

# RP11E

## DRIVE POSITIONING TEST MD-11-DZRPZ-B

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

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

[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]
[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]	[Illegible]

[Illegible]





100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT WILL ENSURE THAT THE DISK DRIVE IS CAPABLE OF PERFORMING SEEKS, THAT THE ACCESS TIMES ARE WITHIN TOLERANCE, AND THAT THE CYLINDER ADDRESSING CAPABILITY OF THE DRIVE IS FUNCTIONING PROPERLY.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH 8K OF MEMORY (WITH OR WITHOUT HARDWARE SWITCH REGISTER); CONSOLE TELETYPE; RP11E DISK CONTROLLER;  
1 TO 8 RPO2, RPRO2, OR RPO3 DISK DRIVES WITH FORMATTED PACKS.

2.2 STORAGE

THIS PROGRAM WILL LOAD AND RUN IN 8K.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZRPW - RP11E DISKLESS LOGIC TEST

3. LOADING PROCEDURE

\*\*\* BEFOR STARTING REFER TO SECTION 5.\*\*\*  
THE PROGRAM MAY BE LOADED FROM EITHER PAPER TAPE OR AN 'XXDP' DEVICE. REFER TO EITHER THE STANDARD PROCEDURES FOR LOADING 'ABS' FORMAT PAPER TAPES OR TO THE 'XXDP' SYSTEM REFERENCE DOCUMENT.

THIS PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS PART OF A CHAIN AND IS LOADED FROM AN RPO2, RPRO2, OR AN RPO3, THE PROGRAM WILL BYPASS TESTING DRIVE 0.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS  
204 SELECT OPERATING PARAMETERS  
210 SELECT RP11 ADDRESS  
214 COMBINATION OF 204 AND 210

4.1.1 START FROM LOCATION 200

E01

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB

MACY11 27(732) 02-NOV-76 15:10 PAGE 5

152  
153  
154  
155

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE PROGRAM WILL  
TEST ALL AVAILABLE DRIVES (SEE SECTION 2.1 FOR DRIVE 0 EXCEPTION)  
USING TESTS 0 - 10, & 12, WITH DEFAULT PARAMETERS. PREVIOUS  
PARAMETER CHANGES ARE OVERLAYED.

156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211

4.1.2 START FROM LOCATION 204

WHEN THE PROGRAM IS STARTED FROM LOCATION 204, THE PARAMETERS ARE RESET TO THEIR DEFAULT VALUES, AND THE PROGRAM ENTERS CONVERSATIONAL MODE. REFER TO SECTION 4.3.2.

4.1.3 START FROM LOCATION 210

WHEN THE PROGRAM IS STARTED FROM LOCATION 210, OPERATION IS THE SAME AS THE START FROM 200, EXCEPT THAT THE PROGRAM ASKS FOR A NEW RP11E ADDRESS, VECTOR ADDRESS, AND PRIORITY. THE OPERATOR MAY ENTER NEW ADDRESS VALUES OR RETAIN THE OLD VALUES. THE TEST PARAMETERS ARE RESET TO THE DEFAULT VALUES. REFER TO SECTION 4.3.1.

4.1.4 START FROM LOCATION 214

PROGRAM START FROM LOCATION 214 IS THE SAME AS THE START FROM LOCATION 204 EXCEPT THAT THE OPERATOR MAY ENTER NEW RP11E ADDRESS, VECTOR ADDRESS, OR PRIORITY. THE TEST PARAMETERS ARE RESET TO THE DEFAULT VALUES. REFER TO SECTIONS 4.3.1 & 4.3.2.

4.2 OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A FORMATTED PACK INTO DRIVE(S) TO BE TESTED
3. BRING DRIVE(S) TO ONLINE STATE.
4. SET 'FORMAT' AND 'MAINTENANCE' SWITCHES ON THE RP11E CONTROL PANEL TO 'NORMAL'.
5. LOAD ADDRESS 200, 204, 210, OR 214.
6. SET SWITCHES (SEE SECTION 5.)
7. PRESS START.

4.3 PROGRAM ACTION

IN AN EFFORT TO ALLOW CONVERSATION WITH A PROGRAM FOR THE PURPOSE OF CONTROLLING ITS OPERATION AND PARAMETERS THE FOLLOWING CONSTRUCTIONS HAVE BEEN ADOPTED.

NOTE1: IN ALL EXAMPLES BRACKETS ARE USED FOR CLARITY AND ARE NOT TYPED BY THE USER.

NOTE2: THE CARRIAGE RETURN TYPED BY THE USER IS INDICATED BY <CR> AND WILL BE ECHOED AS A "CARRIAGE RETURN-LINE FEED."

<.)<CR> PERIOD

A STATEMENT TERMINATOR: WHEN TYPED AT THE END OF A LINE (LEGAL ON ALL LINES) IT TELLS THE PARAMETER STRING INTERPRETER (PSI) THIS IS THE END OF CHANGES TO THE CURRENT PARAMETER STRING.

<..) <CR> PERIOD PERIOD

AN INPUT TERMINATOR: WHEN A PERIOD IS TYPED AFTER A "STATEMENT TERMINATOR PERIOD" IT TELLS

MD-11-DZRPZ-B, RP11E DRIVE POSITIONING TEST  
DZRPZB.CMB

**GO1**  
MACY11 27(732) 02-NOV-76 15:10 PAGE 7

212  
213  
214

THE PSI THIS IS THE END OF ALL CHANGES (LEGAL  
ON ALL LINES) BEGIN EXECUTION.

215  
216  
217  
218  
219  
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  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
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

<,><CR>

COMMA

A CONTINUATION INDICATOR AND FIELD SEPARATOR:

- 1) THIS IS THE CONSTRUCTION USED WHEN NOT USING A PERIOD; THAT IS TO SAY THE COMMA SAYS "TERMINATE THE LINE BUT NOT THE INPUT FOR PARAMETER CHANGES".
- 2) SEPARATE MULTIPLE ENTRIES ON THE SAME TEXT STRING, E.G., A,B,C,D.

</>

SLASH

A MODIFICATION INDICATOR: AT ANY GIVEN PARAMETER STOP IF A SLASH IS TYPED IT SAYS "OPEN THIS PARAMETER FOR CHANGES".

<↑U>

CONTROL-U

DUMP THE PRESENT INPUT STRING AND START A NEW LINE. TYPED BY DEPRESSING THE "CONTROL KEY" (CTRL) AND THEN STRIKING THE "U".

<\>

RUBOUT

DELETE THE LAST CHARACTER FROM THE INPUT STRING. TYPED BY STRIKING THE "RUBOUT" KEY, WHICH WILL BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE CHARACTER DELETED.

#### 4.3.1 ADDRESS SELECTION

STARTING THE PROGRAM FROM 200 (8) WILL RESULT IN AUTOMATIC SELECTION OF THE DEFAULT RP11E VALUES OF BUS ADDRESS, VECTOR ADDRESS, AND PRIORITY LEVEL. IF THE DEFAULT VALUE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) TO ADDRESSING, IT IS REPORTED AS AN ERROR. AFTER THE ERROR IS REPORTED, ONE OF TWO COURSES OF ACTION WILL BE TAKEN:

1. IF THERE IS A MONITOR -- RETURN TO THE MONITOR
2. IF THERE ISN'T A MONITOR -- ASK FOR NEW ADDRESSES

STARTING THE PROGRAM AT EITHER LOCATION 210 OR LOCATION 214 ALLOWS THE OPERATOR TO CHANGE THE RP11E BUS ADDRESS, THE VECTOR ADDRESS, AND (OR) THE PRIORITY.



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

4.3.1.1 ADDRESS SELECTION EXAMPLES

EXAMPLE #1

RPDS=176710/176300..

EXAMPLE #2

RPDS=176710/176300,  
RHVEC=254/260,  
RHPRIO=5/6..

EXAMPLE #3

RPDS=177200,  
RHVEC=254/260..

EXAMPLE #4

RP11 FAILED TO RESPOND TO ADDRESSING  
RPDS ERR PC  
176710 XXXXXX  
RPDS=176710/176300..

EXAMPLE #5

RPDS=176710/1776\67\6300,  
RHVEC=254,  
RHPRIO=5,  
RPDS=176300

4.3.2 DRIVE, TEST, AND PARAMETER SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC  
SELECTION OF THE DRIVES TO TEST AND THE TESTS TO RUN.

STARTING THE PROGRAM AT 210 OR 214 ALLOWS THE OPERATOR  
TO SELECT THE DRIVE(S) TO BE TESTED, THE TESTS TO BE EXECUTED,  
AND THE PARAMETERS TO USE.

304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352

4.3.2.1 DRIVE, TEST, AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED IN THE DESCRIPTION.

"R"	REPEATS (ITERATIONS)
"FC"	FIRST CYLINDER ADDRESS
"LC"	LAST CYLINDER ADDRESS
"IC"	INCREMENT CYLINDER
"TK"	TRACK ADDRESS

SPECIAL CASES OF CONTROL CHARACTERS

IF <...> IS TYPED WHILE A TEST IS OPEN FOR MODIFICATION (</>) AND OTHER TESTS IN THE "TEST COMMAND" STRING ARE TO BE OPENED, THEY WILL RECIEVE THE PARAMETERS OF THE TEST THAT IS OPEN WHEN <...> IS TYPED.

<,> IS ILLEGAL AS A "TEST" LINE TERMINATOR UNLESS A TEST IS TO BE OPENED.

ON STARTING THE PROGRAM ALL TESTS WILL RUN AGAINST ALL DRIVES IN A WORST CASE MANNER. BUT IF THE USER DESIRES TO SELECT THE DRIVES TO BE TESTED, THE TESTS TO BE EXECUTED, AND THE PARAMETERS TO BE USED HE ENTERS CONVERSATION MODE BY TYPING CONTROL C (IC).

THE PROGRAM WILL RESPOND WITH:

DRIVE(S)=

WE WILL ASSUME FOR OUR EXAMPLE THAT THE OPERATOR WISHES TO TEST DRIVE #3 USING TESTS 2 THRU 7 AND TEST 11 AND DOES NOT DESIRE TO CHANGE THE PARAMETERS (INITIAL CYLINDER ADDRESS, FINAL CYLINDER ADDRESS, ETC.). THE USER WOULD TYPE "3<,>" WHICH SAYS "THIS IS THE END OF ANY CHANGES TO THIS LINE BUT TAKE ME DEEPER FOR MORE CHANGES", AND <CR> "TAKE IT."

FOR CLARITY THE LINE WILL BE REPEATED AS IT LOOKS AFTER USER RESPONSE WITH THE LINE BELOW IT ADDED, THE PROGRAMS RESPONSE:

DRIVE(S)=3,<CR>  
TEST=

NOW THE USER INSERTS THE TEST NUMBERS HE DESIRES. IN THIS CASE HE WANTS TESTS 2 THRU 7 AND TEST 11 SO HE TYPES 2-7<,> (TO SEPERATE FIELD), 11<...> (END OF ANY CHANGES START EXECUTION), <CR> (TAKE IT).

IT NOW LOOKS LIKE THIS

DRIVE(S)=3,<CR>  
TEST=2-7,11..<CR>

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

IN OUR NEXT EXAMPLE WE WILL ASSUME THE USER WISHES TO TEST DRIVE 4 AND RUN TESTS 1 AND 3 THRU 11 WITH THE PARAMETER FOR TESTS 3 AND 10 MODIFIED. THIS IS HOW HE WOULD RESPOND:

```
DRIVE(S)=4,<CR>
TEST=
```

THE USER NOW WILL INSERT A LINE TO GET TESTS 1 AND 3 THRU 11 BUT REMEMBER HE WISHES TO "OPEN" TESTS 3 AND 10 FOR PARAMETER CHANGES. THE ENTIRE ENTRY IS SHOWN BELOW:

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
```

NOTICE THIS SAYS SELECT TEST ONE, CONTINUE< >; SELECT TEST 3, OPEN</>; SELECT TESTS 4-7, CONTINUE<,>; SELECT TEST 10, OPEN</>; SELECT TEST 11, END OF INPUT<.>.

THE PSI SCANS THE TEST STRING AND DETERMINES THAT TEST #1'S PARAMETERS WILL REMAIN THE SAME BUT TEST 3'S MUST BE CHANGED. THEREFORE, TEST 3 IS OPENED FOR CHANGE.

RESPONDS: <FOR CLARITY ENTIRE TRANSACTION REPEATED>

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R=X ;WHERE X IS ITERATION
```

AS IN THE NORMAL MANNER TYPING A </> WILL OPEN THIS LINE FOR MODIFICATION OF THE ITERATION COUNT, ENDING THE LINE WITH A <.> (PERIOD) WOULD TERMINATE THE CHANGES FOR THIS TEST, AND AS IN THE EXAMPLE SHOWN USING A <,> (COMMA) WILL GET US THE NEXT PARAMETER.

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11,<CR>
TEST 3
R=1,<CR> ;DO NOT ALTER-BUT CONTINUE
FC=N ;WHERE N IS FIRST CYLINDER ADDRESS
```

398  
399  
400  
401  
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

THE USER DOES NOT WISH TO CHANGE "FC" SO THE FOLLOWING ACTION IS TAKEN:

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R=1,<CR>           ;DO NOT ALTER THIS LINE BUT CONTINUE
FC=0,<CR>         ;DO NOT ALTER THIS LINE BUT CONTINUE
LC=202
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING '202' AS THE EXAMPLE. THIS IS WHAT THE USER INTENDED TO MODIFY AND IS WHY HE OPENED TEST #3. HE WANTS TO CHANGE IT TO CYLINDER 20. THEREFORE HE TYPES </> (OPEN LINE FOR CHANGE), "20" (THE NEW VALUE), <.> (THIS IS THE END OF CHANGES FOR THIS DRIVE), <CR> (TAKE IT).

THE TOTAL TRANSACTION AND RESPONSE:

```
DRIVE(S)=4,<CR>
TEST=1,3/4-7,10/11.<CR>
TEST 3
R= 1,<CR>
FC=0,<CR>
LC= 202/20.<CR>
TEST 10
R=1
```

THE PROGRAM HAS LOADED TEST #3 WITH ITS NEW PARAMETERS, TESTS NUMBER 4 THRU 7 WITH THEIR PREVIOUSLY ASSIGNED PARAMETERS AND IS NOW WAITING FOR CHANGES TO TEST 10.

THE USER TYPES </> (OPEN THIS LINE FOR CHANGE), "10" (MAKE ITERATION COUNT 10), <.> (THIS IS THE END OF CHANGES TO THIS TEST) <CR> (TAKE IT):

THE PROGRAM NOW LOADS TEST 10 WITH THE NEW PARAMETERS, TEST 11 WITH PREVIOUSLY ASSIGNED PARAMETERS AND RESPONDS WITH:

DRIVE(S)=

SINCE THE USER DID NOT END THE CONVERSATION MODE WITH A <...> THE PROGRAM HAS LOOPED BACK TO THE BEGINNING LOOKING FOR MORE CHANGES. THAT IS TO SAY, AFTER THE ENTRY FOR DRIVE SELECTION A <.><CR> WILL CAUSE THE TEST MESSAGE TO BE REPEATED AND FURTHER CHANGES CAN BE MADE. HOWEVER, AT SOME POINT IN ORDER TO EXECUTE THE CHANGES MADE WE MUST NOTIFY THE PROGRAM WE ARE FINISHED BY TYPING <...>.

4.3.3.2 DRIVE, TEST, AND PARAMETER SELECTION EXAMPLES

EXAMPLE #1

DRIVE=4.<CR>

;SELECT DRIVE #4, TERMINATE AND  
;BEGIN EXECUTION USING PREVIOUSLY ASSIGNED  
;PARAMETERS

EXAMPLE #2

DRIVE=0,<CR>  
TEST=1-5.<CR>

;SELECT DRIVE #0 AND MAKE CHANGES ""  
;RUN TEST 1 THRU 5 ONLY, USE DEFAULT  
;PARAMETERS AND TERMINATE AND EXECUTE ""

EXAMPLE #3

DRIVE=2,<CR>  
TEST=1-5,6/7/10/<CR>  
TEST 6  
R=1,<CR>  
FC=0/10.<CR>

;SELECT DRIVE #2 AND MAKE CHANGES ""  
;RUN TEST 1-5 WITH DEFAULT PARAMETERS, OPEN  
;TEST 6, 7 AND 10 FOR CHANGES  
;LEAVE R AS IS MOVE TO NEXT PARAMETER  
;SET "FC" CYLINDER ADDRESS TO 10 "." END CHANGES  
;TO TEST #6

TEST 7

R=1/50,<CR>  
FC=0,<CR>  
LC=202/50.<CR>

;50 ITERATIONS, MOVE TO NEXT PARAMETER  
;DO NOT CHANGE "FC" CYLINDER ADDRESS BUT CONTINUE "",  
;TEST 10 IS STILL PENDING  
;AND WILL THEREFORE BE  
;SET TO THE SAME PARAMETERS  
;AS TEST 7 -- START EXECUTION

EXAMPLE #4

DRIVE=0,1,4,  
TEST=0-5/  
TEST 0  
R=10,  
FC=0,  
LC=202/1..

;TEST DRIVES 0,1, AND 4 IN SEQUENCE  
;CHANGE TEST 0-5

;CHANGE LAST CYLINDER FROM 202  
;TO 1 FOR TEST #0-5; START TESTING

4.4 'CONTROL C' OPERATION

THE OPERATOR MAY TERMINATE THE PROGRAM AT ANY TIME BY TYPING A  
'CONTROL C'. 'CONTROL C' RETURNS THE PROGRAM TO THE PARAMETER  
ENTRY ROUTINE. THE TEST PARAMETERS ARE NOT RESTORED TO THE DEFAULT  
VALUES BY THE 'CONTROL C' RETURN.

CONTROL G OPERATION (REFER TO SECTION 5.)

447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
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

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB

NO1

MACY11 27(732) 02-NOV-76 15:10 PAGE 14

503  
504

5. OPERATING PROCEDURE

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: SW: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.

5.1 OPERATIONAL SWITCH SETTINGS

WITH SW<15:0>=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<11>=1...INHIBIT ITERATIONS
- SW<10>=1...RING BELL ON ERROR
- SW<09>=1...LOOP ON ERROR
- SW<06>=1...TYPE ALL RPII REGISTERS IF ERROR
- SW<04>=1...TYPE THE LONG SEEK TIME GRAPH (TEST #12)
- SW<03>=1...PERFORM STALL BETWEEN SEEK ORDERS
- SW<02>=1...USE RANDOM STALL VALUE BETWEEN SEEK ORDERS
- SW<01>=1...PERFORM INCREMENTING STALL IN TEST 4
- SW<00>=1...DO NOT CHECK POSITION AFTER SEEK ORDERS

505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
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  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600

C02

MD-11-DZRPZ-8, RP11E DRIVE POSITIONING TEST  
DZRPZ8.CMB

MACY11 27(732) 02-NOV-76 15:10 PAGE 16

561  
562  
563



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

6. ERRORS

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

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

6.1 ERROR TYPES

THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO TWO (2) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

6.1.1 NON-FATAL ERROR

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

6.1.2 FATAL ERROR

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND OF ERROR THAT INHIBITS THE PROGRAM FROM TESTING THE DISK.

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

6.2 ERROR RECOVERY

6.2.1 NON-FATAL ERROR

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

6.2.2 FATAL ERROR

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

7. RESTRICTIONS

7.1 SUBSYSTEMS WITH MIXED DRIVE TYPES

THE PROGRAM ASSUMES THAT ONLY ONE KIND OF DRIVE IS ATTACHED TO A SYSTEM (E.G. RPO2'S & RPRO2'S OR RPO3'S). IF BOTH RPO2'S

E02

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB

MACY11 27(732) 02-NOV-76 15:10 PAGE 18

620  
621  
622  
623  
624

(AND RPO2'S) OR RPO3'S ARE ON THE SAME SYSTEM, THE PROGRAM WILL ASSUME THAT ALL OF THE DRIVES ARE RPO3'S. TO TEST DRIVES ON A MIXED SYSTEM, TEST ALL OF THE SAME KIND OF DRIVE AT ONE TIME. THE OTHER DRIVES ON THE CONTROLLER MUST BE POWERED DOWN.

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  
674  
675  
676  
677  
678  
679  
680

7.2 FORMATTED PACKS

EACH OF THE DRIVES BEING TESTED MUST HAVE A FORMATTED PACK. (UNLESS THE PROGRAM IS RUN WITH SWITCH <00> SET.)

8. MISCELLANEOUS

8.1 EXECUTION TIME

TO MAKE ONE PASS OF THE PROGRAM (TESTS 0 - 7) TAKES APPROXIMATELY 1.8 HOURS FOR EACH RPO2 OR RPO2 TESTED AND APPROXIMATELY 6.5 HOURS FOR EACH RPO3 TESTED.

NOTE THAT TEST 7 REQUIRES 1.3 HOURS FOR RPO2 DRIVES AND 5.4 HOURS FOR RPO3 DRIVES. IN SPITE OF THE LENGTHY RUN TIME OF TEST 7, IT IS RECOMMENDED THAT TEST 7 BE RUN OCCASIONALLY TO VERIFY THE SUBTRACTOR IN THE DISK DRIVE. TEST 7 HAS ALSO PROVEN TO BE A VALUABLE LONG TERM SEEK VERIFIER, ALSO.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100 AND EXTENDS DOWNWARD IN MEMORY.

8.3 TIMING TEST PRINTOUTS

8.3.1 TIMING TEST (TESTS 10 & 11) PRINTOUT EXAMPLES

EXAMPLE #1

ONE CYLINDER SEEK TIMES  
(SEEK TIME IS IN MICRO SECONDS)

FORWARD		REVERSE	
# OF SEEKS	SEEK TIME	# OF SEEKS	SEEK TIME
193	10000	60	10000
9	12500	142	12500

EXAMPLE #2

SEEK TIMES BETWEEN CYLINDERS 0 & 202  
(SEEK TIME IS IN MICRO SECONDS)

FORWARD		REVERSE	
# OF SEEKS	SEEK TIME	# OF SEEKS	SEEK TIME

G02

ND-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB

NACY11 27(732) 02-NOV-76 15:10 PAGE 20

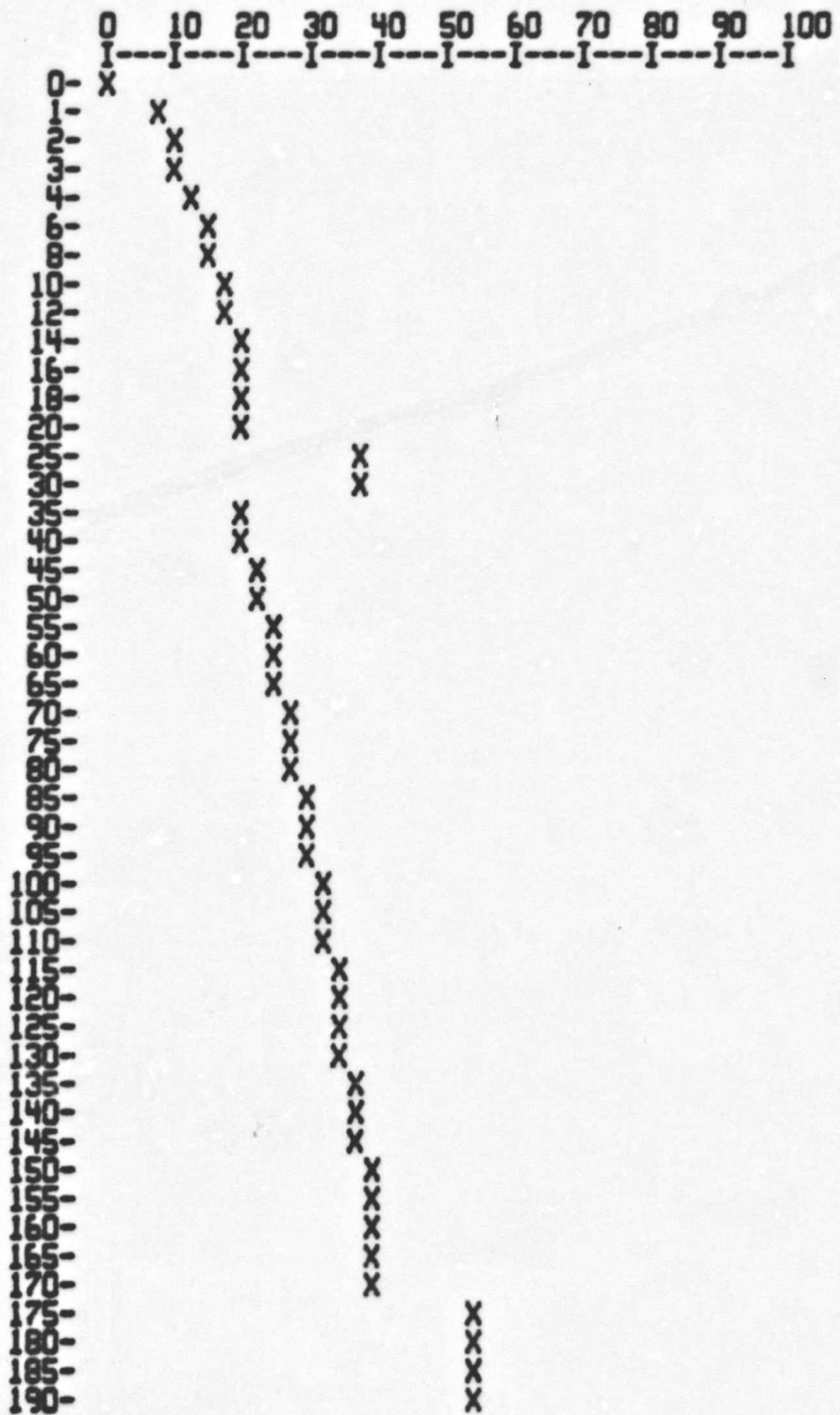
681  
682  
683  
684

200	45000	156	45000
		44	60000

8.3.2 SEEK GRAPH TEST (TEST #12)

EXAMPLE: SMALL SEEK GRAPH (SWITCH <04> = 0)

SEEK TIME GRAPH  
X AXIS - SEEK TIME - MILLI SECS  
Y AXIS - CYLINDER SEEKED FROM 0



685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740

741  
742  
743  
744  
745

195-  
200-  
202-

X X  
X X

746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801

8.4 END OF TEST

WITH ALL SWITCHES ON A "0" AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF TESTING A DRIVE AND THE "END OF TEST" TYPEOUT WILL OCCUR WHEN ALL DRIVES HAVE BEEN TESTED.

8.5 DEFAULT ADDRESSES, TEST SEQUENCE, AND PARAMETERS

8.5.1 DEFAULT ADDRESS

THE PROGRAM IS LOADED FOR THE FOLLOWING ADDRESSES:

RP11E BUS ADDRESS - 176710  
RP11E VECTOR ADDRESS - 254  
RP11E PRIORITY - 5

TO ALTER THESE ADDRESS, START THE PROGRAM AT LOCATION 210 OR 214.

8.5.2 DEFAULT TEST SEQUENCE

THE DEFAULT TEST SEQUENCE IS TESTS 0 - 7. THIS SEQUENCE IS RESTORED ANY TIME A START FROM 200, 204, 210, OR 214 IS INITIATED. 'CONTROL C' RESTART DOES NOT ALTER THE TEST SEQUENCE.

8.5.3 DEFAULT TEST PARAMETERS

THE DEFAULT TEST PARAMETERS ARE GIVEN BELOW. REFER TO SECTION 4.3.1 FOR A GLOSSARY OF THE MNEMONICS USED TO IDENTIFY THE PARAMETERS. THE DEFAULT TEST PARAMETERS ARE RESTORED WHEN A START AT 200, 204, 210, OR 214 IS INITIATED. THE 'CONTROL C' RESTART DOES NOT ALTER THE CURRENT PARAMETER VALUES.

TEST #	PARAMETERS				
	'R'	'FC'	'LC'	'IC'	'TK'
TEST 0	100	-	-	-	0
TEST 1	1000	0	MAXCYL	-	0
TEST 2	10	0	MAXCYL	1	0
TEST 3	10	0	NOTE 3	1	0
TEST 4	10	0	MAXCYL	1	0
TEST 5	10	0	MAXCYL	1	0
TEST 6	10	0	MAXCYL	1	0
TEST 7	1	0	MAXCYL	1	0
TEST 10	(SEE BELOW - NOTE 1)				
TEST 11	1	0	MAXCYL	-	-

K02

MD-11-DZRPZ-B, RP11E DRIVE POSITIONING TEST  
DZRPZB.CMB

MACY11 27(732) 02-NOV-76 15:10 PAGE 24

802  
803  
804  
805  
806

TEST 12 (SEE BELOW - NOTE 2)



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  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850

NOTE 1: TEST 10 ASSUMES THE FOLLOWING DEFAULT PARAMETERS. THESE  
PARAMETERS CANNOT BE ALTERED BY THE OPERATOR.

1 0 MAXCYL 1 -

NOTE 2: TEST 12 ASSUMES THE FOLLOWING DEFAULT PARAMETERS. THESE  
PARAMETERS CANNOT BE ALTERED BY THE OPERATOR.

1 0 MAXCYL 1 -

NOTE 3: 'MAXCYL' USED ABOVE IS USED TO DENOTE THE MAXIMUM CYLINDER  
ADDRESS USED BY THE PROGRAM. THE VALUE OF 'MAXCYL' IS  
DETERMINED BY THE TYPE OF DRIVES USED ON THE SYSTEM. 'MAXCYL'  
IS 405 (10) IF THE DRIVES ARE RPO3'S AND IS 202 (10) IF THE  
DRIVES ARE RPO2'S. IN TEST 3, 'LC' WILL BE 128 (10) IF  
THE DRIVES ARE RPO2'S AND 256 (10) IF THE DRIVES ARE RPO3'S.

8.6 TIMING TEST RESTRICTIONS

THE TIMING TESTS (TESTS 10, 11, & 12) ARE NOT INTENDED TO BE 'PASS-  
FAIL' TESTS. BECAUSE A HIGH RESOLUTION CLOCK IS NOT USED, THE  
TIMES ARE APPROXIMATE AND ARE INTENDED TO BE A GENERAL INDICATION  
OF THE CONDITION OF THE DRIVE'S SERVO SYSTEM. NO ATTEMPT SHOULD  
BE MADE TO ADJUST THE SERVO SYSTEM USING THE OUTPUT FROM THESE  
TESTS.

8.7 POSITIONING CHECK

IF SW<00> IS NOT SET, THE PROGRAM WILL CHECK THE DISK'S POSITION AFTER  
EACH SEEK (EXCEPT IN TESTS 10, 11, & 12).

THE PROGRAM CHECKS POSITION IN THE FOLLOWING MANNER:

USING THE 'SOT' COUNTER TO DETERMINE THE ADDRESS, THE PROGRAM READS  
THE HEADER FROM THE NEXT SECTOR AND COMPARES THE CYLINDER ADDRESS  
IN THE HEADER AGAINST THE CYLINDER THE PROGRAM EXPECTED; IF THE  
CYLINDER ADDRESS IN THE HEADER DOESN'T AGREE WITH THE EXPECTED  
CYLINDER ADDRESS, THE PROGRAM REPORTS A POSITIONING ERROR AND ISSUES  
A HOME SEEK TO THE DRIVE. (THE PROGRAM ALSO CHECKS POSITION AFTER  
A HOME SEEK OPERATION.)

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  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903

9. PROGRAM DESCRIPTION

-----  
THE RPIIE DRIVE POSITIONING TEST CONTAINS 11 TESTS. TESTS 0 - 6 TEST THE OPERATION OF THE DRIVE'S SERVO SYSTEM. POSITIONING ACCURACY IS CHECKED BY READING A HEADER FROM THE DESTINATION CYLINDER USING A 'READ WITHOUT IMPLIED SEEK' INSTRUCTION AFTER THE SEEK TERMINATES.

TESTS 10), & 11, 12 MEASURE SEEK TIME. TEST 10 & 11 MEASURE THE TIME FOR SPECIFIC SEEK OPERATIONS; TEST 12 MEASURES THE TIME REQUIRED TO SEEK FROM CYLINDER 0 TO ALL OTHER CYLINDERS. TESTS 10 & 11 TYPE OUT THE MEASURED SEEK TIME IN MICROSECONDS WHILE TEST 12 PRODUCES A SEEK TIME GRAPH. IN THE SEEK TIMING TESTS, POSITIONING ACCURACY IS NOT CHECKED.

SEEK TIME IN TESTS 10, 11, & 12 IS DETERMINED BY COUNTING THE NUMBER OF SECTOR PULSES OCCURING BETWEEN START OF SEEK AND SEEK DONE. THE 'SOT' COUNTER IS USED AS THE REFERENCE IN THESE TESTS. THE RESOLUTION OF THE TIMING TESTS IS 2.5 MS.

THE PROGRAM WILL START BY IDENTIFYING ITSELF AND DETERMINING ALL DRIVES THAT ARE AVAILABLE FOR TESTING. THEN BEGINNING WITH THE LOWEST NUMERICAL DRIVE AND PROCEEDING IN SEQUENTIAL ORDER ALL OF THE DRIVES WILL BE TESTED. ONE PASS THROUGH THE TEST SEQUENCE (TESTS 0 - 7) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. THE DRIVE TO BE TESTED WILL BE TYPED AT THE BEGINNING OF EACH PASS, AN "END OF PASS" MESSAGE WILL BE TYPED AT THE COMPLETION OF EACH PASS, AND AN "END OF TEST" MESSAGE WILL BE TYPED AFTER TESTING ALL DRIVES.

REFER TO THE FOLLOWING SECTION FOR A DETAILED DESCRIPTION OF EACH TEST.

9.1 TEST DESCRIPTIONS

-----  
IN THE DESCRIPTIONS OF THE FOLLOWING TESTS, THE VARIABLES USED AND THEIR DEFAULT VALUES (UNLESS SPECIFIED OTHERWISE) ARE:

MNEMONIC	VARIABLE
-----	-----
FC	FIRST CYLINDER ADDRESS
LC	LAST CYLINDER ADDRESS
IC	INCREMENT VALUE
NC OR NC1	NEW OR MODIFIED CYLINDER (FC+IC) ADDRESS
NC2	NEW OR MODIFIED CYLINDER (LC-IC) ADDRESS
TK	TRACK ADDRESS

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  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959

9.1.1 TEST 0 - HOME SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A HOME SEEK COMMAND CYCLE. AT THE COMPLETION OF THE COMMAND, THE STATUS INDICATORS ARE CHECKED TO VERIFY THAT NO ERRORS HAVE OCCURED.

9.1.2 TEST 1 - SEEK/SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK CYCLE TO "LC" "LT" "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO "FC" "FT" "FS". AT THE COMPLETION OF EACH SEEK, THE PROPER INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

9.1.3 TEST 2 - INCREMENTAL SEEK TEST

THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC". WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC" UNTIL "NC" IS LESS THAN "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

9.1.4 TEST 3 - STEPPING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4, 8, 16, 32, 64, AND 128 (AND 256 IF AN RPO3). AT THE COMPLETION OF EACH SEEK COMMAND, THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

9.1.5 TEST 4 - OSCILLATING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

9.1.6 TEST 5 - CONVERGING/DIVERGING SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE SEEKS FROM "NC1" AND "NC2" RESPECTIVELY. "NC1" WILL BE INCREMENTED BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS LESS THAN THE INITIAL VALUE OF "NC1". AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION. "NC1" AND "NC2" DEFAULT TO "FC" AND "LC" RESPECTIVELY.

9.1.7 TEST 6 - SERVO ADDRESSING LOGIC NOSE GENERATION

IN THIS TEST A SEEK IS DONE TO CYL "NC" THEN A SEEK TO NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5. NOW "NC" IS UPDATED BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNTIL "LC" IS

803

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB

MACY11 27(732) 02-NOV-76 15:10 PAGE 28

960  
961  
962  
963

EXCEEDED BY ANY OF THE ABOVE VALUES. THE INITIAL VALUE OF "NC"  
IS "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE  
PROPER INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

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  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010

## 9.1.8 TEST 7 - CYLINDER ADDRESSING TEST

THIS TEST WILL CAUSE THE DRIVE TO SEEK FROM EACH CYLINDER TO EVERY OTHER CYLINDER BETWEEN CYLINDERS "FC" AND "LC". SEEK CYCLES ARE COMMANDED FROM CYLINDER "NC" TO CYLINDER "NC1" AND BACK TO CYLINDER "NC". "NC" AND "NC1" START AT "FC". "NC1" IS INCREMENTED BY "IC" UNTIL "NC1" IS EQUAL TO "LC"; WHEN "NC1" IS EQUAL TO "LC", IT IS RESET TO "FC" AND "NC" IS INCREMENTED BY "IC". THE CYCLE IS COMPLETE WHEN "NC" AND "NC1" BOTH EQUAL "LC". AT THE COMPLETION OF EACH SEEK COMMAND, THE INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

## 9.1.9 TEST 10 - ONE CYLINDER SEEK TIMER TEST

THIS TEST PERFORMS ONE CYLINDER SEEKS FROM CYLINDER 0 TO THE MAXIMUM CYLINDER FOR THE DRIVE (E.G. 202 FOR RP02'S AND 405 FOR RP03'S). THE SEEK TIMES ARE MEASURED RELATIVE TO THE SECTOR PULSES FROM THE DRIVE. THE TEST DISPLAYS THE FORWARD AND REVERSE TIMES.

## 9.1.10 TEST 11 - SEEK-SEEK TIMER TEST

THIS TEST IS A GENERAL PURPOSE TIMING TEST WHICH PERFORMS 200 ITERATIONS BETWEEN THE CYLINDER IN 'FC' AND THE CYLINDER IN 'LC'. BOTH FORWARD AND REVERSE SEEKS ARE TIMED AND DISPLAYED. THE SEEK TIMES DISPLAYED ARE RELATIVE TO THE SECTOR COUNTER. 'FC' DEFAULTS TO 0; 'LC' DEFAULTS TO 'MAXCYL' (202 IF RP02'S OR 405 IF RP03'S).

## 9.1.11 TEST 12 - SEEK TIME GRAPH TEST

THIS TEST PRODUCES TWO SEEK TIME GRAPHS. THE FIRST GRAPH IS TYPED IS SWITCH <04> IS 0. THIS GRAPH IS A PLOT OF SEEK TIME AGAINST CYLINDERS SEEKED TO FROM CYLINDER 0. THE CYLINDERS PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25...100,105...MAXCYL. THE OTHER GRAPH WHICH IS PRODUCED (SWITCH <04> SET) IS TIME TO SEEK TO ALL CYLINDERS FROM CYLINDER 0.

IN BOTH SEEK GRAPHS, ONLY THE FORWARD SEEK TIMES ARE MEASURED.

THIS TEST USES THE SECTOR COUNTER ('SOT') IN THE RP11 AS THE TIME BASE FOR THE MEASUREMENTS, USE OF THE 'SOT' PROVIDES A RESOLUTION OF 2.5 MS.

10. PROGRAM LISTING

1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066

