IT THE FIRST CHAR?		
3263	010652	001403				BEQ	11\$		:: BRANCH IF YES		
3264	010654	016677	000002	170256		MOV	2(SP), @SWR		:: SAVE NEW SWR		
3265	010662	062706	000006		11\$:	ADD	#6, SP		:: CLEAR UP STACK		
3266	010666	104400	001203		14\$:	TYPE	\$CRLF		:: ECHO <CR> AND <LF>		
3267	010672	126727	170237	000001		CMPB	\$INTAG, #1		:: RE-ENABLE TTY KBD INTERRUPTS?		
3268	010700	001003				BNE	15\$		:: BRANCH IF NOT		
3269	010702	012777	000100	170234		MOV	#100, @STKS		:: RE-ENABLE TTY KBD INTERRUPTS		
3270	010710	000002			15\$:	RTI			:: RETURN		
3271	010712	004767	177522		16\$:	JSR	PC, \$TYPEC		:: ECHO CHAR		
3272	010716	021627	000060			CMP	(SP), #60		:: CHAR < 0?		
3273	010722	002420				BLT	18\$		:: BRANCH IF YES		
3274	010724	021627	000067			CMP	(SP), #67		:: CHAR > 7?		
3275	010730	003015				BGT	18\$		:: BRANCH IF YES		
3276	010732	042726	000060			BIC	#60, (SP)+		:: STRIP-OFF ASCII		
3277	010736	005766	000002			TST	2(SP)		:: IS THIS THE FIRST CHAR		
3278	010742	001403				BEQ	17\$		:: BRANCH IF YES		
3279	010744	006316				ASL	(SP)		:: NO, SHIFT PRESENT		
3280	010746	006316				ASL	(SP)		:: CHAR OVER TO MAKE		
3281	010750	006316				ASL	(SP)		:: ROOM FOR NEW ONE.		
3282	010752	005266	000002		17\$:	INC	2(SP)		:: KEEP COUNT OF CHAR		
3283	010756	056616	177776			BIS	-2(SP), (SP)		:: SET IN NEW CHAR		
3284	010762	000707				BR	7\$		:: GET THE NEXT ONE		
3285	010764	104400	001202		18\$:	TYPE	\$QUES		:: TYPE ?<CR><LF>		
3286	010770	000720				BR	20\$		:: SIMULATE CONTROL-U		
3287					.DSABL	LSB					

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

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

::\*CALL:

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

3297											
3298											
3299	010772	011646			\$RDCHR:	MOV	(SP), -(SP)		:: PUSH DOWN THE PC		
3300	010774	016666	000004	000002		MOV	4(SP), 2(SP)		:: SAVE THE PS		
3301	011002	105777	170136		1\$:	TSTB	@STKS		:: WAIT FOR		
3302	011006	100375				BPL	1\$		:: A CHARACTER		
3303	011010	117766	170132	000004		MOVB	@STKB, 4(SP)		:: READ THE TTY		
3304	011016	042766	177600	000004		BIC	#1C<177>, 4(SP)		:: GET RID OF JUNK IF ANY		
3305	011024	026627	000004	000023		CMP	4(SP), #23		:: IS IT A CONTROL-S?		
3306	011032	001013				BNE	3\$		:: BRANCH IF NO		
3307	011034	105777	170104		2\$:	TSTB	@STKS		:: WAIT FOR A CHARACTER		
3308	011040	100375				BPL	2\$		:: LOOP UNTIL ITS THERE		
3309	011042	117746	170100			MOVB	@STKB, -(SP)		:: GET CHARACTER		
3310	011046	042716	177600			BIC	#1C177, (SP)		:: MAKE IT 7-BIT ASCII		
3311	011052	022627	000021			CMP	(SP)+, #21		:: IS IT A CONTROL-Q?		
3312	011056	001366				BNE	2\$		:: IF NOT DISCARD IT		
3313	011060	000750				BR	1\$		:: YES, RESUME		
3314	011062	026627	000004	000140	3\$:	CMP	4(SP), #140		:: IS IT UPPER CASE?		
3315	011070	002407				BLT	4\$		:: BRANCH IF YES		

```

3315 011072 026627 000004 000175      CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
3316 011100 003003                    BGT      4$             ;; BRANCH IF YES
3317 011102 042766 000040 000004      BIC      #40,4(SP)      ;; MAKE IT UPPER CASE
3318 011110 000002                    RTI                      ;; GO BACK TO USER
3319                                     ;; *****
3320                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3321                                     ;; *CALL:
3322                                     ;; *      RDLIN              ;; INPUT A STRING FROM THE TTY
3323                                     ;; *      RETURN HERE      ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3324                                     ;; *                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
3325
3326 011112 010346                    $RDLIN: MOV      R3,-(SP)      ;; SAVE R3
3327 011114 012703 011220             1$:      MOV      #$TTYIN,R3  ;; GET ADDRESS
3328 011120 022703 011230             2$:      CMP      #$TTYIN+8.,R3  ;; BUFFER FULL?
3329 011124 101405                    BLOS     4$             ;; BR IF YES
3330 011126 104406                    RDCHR    ;; GO READ ONE CHARACTER FROM THE TTY
3331 011130 112613                    MOV      (SP)+,(R3)     ;; GET CHARACTER
3332 011132 122713 000177             10$:    CMP      #177,(R3)     ;; IS IT A RUBOUT
3333 011136 001003                    BNE     3$             ;; SKIP IF NOT
3334 011140 104400 001202             4$:      TYPE     $QUES     ;; TYPE A '?'
3335 011144 000763                    BR       1$            ;; CLEAR THE BUFFER AND LOOP
3336 011146 111367 000044             3$:      MOV      (R3),9$    ;; ECHO THE CHARACTER
3337 011152 104400 011216                    TYPE     9$
3338 011156 122723 000015                    CMP      #15,(R3)+     ;; CHECK FOR RETURN
3339 011162 001356                    BNE     2$            ;; LOOP IF NOT RETURN
3340 011164 105063 177777                    CLRB    -1(R3)        ;; CLEAR RETURN (THE 15)
3341 011170 104400 001204                    TYPE     $LF          ;; TYPE A LINE FEED
3342 011174 012603                    MOV      (SP)+,R3      ;; RESTORE R3
3343 011176 011646                    MOV      (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3344 011200 016666 000004 000002      MOV      4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
3345 011206 012766 011220 000004      MOV      #$TTYIN,4(SP)
3346 011214 000002                    RTI
3347 011216 000                    9$:      .BYTE    0           ;; RETURN
3348 011217 000                    .BYTE    0           ;; STORAGE FOR ASCII CHAR. TO TYPE
3349 011220 000010                    $TTYIN: .BLKB    8.   ;; TERMINATOR
3350 011230 052536 005015 000          $CNTLU: .ASCIZ  /↑U/<15><12>  ;; RESERVE 8 BYTES FOR TTY INPUT
3351 011235 0136 006507 000012        $CNTLG: .ASCIZ  /↑G/<15><12>  ;; CONTROL "U"
3352 011242 005015 053523 020122        $MSWR:  .ASCIZ  <15><12>/SWR = /  ;; CONTROL "G"
3353 011250 020075 000
3354 011253 040 047040 053505        $MNEW:  .ASCIZ  / NEW = /
3355 011260 036440 000040

```

3356  
3357  
3358  
3359  
3360  
3361  
3362  
3363  
3364  
3365  
3366  
3367  
3368  
3369  
3370  
3371  
3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383  
3384  
3385  
3386  
3387  
3388  
3389  
3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    N              ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    N              ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    N              ;;CALL FOR TYPEOUT
3381 011264 017546 000000      $TYPOS: MOV      2(SP),-(SP)      ;; PICKUP THE MODE
3382 011270 116667 000001 000211  MOVB     1(SP),$OFILL      ;; LOAD ZERO FILL SWITCH
3383 011276 112667 000207      MOVB     (SP)+,$SOMODE+1    ;; NUMBER OF DIGITS TO TYPE
3384 011302 062716 000002      ADD      #2,(SP)          ;; ADJUST RETURN ADDRESS
3385 011306 000406      BR      $TYPON
3386 011310 112767 000001 000171  $TYPOC: MOVB     #1,$OFILL      ;; SET THE ZERO FILL SWITCH
3387 011316 112767 000006 000165  MOVB     #6,$SOMODE+1      ;; SET FOR SIX(6) DIGITS
3388 011324 112767 000005 000154  $TYPON: MOVB     #5,$OCNT      ;; SET THE ITERATION COUNT
3389 011332 010346      MOV      R3,-(SP)        ;; SAVE R3
3390 011334 010446      MOV      R4,-(SP)        ;; SAVE R4
3391 011336 010546      MOV      R5,-(SP)        ;; SAVE R5
3392 011340 116704 000145      MOVB     $SOMODE+1,R4     ;; GET THE NUMBER OF DIGITS TO TYPE
3393 011344 005404      NEG      R4
3394 011346 062704 000006      ADD      #6,R4           ;; SUBTRACT IT FOR MAX. ALLOWED
3395 011352 110467 000132      MOVB     R4,$SOMODE       ;; SAVE IT FOR USE
3396 011356 116704 000125      MOVB     $OFILL,R4        ;; GET THE ZERO FILL SWITCH
3397 011362 016605 000012      MOV      12(SP),R5       ;; PICKUP THE INPUT NUMBER
3398 011366 005003      CLR      R3              ;; CLEAR THE OUTPUT WORD
3399 011370 006105      1$:     ROL      R5        ;; ROTATE MSB INTO "C"
3400 011372 000404      BR      3$              ;; GO DO MSB
3401 011374 006105      2$:     ROL      R5        ;; FORM THIS DIGIT
3402 011376 006105      ROL      R5
3403 011400 006105      ROL      R5
3404 011402 010503      MOV      R5,R3
3405 011404 006103      3$:     ROL      R3        ;; GET LSB OF THIS DIGIT
3406 011406 105367 000076      DECB     $SOMODE         ;; TYPE THIS DIGIT?
3407 011412 100016      BPL      7$              ;; BR IF NO
3408 011414 042703 177770      BIC      #177770,R3      ;; GET RID OF JUNK
3409 011420 001002      BNE      4$              ;; TEST FOR 0
3410 011422 005704      TST     R4              ;; SUPPRESS THIS 0?