```
%
.TITLE MD-11-DZRPZ-B, RP11E DRIVE POSITIONING TEST
.*COPYRIGHT (C) 1975,1976
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY C. HESS/F. ROEMER
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZGAC-C1),MAR 24, 1976.
.*
```

```
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.*      SWITCH          USE
.*      -----
.*      15             HALT ON ERROR
.*      14             LOOP ON TEST
.*      13             INHIBIT ERROR TYPEOUTS
.*      11             INHIBIT ITERATIONS
.*      10             BELL ON ERROR
.*      9              LOOP ON ERROR
.*      6              TYPE ALL THE RP11 REGISTERS IF ERROR
.*      4              TYPE THE LONG SEEK TIME GRAPH
.*      3              PERFORM STALL BETWEEN SEEK ORDERS
.*      2              USE RANDOM STALL VALUE BETWEEN SEEK ORDERS
.*      1              PERFORM INCREMENTING STALL IN TEST 4
.*      0              DO NOT CHECK POSITION AFTER SEEK ORDERS
```

```
.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 ACT11 HOOKS
.*
.*      *****
.*      ;HOOKS REQUIRED BY ACT11
.*      $SVPC=.          ;SAVE PC
.*      =46             ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
SENDAD
.*      =52             ;;2)SET LOC.52 TO ZERO
.*      .WORD 0
.*      =$SVPC          ;; RESTORE PC
```

```
.SBTTL STARTING ADDRESSES
.*      =200
.*      ;*200 = NORMAL START
.*      JMP      @START1
```

```
000000
000174 000000
000176 000000
000200
000200 000137 002246
```

```

1067      ;#204 = SELECT OPERATING PARAMETERS
1068 000204 000137 002270      JMP      @#START2
1069      ;#210 = SELECT RP11 ADDRESSES
1070 000210 000137 002236      JMP      @#START3
1071      ;#214 = COMBINATION OF 204 AND 210
1072 000214 000137 002260      JMP      @#START4
1073
1074      .SBTTL BASIC DEFINITIONS
1075
1076      ;#INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1077      STACK= 1100
1078      .EQUIV ENT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
1079      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
1080
1081      ;#MISCELLANEOUS DEFINITIONS
1082      AT= 11      ;;CODE FOR HORIZONTAL TAB
1083      LF= 12      ;;CODE FOR LINE FEED
1084      CR= 15      ;;CODE FOR CARRIAGE RETURN
1085      CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
1086      PS= 177776      ;;PROCESSOR STATUS WORD
1087      .EQUIV PS,PSW
1088      STKLMT= 177774      ;;STACK LIMIT REGISTER
1089      PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
1090      DSMR= 177570      ;;HARDWARE SWITCH REGISTER
1091      DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
1092
1093      ;#GENERAL PURPOSE REGISTER DEFINITIONS
1094      R0= %0      ;;GENERAL REGISTER
1095      R1= %1      ;;GENERAL REGISTER
1096      R2= %2      ;;GENERAL REGISTER
1097      R3= %3      ;;GENERAL REGISTER
1098      R4= %4      ;;GENERAL REGISTER
1099      R5= %5      ;;GENERAL REGISTER
1100      R6= %6      ;;GENERAL REGISTER
1101      R7= %7      ;;GENERAL REGISTER
1102      .EQUIV R6,SP      ;;STACK POINTER
1103      .EQUIV R7,PC      ;;PROGRAM COUNTER
1104
1105      ;#PRIORITY LEVEL DEFINITIONS
1106      PRO= 0      ;;PRIORITY LEVEL 0
1107      PR1= 40      ;;PRIORITY LEVEL 1
1108      PR2= 100      ;;PRIORITY LEVEL 2
1109      PR3= 140      ;;PRIORITY LEVEL 3
1110      PR4= 200      ;;PRIORITY LEVEL 4
1111      PR5= 240      ;;PRIORITY LEVEL 5
1112      PR6= 300      ;;PRIORITY LEVEL 6
1113      PR7= 340      ;;PRIORITY LEVEL 7
1114
1115      ;#"SWITCH REGISTER" SWITCH DEFINITIONS
1116      SW15= 100000
1117      SW14= 40000
1118      SW13= 20000
1119      SW12= 10000
1120      SW11= 4000
1121      SW10= 2000
1122      SW09= 1000

```

1123 000400  
1124 000200  
1125 000100  
1126 000040  
1127 000020  
1128 000010  
1129 000004  
1130 000002  
1131 000001

SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

1143 100000  
1144 040000  
1145 020000  
1146 010000  
1147 004000  
1148 002000  
1149 001000  
1150 000400  
1151 000200  
1152 000100  
1153 000040  
1154 000020  
1155 000010  
1156 000004  
1157 000002  
1158 000001

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

1171 000004  
1172 000010  
1173 000014  
1174 000014  
1175 000014  
1176 000014  
1177 000020  
1178 000024

.\*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4  
RESVEC= 10  
TBITVEC= 14  
TRTVEC= 14  
BPTVEC= 14  
IOTVEC= 20  
PWRVEC= 24  
: TIME OUT AND OTHER ERRORS  
: RESERVED AND ILLEGAL INSTRUCTIONS  
: "T" BIT  
: TRACE TRAP  
: BREAKPOINT TRAP (BPT)  
: INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
: POWER FAIL



```

1179      000030      EMTVEC= 30      ;;EMULATOR TRAP (EMT) **ERROR**
1180      000034      TRAPVEC=34      ;;"TRAP" TRAP
1181      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
1182      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
1183      000240      PIRQVEC=240      ;;PROGRAM INTERRUPT REQUEST VECTOR
1184
1185      ;OP CODE DEFINITIONS
1186
1187      000001      CLEAR=1      ;CONTROLLER CLEAR
1188      000003      WRTSEK=3      ;WRITE WITH IMPLIED SEEK AND HEAD SELECTION
1189      000005      RDSEK=5      ;READ WITH IMPLIED SEEK AND HEAD SELECT
1190      000007      WCKSEK=7      ;WRITE CHECK WITH IMPLIED SEEK AND HEAD SELECT
1191      000011      SEEK=11      ;SEEK
1192      000013      WRTSE=13      ;WRITE (NO IMPLIED SEEK OR HEAD SELECT)
1193      000015      HOMSEK=15      ;HOME SEEK
1194      000017      READ=17      ;READ (NO IMPLIED SEEK OR HEAD SELECT)
1195
1196      ;RP11 REGISTER EQUATES
1197
1198      ;RPDS (DRIVE STATUS REGISTER)
1199
1200      ;ATTENTION: BITS ARE REFERENCED BY BIT NUMBER (BIT00 - BIT07)
1201      000400      SUMP=400      ;SELECTED UNIT WRITE PROTECTED
1202      001000      SUFU=1000      ;SELECTED UNIT FILE UNSAFE
1203      002000      SUSU=2000      ;SELECTED UNIT SEEK UNDERWAY
1204      004000      SUSI=4000      ;SELECTED UNIT SEEK INCOMPLETE
1205      010000      HNF=10000      ;HEADER NOT FOUND
1206      020000      SURP03=20000      ;SELECTED UNIT IS AN RPO3
1207      040000      SUOL=40000      ;SELECTED UNIT IS ONLINE
1208      100000      SURDY=100000      ;SELECTED UNIT IS READY
1209
1210      ;RPER (ERROR REGISTER)
1211
1212      000001      DSKERR=1      ;DISK ERROR
1213      000002      EOP=2      ;END OF PACK
1214      000004      NXME=4      ;NON-EXISTENT MEMORY
1215      000010      WCE=10      ;WRITE CHECK ERROR
1216      000020      TIMEE=20      ;TIMING ERROR
1217      000040      CSME=40      ;CHECKSUM ERROR
1218      000100      WPE=100      ;WORD PARITY ERROR
1219      000200      LPE=200      ;LONGITUDINAL PARITY ERROR
1220      000400      MODERR=400      ;MODE ERROR
1221      001000      FMTE=1000      ;FORMAT ERROR
1222      002000      PROG=2000      ;PROGRAM ERROR
1223      004000      NXS=4000      ;NON-EXISTENT SECTOR
1224      010000      NXT=10000      ;NON-EXISTENT TRACK
1225      020000      NXC=20000      ;NON-EXISTENT CYLINDER
1226      040000      FUV=40000      ;FILE UNSAFE VIOLATION
1227      100000      WPV=100000      ;WRITE PROTECT VIOLATION
1228
1229      ;RPCS (CONTROL REGISTER)
1230
1231      ;'GO' BIT AND FUNCTION BITS ARE LOADED TOGETHER AND ARE DEFINED ELSEWHERE.
1232      ;EXTENDED MEMORY ADDRESS BITS ARE REFERED TO BY BIT NUMBER
1233      000100      IDE=100      ;INTERRUPT ON DONE
1234      000200      RDY=200      ;RP11 READY

```

1235		:DRIVE SELECT BITS ARE REFERED TO BY BIT NUMBER	
1236	004000	HDR=4000	:HEADER
1237	010000	MODE=10000	:MODE
1238	020000	AIE=20000	:ATTENTION INTERRUPT ENABLE
1239	040000	HE=40000	:HARD ERROR
1240	100000	ERR=100000	:ERROR
1241			

.SBTTL COMMON TAGS

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

1243			
1244			
1245			
1246			
1247			
1248		001100	
1249	001100		
1250	001100	000000	
1251	001102	000	
1252	001103	000	
1253	001104	000000	
1254	001106	000000	
1255	001110	000000	
1256	001112	000000	
1257	001114	000	
1258	001115	001	
1259	001116	000000	
1260	001120	000000	
1261	001122	000000	
1262	001124	000000	
1263	001126	000000	
1264	001130	000000	
1265	001132	000000	
1266	001134	000	
1267	001135	000	
1268	001136	000000	
1269	001140	177570	
1270	001142	177570	
1271	001144	177560	
1272	001146	177562	
1273	001150	177564	
1274	001152	177566	
1275	001154	000	
1276	001155	002	
1277	001156	012	
1278	001157	000	
1279	001160	000000	
1280			
1281	001162	000000	
1282	001164	000000	
1283	001166	000000	
1284	001170	000000	
1285	001172	000000	
1286	001174	000000	
1287	001176	000000	
1288	001200	000000	
1289	001202	000000	
1290	001204	000000	
1291	001206	000000	
1292	001210	000000	
1293	001212	177607	000377
1294	001216	077	
1295	001217	015	
1296	001220	000012	
1297			

```

.=1100
SCMTAG:
SPASS: .WORD 0
STSTNM: .BYTE 00
SERFLG: .BYTE 000
SICNT: .WORD 000000
SLPADR: .WORD 000000
SLPERR: .WORD 000000
SERTTL: .WORD 000000
SITEMB: .BYTE 000
SERMAX: .BYTE 1
SERRPC: .WORD 000000
SGDADR: .WORD 000000
SBDADR: .WORD 000000
SGDOAT: .WORD 000000
SBD0AT: .WORD 000000
SAUTOB: .BYTE 000
SINTAG: .BYTE 000
SMR: .WORD DSMR
DISPLAY: .WORD DDISP
$TKS: 177560
$TKB: 177562
$TPS: 177564
$TPB: 177566
$NULL: .BYTE 0
$FILLS: .BYTE 2
$FILLC: .BYTE 12
$STPFLG: .BYTE 0
$REGAD: .WORD 0
$REG0: .WORD 0
$REG1: .WORD 00
$REG2: .WORD 00
$REG3: .WORD 00
$REG4: .WORD 00
$REG5: .WORD 00
$REG6: .WORD 00
$REG7: .WORD 00
$REG10: .WORD 00
$TMPO: .WORD 0
$TIMES: 0
$ESCAPE: 0
$BELL: .ASCIZ <207><377><377>
$QUES: .ASCIZ /?/
$CRLF: .ASCIZ <15>
$LF: .ASCIZ <12>

```

```

: 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 (($REGAD)+6)
: CONTAINS (($REGAD)+10)
: CONTAINS (($REGAD)+12)
: CONTAINS (($REGAD)+14)
: CONTAINS (($REGAD)+16)
: CONTAINS (($REGAD)+20)
: USER DEFINED
: MAX. NUMBER OF ITERATIONS
: ESCAPE ON ERROR ADDRESS
: CODE FOR BELL
: QUESTION MARK
: CARRIAGE RETURN
: LINE FEED

```