3411 011424 001403      BEQ     5$              ;; BR IF YES
    
```

3412	011426	005204		4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
3413	011430	052703	000060		BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
3414	011434	052703	000040	5\$:	BIS	#',R3	:: MAKE ASCII IF NOT ALREADY
3415	011440	110367	000040		MOVB	R3,B\$	:: SAVE FOR TYPING
3416	011444	104400	011504		TYPE	B\$	:: GO TYPE THIS DIGIT
3417	011450	105367	000032	7\$:	DECB	\$OCNT	:: COUNT BY 1
3418	011454	003347			BGT	2\$	:: BR IF MORE TO DO
3419	011456	002402			BLT	6\$	:: BR IF DONE
3420	011460	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
3421	011462	000744			BR	2\$	:: GO DO THE LAST DIGIT
3422	011464	012605		6\$:	MOV	(SP)+,R5	:: RESTORE R5
3423	011466	012604			MOV	(SP)+,R4	:: RESTORE R4
3424	011470	012603			MOV	(SP)+,R3	:: RESTORE R3
3425	011472	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
3426	011500	012616			MOV	(SP)+,(SP)	
3427	011502	000002			RTI		:: RETURN
3428	011504	000		8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
3429	011505	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
3430	011506	000		\$OCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
3431	011507	000		\$DFILL:	.BYTE	0	:: ZERO FILL SWITCH
3432	011510	000000		\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE



3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469

.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.

011512 010046  
011514 016600 000002  
011520 005740  
011522 111000  
011524 006300  
011526 016000 011534  
011532 000200

\$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

.SBTTL TRAP TABLE

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

011534  
011534 010270  
011536 011310  
011540 011264  
011542 011324  
011544 010560  
011546 010510  
011550 010772  
011552 011112  
011554 007452  
011556 006616  
011560 006654

ROUTINE  
-----  
\$TRPAD: \$TYPE ;; CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE  
\$TYPOC ;; CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)  
\$TYPOS ;; CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
\$TYPON ;; CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)  
\$GTSWR ;; CALL=GTSWR TRAP+4(104404) GET SOFT-SWR SETTING  
\$CKSWR ;; CALL=CKSWR TRAP+5(104405) TEST FOR CHANGE IN SOFT-SWR  
\$RDCHR ;; CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE  
\$RDLIN ;; CALL=RDLIN TRAP+7(104407) TTY TYPEIN STRING ROUTINE  
\$RDOCT ;; CALL=RDOCT TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY  
\$SAVREG ;; CALL=SAVREG TRAP+11(104411) SAVE R0-R5 ROUTINE  
\$RESREG ;; CALL=RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE

.SBTTL POWER DOWN AND UP ROUTINES

3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510  
3511  
3512  
3513  
3514

011562 012737 011726 000024  
011570 012737 000340 000026  
011576 010046  
011600 010146  
011602 010246  
011604 010346  
011606 010446  
011610 010546  
011612 017746 167322  
011616 010667 000110  
011622 012737 011634 000024  
011630 000000  
011632 000776  
  
011634 012737 011726 000024  
011642 016706 000064  
011646 005067 000060  
011652 005267 000054  
011656 001375  
011660 012677 167254  
011664 012605  
011666 012604  
011670 012603  
011672 012602  
011674 012601  
011676 012600  
011700 012737 011562 000024  
011706 012737 000340 000026  
011714 104400  
011716 011734  
011720 012716  
011722 004406  
011724 000002  
011726 000000  
011730 000776  
011732 000000  
011734 005015 047520 042527  
011742 000122

```
*****  
:POWER DOWN ROUTINE  
$PWRDN: MOV $SILLUP, @PWRVEC ;; SET FOR FAST UP  
MOV #340, @PWRVEC+2 ;; PRIO:7  
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 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 (PC)+, (SP) ;; POWER FAIL MESSAGE POINTER  
$PWRAD: .WORD $EOP ;; RESTART AT $EOP  
RTI ;; RESTART ADDRESS  
$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
```

```

3515      ;:*****
3516
3517      ;DATA POINTERS
3518
3519 011744 001116 001162 001166 DT1:  .WORD  $ERRPC,$REG0,$REG2,BYTNUM,$GDDAT,$BDDAT,$GDADR,$BDADR,0
3520 011752 001230 001124 001126
3521 011760 001120 001122 000000
3522
3523 011766 001116 001162 001166 DT2:  .WORD  $ERRPC,$REG0,$REG2,$TMPO,$TMP2,0
3524 011774 001172 001176 000000
3525
3526 012002 001116 001162 001166 DT101: .WORD  $ERRPC,$REG0,$REG2,$TMPO,0
3527 012010 001172 000000
3528
3529 012014 001116 001240 000000 DT201: .WORD  $ERRPC,TACSL,0
3530
3531 012022 001116 000000      DT202: .WORD  $ERRPC,0
3532
3533      ;DATA FORMATS
3534      000000      ON=0      ;OCTAL NUMBER