```

1298 001222 000000 CNTRLC: .WORD 0 ;CONTROL "C" FLAG
1299 001224 000000 BUSADR: .WORD 0 ;GET ADDRESSES FROM THE TTY FLAG (0=NO, -1=YES)
1300 001226 000000 INADR: .WORD 0 ;CONTAINS INNER ADDRESS FOR TIMING TEST
1301 001230 000000 OUTADR: .WORD 0 ;CONTAINS OUTER ADDRESS FOR TIMING TEST
1302 001232 000000 DRVSEL: .WORD 0 ;DRIVES SELECTED FOR TESTING
1303 001234 000777 TSTNMS: .WORD 777 ;RUN TESTS 0-10
1304 001236 000000 MAXCYL: .WORD 0 ;MAXIMUM CYLINDER ADDRESS
1305 001240 000000 DRVTYP: .WORD 0 ;CONTAINS A BIT FOR EACH RPO3 DRIVE
1306 ; ON THE SYSTEM. BIT00=DRIVE 0, ETC.
1307 001242 000000 SPSAV: .WORD 0 ;SAVE THE STACK POINTER HERE
1308 001244 000000 BYPASS: .WORD 0 ;BYPASS CURRENT TEST ADDRESS
1309 001246 000000 CHKDRV: .WORD 0 ;DRIVE UNDER TEST
1310 001250 000000 DRVMSK: .WORD 0 ;DRIVE MASK BIT
1311 001252 000000 DRVBAD: .WORD 0 ;CONTAINS BITS FOR UNSAFE OR OFFLINE DRIVES
1312 001254 000000 CYL.CR: .WORD 0 ;CURRENT CYLINDER
1313 001256 000000 TRK.RD: .WORD 0 ;TRACK READ
1314 001260 000000 SEC.RD: .WORD 0 ;SECTOR READ
1315 001262 000000 CYL.DS: .WORD 0 ;CYLINDER DESIRED
1316 001264 000000 STR20X: .WORD 0 ;START FROM 200, 204, 210, 214 FLAG
1317
1318 ;CONSTANTS STORAGE
1319 001266 176710 RPADR: .WORD 176710 ;RP11 ADDRESS
1320 001270 000254 RPVEC: .WORD 254 ;RP11 VECTOR ADDRESS
1321 001272 000240 RPPRIO: .WORD 5*32. ;RP11 PRIORITY
1322 001274 000000 STALLO: .WORD 0 ;VARIABLE STALL (TEST 4)
1323 001276 000031 STALL1: .WORD 25. ;25 MILLISECONDS STALL (TEST 0-6)
1324 001300 000000 STALLG: .WORD 0 ;GENERAL STALL VALUE (VARIABLE)
1325 001302 000062 MXSTAL: .WORD 50. ;MAX INCREMENTING STALL ALLOWED IN TEST 4
1326 .EVEN
1327
1328 ;#TABLE OF DRIVE STATUS INDICATORS
1329 ;#DRVSTA>0 - DRIVE IS ONLINE
1330 ;#DRVSTA<0 - DRIVE IS OFFLINE, UNSAFE, OR AN RPO3
1331
1332 001304 000 DRVSTA: .BYTE 0 ;DRIVE 0
1333 001305 000 ;DRIVE 1
1334 001306 000 ;DRIVE 2
1335 001307 000 ;DRIVE 3
1336 001310 000 ;DRIVE 4
1337 001311 000 ;DRIVE 5
1338 001312 000 ;DRIVE 6
1339 001313 000 ;DRIVE 7
1340
1341 ;ATTENTION BIT TABLE
1342
1343 001314 001 ATABIT: .BYTE 1 ;DRIVE 0
1344 001315 002 ;DRIVE 1
1345 001316 004 ;DRIVE 2
1346 001317 010 ;DRIVE 3
1347 001320 020 ;DRIVE 4
1348 001321 040 ;DRIVE 5
1349 001322 100 ;DRIVE 6
1350 001323 200 ;DRIVE 7
1351
1352 ;BIT TABLE
1353 001324 000001 BITS: .WORD BIT00

```

1354 001326 000002  
 1355 001330 000004  
 1356 001332 000010  
 1357 001334 000020  
 1358 001336 000040  
 1359 001340 000100  
 1360 001342 000200  
 1361 001344 000400  
 1362 001346 001000  
 1363 001350 002000  
 1364 001352 004000  
 1365 001354 010000  
 1366 001356 020000  
 1367 001360 040000  
 1368 001362 100000

.WORD BIT01  
 .WORD BIT02  
 .WORD BIT03  
 .WORD BIT04  
 .WORD BIT05  
 .WORD BIT06  
 .WORD BIT07  
 .WORD BIT08  
 .WORD BIT09  
 .WORD BIT10  
 .WORD BIT11  
 .WORD BIT12  
 .WORD BIT13  
 .WORD BIT14  
 .WORD BIT15

;DRIVE OPERATION PARAMETER BLOCK

DPB: .BYTE 0 ;OP CODE  
 .BYTE 0 ;DRIVE ADDRESS  
 .WORD 0 ;CYLINDER ADDRESS  
 .BYTE 0 ;SECTOR ADDRESS  
 .BYTE 0 ;TRACK ADDRESS

;COMMON STORAGE FOR TEST PARAMETER

PRM: .WORD 0  
 RPT: .WORD 0 ;REPEAT COUNTS FOR ALL TESTS  
 FC: .WORD 0 ;FIRST CYLINDER  
 LC: .WORD 0 ;LAST CYLINDER  
 IC: .WORD 0 ;INCREMENT CYLINDER  
 TK: .WORD 0 ;TRACK ADDRESS

;TABLE OF PARAMETER POINTERS

PRMPT: .WORD PRM0  
 .WORD PRM1  
 .WORD PRM2  
 .WORD PRM3  
 .WORD PRM4  
 .WORD PRM5  
 .WORD PRM6  
 .WORD PRM7  
 .WORD PRM10  
 .WORD PRM11  
 .WORD PRM12

;PARAMETER UPPER LIMIT

PRMLMT: .WORD 32767. ;"R"  
 .WORD 202. ;"FC"  
 .WORD 202. ;"LC"  
 .WORD 202. ;"IC"  
 .WORD 19. ;"TK"

;TABLE OF MESSAGE POINTERS

PRMMSG: .WORD MSG.R  
 .WORD MSG.FC

1369  
 1370  
 1371  
 1372  
 1373 001364 000  
 1374 001365 000  
 1375 001366 000000  
 1376 001370 000  
 1377 001371 000  
 1378  
 1379  
 1380 001372 000000  
 1381 001374 000000  
 1382 001376 000000  
 1383 001400 000000  
 1384 001402 000000  
 1385 001404 000000  
 1386  
 1387  
 1388 001406 001632  
 1389 001410 001640  
 1390 001412 001654  
 1391 001414 001670  
 1392 001416 001704  
 1393 001420 001720  
 1394 001422 001734  
 1395 001424 001750  
 1396 001426 001764  
 1397 001430 001770  
 1398 001432 002000  
 1399  
 1400  
 1401 001434 077777  
 1402 001436 000312  
 1403 001440 000312  
 1404 001442 000312  
 1405 001444 000023  
 1406  
 1407  
 1408 001446 021370  
 1409 001450 021372

1410 001452 021375  
 1411 001454 021400  
 1412 001456 021403  
 1413  
 1414  
 1415 001460 000021 000144 000000  
 1416 001466 000037 001750 000000  
 1417 001474 000312 000000 000000  
 1418 001502 000037 000010 000000  
 1419 001510 000312 000001 000000  
 1420 001516 000037 000010 000000  
 1421 001524 000200 000001 000000  
 1422 001532 000037 000010 000000  
 1423 001540 000312 000001 000000  
 1424 001546 000037 000010 000000  
 1425 001554 000312 000001 000000  
 1426 001562 000037 000010 000000  
 1427 001570 000312 000001 000000  
 1428 001576 000037 000001 000000  
 1429 001604 000312 000001 000000  
 1430 001612 000001 000001 000000  
 1431 001616 000007 000001 000000  
 1432 001624 000312  
 1433 001626 000001 000001  
 1434  
 1435  
 1436  
 1437  
 1438 001632 000021  
 1439 001634 000144  
 1440 001636 000000  
 1441  
 1442  
 1443 001640 000037  
 1444 001642 001750  
 1445 001644 000000  
 1446 001646 000312  
 1447 001650 000000  
 1448 001652 000000  
 1449  
 1450  
 1451 001654 000037  
 1452 001656 000012  
 1453 001660 000000  
 1454 001662 000312  
 1455 001664 000001  
 1456 001666 000000  
 1457  
 1458  
 1459 001670 000037  
 1460 001672 000012  
 1461 001674 000000  
 1462 001676 000200  
 1463 001700 000001  
 1464 001702 000000  
 1465

:DEFAULT VALUES OF TESTS PARAMETERS

DFLT: .WORD 21,100.,0 ;HOME SEEK (T0)  
 .WORD 37,1000.,0,202.,0,0 ;SEEK/SEEK (T1)  
 .WORD 37,10,0,202.,1,0 ;INCREMENT SEEK (T2)  
 .WORD 37,10,0,128.,1,0 ;STEPPING SEEK (T3)  
 .WORD 37,10,0,202.,1,0 ;OSCILLATING SEEK (T4)  
 .WORD 37,10,0,202.,1,0 ;CONVERGING/DIVERGING SEEK (T5)  
 .WORD 37,10,0,202.,1,0 ;SERVO ADDRESSING LOGIC NOISE (T6)  
 .WORD 37,1,0,202.,1,0 ;CYLINDER ADDRESSING TEST (T7)  
 .WORD 1,1 ;ONE CYLINDER SEEK TIMER TEST (T10)  
 .WORD 7,1,0,202. ;SEEK-SEEK TIMER TEST (T11)  
 .WORD 1,1 ;SEEK TIME GRAPH TEST (T12)

;PARAMETER TABLES

:HOME SEEK (T0)

PRM0: .WORD 21  
 RPT0: .WORD 100.  
 TK0: .WORD 0

:SEEK-SEEK (T1)

PRM1: .WORD 37  
 RPT1: .WORD 1000.  
 FC1: .WORD 0  
 LC1: .WORD 202.  
 IC1: .WORD 0  
 TK1: .WORD 0

:INCREMENT SEEK (T2)

PRM2: .WORD 37  
 RPT2: .WORD 10.  
 FC2: .WORD 0  
 LC2: .WORD 202.  
 IC2: .WORD 1  
 TK2: .WORD 0

:STEPPING SEEK (T3)

PRM3: .WORD 37  
 RPT3: .WORD 10.  
 FC3: .WORD 0  
 LC3: .WORD 128.  
 IC3: .WORD 1  
 TK3: .WORD 0

1466  
1467 001704 000037  
1468 001706 000012  
1469 001710 000000  
1470 001712 000312  
1471 001714 000001  
1472 001716 000000  
1473  
1474  
1475 001720 000037  
1476 001722 000012  
1477 001724 000000  
1478 001726 000312  
1479 001730 000001  
1480 001732 000000  
1481  
1482  
1483 001734 000037  
1484 001736 000012  
1485 001740 000000  
1486 001742 000312  
1487 001744 000001  
1488 001746 000000  
1489  
1490  
1491 001750 000037  
1492 001752 000001  
1493 001754 000000  
1494 001756 000312  
1495 001760 000001  
1496 001762 000000  
1497  
1498  
1499 001764 000001  
1500 001766 000001  
1501  
1502  
1503 001770 000007  
1504 001772 000001  
1505 001774 000000  
1506 001776 000312  
1507  
1508  
1509 002000 000001  
1510 002002 000001  
1511  
1512  
1513  
1514 002004 000000  
1515 002006 000000  
1516 002010 000000  
1517 002012 000000  
1518 002014 000000  
1519 002016 000000  
1520 002020 000000  
1521 002022 000000

:OSCILLATING SEEK (T4)

PRM4: .WORD 37  
RPT4: .WORD 10.  
FC4: .WORD 0  
LC4: .WORD 202.  
IC4: .WORD 1  
TK4: .WORD 0

:CONVERGING/DIVERGING SEEK (T5)

PRM5: .WORD 37  
RPT5: .WORD 10.  
FC5: .WORD 0  
LC5: .WORD 202.  
IC5: .WORD 1  
TK5: .WORD 0

:SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)

PRM6: .WORD 37  
RPT6: .WORD 10.  
FC6: .WORD 0  
LC6: .WORD 202.  
IC6: .WORD 1  
TK6: .WORD 0

:CYLINDER ADDRESSING (T7)

PRM7: .WORD 37  
RPT7: .WORD 1  
FC7: .WORD 0  
LC7: .WORD 202.  
IC7: .WORD 1  
TK7: .WORD 0

:ONE CYLINDER SEEK TIMER TEST (T10)

PRM10: .WORD 1  
RPT10: .WORD 1

:SEEK-SEEK TIMING TEST (T11)

PRM11: .WORD 7  
RPT11: .WORD 1  
FC11: .WORD 0  
LC11: .WORD 202.

:SEEK TIME GRAPH TEST (T12)

PRM12: .WORD 1  
RPT12: .WORD 1

;SAVE THE RPI1E REGISTERS HERE

SRPOS: .WORD 0  
SRPER: .WORD 0  
SRPCS: .WORD 0  
SRPWC: .WORD 0  
SRPBA: .WORD 0  
SRPCA: .WORD 0  
SRPDA: .WORD 0  
SRPM1: .WORD 0

:DRIVE STATUS REGISTER  
:ERROR REGISTER  
:COMMAND & STATUS REGISTER  
:WORD COUNT REGISTER  
:BUFFER ADDRESS REGISTER  
:CURRENT CYLINDER ADDRESS REGISTER  
:TRACK-SECTOR ADDRESS REGISTER  
:MAINTENANCE REGISTER #1

1522	002024	000000		\$SUCA: .WORD	0					
1523	002026	000000		\$SILO: .WORD	0					
1524										
1525										
1526										
1527		000000		RPDS=00						
1528		000002		RPER=02						
1529		000004		RPCS=04						
1530		000006		RPWC=06						
1531		000010		RPBA=10						
1532		000012		RPCA=12						
1533		000014		RPOA=14						
1534		000016		RPM1=16						
1535		000020		RPM2=20						
1536		000022		RPM3=22						
1537		000024		SUCA=24						
1538		000026		SILO=26						
1539										
1540										
1541	002030	020040	040	BLNK13: .ASCII	/	/	/			
1542	002033	040		BLNK10: .ASCII	/	/	/			
1543	002034	040		BLNK9: .ASCII	/	/	/			
1544	002035	040		BLNK8: .ASCII	/	/	/			
1545	002036	040		BLNK7: .ASCII	/	/	/			
1546	002037	040		BLNK6: .ASCII	/	/	/			
1547	002040	040		BLNK5: .ASCII	/	/	/			
1548	002041	040		BLNK4: .ASCII	/	/	/			
1549	002042	040		BLNK3: .ASCII	/	/	/			
1550	002043	040		BLNK2: .ASCII	/	/	/			
1551	002044	000040		BLNK1: .ASCII	/	/	/			
1552										
1553				.EVEN						

;SELECTED UNIT CYLINDER ADDRESS REGISTER  
 ;SILO REGISTER  
 ;REGISTER INDEX VALUES  
 ;DRIVE STATUS REGISTER  
 ;ERROR REGISTER  
 ;CONTROL REGISTER  
 ;WORD COUNT REGISTER  
 ;BUFFER ADDRESS REGISTER  
 ;CYLINDER ADDRESS REGISTER  
 ;SECTOR/TRACK ADDRESS REGISTER  
 ;MAINTENANCE REGISTER #1  
 ;MAINTENANCE REGISTER #2  
 ;MAINTENANCE REGISTER #3  
 ;SELECTED UNIT CYLINDER ADDRESS REGISTER  
 ;SILO REGISTER



1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609

.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
    
```

SERRTB:

;\*EM AND DH ARE ASCIZ MESSAGES, DT IS A STRING OF WORDS THAT POINT TO THE  
;\*DATA TO BE TYPED AND DF IS A STRING OF DATA THAT TELL HOW THE DT WORDS  
;\*ARE TO BE TYPED. IF ANY OF THE POINTERS ARE NOT NEEDED FOR A PARTICULAR  
;\*ERROR IT IS REPLACED WITH A ZERO.  
;\*EACH OF THE ITEMS BELOW REFER TO THE ERROR NUMBER AND INDICATE  
;\*THE INFORMATION THAT WILL BE TYPED WHEN THE ERROR OCCURS.  
;\*UNLESS STATED OTHER ALL NUMBERS ARE OCTAL

;ERROR 1

```

EM1          ;RP11 FAILED TO RESPOND TO ADDRESSING
DH1
DT1
DF1
    
```

;ERROR 2

```

EM2          ;DRIVE UNSAFE
DH2
DT2
DF2
    
```

;ERROR 3

```

EM3          ;DRIVE OFFLINE
DH2
DT2
DF2
    
```

;ERROR 4

```

EM4          ;DISK ERROR DURING TIMING TEST
DH2
DT2
DF2
    
```

;ERROR 5

```

EM5          ;NO INTERRUPT FROM POSITIONING AFTER 1 SECOND
    
```

002046

002046 021612  
002050 022553  
002052 023306  
002054 023442

002056 021657  
002060 022570  
002062 023312  
002064 023446

002066 021674  
002070 022570  
002072 023312  
002074 023446

002076 021712  
002100 022570  
002102 023312  
002104 023446

002106 021751

1610	002110	022570	DH2	
1611	002112	023312	DT2	
1612	002114	023446	DF2	
1613				
1614				;ERROR 6
1615				
1616	002116	022026	EM6	;DISK ERROR AFTER POSITIONING
1617	002120	022570	DH2	
1618	002122	023312	DT2	
1619	002124	023446	DF2	
1620				
1621				;ERROR 7
1622				
1623	002126	022063	EM7	;ATTN BIT NOT SET AFTER POSITIONING
1624	002130	022570	DH2	
1625	002132	023312	DT2	
1626	002134	023446	DF2	
1627				
1628				;ERROR 10
1629				
1630	002136	022136	EM10	;DRIVE OFFLINE AFTER POSITIONING
1631	002140	022570	DH2	
1632	002142	023312	DT2	
1633	002144	023446	DF2	
1634				
1635				;ERROR 11
1636				
1637	002146	022176	EM11	; 'SUCA' NOT CORRECT AFTER POSITIONING
1638	002150	022645	DH11	
1639	002152	023326	DT11	
1640	002154	023452	DF11	
1641				
1642				;ERROR 12
1643				
1644	002156	022243	EM12	;DISK ERROR WHILE VERIFYING POSITION
1645	002160	022732	DH12	
1646	002162	023344	DT12	
1647	002164	023456	DF12	
1648				
1649				;ERROR 13
1650				
1651	002166	022307	EM13	;DISK POSITIONED TO WRONG CYLINDER
1652	002170	023027	DH13	
1653	002172	023364	DT13	
1654	002174	023462	DF13	
1655				
1656				;ERROR 14
1657				
1658	002176	022351	EM14	;NO INTERRUPT FROM I/O AFTER 1 SECOND
1659	002200	022570	DH2	
1660	002202	023312	DT2	
1661	002204	023446	DF2	
1662				
1663				;ERROR 15
1664				
1665	002206	022416	EM15	;SEEK INCOMPLETE

1666 002210 022570  
1667 002212 023312  
1668 002214 023446  
1669  
1670  
1671  
1672 002216 022436  
1673 002220 022570  
1674 002222 023312  
1675 002224 023446  
1676  
1677  
1678  
1679 002226 022500  
1680 002230 022570  
1681 002232 023312  
1682 002234 023446  
1683  
1684