3535      000001      DN=1      ;DECIMAL NUMBER
3536
3537 012026      000      DF1:  .BYTE  ON      ;$ERRPC
3538 012027      001      .BYTE  DN      ;$REG0
3539 012030      001      .BYTE  DN      ;$REG2
3540 012031      001      .BYTE  DN      ;BYTNUM
3541 012032      000      .BYTE  ON      ;$GDDAT
3542 012033      000      .BYTE  ON      ;$BDDAT
3543 012034      000      .BYTE  ON      ;$GDADR
3544 012035      000      .BYTE  ON      ;$BDADR
3545
3546 012036      000      DF2:  .BYTE  ON      ;$ERRPC
3547 012037      001      .BYTE  DN      ;$REG0
3548 012040      001      .BYTE  DN      ;$REG2
3549 012041      001      .BYTE  DN      ;$TMPO
3550 012042      001      .BYTE  DN      ;$TMP1
3551

```

```

3552                                     ;*****
3553                                     ;
3554                                     ;MESSAGES
3555
3556 012043      127 044522 042524 MXWFG: .ASCIZ /WRITE-FILE-GAP/
3557 012050      043055 046111 026505
3558 012056      040507 000120
3559 012062      051127 052111 000105 MXWRIT: .ASCIZ /WRITE/
3560 012070      042522 042101 000 MXREAD: .ASCIZ /READ/
3561 012075      102 041501 026513 MXBSFG: .ASCIZ /BACK-SPACE-FILE-GAP/
3562 012102      050123 041501 026505
3563 012110      044506 042514 043455
3564 012116      050101 000
3565 012121      102 041501 026513 MXBSBG: .ASCIZ /BACK-SPACE-BLK-GAP/
3566 012126      050123 041501 026505
3567 012134      046102 026513 040507
3568 012142      000120
3569 012144      050123 041501 026505 MXSFFG: .ASCIZ /SPACE-FWD-FILE-GAP/
3570 012152      053506 026504 044506
3571 012160      042514 043455 050101
3572 012166      000
3573 012167      123 040520 042503 MXSFBG: .ASCIZ /SPACE-FWD-BLK-GAP/
3574 012174      043055 042127 041055
3575 012202      045514 043455 050101
3576 012210      000
3577 012211      122 053505 047111 MXRWND: .ASCIZ /REWIND/
3578 012216      000104
3579 012220      040504 040524 042440 EM1: .ASCIZ "DATA ERROR"
3580 012226      051122 051117 000
3581 012233      123 047131 020103 EM2: .ASCIZ "SYNC ERROR"
3582 012240      051105 047522 000122
3583 012246      051104 053111 020105 EM101: .ASCIZ "DRIVE IS OFF-LINE"
3584 012254      051511 047440 043106
3585 012262      046055 047111 000105
3586 012270      051104 053111 020105 EM102: .ASCIZ "DRIVE IS WRITE-LOCK"
3587 012276      051511 053440 044522
3588 012304      042524 046055 041517
3589 012312      000113
3590 012314      046103 040505 020122 EM103: .ASCIZ "CLEAR LEADER ERROR"
3591 012322      042514 042101 051105
3592 012330      042440 051122 051117
3593 012336      000
3594 012337      106 046111 020105 EM104: .ASCIZ "FILE GAP ERROR"
3595 012344      040507 020120 051105
3596 012352      047522 000122
3597 012356      044524 044515 043516 EM105: .ASCIZ "TIMING ERROR"
3598 012364      042440 051122 051117
3599 012372      000
3600 012373      102 047514 045503 EM106: .ASCIZ "BLOCK CHECK ERROR"
3601 012400      041440 042510 045503
3602 012406      042440 051122 051117
3603 012414      000
3604 012415      125 045516 047516 EM107: .ASCIZ "UNKNOWN INTERRUPT"
3605 012422      047127 044440 052116
3606 012430      051105 052522 052120
3607 012436      000

```



3664	013103	200	054502	042524	MSG5:	.ASCIZ	<CRLF>"BYTES WRITTEN="
3665	013110	020123	051127	052111			
3666	013116	042524	036516	000			
3667	013123	200	040524	042520	MSG6:	.ASCIZ	<CRLF>"TAPE PASSES="
3668	013130	050040	051501	042523			
3669	013136	036523	000				
3670	013141	200	044506	042514	MSG7:	.ASCIZ	<CRLF>"FILES/PASS="
3671	013146	027523	040520	051523			
3672	013154	000075					
3673	013156	041200	047514	045503	MSG8:	.ASCIZ	<CRLF>"BLOCKS/FILE="
3674	013164	027523	044506	042514			
3675	013172	000075					
3676	013174	040440	042116	000	MSG9:	.ASCIZ	" AND"
3677	013201	040	044527	046114	MSG10:	.ASCIZ	" WILL BE TESTED"<CRLF>
3678	013206	041040	020105	042524			
3679	013214	052123	042105	000200			
3680	013222	025052	020052	047506	MSG11:	.ASCIZ	"*** FORMAT ***"
3681	013230	046522	052101	025040			
3682	013236	025052	000				
3683	013241	052	025052	051040	MSG12:	.ASCIZ	"*** READ ***"
3684	013246	040505	020104	020040			
3685	013254	025052	000052				
3686	013260	025052	020052	051127	MSG13:	.ASCIZ	"*** WRITE ***"
3687	013266	052111	020105	025040			
3688	013274	025052	000				
3689	013277	040	000040		MSG14:	.ASCIZ	" "
3690	013302	042040	044522	042526	MSG15:	.ASCIZ	" DRIVE "
3691	013310	000040					
3692	013312	042200	044522	042526	MSGDRV:	.ASCIZ	<CRLF>"DRIVE(S)? "
3693	013320	051450	037451	000040			
3694	013326	052200	041501	037523	MSGASK:	.ASCIZ	<CRLF>"TACS?"
3695	013334	000					
3696	013335	200	042526	052103	MSGVEC:	.ASCIZ	<CRLF>"VECTOR?"
3697	013342	051117	000077				
3698	013346	050200	044522	051117	MSGPRI:	.ASCIZ	<CRLF>"PRIORITY?"
3699	013354	052111	037531	000			
3700	013361	200	040524	051503	MTACS:	.ASCIZ	<CRLF>"TACS="
3701	013366	000075					
3702	013370	052040	042101	036502	MTADB:	.ASCIZ	/ TADB=/
3703	013376	000					
3704	013377	040	053040	041505	MTAVEC:	.ASCIZ	" VECTOR="
3705	013404	047524	036522	000			
3706	013411	040	050040	044522	MTAPRI:	.ASCIZ	" PRIORITY="
3707	013416	051117	052111	036531			
3708	013424	000					
3709	013425	200	045517	000077	MSGOK:	.ASCIZ	<CRLF>"OK?"
3710	013432	042531	100123	000	MYES:	.ASCIZ	"YES"<CRLF>
3711		013440					.EVEN
3712							
3713							.SBTTL
3714							READ AND WRITE BUFFER
3715	013440	002004			BUFFER:	.BLKB	1028.
3716		015444			LASTADDRESS=		.
3717							
3718		000001					.END





















SGOTO	1161#	2452												
CALL	1161#	1797	1807	1821	1832	1844	1875	1947	1963	1974	2028	2075	2085	2099
COMMEN	1#	1130#												
DECBLK	1161#	2482												
DECFIL	1161#	2471												
ENDCOM	1#	1130#												
EOPCOD	1161#	2175												
ERROR	1024#	1589	1652	1857	1864	1986	1993	2463	2466	2493	2496	2499	2502	2508
ESCAPE	1#	1130#												
GETPRI	1#	1130#												
GETSWR	1#	1130#	1534#											
INCBK	1161#	1887	2013	2112										
INCFIL	1161#	1892	2019	2117										
JGOTO	1161#	1733												
MORETA	1161#	1227												
MULT	1#	1130#												
NEWTST	1#	1130#												
OUTDBL	1161#	2194	2199											
OUTSGL	1161#	2184	2189	2204	2216									
POP	1#	1130#	2571	2755	2831	2846	2997	3495	3496					
PUSH	1#	1130#	2554	2735	2819	2976	3476	3482						
REMARK	1161#	1794	1804	1816	1829	1840	1852	1859	1870	1884	1944	1954	1959	1970
	1988	2000	2010	2025	2072	2082	2094	2109	2176	2221				1981
REPORT	1#	1130#												
SCOPE	1025#	1851	1999	2108	2144	2162	2175							
SETBLK	1161#	1788	1895	1938	2022	2066	2120							
SETFIL	1161#	1791	1941	2069										
SETPRI	1#	1130#												
SETTRA	3449#	3458	3459	3460	3462	3464	3465	3466	3467	3468	3469			
SETUP	1#	1130#	1492											
SKIP	1#	1130#												
SLASH	1#	1130#	1263	1284	1306	1360	1472	1659	1743	1747	1794	1796	1804	1806
	1818	1829	1831	1840	1842	1852	1854	1859	1861	1870	1872	1884	1886	1899
	1946	1954	1956	1959	1961	1970	1972	1981	1983	1988	1990	2000	2002	2010
	2025	2027	2036	2072	2074	2082	2084	2094	2096	2109	2111	2124	2176	2178
	2223	2352	2372	2406	2415	2422								2221
SPACE	1130#													
STARS	1#	957	1130#	1174	1178	1226	1386	1408	1456	1474	1480	1546	1555	1563
	1598	1620	1661	1671	1699	1717	1745	1754	1782	1906	1932	2043	2059	2129
	2153	2248	2281	2320	2510	2533	2577	2596	2635	2684	2701	2715	2719	2764
	2860	2863	2925	2943	2966	3002	3019	3055	3115	3149	3219	3222	3290	3319
	3435	3472	3488	3515	3552									3358
SWRSU	1#	1130#	1511#											
TRMTRP	3449#													
TYPBIN	1#	1130#												
TYPDEC	1#	1130#												
TYPNAM	1#	1130#	1527											
TYPNUM	1#	1130#												
TYPOCS	1#	1130#												
TYPOCT	1#	1130#	2362	2374	2410	3092	3095	3098	3101	3240				
TYPTXT	1#	1130#												
SSCHRE	1176#	1215	1216	1217	1218									
SSCHTM	1176#	1219	1220	1221	1222									
SSESCA	1#	1130#												
SSNEWT	1#	1130#												
SSSET	3449#	3458	3459	3460	3462	3464	3465	3466	3467	3468	3469			



SSSKIP	1#	1130#	
.EQUAT	1#	947#	1020
.HEADE	1#	947#	
.KT11	1#		
.SETUP	1#	947#	1482
.SWRHI	1#	947#	1002
.SWRLO	1012#		
.SACT1	1#		
.SAPT8	1#		
.SAPTH	1#		
.SAPTY	1#		
.SASTA	1#		
.SCATC	1#	947#	1161
.SCMTA	1#	947#	1176
.SDB2D	1#	947#	2861
.SDB20	1#		
.SDIV	1#		