DH2  
DT2  
DF2

;ERROR 16

;DRIVE NOT READY AFTER POSITIONING

EM16  
DH2  
DT2  
DF2

;ERROR 17

;DRIVE NOT READY/OFFLINE DURING TIMING TEST

EM17  
DH2  
DT2  
DF2

```

1685
1686
1687
1688
1689 002236 012737 177777 001224 START3: MOV      8-1,2#BUSADR ;GET BUSADR FLAG
1690 002244 000402          BR          STRT1A
1691 002246 005037 001224 START1: CLR      2#BUSADR ;CLR BUSADR FLAG
1692 002252 005037 001222 STRT1A: CLR      2#CNTRLC ;NO CONTROL "C"
1693 002256 000411          BR          START
1694 002260 012737 177777 001224 START4: MOV      8-1,2#BUSADR ;SET BUSADR FLAG
1695 002266 000402          BR          STRT2A
1696 002270 005037 001224 START2: CLR      2#BUSADR ;CLR BUSADR FLAG
1697 002274 012737 177777 001222 STRT2A: MOV      8-1,2#CNTRLC ;SET CONTROL "C" FLAG
1698 002302 012737 177777 001264 START:  MOV      8-1,STR20X ;SET START FLAG
1699 002310 000005          RESET
1700
1701 .SBTTL INITIALIZE THE COMMON TAGS
1702 ;;CLEAR THE COMMON TAGS ($CNTAG) AREA
1703 002312 012706 001100 MOV      #SCNTAG,R6 ;:FIRST LOCATION TO BE CLEARED
1704 002316 005026          CLR      (R6)+ ;:CLEAR MEMORY LOCATION
1705 002320 022706 001140 CMP      #SMR,R6 ;:DONE?
1706 002324 001374          BNE     -6 ;:LOOP BACK IF NO
1707 002326 012706 001100 MOV      #STACK,SP ;:SETUP THE STACK POINTER
1708 ;;INITIALIZE A FEW VECTORS
1709 002332 012737 012502 000020 MOV      #SSCOPE,2#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
1710 002340 012737 000340 000022 MOV      #340,2#IOTVEC+2 ;:LEVEL 7
1711 002346 012737 007732 000030 MOV      #ERROR,2#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
1712 002354 012737 000340 000032 MOV      #340,2#EMTVEC+2 ;:LEVEL 7
1713 002362 012737 013134 000034 MOV      #STRAP,2#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
1714 002370 012737 000340 000036 MOV      #340,2#TRAPVEC+2 ;:LEVEL 7
1715 002376 005037 001206          CLR      $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
1716 002402 005037 001210          CLR      $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
1717 002406 112737 000001 001115 MOVB     #1,$SERMAX ;:ALLOW ONE ERROR PER TEST
1718 002414 012737 002414 001106 MOV      #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
1719 002422 012737 002422 001110 MOV      #,$SLPERR ;:SETUP THE ERROR LOOP ADDRESS
1720 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1721 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1722 002430 013746 000004          MOV      2#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
1723 002434 012737 002470 000004 MOV      #64,$ERRVEC ;:SET UP ERROR VECTOR
1724 002442 012737 177570 001140 MOV      #DSMR,SMR ;:SETUP FOR A HARDWARE SWICH REGISTER
1725 002450 012737 177570 001142 MOV      #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
1726 002456 022777 177777 176454 CMP      8-1,$SMR ;:TRY TO REFERENCE HARDWARE SMR
1727 002464 001012          BNE     66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
1728 002466 000403          BR          65$ ;:AND THE HARDWARE SMR IS NOT = -1
1729 002470 012716 002476          BR          64$: MOV      #65,$(SP) ;:BRANCH IF NO TIMEOUT
1730 002474 000002          RTI ;:SET UP FOR TRAP RETURN
1731 002476 012737 000176 001140 65$: MOV      #SMREG,SMR ;:POINT TO SOFTWARE SMR
1732 002504 012737 000174 001142 MOV      #DISPREG,DISPLAY
1733 002512 012637 000004          66$: MOV      (SP)+,2#ERRVEC ;:RESTORE ERROR VECTOR
1734
1735 002516 012700 001160          MOV      #SREGAD,RO ;:FIRST ADDRESS
1736 002522 005020          1$: CLR      (RO)+ ;:CLEAR VARIABLE STORAGE
1737 002524 022700 001212          CMP      #SBELL,RO ;:DONE?
1738 002530 001374          BNE     1$ ;:NO--BRANCH
1739 002532 005227 177777          INC     8-1 ;:FIRST START ?
1740 002536 001032          BNE     2$ ;:BR IF NOT

```

1741	002540	104401	002546		TYPE	685	::TYPE ASCIZ STRING
1742	002544	000427			BR	675	::GET OVER THE ASCIZ
1743				::685:	.ASCIZ	<15><12>/MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST/	
1744	002624			675:			
1745	002624	004737	011232	25:	JSR	PC,STKINT	::SETUP THE TTY KEYBOARD
1746				.SBTTL	GET	VALUE FOR SOFTWARE SWITCH REGISTER	
1747	002630	005737	000042		TST	3#42	::ARE WE RUNNING UNDER XXDP/ACT?
1748	002634	001006			BNE	695	::BRANCH IF YES
1749	002636	023727	001140	000176	CMP	SWR,#SWREG	::SOFTWARE SWITCH REG SELECTED?
1750	002644	001005			BNE	705	::BRANCH IF NO
1751	002646	104406			GTSWR		::GET SOFT-SWR SETTINGS
1752	002650	000403			BR	705	
1753	002652	112737	000001	001134	695:	MOV8	#1,SAUTO8
1754	002660				705:		::SET AUTO-MODE INDICATOR
1755	002660	012737	000377	001234	MOV	#377,TSTNMS	::SELECT TESTS 0-7
1756	002666	012700	001460		MOV	#DFLT,RO	::DEFAULT PARAMETERS POINTER
1757	002672	012701	001632		MOV	#PRND,R1	::TABLE POINTER
1758	002676	010102			MOV	R1,R2	::STOP ADDRESS
1759	002700	012021		35:	MOV	(R0)+,(R1)+	::MOVE DEFAULT PARAMETERS INTO
1760	002702	020002			CMP	RO,R2	::RUN TIME TABLES ** DONE?
1761	002704	103775			BLO	35	::NO--BRANCH
1762	002706	000406			BR	SRTINT	::FINISH SETUP
1763	002710	000005			STARTS:	RESET	::'CONTROL C' RESET
1764	002712	004737	011232		JSR	PC,STKINT	
1765	002716	012737	177777	001222	MOV	#-1,CNTRLC	::SET 'CONTROL C' SWITCH
1766	002724	012737	000340	000062	SRTINT:	MOV	#PR7,TKVEC+2
1767	002732	005037	177776		CLR	PS	::SET PRIORITY TO ZERO
1768	002736	005037	001252		CLR	DRVBD	::CLEAR OFFLINE/UNSAFE DRIVE BITS
1769	002742	005037	001206		CLR	STIMES	::INITIALIZE NUMBER OF ITERATIONS
1770	002746	005037	001210		CLR	SESCAPE	::CLEAR THE ESCAPE ON ERROR ADDRESS
1771	002752	005037	001244		CLR	BYPASS	::CLEAR THE BYPASS ADDRESS
1772	002756	012737	000001	001104	MOV	#1,SICNT	::PRESET ITERATION COUNT TO 1
1773	002764	112737	000001	001115	MOV8	#1,SERMAX	::ALLOW ONE ERROR PER TEST
1774	002772	012737	002772	001106	MOV	#.,SLPADR	::INITIALIZE THE LOOP ADDRESS FOR SCOPE
1775	003000	012737	003000	001110	MOV	#.,SLPERR	::SETUP THE ERROR LOOP ADDRESS
1776	003006	004737	017226		JSR	PC,GETADR	::CHECK RPII ADDRESS
1777	003012	004737	013612		JSR	PC,RPINIT	::FIND OUT WHICH DRIVES ARE ON SYSTEM
1778	003016	012737	000312	001236	MOV	#202,MAXCYL	::ASSUME RPO2'S
1779	003024	005737	001240		TST	DRVTYP	::WHICH DRIVES ?
1780	003030	001403			BEQ	15	::BR IF THEY REALLY WERE RPO2'S
1781	003032	012737	000625	001236	MOV	#405,MAXCYL	::SET MAX CYLINDER FOR RPO3'S
1782	003040	013737	001236	001436	15:	MOV	MAXCYL,PRMLMT+2
1783	003046	013737	001236	001440	MOV	MAXCYL,PRMLMT+4	::UPDATE 'LC' LIMIT
1784	003054	005737	001264		TST	STR20X	::COMING FROM A 'CONTROL C' RESTART ?
1785	003060	001002			BNE	95	::BR IF NOT
1786	003062	000137	003432		JMP	START6	::BYPASS 'LC' UPDATE AND DRIVE STATUS TYPEOUT
1787	003066				95:		
1788	003066	013737	001236	001646	MOV	MAXCYL,LC1	::SETUP MAX CYLINDER FOR TEST 1
1789	003074	013737	001236	001662	MOV	MAXCYL,LC2	::SETUP MAX CYLINDER FOR TEST 2
1790	003102	013737	001236	001712	MOV	MAXCYL,LC4	::SETUP MAX CYLINDER FOR TEST 4
1791	003110	013737	001236	001726	MOV	MAXCYL,LC5	::SETUP MAX CYLINDER FOR TEST 5
1792	003116	013737	001236	001742	MOV	MAXCYL,LC6	::SETUP MAX CYLINDER FOR TEST 6
1793	003124	013737	001236	001756	MOV	MAXCYL,LC7	::SETUP MAX CYLINDER FOR TEST 7
1794	003132	013737	001236	001776	MOV	MAXCYL,LC11	::SETUP MAX CYLINDER FOR TEST 11
1795	003140	012737	000200	001676	MOV	#128,LC3	::ASSUME RPO2
1796	003146	005737	001240		TST	DRVTYP	::RPO2 OR RPO3 ?

```

1797 003152 001403          BEQ      25          ;BR IF RPO2'S
1798 003154 012737 000400 001676  MOV      #256.,LC3  ;CHANGE VALUE FOR RPO3
1799 003162 005227 177777      25:     INC      #1      ;FIRST START ?
1800 003166 001404          BEQ      105         ;BR IF IT IS
1801 003170 032777 000200 175742  BIT      #SW07,SWR   ;INHIBIT THE STATUS TYPEOUT
1802 003176 001115          BNE      START6     ;BR IF YES
1803 003200          105:     TYPE      655         ;;TYPE ASCIZ STRING
1804 003200 104401 003206      BR       645         ;;GET OVER THE ASCIZ
1805 003204 000412          ;;655:  .ASCIZ  <15><12><12>/DRIVE STATUS:/<15><12><12>
1806          645:
1807 003232          CLR      R1         ;CLEAR TABLE POINTER
1808 003232 005001          MOV      #8.,R2     ;COUNTER
1809 003234 012702 000010      35:     MOV      R1,-(SP) ;;SAVE R1 FOR TYPEOUT
1810 003240          ;;TYPE DRIVE NUMBER
1811 003240 010146          ;;GO TYPE--OCTAL ASCII
1812          ;;TYPE 1 DIGIT(S)
1813 003242 104403          .BYTE    1         ;SUPPRESS LEADING ZEROS
1814 003244      001          .BYTE    0         ;CHECK DRIVE'S STATUS
1815 003245      000          TSTB    DRVSTA(R1) ;DRIVE IS OFFLINE OR NON-EXISTENT
1816 003246 105761 001304          BEQ      75         ;DRIVE IS UNSAFE
1817 003252 001450          BMI      45         ;TYPE ASCIZ STRING
1818 003254 100412          TYPE      675         ;GET OVER THE ASCIZ
1819 003256 104401 003264      BR       665
1820 003262 000406          ;;675:  .ASCIZ  / ONLINE/
1821          665:
1822 003300          BR       55         ;
1823 003300 000411          45:     TYPE      695         ;;TYPE ASCIZ STRING
1824 003302          BR       685         ;;GET OVER THE ASCIZ
1825 003302 104401 003310      ;;695:  .ASCIZ  / UNSAFE/
1826 003306 000406          685:
1827          55:     BITB    ATABIT(R1),DRV TYP ;SEE WHICH DRIVE TYPE
1828 003324          BNE      65         ;BR IF RPO3
1829 003324 136137 001314 001240  TYPE      715         ;TYPE ASCIZ STRING
1830 003332 001010          BR       705         ;GET OVER THE ASCIZ
1831 003334 104401 003342      ;;715:  .ASCIZ  / RPO2/
1832 003340 000404          705:
1833          BR       85         ;
1834 003352          65:     TYPE      735         ;;TYPE ASCIZ STRING
1835 003352 000422          BR       725         ;;GET OVER THE ASCIZ
1836 003354          ;;735:  .ASCIZ  / RPO3/
1837 003354 104401 003362      725:
1838 003360 000404          BR       85         ;
1839          75:     TYPE      755         ;;TYPE ASCIZ STRING
1840 003372          BR       745         ;;GET OVER THE ASCIZ
1841 003372 000412          ;;755:  .ASCIZ  / OFFLINE/
1842 003374          745:
1843 003374 104401 003402          BR       85         ;
1844 003400 000407          85:     TYPE      85         ;CR-LF
1845          INC      R1         ;INCREMENT TABLE POINTER
1846 003420          DEC      R2         ;DECREMENT THE COUNTER
1847 003420 104401 001217          BNE      35         ;CONTINUE
1848 003424 005201          START6: CLR     STR20X ;CLEAR MANUAL START FLAG
1849 003426 005302          TST     CNTRLC     ;CONTROL "C" START/RESTART?
1850 003430 001303
1851 003432 005037 001264
1852 003436 005737 001222
    
```

1853	003442	001403			BEQ	15		:NO--BRANCH
1854	003444	004737	017560		JSR	PC,2#GT.PRM		:YES--GET PARAMETERS
1855	003450	000416			BR	55		
1856	003452	005037	001232		CLR	DRVSEL		:NO DRIVES SELECTED
1857	003456	005000		15:	CLR	R0		:DETERMINE THE DRIVES THAT
1858	003460	012701	000001		MOV	#1,R1		:ARE AVAILABLE FOR TESTING
1859	003464	105760	001304		TSTB	DRVSTA(R0)		
1860	003470	003403		25:	BLE	35		
1861	003472	156037	001314	001232	BISB	ATABIT(R0),2#DRVSEL		
1862	003500	005200			INC	R0		
1863	003502	106301		35:	ASLB	R1		
1864	003504	001367			BNE	25		
1865	003506			55:				
1866	003506	104401	003514		TYPE	,655		::TYPE ASCIZ STRING
1867	003512	000415			BR	645		::GET OVER THE ASCIZ
1868								
1869	003546							
1870	003546	005037	007646		.ASCIZ	<15><12>/DRIVE(S) TO BE TESTED /		
1871	003552	005000		645:				
1872	003554	013701	001232		CLR	2#SENDCT		:DETERMINE PASSES TO MAKE AND
1873	003560	001010			CLR	R0		:THE DRIVES TO BE TESTED
1874	003562	104401	003570		MOV	2#DRVSEL,R1		:ANY DRIVES SELECTED?
1875	003566	000403			BNE	95		:YES--BRANCH
1876					TYPE	,675		:TYPE ASCIZ STRING
1877	003576				BR	665		:GET OVER THE ASCIZ
1878	003576	000137	007464		.ASCIZ	/NONE/		
1879	003602	005737	001222					
1880	003606	001010		665:	JMP	2#SEOP		:GO TO END OF PROGRAM
1881	003610	005737	000042		TST	CNTRLC		:CONTROL "C" START/RESTART ?
1882	003614	001405			BNE	65		:BR IF NOT
1883	003616	122737	000007	000041	TST	42		:UNDER MONITOR CONTROL ?
1884	003624	001001			BEQ	65		:BR IF NOT
1885	003626	006201			CMPB	#7,41		:LOADED BY RPO2/RPO3 ?
1886	003630	006201			BNE	65		:BR IF NOT
1887	003632	103013			ASR	R1		:EXCLUDE DRIVE 0 FROM TESTING
1888	003634	005237	007646		ASR	R1		:REPORT THE DRIVES TO BE TESTED
1889	003640	010046		65:	BCC	75		
1890	003642	104403			INC	2#SENDCT		:GIVE THIS DRIVE A PASS
1891	003644	001			MOV	R0,-(SP)		:SAVE R0 FOR TYPEOUT
1892	003645	000			TYPOS			:GO TYPE--OCTAL ASCII
1893	003646	005701			.BYTE	1		:TYPE 1 DIGIT(S)
1894	003650	001406			.BYTE	0		:SUPPRESS LEADING ZEROS
1895	003652	104401	003660		TST	R1		:MORE DRIVES?
1896	003656	000401			BEQ	85		:NO--BRANCH
1897					TYPE	,695		:TYPE ASCIZ STRING
1898	003662				BR	685		:GET OVER THE ASCIZ
1899	003662	005200			.ASCIZ	,		
1900	003664	000761		685:				
1901	003666	013737	007646	007640	75:	INC	R0	:INCREMENT DRIVE NUMBER
1902	003674	005037	001246		BR	65		
1903	003700	012737	000001	001250	MOV	2#SENDCT,2#SEOPCT		
1904	003706	033737	001250	001232	RSTRT1:	CLR	CHKDRV	:INIT. THE CHECK DRIVE KEY
1905	003714	001011			MOV	#1,DRVMSK		:START TO CHECK DESIRED DRIVES
1906	003716	005237	001246		RSTRT2:	BIT	DRVMSK,DRVSEL	:IS THIS DRIVE SELECTED?
1907	003722	106337	001250		RSTRT3:	BNE	DRVOK	:YES--GO CHECK IF DRIVE IS READY FOR TESTING
1908	003726	103367			RESTART:	INC	CHKDRV	:MOVE TO NEXT DRIVE NUMBER
						ASLB	DRVMSK	:POSITION THE MASK
						BCC	RSTRT2	:BR IF MORE DRIVES

1909	003730	043737	001252	001232	BIC	DRVBAD,DRVSEL	::CLEAR SELECTION BITS FOR ANY OFFLINE/UNSAFE
1910							::DRIVES
1911	003736	000756			BR	RSTART1	::CONTINUE WITH CYCLE
1912	003740	013702	001246		DRVOK: MOV	2#CHKDRV,R2	::PICKUP THE DRIVE NUMBER
1913	003744	110237	001365		MOV	R2,DPB+1	::LOAD DRIVE NUMBER
1914	003750	104401	003756		TYPE	655	::TYPE ASCIZ STRING
1915	003754	000411			BR	645	::GET OVER THE ASCIZ
1916					::655: .ASCIZ	<15><12><12>/TESTING DRIVE /	
1917	004000				645: MOV	R2,-(SP)	::SAVE R2 FOR TYPEOUT
1918	004000	010246			TYPOS		::GO TYPE--OCTAL ASCII
1919	004002	104403			.BYTE	1	::TYPE 1 DIGIT(S)
1920	004004	001			.BYTE	0	::SUPPRESS LEADING ZEROS
1921	004005	000			TYPE	SCRLF	
1922	004006	104401	001217		MOV	RPADR,RO	::RP11 ADDRESS
1923	004012	013700	001266		MOV	RPCA(RO),CYL.CR	::INITIALIZE CURRENT STORAGE
1924	004016	016037	000012	001254			