.SEOP	1#	947#	2151
.SERRO	1#	947#	2279
.SERRT	1#		
.SMULT	1#		
.SPOWE	1#	947#	3470
.SRAND	1#		
.SRDDE	1#		
.SRDOC	1#	947#	2964
.SREAD	1#	947#	3217
.SR2AZ	1#		
SSAVE	1#	947#	2717
SSB2D	1#	947#	2923
SSB20	1#		
SSCOP	1#	947#	2246
SSIZE	1#		
SSPAC	947#		
SSUPR	1#	947#	2941
STRAP	1#	947#	3433
STYPB	1#		
STYPD	1#		
STYPE	1#	947#	3147
STYPO	1#	947#	3356
S4OCA	1#		
.117D	1#		

ADC	2527	2592	2891												
ADD	1724	1738	1887	1892	2014	2019	2112	2117	2333	2347	2367	2385	2454	2526	2591
	2616	2621	2632	2672	2676	2681	2890	2892	2937	2992	3013	3068	3077	3145	3174
	3256	3265	3384	3394											
ASL	2328	2329	2330	2525	2590	2646	2985	2987	2989	3279	3280	3281	3445		
ASLB	1736	1737	2453												
ASR	1680	2793	3086	3087	3088										
ASRB	1643	1648													
BEQ	1532	1588	1594	1625	1640	1646	1657	1800	1811	1824	1836	1847	1878	1950	1966
	1977	2031	2078	2089	2103	2174	2225	2234	2295	2298	2359	2382	2409	2489	2627
	2795	2954	2956	2984	3032	3046	3052	3064	3073	3082	3107	3109	3128	3131	3137
	3139	3141	3177	3212	3236	3263	3278	3411							
BGE	2146	2695													
BGT	1697	1891	2005	2007	2018	2116	2168	2663	3275	3316	3418				
BHI	3048	3084													
BHIS	3075														
BIC	1681	2165	2429	2524	2589	2645	2991	3012	3089	3232	3249	3276	3303	3309	3317
	3408														
BICB	1719														
BIS	2828	2845	2894	3129	3283	3413	3414								
BISB	2803	2809													
BIT	2147	2224	2259	2297	2304	2434	2436	2438	2440	2442	2444	2794	2837	3130	3134
	3136	3143													
BITB	3200														
BLO	3066														
BLOS	3329														
BLT	1558	1561	1722	1874	2659	2712	2887	3191	3273	3314	3419				
BMI	1720	2825	2836												
BTT	1485	1487	1498	1518	1530	1536	1538	1592	1637	1655	1684	1727	2148	2229	2260
	2305	2315	2361	2435	2437	2439	2441	2443	2445	2470	2480	2504	2570	2618	2623
	2625	2653	2655	2661	2838	2898	3035	3039	3043	3135	3144	3171	3179	3187	3201
	3208	3228	3234	3254	3261	3268	3305	3311	3333	3339	3409	3494			
BPL	1801	1812	1825	1837	1848	1879	1951	1967	1978	2032	2079	2090	2104	2310	2680
	3133	3165	3205	3230	3246	3301	3307	3407							
BR	1488	1490	1520	1540	1543	1583	1623	1627	1629	1632	1651	1694	1730	1856	1863
	1866	1868	1898	1958	1985	1992	1995	1997	2009	2123	2262	2268	2271	2334	2348
	2349	2350	2356	2364	2371	2389	2394	2413	2455	2456	2457	2458	2459	2460	2461
	2478	2492	2664	2677	2778	2780	2782	2784	2786	2788	2790	2840	2843	2889	2993
	3037	3041	3050	3111	3142	3167	3184	3194	3203	3210	3257	3284	3286	3312	3335
	3385	3400	3421	3486	3510										
CLR	1482	1496	1581	1597	1628	1692	1693	1695	1723	1729	1734	2163	2325	2433	2665
	2666	2883	2981	2982	3028	3126	3243	3244	3398	3492					
CLRB	1633	1638	2275	2801	2830	2899	3183	3209	3340						
CMP	1486	1497	1517	1531	1537	1585	1721	1726	2145	2228	2269	2314	2469	2479	2490
	2503	2505	2624	2626	2711	2893	3047	3065	3074	3083	3106	3108	3227	3233	3253
	3260	3272	3274	3304	3310	3313	3315	3328							
CMPB	1484	1636	1890	2004	2006	2017	2115	2652	2660	2955	3034	3038	3042	3127	3138
	3140	3176	3178	3186	3207	3211	3235	3267	3332	3338					
COM	2488	2617	2622												
DEC	1557	1560	1696	1873	2166	2327	2380	2477	2569	2651	2654	2658	2662	2694	2698
	2835	2896	2957												
DECB	3190	3193	3406	3417											
EMT	1024														
HALT	1167	2226	2311	2316	3166	3485	3509								
INC	1529	1584	1630	1725	1731	2008	2164	2300	2446	2447	2448	2449	2450	2451	2491
	2693	2710	2802	2842	2888	3282	3412	3420	3493						



.BYTE	1185	1186	1191	1192	1200	1201	1209	1210	1211	1212	1257	1258	1259	1260	1310
	1313	1316	1319	1322	1325	1328	1331	1334	1337	1340	1343	1346	1349	1352	1355