1981	004126	113737	001636	001371
1982	004134	012706	001100	
1983	004140	004737	014006	
1984	004144	000004		

```

TESTO:  MOVB   TK0,DPB+5      ;LOAD USER SPECIFIED TRACK
        MOV    #STACK,SP     ;SET UP STACK POINTER
        JSR    PC,SEEKS      ;DO THE SPECIFIED SEEK
EXITO:  SCOPE                ;LOOP

```

1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994

```

;*****
;TEST 1      SEEK/SEEK TEST

```

```

;*      THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SEEK
;*      CYCLE TO "LC", "LT", "LS" FOLLOWED BY A REVERSE SEEK CYCLE TO
;*      "FC", "FT", "FS".  AT THE COMPLETION OF EACH SEEK, THE PROPER
;*      INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

```

1995	004146			
1996	004146	033737	001326	001234
1997	004154	001002		
1998	004156	000137	004304	
1999	004162	013737	001642	001206
2000	004170	012737	004252	001106
2001	004176	012737	004146	001110
2002	004204	012737	000001	001102

```

;*****
;TST1:
        BIT    @#BITS+(1*2),TSTNMS ;DO THIS TEST?
        BNE   64$                ;YES--BRANCH
        JMP   TST2                ;NO--GO TO THE NEXT TEST
64$:   MOV    @#RPT1,$TIMES        ;GET THE ITERATION COUNT
        MOV    #TEST1,@#SLPADR
        MOV    #TST1,@#SLPERR     ;SETUP THE ERROR LOOP ADDRESS
        MOV    #1,@#TSTNM        ;SET UP TEST NUMBER AND
        ;CLEAR THE ERROR FLAG ($ERFLG)
        MOV    $TSTNM,@DISPLAY   ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
        MOV    #EXIT1,BYPASS     ;SETUP BYPASS ADDRESS
        MOV    RPADR,R0          ;RP11 ADDRESS
        MOVB   #SEEK,DPB         ;LOAD SEEK OPCODE
        CLR    DPB+4            ;CLEAR SECTOR/TRACK ADDRESS
        MOVB   TK1,DPB+5        ;LOAD USER SUPPLIED TRACK ADDRESS
TEST1: MOV    #STACK,SP         ;SET THE STACK POINTER
        MOV    FC1,DPB+2        ;STARTING CYLINDER
        JSR    PC,SEEKS         ;SEEK TO FC
        MOV    LC1,DPB+2        ;ENDING CYLINDER
        JSR    PC,SEEKS         ;SEEK TO LC
EXIT1:  SCOPE                    ;LOOP

```

2003				
2004	004212	013777	001102	174722
2005	004220	012737	004302	001244
2006	004226	013700	001266	
2007	004232	112737	000011	001364
2008	004240	005037	001370	
2009	004244	113737	001652	001371
2010	004252	012706	001100	
2011	004256	013737	001644	001366
2012	004264	004737	014006	
2013	004270	013737	001646	001366
2014	004276	004737	014006	
2015	004302	000004		

```

;*****
;TEST 2      INCREMENTAL SEEK TEST

```

```

;*      THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
;*      CYLINDER ADDRESS FROM "FC" TO "LC" BY THE INCREMENT "IC".
;*      WHEN THE RESULTANT CYLINDER ADDRESS (NC) EXCEEDS
;*      "LC" REVERSE SEEK CYCLES ARE INITIATED; STARTING
;*      AT THE LAST LEGAL "NC" AND DECREMENTING BY "IC"
;*      UNTIL "NC" IS LESS THAN "FC".  AT THE COMPLETION OF EACH
;*      SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
;*      INSURE PROPER OPERATION.

```

2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029

2030	004304			
2031	004304	033737	001330	001234
2032	004312	001002		
2033	004314	000137	004476	
2034	004320	013737	001656	001206
2035	004326	012737	004410	001106
2036	004334	012737	004304	001110

```

;*****
;TST2:
        BIT    @#BITS+(2*2),TSTNMS ;DO THIS TEST?
        BNE   64$                ;YES--BRANCH
        JMP   TST3                ;NO--GO TO THE NEXT TEST
64$:   MOV    @#RPT2,$TIMES        ;GET THE ITERATION COUNT
        MOV    #TEST2,@#SLPADR
        MOV    #TST2,@#SLPERR     ;SETUP THE ERROR LOOP ADDRESS

```

```

2037 004342 012737 000002 001102      MOV      #2,2#STSTNM      ;SET UP TEST NUMBER AND
2038                                     ;CLEAR THE ERROR FLAG (SERFLG)
2039 004350 013777 001102 174564      MOV      STSTNM,2DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2040 004356 012737 004474 001244      MOV      #EXIT2,BYPASS  ;SETUP BYPASS ADDRESS
2041 004364 013700 001266                                     MOV      RPADR,R0        ;RP11 ADDRESS
2042 004370 112737 000011 001364      MOVVB   #SEEK,DPB       ;LOAD SEEK OPCODE
2043 004376 005037 001370                                     CLR      DPB+4          ;CLEAR SECTOR/TRACK ADDRESS
2044 004402 113737 001666 001371      MOVVB   TK2,DPB+5       ;LOAD USER SUPPLIED TRACK ADDRESS
2045 004410 012706 001100 TEST2:  MOV      #STACK,SP      ;SET UP THE STACK POINTER
2046 004414 013737 001660 001366      MOV      FC2,DPB+2      ;STARTING CYLINDER
2047 004422 004737 014006 INCSK:  JSR      PC,SEEKS      ;SEEK TO FC
2048 004426 063737 001664 001366      ADD      IC2,DPB+2      ;MOVE TO NEXT CYLINDER
2049 004434 023737 001662 001366      CMP      LC2,DPB+2      ;OUT OF CYLINDERS?
2050 004442 002367                                     BGE      INCSK          ;NO--BRANCH
2051 004444 013737 001662 001366      MOV      LC2,DPB+2      ;LOAD LC
2052 004452 004737 014006 DECSK:  JSR      PC,SEEKS      ;SEEK TO LC
2053 004456 163737 001664 001366      SUB      IC2,DPB+2
2054 004464 023737 001660 001366      CMP      FC2,DPB+2
2055 004472 003767                                     BLE      DECSK
2056 004474 000004 EXIT2:  SCOPE                                     ;LOOP
2057
2058                                     ;*****
2059                                     ;*TEST 3          STEPPING SEEK TEST
2060
2061                                     ;*
2062                                     ;* THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0,1,2,4,
2063                                     ;* 8, 16, 32, 64, AND 128 (AND 256 IF AN RPO3). AT THE
2064                                     ;* COMPLETION OF EACH SEEK COMMAND, THE PROPER INDICATORS ARE
2065                                     ;* EXAMINED TO VERIFY PROPER OPERATION.
2066                                     ;*****
2067 004476                                     †ST3:
2068 004476 033737 001332 001234      BIT      2#BITS+(3*2),TSTNMS ;DO THIS TEST?
2069 004504 001002                                     BNE      64$           ;YES--BRANCH
2070 004506 000137 004646                                     JMP      TST4          ;NO--GO TO THE NEXT TEST
2071 004512 013737 001672 001206 64$:  MOV      2#RPT3,$TIMES  ;GET THE ITERATION COUNT
2072 004520 012737 004602 001106      MOV      #TEST3,2#SLPADR
2073 004526 012737 004476 001110      MOV      #TST3,2#SLPERR  ;SETUP THE ERROR LOOP ADDRESS
2074 004534 012737 000003 001102      MOV      #3,2#STSTNM    ;SET UP TEST NUMBER AND
2075                                     ;CLEAR THE ERROR FLAG (SERFLG)
2076 004542 013777 001102 174372      MOV      STSTNM,2DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2077 004550 012737 004644 001244      MOV      #EXIT3,BYPASS  ;SETUP BYPASS ADDRESS
2078 004556 013700 001266                                     MOV      RPADR,R0        ;RP11 ADDRESS
2079 004562 112737 000011 001364      MOVVB   #SEEK,DPB       ;LOAD SEEK OPCODE
2080 004570 005037 001370                                     CLR      DPB+4          ;CLEAR SECTOR/TRACK ADDRESS
2081 004574 113737 001702 001371      MOVVB   TK3,DPB+5       ;LOAD USER SUPPLIED TRACK ADDRESS
2082 004602 012706 001100 TEST3:  MOV      #STACK,SP      ;SET UP THE STACK
2083 004606 013737 001674 001366      MOV      FC3,DPB+2      ;FC
2084 004614 004737 014006 1$:      JSR      PC,SEEKS      ;SEEK TO FC
2085 004620 013701 001700                                     MOV      IC3,R1         ;CYLINDER 1
2086 004624 010137 001366                                     MOV      R1,DPB+2      ;DESIRED CYLINDER
2087 004630 004737 014006                                     JSR      PC,SEEKS      ;SEEK TO NC
2088 004634 006301                                     ASL      R1             ;MOVE TO NEXT CYLINDER
2089 004636 020137 001676                                     CMP      R1,LC3        ;DONE?
2090 004642 003770                                     BLE      1$            ;NO--LOOP
2091 004644 000004 EXIT3:  SCOPE
2092

```

```

2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103 004646
2104 004646 033737 001334 001234
2105 004654 001002
2106 004656 000137 005210
2107 004662 013737 001706 001206
2108 004670 012737 004766 001106
2109 004676 012737 004646 001110
2110 004704 012737 000004 001102
2111
2112 004712 013777 001102 174222
2113 004720 012737 005206 001244
2114 004726 013700 001266
2115 004732 112737 000011 001364
2116 004740 005037 001370
2117 004744 113737 001716 001371
2118 004752 005002
2119 004754 032777 000002 174156
2120 004762 001401
2121 004764 005102
2122 004766 012706 001100
2123 004772 013701 001710
2124 004776 005037 001274
2125 005002 010137 001366
2126 005006 004737 014006
2127 005012 005702
2128 005014 001403
2129 005016 004037 015546
2130 005022 001274
2131 005024 013737 001710 001366
2132 005032 004737 014006
2133 005036 005702
2134 005040 001413
2135 005042 004037 015546
2136 005046 001274
2137 005050 005237 001274
2138 005054 023737 001302 001274
2139 005062 003347
2140 005064 005037 001274
2141 005070 063701 001714
2142 005074 020137 001712
2143 005100 003740
2144 005102 013701 001712
2145 005106 010137 001366
2146 005112 004737 014006
2147 005116 005702
2148 005120 001403

```

```

*****
:TEST 4 OSCILLATING SEEK TEST
*****
: THIS TEST WILL COMMAND SEEK CYCLES FROM "FC" TO "NC" AND BACK
: TO "FC". "NC" STARTS AT "FC" AND INCREMENTS BY "IC" UP TO CYLINDER
: "LC", THEN IS DECREMENTED BY "IC" BACK TO CYLINDER "FC". AT THE
: COMPLETION OF EVERY SEEK COMMAND THE PROPER INDICATORS ARE
: EXAMINED TO VERIFY PROPER OPERATION.
*****
TST4:
BIT 2#BITS+(4*2),TSTNMS ;DO THIS TEST?
BNE 64$ ;YES--BRANCH
JMP TST5 ;NO--GO TO THE NEXT TEST
64$: MOV 2#RPT4,$TIMES ;GET THE ITERATION COUNT
MOV #TEST4,2#SLPADR
MOV #TST4,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #4,2#$TSTNM ;SET UP TEST NUMBER AND
;CLEAR THE ERROR FLAG (SERFLG)
MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
MOV #EXIT4,BYPASS ;SETUP BYPASS ADDRESS
MOV RPADR,R0 ;RP11 ADDRESS
MOV #SEEK,DPB ;LOAD SEEK OPCODE
CLR DPB+4 ;CLEAR SECTOR/TRACK ADDRESS
MOV #TK4,DPB+5 ;LOAD USER SUPPLIED TRACK ADDRESS
CLR R2 ;CLEAR STALL SWITCH (NO STALL)
BIT #SW1,2#SWR ;STALL REQUIRED?
BEQ TEST4 ;NO--BRANCH
COM R2 ;YES--SET SWITCH
TEST4: MOV #STACK,SP ;SET UP THE STACK POINTER
MOV FC4,R1 ;SET NC TO FC
CLR STALLO ;START AT ZERO IF STALLS REQUIRED
1$: MOV R1,DPB+2 ;NC
JSR PC,SEEKS ;SEEK TO NC
TST R2 ;STALL?
BEQ 2$ ;NO--BRANCH
JSR R0,STALL ;YES--GO TO STALL ROUTINE
.WORD STALLO ;TIME POINTER
2$: MOV FC4,DPB+2 ;FC
JSR PC,SEEKS ;SEEK TO FC
TST R2 ;STALL?
BEQ 3$ ;NO--BRANCH
JSR R0,STALL ;YES--GO TO STALL ROUTINE
.WORD STALLO ;TIME POINTER
INC STALLO ;UPDATE THE TIME
CMP MXSTAL,STALLO ;TIME TO BIG?
BGT 1$ ;NO--BRANCH
CLR STALLO ;YES--START OVER AT ZERO
3$: ADD IC4,R1 ;MOVE TO NEXT CYLINDER
CMP R1,LC4 ;LAST CYLINDER COMPLETED?
BLE 1$ ;NO--BRANCH
MOV LC4,R1 ;SET NC TO LC
4$: MOV R1,DPB+2 ;NC
JSR PC,SEEKS ;SEEK TO NC
TST R2 ;STALL?
BEQ 5$ ;NO--BRANCH

```

```

2149 005122 004037 015546 JSR RD,STALL ;YES--GO TO STALL ROUTINE
2150 005126 001274 .WORD STALLO ;TIME POINTER
2151 005130 013737 001712 001366 5S: MOV LC4,DPB+2 ;LC
2152 005136 004737 014006 JSR PC,SEEKS ;SEEK TO LC
2153 005142 005702 TST R2 ;STALL?
2154 005144 001413 BEQ 6S ;NO--BRANCH
2155 005146 004037 015546 JSR RD,STALL ;YES--GO TO STALL ROUTINE
2156 005152 001274 .WORD STALLO ;TIME POINTER
2157 005154 005237 001274 INC STALLO ;UPDATE STALL TIME
2158 005160 023737 001302 001274 CMP MXSTAL,STALLO ;TIME TOO BIG?
2159 005166 003347 BGT 4S ;NO--BRANCH
2160 005170 005037 001274 CLR STALLO ;YES--SET STALL TIME BACK TO ZERO
2161 005174 163701 001714 6S: SUB IC4,R1 ;NEXT CYLINDER
2162 005200 020137 001710 CMP R1,FC4 ;DONE?
2163 005204 002373 BGE 6S ;NO--BRANCH
2164 005206 000004 EXIT4: SCOPE ;LOOP

```

```

*****
;TEST 5 CONVERGING/DIVERGING SEEK TEST

```

```

;* THIS TEST WILL CAUSE THE DRIVE TO EXECUTE FORWARD AND REVERSE
;* SEEKS FROM "NC1" AND "NC2" RESPECTIVELY, "NC1" WILL BE INCREMENTED
;* BY "IC" AND "NC2" WILL BE DECREMENTED BY "IC" UNTIL "NC1" IS
;* GREATER THAN THE INITIAL VALUE OF "NC2" AND "NC2" IS
;* LESS THAN THE INITIAL VALUE OF "NC1". AT THE COMPLETION OF
;* EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO
;* VERIFY PROPER OPERATION. "NC1" AND "NC2" DEFAULT TO
;* "FC" AND "LC" RESPECTIVELY.

```

```

*****
;TESTS:

```

```

2179 005210 033737 001336 001234 BIT 2#BITS+(5*2),TSTNMS ;DO THIS TEST?
2180 005210 001002 BNE 64S ;YES--BRANCH
2181 005216 000137 005376 JMP TST6 ;NO--GO TO THE NEXT TEST
2182 005220 013737 001722 001206 64S: MOV 2#RPTS,$TIMES ;GET THE ITERATION COUNT
2183 005224 012737 005314 001106 MOV #TESTS,2#SLPADR
2184 005232 012737 005210 001110 MOV #TSTS,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
2185 005240 012737 000005 001102 MOV #5,2#STSTNM ;SET UP TEST NUMBER AND
2186 ;CLEAR THE ERROR FLAG (SERFLG)
2187 MOV $TSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2188 ;SETUP BYPASS ADDRESS
2189 005262 012737 005374 001244 MOV #EXITS,BYPASS ;RP11 ADDRESS
2190 005270 013700 001266 MOV RPADR,R0 ;LOAD SEEK OPCODE
2191 005274 112737 000011 001364 MOV#B #SEEK,DPB ;CLEAR SECTOR/TRACK ADDRESS
2192 005302 005037 001370 CLR DPB+4 ;LOAD USER SUPPLIED TRACK ADDRESS
2193 005306 113737 001732 001371 MOV#B TK5,DPB+5
2194 005314 012706 001100 TESTS: MOV #STACK,SP
2195 005320 013701 001724 MOV FCS,R1 ;START NC1 AT FC
2196 005324 013702 001726 MOV LCS,R2 ;START NC2 AT LC
2197 005330 010137 001366 1S: MOV R1,DPB+2 ;NC1
2198 005334 004737 014006 JSR PC,SEEKS ;SEEK TO NC1
2199 005340 010237 001366 MOV R2,DPB+2 ;NC2
2200 005344 004737 014006 JSR PC,SEEKS ;SEEK TO NC1
2201 005350 063701 001730 ADD IC5,R1 ;NEXT NC1
2202 005354 163702 001730 SUB IC5,R2 ;NEXT NC2
2203 005360 020137 001726 CMP R1,LCS ;DONE?
2204 005364 003003 BGT EXITS ;YES--BRANCH

```

2205 005366 020237 001724  
2206 005372 002356  
2207 005374 000004

CMP R2,FC5  
BGE 15  
EXITS: SCOPE ;NO--BRANCH ;LOOP

\*\*\*\*\*  
;#TEST 6 SERVO ADDRESSING LOGIC NOISE GENERATOR

;\* IN THIS TEST A SEEK IS DONE TO CYL "NC" THEN A SEEK TO  
;\* NC+4 THEN NC+1 THEN NC+3 THEN NC+2 THEN NC+5. NOW "NC" IS UPDATED  
;\* BY "IC" AND THE ABOVE SEQUENCE IS REPEATED UNTIL "LC" IS  
;\* EXCEEDED BY ANY OF THE ABOVE VALUES. THE INITIAL VALUE OF "NC"  
;\* IS "FC". AT THE COMPLETION OF EACH SEEK COMMAND THE  
;\* PROPER INDICATORS ARE EXAMINED TO INSURE PROPER OPERATION.

\*\*\*\*\*

2220 005376  
2221 005376 033737 001340 001234  
2222 005404 001002  
2223 005406 000137 005630  
2224 005412 013737 001736 001206  
2225 005420 012737 005502 001106  
2226 005426 012737 005376 001110  
2227 005434 012737 000006 001102  
2228  
2229 005442 013777 001102 173472  
2230 005450 012737 005626 001244  
2231 005456 013700 001266  
2232 005462 112737 000011 001364  
2233 005470 005037 001370  
2234 005474 113737 001746 001371  
2235 005502 012706 001100  
2236 005506 013701 001740  
2237 005512 013702 001742  
2238 005516 162702 000005  
2239 005522 020102  
2240 005524 003040  
2241 005526 010137 001366  
2242 005532 004737 014006  
2243 005536 062737 000004 001366  
2244 005544 004737 014006  
2245 005550 162737 000003 001366  
2246 005556 004737 014006  
2247 005562 062737 000002 001366  
2248 005570 004737 014006  
2249 005574 162737 000001 001366  
2250 005602 004737 014006  
2251 005606 062737 000003 001366  
2252 005614 004737 014006  
2253 005620 063701 001744  
2254 005624 000736  
2255 005626 000004

TST6: BIT 2#BITS+(6\*2),TSTNMS ;DO THIS TEST?  
BNE 645 ;YES--BRANCH  
JMP TST7 ;NO--GO TO THE NEXT TEST  
645: MOV 2#RPT6,STINES ;GET THE ITERATION COUNT  
MOV 2#TEST6,2#SLPADR  
MOV 2#ST6,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS  
MOV 2#6,2#STSTNM ;SET UP TEST NUMBER AND  
;CLEAR THE ERROR FLAG (SERFLG)  
MOV 2#STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER  
MOV 2#EXIT6,BYPASS ;SETUP BYPASS ADDRESS  
MOV 2#RPAOR,R0 ;RPI1 ADDRESS  
MOV 2#B,DPB ;LOAD SEEK OPCODE  
CLR 2#DPB+4 ;CLEAR SECTOR/TRACK ADDRESS  
MOV 2#TK6,DPB+5 ;LOAD USER SUPPLIED TRACK ADDRESS  
TEST6: MOV 2#STACK,SP ;SETUP STACK  
MOV 2#FC6,R1 ;PICKUP "FC"  
MOV 2#LC6,R2 ;FORM LAST CYLINDER THAT  
SUB 2#5,R2 ;IS AVAILABLE FOR TESTING  
15: CMP 2#R1,R2 ;LAST CYLINDER  
BGT 2#EXIT6 ;YES--BRANCH  
MOV 2#R1,DPB+2 ;NC  
JSR 2#PC,SEEKS ;SEEK TO NC  
ADD 2#4,DPB+2 ;NC+4  
JSR 2#PC,SEEKS ;SEEK TO NC+4  
SUB 2#3,DPB+2 ;NC+1  
JSR 2#PC,SEEKS ;SEEK TO NC+1  
ADD 2#2,DPB+2 ;NC+3  
JSR 2#PC,SEEKS ;SEEK TO NC+3  
SUB 2#1,DPB+2 ;NC+2  
JSR 2#PC,SEEKS ;SEEK TO NC+2  
ADD 2#3,DPB+2 ;NC+5  
JSR 2#PC,SEEKS ;SEEK TO NC+5  
ADD 2#IC6,R1  
BR 15  
EXIT6: SCOPE ;LOOP

\*\*\*\*\*  
;#TEST 7 CYLINDER ADDRESSING TEST

;\* THIS TEST WILL CAUSE THE DRIVE TO SEEK FROM EACH CYLINDER TO EVERY

2256  
2257  
2258  
2259  
2260

```

2261 :# OTHER CYLINDER BETWEEN CYLINDERS "FC" AND "LC". SEEK CYCLES ARE
2262 :# COMMANDED FROM CYLINDER "NC" TO CYLINDER "NC1" AND BACK TO CYLINDER
2263 :# "NC". "NC" AND "NC1" START AT "FC". "NC1" IS INCREMENTED BY "IC"
2264 :# UNTIL "NC1" IS EQUAL TO "LC"; WHEN "NC1" IS EQUAL TO "LC", IT IS
2265 :# RESET TO "FC" AND "NC" IS INCREMENTED BY "IC". THE CYCLE IS COMPLETE
2266 :# WHEN "NC" AND "NC1" BOTH EQUAL "LC". AT THE COMPLETION OF EACH
2267 :# SEEK COMMAND, THE INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

```

```

2268 :#
2269 :#
2270 :#*****
2271 :#TST7:
2272 :# BIT 2#BITS+(7*2),TSTNMS ;DO THIS TEST?
2273 :# BNE 645 ;YES--BRANCH
2274 :# JMP TST10 ;NO--GO TO THE NEXT TEST
2275 :# 645: MOV 2#RPT7,STIMES ;GET THE ITERATION COUNT
2276 :# MOV 2#TEST7,2#SLPADR
2277 :# MOV 2#TST7,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
2278 :# MOV 2#7,2#STSTNM ;SET UP TEST NUMBER AND
2279 :# ;CLEAR THE ERROR FLAG (SERFLG)
2280 :# MOV 2#STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2281 :# MOV 2#EXIT7,BYPASS ;SETUP BYPASS ADDRESS
2282 :# MOV 2#RPADR,R0 ;RPII ADDRESS
2283 :# MOV 2#SEEK,DPB ;SETUP FOR SEEKING
2284 :# CLR 2#DPB+4 ;CLEAR TRACK/SECTOR STORAGE
2285 :# MOV 2#TK7,DPB+5 ;LOAD USER SPECIFIED TRACK ADDRESS
2286 :# TEST7: MOV 2#STACK,SP ;LOAD STACK POINTER
2287 :# MOV 2#FC7,R1 ;START NC AT FC
2288 :# MOV 2#FC7,R2 ;START NC1 AT FC
2289 :# 1S: MOV 2#R1,DPB+2 ;NC
2290 :# JSR 2#PC,SEEKS ;SEEK TO NC
2291 :# MOV 2#R2,DPB+2 ;NC1
2292 :# JSR 2#PC,SEEKS ;SEEK TO NC1
2293 :# ADD 2#IC7,R2 ;INCREMENT NC1
2294 :# CMP 2#LC7,R2 ;EXCEEDED LC ?
2295 :# BGE 1S ;BR IF NOT
2296 :# MOV 2#FC7,R2 ;RESET NC1
2297 :# ADD 2#IC7,R1 ;INCREMENT NC
2298 :# CMP 2#LC7,R1 ;EXCEEDED LC ?
2299 :# BGE 1S ;BR IF NOT
2300 :# EXIT7: SCOPE ;LOOP ?

```

```

2301 :#*****
2302 :#TEST 10 ONE CYLINDER SEEK TIMER TEST
2303 :#
2304 :# THIS TEST PERFORMS ONE CYLINDER SEEKS FROM CYLINDER 0 TO THE
2305 :# MAXIMUM CYLINDER FOR THE DRIVE (E.G. 202 FOR RPO2'S AND 405 FOR
2306 :# RPO3'S). THE SEEK TIMES ARE MEASURED RELATIVE TO THE SECTOR
2307 :# PULSES FROM THE DRIVE. THE TEST DISPLAYS THE FORWARD AND REVERSE
2308 :# TIMES.

```

```

2309 :#*****
2310 :#TST10:
2311 :# BIT 2#BITS+(10*2),TSTNMS ;DO THIS TEST?
2312 :# BNE 645 ;YES--BRANCH
2313 :# JMP TST11 ;NO--GO TO THE NEXT TEST
2314 :# 645: MOV 2#RPT10,STIMES ;GET THE ITERATION COUNT
2315 :# MOV 2#TEST10,2#SLPADR
2316 :#

```

2317	006052	012737	006022	001110	MOV	#TST10, @#SLPERR	: SETUP THE ERROR LOOP ADDRESS
2318	006060	012737	000010	001102	MOV	#10, @#STSTNM	: SET UP TEST NUMBER AND
2319							: CLEAR THE ERROR FLAG (SERFLG)
2320	006066	013777	001102	173046	MOV	STSTNM, @DISPLAY	: LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2321	006074	012737	006274	001244	MOV	#EXIT10, BYPASS	: SETUP BYPASS ADDRESS
2322	006102	013700	001266		MOV	RPADR, R0	: RP11 ADDRESS
2323	006106	112737	000011	001364	MOVB	#SEEK, DPB	: OPCODE FOR THE TEST
2324	006114	005037	001370		CLR	DPB+4	: TRACK ADDRESS



E05

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST MACY11 27(732) 02-NOV-76 15:10 PAGE 57  
DZRPZB.CMB T10 ONE CYLINDER SEEK TIMER TEST

2325 006120 012706 001100

TEST10: MOV #STACK,SP

;RESTORE THE STACK POINTER

# F05

```

2326 006124 004737 015264 JSR PC,RESTOR ;DO A HOME SEEK
2327 006130 012704 024010 MOV #FRMSTR,R4 ;FORWARD TIME STORAGE POINTER
2328 006134 012737 000001 001366 MOV #1,DPB+2 ;SEEK TO THIS CYLINDER
2329 006142 004737 016476 15: JSR PC,TIMSEK ;START THE SEEK, AND TIME IT
2330 006146 010324 MOV R3,(R4)+ ;STORE THE TIME
2331 006150 005237 001366 INC DPB+2 ;INCREMENT THE CYLINDER ADDRESS
2332 006154 023737 001366 001236 CMP DPB+2,MAXCYL ;AT THE MAXIMUM?
2333 006162 003767 BLE 15 ;BR IF NOT
2334 006164 013737 001236 001366 MOV MAXCYL,DPB+2 ;SETUP TO SEEK BACK TO MINIMUM
2335 006172 005337 001366 DEC DPB+2 ;SEEK TO CYLINDER
2336 006176 012704 025464 MOV #RVSTR,R4 ;REVERSE TIME STORAGE POINTER
2337 006202 004737 016476 25: JSR PC,TIMSEK ;START THE SEEK, AND TIME IT
2338 006206 010324 MOV R3,(R4)+ ;STORE THE TIME
2339 006210 005337 001366 DEC DPB+2 ;DECREMENT THE CYLINDER ADDRESS
2340 006214 100372 BPL 25 ;BR IF NOT AT MINIMUM CYLINDER
2341 006216 104401 006224 TYPE ,655 ;TYPE ASCIZ STRING
2342 006222 000416 BR 645 ;GET OVER THE ASCIZ
2343 ;:655: .ASCIZ <15><12><12>/ONE CYLINDER SEEK TIMES/
2344 006260 645:
2345 006260 013737 001236 006272 MOV MAXCYL,35 ;NUMBER OF SEEKS PERFORMED
2346 006266 004037 015630 JSR RD,SORT ;SORT AND TYPE OUT THE TIMES
2347 006272 000000 35: .WORD 0 ;NUMBER OF TIMES GOES HERE
2348 006274 000004 EXIT10: SCOPE ;LOOP?
2349
2350 ;:*****
2351 ;#TEST 11 SEEK-SEEK TIMER TEST
2352
2353 ;# THIS TEST IS A GENERAL PURPOSE TIMING TEST WHICH PERFORMS 200
2354 ;# ITERATIONS BETWEEN THE CYLINDER IN 'FC' AND THE CYLINDER IN 'LC'.
2355 ;# BOTH FORWARD AND REVERSE SEEKS ARE TIMED AND DISPLAYED. THE
2356 ;# SEEK TIMES DISPLAYED ARE RELATIVE TO THE SECTOR COUNTER.
2357 ;# 'FC' DEFAULTS TO 0; 'LC' DEFAULTS TO 'MAXCYL' (202 IF RPO2'S OR
2358 ;# 405 IF RPO3'S).
2359
2360 ;:*****
2361 006276 TST11:
2362 006276 033737 001346 001234 BIT #BITS+(11*2),TSTNMS ;DO THIS TEST?
2363 006304 001002 BNE 645 ;YES--BRANCH
2364 006306 000137 006610 JMP TST12 ;NO--GO TO THE NEXT TEST
2365 006312 013737 001772 001206 645: MOV #R7/11,STIMES ;GET THE ITERATION COUNT
2366 006320 012737 006374 001106 MOV #TEST11,#SLPADR
2367 006324 012737 006276 001110 MOV #TST11,#SLPERR ;SETUP THE ERROR LOOP ADDRESS
2368 006334 012737 000011 001102 MOV #11,#STSTNM ;SET UP TEST NUMBER AND
2369 ;CLEAR THE ERROR FLAG (SERFLG)
2370 006342 013777 001102 172572 MOV STSTNM,#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2371 006350 012737 006606 001244 MOV #EXIT11,BYPASS ;SETUP BYPASS ADDRESS
2372 006356 013700 001266 MOV RPADR,R0 ;RPI1 ADDRESS
2373 006362 012737 000011 001364 MOV #SEEK,DPB ;OPCODE FOR THE TEST
2374 006370 005037 001370 CLR DPB+4 ;CLEAR THE TRACK ADDRESS STORAGE
2375 006374 012706 001100 TEST11: MOV #STACK,SP ;RESTORE THE STACK POINTER
2376 006400 004737 015264 JSR PC,RESTOR ;DO A HOME SEEK
2377 006404 012701 000311 MOV #201,R1 ;ITERATION COUNT
2378 006410 012704 025464 MOV #RVSTR,R4 ;REVERSE TIME STORAGE POINTER
2379 006414 012705 024010 MOV #FRMSTR,R5 ;FORWARD TIME STORAGE POINTER
2380 006420 013737 001774 001366 MOV FC11,DPB+2 ;POSITION THE DISK TO THE FIRST CYLINDER
2381 006426 004737 016476 JSR PC,TIMSEK ;DO THE SEEK

```

```

2382 006432 013737 001776 001366 1S:  MOV LC11,DPB+2      ;TIME TO THIS CYLINDER
2383 006440 004737 016476              JSR PC,TIMSEK      ;START THE SEEK AND TIME IT
2384 006444 010325              MOV R3,(R5)+       ;STORE THE TIME
2385 006446 013737 001774 001366      MOV FC11,DPB+2     ;MINIMUM CYLINDER
2386 006454 004737 016476              JSR PC,TIMSEK      ;START THE SEEK AND TIME IT
2387 006460 010324              MOV R3,(R4)+       ;STORE THE TIME
2388 006462 005301              DEC R1             ;DECREMENT THE ITERATION COUNTER
2389 006464 001362              BNE 1S            ;BR IF NOT FINISHED
2390 006466 104401 006474              TYPE 65S          ;TYPE ASCIZ STRING
2391 006472 000421              BR 64S            ;GET OVER THE ASCIZ
2392              ;:65S: .ASCIZ <15><12><12>/SEEK TIMES BETWEEN CYLINDERS /
2393              64S:
2394 006536 013746 001774              MOV FC11,-(SP)    ;PUT STARTING CYLINDER ADDRESS ON THE STACK
2395 006542 004737 013220              JSR PC,$S82D      ;CONVERT IT TO DECIMAL
2396 006546 004737 013450              JSR PC,$SUPRS     ;TYPE IT
2397 006552 104401 006560              TYPE 67S          ;TYPE ASCIZ STRING
2398 006556 000402              BR 66S            ;GET OVER THE ASCIZ
2399              ;:67S: .ASCIZ / & /
2400              66S:
2401 006564 013746 001776              MOV LC11,-(SP)    ;PUT ENDING CYLINDER ADDRESS ON THE STACK
2402 006570 004737 013220              JSR PC,$S82D      ;CONVERT IT TO DECIMAL
2403 006574 004737 013450              JSR PC,$SUPRS     ;TYPE IT
2404 006600 004037 015630              JSR RD, SORT      ;TYPE THE TIMES
2405 006604 000310              .WORD 200.
2406 006506 000004              EXIT11: SCOPE     ;LOOP ?

```

\*\*\*\*\*  
;\*TEST 12 SEEK TIME GRAPH TEST

```

;*
;* THIS TEST PRODUCES TWO SEEK TIME GRAPHS. THE FIRST GRAPH IS
;* TYPED IS SWITCH <04> IS 0. THIS GRAPH IS A PLOT OF SEEK TIME
;* AGAINST CYLINDERS SEEKED TO FROM CYLINDER 0. THE CYLINDERS
;* PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25,...100,105,...MAXCYL.
;* THE OTHER GRAPH WHICH IS PRODUCED (SWITCH <04> SET) IS TIME TO
;* SEEK TO ALL CYLINDERS FROM CYLINDER 0.
;*
;* IN BOTH SEEK GRAPHS, ONLY THE FORWARD SEEK TIMES ARE MEASURED.
;*
;* THIS TEST USES THE SECTOR COUNTER ('SOT') IN THE RP11 AS THE
;* TIME BASE FOR THE MEASUREMENTS, USE OF THE 'SOT' PROVIDES A
;* RESOLUTION OF 2.5 MS.