	1364	1365	1789	1792	1888	1893	1896	1939	1942	2015	2020	2023	2067	2070	2113
	2118	2121	2243	2473	2483	3347	3348	3428	3429	3430	3431	3537	3538	3539	3540
	3541	3542	3543	3544	3546	3547	3548	3549	3550						
.DSABL	3287														
.ENABL	1	947	3220												
.END	3718														
.ENDC	952	960	1009	1010	1011	1012	1024	1116	1130	1131	1172	1175	1179	1183	1185
	1213	1219	1223	1227	1264	1285	1307	1361	1370	1389	1411	1459	1474	1475	1481
	1482	1499	1500	1503	1505	1507	1509	1510	1511	1527	1531	1533	1537	1543	1545
	1548	1556	1565	1580	1600	1621	1661	1663	1672	1701	1718	1736	1745	1747	1751
	1756	1783	1795	1797	1805	1807	1817	1819	1830	1832	1841	1843	1853	1855	1860
	1862	1871	1873	1885	1887	1903	1908	1933	1945	1947	1955	1957	1960	1962	1971
	1973	1982	1984	1989	1991	2001	2003	2011	2013	2026	2028	2040	2045	2060	2073
	2075	2083	2085	2095	2097	2110	2112	2126	2131	2144	2154	2155	2156	2158	2160
	2163	2168	2171	2172	2173	2175	2177	2179	2222	2224	2233	2235	2241	2243	2246
	2249	2252	2254	2259	2261	2272	2273	2274	2275	2279	2282	2285	2294	2301	2306
	2307	2308	2309	2314	2318	2319	2320	2321	2353	2373	2407	2416	2424	2453	2512
	2534	2579	2597	2636	2685	2702	2716	2720	2765	2815	2861	2864	2926	2944	2967
	2969	3002	3003	3020	3056	3116	3150	3170	3220	3221	3223	3251	3287	3291	3319
	3320	3327	3329	3332	3334	3350	3356	3359	3436	3442	3445	3457	3458	3459	3460
	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3473	3482	3483	3489	3495
	3496	3506	3508	3515	3516	3553									
.EQUIV	1024	1025	1033	1048	1049	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115					
.EVEN	1545	3514	3711												
.IF	948	957	1009	1010	1011	1022	1088	1116	1130	1170	1174	1178	1182	1184	1213
	1219	1223	1226	1227	1263	1284	1306	1360	1386	1408	1456	1472	1474	1480	1482
	1494	1499	1501	1503	1505	1507	1509	1510	1527	1530	1531	1532	1534	1537	1544
	1546	1555	1563	1579	1598	1620	1659	1661	1671	1699	1717	1734	1735	1743	1745
	1747	1754	1782	1794	1796	1804	1806	1816	1818	1829	1831	1840	1842	1852	1854
	1859	1861	1870	1872	1884	1886	1899	1906	1932	1944	1946	1954	1956	1959	1961
	1970	1972	1981	1983	1988	1990	2000	2002	2010	2012	2025	2027	2036	2043	2059
	2072	2074	2082	2084	2094	2096	2109	2111	2124	2129	2143	2153	2154	2155	2156
	2157	2158	2162	2167	2170	2172	2173	2175	2176	2178	2221	2223	2233	2241	2243
	2244	2246	2248	2251	2254	2259	2271	2273	2274	2275	2278	2279	2281	2284	2294
	2297	2304	2306	2307	2309	2313	2314	2318	2319	2320	2352	2372	2406	2415	2422
	2453	2510	2533	2577	2596	2635	2684	2701	2715	2719	2764	2814	2860	2863	2925
	2943	2966	2969	2981	3002	3019	3055	3115	3149	3170	3219	3221	3222	3223	3251
	3290	3291	3319	3327	3328	3332	3333	3349	3350	3356	3358	3435	3441	3445	3449
	3458	3459	3460	3461	3462	3464	3465	3466	3467	3468	3469	3470	3472	3482	3483
	3488	3495	3496	3504	3506	3508	3512	3515	3552						
.IFF	957	1009	1010	1011	1022	1131	1175	1179	1182	1184	1213	1227	1264	1285	1307
	1361	1386	1408	1456	1472	1475	1481	1499	1530	1532	1546	1556	1563	1580	1598
	1621	1659	1661	1672	1699	1718	1735	1743	1745	1747	1754	1783	1795	1797	1805
	1807	1817	1819	1830	1832	1841	1843	1853	1855	1860	1862	1871	1873	1885	1887
	1899	1906	1933	1945	1947	1955	1957	1960	1962	1971	1973	1982	1984	1989	1991
	2001	2003	2011	2013	2026	2028	2036	2043	2060	2073	2075	2083	2085	2095	2097
	2110	2112	2124	2129	2144	2154	2157	2163	2168	2171	2177	2179	2222	2224	2243
	2249	2271	2273	2279	2282	2284	2297	2313	2314	2319	2321	2353	2373	2407	2416
	2422	2453	2510	2534	2577	2597	2636	2685	2702	2716	2720	2765	2815	2861	2864
	2926	2944	2967	3003	3020	3056	3116	3150	3220	3223	3291	3293	3298	3319	3320
	3329	3333	3350	3359	3436	3442	3473	3489	3506	3516	3553				