```

\*\*\*\*\*  
;TST12:

```

2407 006610 033737 001350 001234      BIT 2#BITS+(12*2),TSTNMS ;DO THIS TEST?
2408 006616 001002              BNE 64S           ;YES--BRANCH
2409 006620 000137 007464              JMP SEOP          ;NO--GO TO THE END OF THE PROGRAM
2410 006624 013737 002002 001206      64S: MOV 2#RPT12,$TIMES ;GET THE ITERATION COUNT
2411 006632 012737 006706 001106      MOV #TEST12,2#SLPADR
2412 006640 012737 006610 001110      MOV #TST12,2#SLPERR ;SETUP THE ERROR LOOP ADDRESS
2413 006646 012737 000012 001102      MOV #12,2#STSTNM  ;SET UP TEST NUMBER AND
2414              ;CLEAR THE ERROR FLAG (SERFLG)
2415 006654 013777 001102 172260      MOV STSTNM,2#DISPLAY ;LOAD THE TEST NUMBER INTO THE DISPLAY REGISTER
2416 006662 012737 007462 001244      MOV #EXIT12,BYPASS ;SETUP BYPASS ADDRESS
2417 006670 013700 001266              MOV RPADR,R0     ;RP11 ADDRESS

```

```

2438 006674 112737 000011 001364      MOVB    #SEEK,DPB      ;OPCODE FOR THE TEST
2439 006702 005037 001370      CLR     DPB+4         ;CLEAR THE TRACK ADDRESS STORAGE
2440 006706 012706 001100      TEST12: MOV    #STACK,SP ;INITIALIZE THE STACK POINTER
2441 006712 004737 015264      JSR    PC,RESTOR     ;DO A HOME SEEK
2442 006716 012704 024010      MOV     #FRMSTR,R4   ;STORE 'SEEK TIME' HERE
2443 006722 005001      CLR     R1           ;CLEAR 'FC' STORAGE
2444 006724 005002      CLR     R2           ;CLEAR 'NC' STORAGE
2445 006726 010237 001366      IS:    MOV    R2,DPB+2 ;ADDRESS OF NEW CYLINDER 'NC'
2446 006732 004737 016476      JSR    PC,TIMSEK     ;SEEK TO 'NC' AND TIME IT
2447 006736 010324      MOV     R3,(R4)+     ;STORE MEASURED SEEK TIME
2448 006740 010137 001366      MOV     R1,DPB+2    ;'FC' ADDRESS
2449 006744 004737 016476      JSR    PC,TIMSEK     ;RETURN TO 'FC'
2450 006750 005202      INC     R2           ;INCREMENT 'NC'
2451 006752 023702 001236      CMP     MAXCYL,R2   ;IS 'NC' AT MAXIMUM ?
2452 006756 002363      BGE     IS          ;BR IF NOT

;PLOT A GRAPH USING MEASURED SEEK TIMES
PLOT:
2456 006760      TYPE    65$         ;TYPE ASCIZ STRING
2457 006760 104401 006766      BR     64$         ;GET OVER THE ASCIZ
2458 006764 000412      ;:65$: .ASCIZ <15><12><12>/SEEK TIME GRAPH/
2459 64$:
2460 007012      TYPE    67$         ;TYPE ASCIZ STRING
2461 007012 104401 007020      BR     66$         ;GET OVER THE ASCIZ
2462 007016 000421      ;:67$: .ASCIZ <15><12>/X AXIS - SEEK TIME - MILLI SECS/
2463 66$:
2464 007062      TYPE    69$         ;TYPE ASCIZ STRING
2465 007062 104401 007070      BR     68$         ;GET OVER THE ASCIZ
2466 007066 000423      ;:69$: .ASCIZ <15><12>/Y AXIS - CYLINDER SEEKED FROM 0/<15><12><12>
2467 68$:
2468 007136      TYPE    GRAPH1      ;TYPE THE GRAPH HEADING
2469 007136 104401 021443      BIT    #SM4,SMR     ;TYPE COMPLETE GRAPH?
2470 007142 032777 000020 171770      BNE    CMPGRP       ;YES BRANCH; IF NOT, TYPE SMALL GRAPH
2471 007150 001064      ;IN THE SMALL GRAPH, SEEK TIMES ARE PLOTTED ONLY FOR SELECTED CYLINDERS.
2472 ;THE CYLINDERS PLOTTED ARE: 0,1,2,3,4,6,8,10,12,14,16,18,20,25,30
2473 ;35,40...100,105...MAXCYL
2474
2475
2476
2477 007152 005000      SMGRP: CLR     R0
2478 007154 005037 024010      CLR     FRMSTR
2479 007160 104401 001217      IS:    TYPE    SCALF ;FORCE CYL 0 CYL 0 SEEK TO ZERO TIME
2480 007164 010046      MOV     R0,-(SP)    ;CR-LF
2481 007166 104405      TYPDS ;TYPE THE MARKERS
2482 007170 104401 007176      TYPE    65$         ;TYPE ASCIZ STRING
2483 007174 000401      BR     64$         ;GET OVER THE ASCIZ
2484 64$:
2485 007200      ;:65$: .ASCIZ /-/
2486 64$:
2487 007200      MOV     R0,R1      ;FORM THE ADDRESS OF 'SEEK TIME'
2488 007202 006301      ASL    R1
2489 007204 016103 024010      MOV     FRMSTR(R1),R3 ;GET THE SEEK TIME
2490 007210 004737 007422      JSR    PC,PLTPT     ;GO PLOT IT
2491 007214 022700 000004      CMP     #4,R0       ;PLOTTED UPTO CYL 4?
2492 007220 003402      BLE    2$          ;YES
2493 007222 005200      INC    R0           ;INCREMENT BY 1
2494 007224 000755      BR     IS

```

```

2494 007226 022700 000024 2S:  CMP      #20.,R0      ;PAST CYLINDER 20 ?
2495 007232 003403          BLE      3S      ;BR IF PAST 20
2496 007234 062700 000002      ADD      #2,R0      ;INCREMENT BY 2
2497 007240 000747          BR       1S
2498 007242 013746 001236 3S:  MOV      MAXCYL,-(SP) ;CHECK FOR END
2499 007246 042716 000007      BIC      #7,(SP)    ;CLEAR LOWER CYLINDER BITS
2500 007252 022600          CMP      (SP)+,R0   ;ALMOST MAX CYL ?
2501 007254 003403          BLE      4S      ;BR IF ALMOST AT MAX
2502 007256 062700 000005      ADD      #5,R0      ;INCREMENT BY 5
2503 007262 000736          BR       1S
2504 007264 023700 001236 4S:  CMP      MAXCYL,R0   ;PLOTTED ALL CYLS?
2505 007270 001412          BEQ      6S      ;BR IF DONE
2506 007272 022737 000312 001236  CMP      #202.,MAXCYL ;MAXCYL = 202. ?
2507 007300 001003          BNE      5S      ;BR IF NOT
2508 007302 062700 000002      ADD      #2,R0      ;INCREMENT TO CYLINDER 202
2509 007306 000724          BR       1S
2510 007310 062700 000005 5S:  ADD      #5,R0      ;INCREMENT TO CYLINDER 405
2511 007314 000721          BR       1S
2512 007316 000137 007462 6S:  JMP      EXIT12     ;GO TO THE NEXT TEST

;IF SW 4 IS SET THE COMPLETE GRAPH IS PRINTED OUT. IT GIVES TIMES FOR
;SEEKS FROM CYLINDER 0 TO ALL OTHER CYLINDERS (0,1,2,3...MAXCYL).

007322 005000          CMPGRP: CLR      R0      ;INITIALIZE COUNT
007324 012701 177773      MOV      #5,R1      ;INITIALIZE COUNT FOR Y-AXIS MARKER
007330 012702 024010      MOV      #FRWSTR,R2 ;INITIALIZE POINTER TO SEEK TIMES
007334 104401 001217      TYPE    ,SCLRF
007340 000404          BR       3S
007342 005201 2S:  INC      R1      ;TYPE OUT Y-AXIS MARKER 'CYL #'
007344 001005          BNE      4S      ;IF REQUIRED
007346 012701 177773      MOV      #5,R1      ;RESET INDEX
007352 010046 3S:  MOV      R0,-(SP)   ;TYPE 'CYL #' ON Y-AXIS
007354 104405          TYPDS   ;(IN DECIMAL)
007356 000402          BR       5S
007360 104401 002037 4S:  TYPE    ,BLNKS6    ;6 BLANKS
007364 5S:
007364 104401 007372      TYPE    ,65S      ;;TYPE ASCIZ STRING
007370 000401          BR      64S      ;;GET OVER THE ASCIZ
64S:  .ASCIZ  /-/
007374 012203          MOV      (R2)+,R3   ;GET SEEK TIME
007376 004737 007422      JSR     PC,PLTPT   ;GO PLOT THE POINT
007402 104401 001217      TYPE    ,SCLRF
007406 005200          INC      R0      ;ALL DONE?
007410 022700 001236      CMP      #MAXCYL,R0 ;LAST CYLINDER ?
007414 001352          BNE      2S      ;IF NOT, GO BACK
007416 000137 007462 6S:  JMP      EXIT12     ;GO TO THE NEXT TEST

;ROUTINE TO PLOT THE SEEK TIME ON THE GRAPH. ENTER WITH R3 CONTAINING
;THE HORIZONTAL AXIS COORDINATE (I.E. SEEK TIME).

007422 010546          PLTPT: MOV      R5,-(SP) ;STORE R5
007424 012705 000031      MOV      #25.,R5   ;HORIZONTAL AXIS INDEX
007430 162703 000372 1S:  SUB      #250.,R3   ;FIND OUT HOW MANY BLANKS TO
007434 002404          BLT     2S      ;INSERT TO PLOT THE POINT
;NOTE THE FIRST CELL = 0 MS

```

```

2550 007436 104401 002044      TYPE      BLNKS1      ;: BLANK
2551 007442 005305              DEC      R5           ;: AT THE END OF THE HORIZONTAL AXIS ?
2552 007444 100371              BPL      1$          ;: BR IF NOT
2553 007446 104401 007454      2$:      TYPE      65$      ;: TYPE ASCIZ STRING
2554 007446 000401              BR      64$          ;: GET OVER THE ASCIZ
2555 007452 000401              ;: 65$: .ASCIZ  /X/
2556 007456 012605              64$:      MOV      (SP)+,R5      ;: RESTORE R5
2557 007456 000207              RTS      PC
2558 007460 000004              EXIT12: SCOPE          ;: LOOP ?
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572 007464 104401 007472      SEOP:    TYPE      65$      ;: TYPE ASCIZ STRING
2573 007464 000410              BR      64$          ;: GET OVER THE ASCIZ
2574 007470 000410              ;: 65$: .ASCIZ <15><12><12>/END OF PASS/
2575
2576 007512 005737 001232      64$:    TST      2#DRVSEL      ;: ANY DRIVES SELECTED?
2577 007512 001434              BEQ      1$          ;: NO--BRANCH
2578 007516 104401 007526      TYPE      67$      ;: TYPE ASCIZ STRING
2579 007520 000405              BR      66$          ;: GET OVER THE ASCIZ
2580 007524 000405              ;: 67$: .ASCIZ / ON DRIVE/
2581
2582 007540 013746 001246      66$:    MOV      2#CHKDRV,-(SP) ;: SAVE 2#CHKDRV FOR TYPEOUT
2583 007540 104403              TYPOS
2584 007544 002              .BYTE 2              ;: GO TYPE--OCTAL ASCII
2585 007546 000              .BYTE 0              ;: TYPE 2 DIGIT(S)
2586 007547 000              .BYTE 0              ;: SUPPRESS LEADING ZEROS
2587 007550 104401 007556      TYPE      69$      ;: TYPE ASCIZ STRING
2588 007554 000412              BR      68$          ;: GET OVER THE ASCIZ
2589 007602 013746 001112      ;: 69$: .ASCIZ / ERRORS DETECTED=/
2590 007602 104402              68$:    MOV      2#SERTTL,-(SP) ;: SAVE 2#SERTTL FOR TYPEOUT
2591 007606 005037 001112      TYPOC
2592 007610 005037 001112      1$:    CLR      2#SERTTL      ;: GO TYPE--OCTAL ASCII(ALL DIGITS)
2593 007614 005037 001102      CLR      $STNM        ;: ZERO ERROR TOTAL
2594 007620 005037 001206      CLR      $TIMES       ;: ZERO THE TEST NUMBER
2595 007624 005237 001100      CLR      $SPASS       ;: ZERO THE NUMBER OF ITERATIONS
2596 007630 042737 100000 001100      INC      $PC          ;: INCREMENT THE PASS NUMBER
2597 007636 005327              BIC      #100000,$SPASS ;: DON'T ALLOW A NEG. NUMBER
2598 007640 000001              DEC      (PC)+        ;: LOOP?
2599 007642 003027              SEOPCT: .WORD 1
2600 007644 012737              BGT      $DOAGN       ;: YES
2601 007646 000001              MOV      (PC)+,2(PC)+ ;: RESTORE COUNTER
2602 007650 007640              SENDCT: .WORD 1
2603 007652 104401 007660      TYPE      65$      ;: TYPE ASCIZ STRING
2604 007656 000407              BR      64$          ;: GET OVER THE ASCIZ

```

# K05

```

2606      ::655: .ASCIZ  <15><12>/END OF TEST/
2607 007676 645:
2608 007676 104401 007726      TYPE      SNULL      ;TYPE NULL CHARACTER
2609 007702 013700 000042  $GET42: MOV      @#42,RO  ;GET MONITOR ADDRESS
2610 007706 001405      BEQ      SDOAGN     ;BRANCH IF NO MONITOR
2611 007710 000005      RESET     ;CLEAR THE WORLD
2612 007712 004710  SENDAD: JSR      PC,(RO) ;GO TO MONITOR
2613 007714 000240      NOP      ;SAVE ROOM
2614 007716 000240      NOP      ;FOR
2615 007720 000240      NOP      ;ACT11
2616 007722  SDOAGN:
2617 007722 000137      JMP      @PC+      ;;RETURN
2618 007724 003716  SRTNAD: .WORD    RESTART
2619 007726 377      000  SNULL: .BYTE    -1,-1,0  ;;NULL CHARACTER STRING
2620 007732  .EVEN
2621
2622      .SBTTL  *** SUBROUTINES ***
2623      .SBTTL  ERROR HANDLER ROUTINE
2624
2625      ;*****
2626      ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2627      ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL,
2628      ;AND GO TO TYPERR ON ERROR
2629      ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2630      ;SW15=1      HALT ON ERROR
2631      ;SW13=1      INHIBIT ERROR TYPEOUTS
2632      ;SW10=1      BELL ON ERROR
2633      ;SM09=1      LOOP ON ERROR
2634      ;CALL
2635      ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2636
2637 007732  SERROR:
2638 007732 104407      CKSWR      ;TEST FOR CHANGE IN SOFT-SWR
2639 007734 105237 001103  75:      INCB      SERFLG      ;SET THE ERROR FLAG
2640 007740 001775      BEQ      75      ;DON'T LET THE FLAG GO TO ZERO
2641 007742 013777 001102 171172  MOV      $STNM,$DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
2642 007750 032777 002000 171162  BIT      #BIT10,$SWR    ;BELL ON ERROR?
2643 007756 001402      BEQ      15      ;NO - SKIP
2644 007760 104401 001212      TYPE      $BELL      ;RING BELL
2645 007764 005237 001112  15:      INC      $ERTTL     ;COUNT THE NUMBER OF ERRORS
2646 007770 011637 001116      MOV      (SP),$ERRPC  ;GET ADDRESS OF ERROR INSTRUCTION
2647 007774 162737 000002 001116  SUB      #2,$ERRPC
2648 010002 117737 171110 001114  MOVB    $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
2649 010010 032777 020000 171122  BIT      #BIT13,$SWR    ;SKIP TYPEOUT IF SET
2650 010016 001004      BNE      20$      ;SKIP TYPEOUTS
2651 010020 004737 010072      JSR      PC,TYPERR   ;GO TO USER ERROR ROUTINE
2652 010024 104401 001217      TYPE      ,SCLF
2653 010030  20$:
2654 010030 005777 171104  25:      TST      $SWR      ;HALT ON ERROR
2655 010034 100002      BPL      35      ;SKIP IF CONTINUE
2656 010036 000000      HALT     ;HALT ON ERROR!
2657 010040 104407      CKSWR      ;TEST FOR CHANGE IN SOFT-SWR
2658 010042 032777 001000 171070  35:      BIT      #BIT09,$SWR  ;LOOP ON ERROR SWITCH SET?
2659 010050 001402      BEQ      45      ;BR IF NO
2660 010052 013716 001110      MOV      $LPERR,(SP) ;FUDGE RETURN FOR LOOPING
2661 010056 005737 001210  45:      TST      $ESCAPE    ;CHECK FOR AN ESCAPE ADDRESS
  
```

```

2662 010062 001402          BEQ      5$          ;;BR IF NONE
2663 010064 013716 001210    MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
2664 010070          5$:          RTI              ;;RETURN
2665 010070 000002
2666
2667 ;;*****
2668 .SBTTL  TYPERR - TYPE ERROR ROUTINE
2669 ;;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
2670 ;;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS, FROM THE "ERROR
2671 ;;TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
2672 ;;CONCERNING THE ERROR.
2673 ;;CALL
2674     JSR      PC,@TYPERR
2675     RETURN
2676
2677 010072 104412          TYPERR: SAVREG          ;;SAVE R0-R5
2678 010074 005037 001204    CLR      $TMPD          ;;CLEAR LOCATION FOR TEST NUMBER
2679 010100 113737 001102 001204  MOVB     $STNM,$TMPD    ;;LOAD TEST NUMBER FOR TYPEOUT
2680 010106 005000          CLR      R0            ;;CLEAR R0 FOR ERROR NUMBER
2681 010110 113700 001114    MOVB     $ITEMB,R0     ;;ERROR NUMBER
2682 010114 005300          DEC      R0            ;;FORM INDEX FOR ERROR TABLE
2683 010116 006300          ASL     R0
2684 010120 006300          ASL     R0
2685 010122 006300          ASL     R0
2686 010124 062700 002046    1$:     ADD     $ERRTB,R0  ;;FORM ADDRESS
2687 010130 012037 010144    MOV      (R0)+,2$      ;;GET ERROR MESSAGE (EM) POINTER
2688 010134 001412          BEQ     3$            ;;BRANCH IF THERE ISN'T ONE
2689 010136 104401 001217    TYPE    ,SCRLF        ;;CARRIAGE RETURN - LINE FEED
2690 010142 104401
2691 010144 000000          2$:     .WORD   0        