.IFT	1545	2273	2307	2985	3001	3002	3293	3298							
.IFTF	1545	2273	2306	2981	2985	3001	3238	3291	3294						

.IIF	947	952	957	1006	1007	1008	1009	1012	1013	1167	1226	1500	1503	1509	1510
	1511	1531	2155	2156	2163	2164	2243	2246	2252	2253	2254	2258	2275	2279	2285
	2286	2287	2288	2289	2293	2312	2314	2319	2320	2363	2375	2411	3093	3096	3099
	3102	3217	3220	3241	3342	3350	3356	3457	3458	3459	3460	3462	3464	3465	3466
	3467	3468	3469												
.IRP	1227	1482	1739	2175	2455	2555	2571	2735	2755	2820	2831	2847	2976	2997	3476
	3482	3495	3496												
.LIST	1	947	957	958	959	960	1130	1161	1167	1213	1215	1216	1217	1218	1219
	1220	1221	1222	1223	1386	1387	1388	1389	1408	1409	1410	1411	1456	1457	1458
	1459	1472	1473	1474	1482	1511	1531	1534	1545	1546	1547	1548	1563	1564	1565
	1598	1599	1600	1659	1660	1661	1662	1663	1699	1700	1701	1734	1743	1744	1745
	1746	1747	1748	1749	1750	1751	1754	1755	1756	1899	1900	1901	1902	1903	1906
	1907	1908	2036	2037	2038	2039	2040	2043	2044	2045	2124	2125	2126	2129	2130
	2131	2163	2175	2233	2254	2279	2314	2320	2422	2423	2424	2453	2510	2511	2512
	2577	2578	2579	3319	3449	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466
	3467	3468	3469	3470	3515	3552									
.MACRO	1	1012	1161	1176	3449										
.MCALL	947	1130	1511	1534											
.NLIST	1	947	957	958	959	960	1130	1161	1167	1213	1215	1216	1217	1218	1219
	1220	1221	1222	1223	1386	1387	1388	1389	1408	1409	1410	1411	1456	1457	1458
	1459	1472	1473	1474	1482	1511	1531	1534	1545	1546	1547	1548	1563	1564	1565
	1598	1599	1600	1659	1660	1661	1662	1663	1699	1700	1701	1734	1743	1744	1745
	1746	1747	1748	1749	1750	1751	1754	1755	1756	1899	1900	1901	1902	1903	1906
	1907	1908	2036	2037	2038	2039	2040	2043	2044	2045	2124	2125	2126	2129	2130
	2131	2163	2175	2233	2254	2279	2314	2320	2422	2423	2424	2453	2510	2511	2512
	2577	2578	2579	3319	3449	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466
	3467	3468	3469	3470	3515	3552									
.NTYPE	1734	2453													
.PAGE	1002	1161	1176	1262	1305	1359	1370	1472	1598	1659	1699	1747	1899	2036	2124
	2151	2246	2279	2320	2372	2422	2469	2510	2715	2764	2814	2860	3002	3019	3055
	3115	3147	3356	3433	3470	3515	3552								
.REM	1	960													
.REPT	957	1167	1215	1219	1386	1408	1456	1472	1546	1563	1598	1659	1661	1699	1743
	1745	1747	1754	1899	1906	2036	2043	2124	2129	2422	2510	2577			
.SBTTL	957	1002	1020	1161	1170	1176	1265	1286	1308	1362	1370	1472	1493	1527	1534
	1752	1904	2041	2127	2151	2246	2279	2320	2424	2512	2534	2579	2597	2636	2686
	2703	2717	2766	2816	2861	2923	2941	2964	3003	3021	3056	3117	3147	3217	3356
	3433	3449	3470	3515	3552	3713									
.TITLE	947														
.WORD	1167	1168	1169	1184	1187	1188	1189	1190	1193	1194	1195	1196	1197	1198	1199
	1202	1203	1204	1213	1215	1216	1217	1218	1219	1220	1221	1222	1227	1228	1229
	1230	1231	1232	1233	1234	1236	1238	1248	1249	1250	1251	1252	1253	1254	1255
	1256	1261	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
	1280	1281	1282	1366	1367	1368	1790	1793	1889	1894	1897	1940	1943	2016	2021
	2024	2068	2071	2114	2119	2122	2167	2170	2242	2370	2381	2388	2393	2474	2484
	2854	2855	2856	2857	2897	2940	2960	3001	3018	3214	3432	3505	3507	3519	3523
	3526	3529	3531												

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

\* ,DZTAEB.SEG/SOL/CRF/PAGNUM/NL:TOC=SYSMAC.CO,DZTAEB.P11  
 RUN-TIME: 38 52 6 SECONDS

K08

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-B MACY11 27(732) 25-SEP-76 11:17 PAGE 105  
DZTAE.B.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

RUN-TIME RATIO: 188/97=1.9  
CORE USED: 35K (69 PAGES)

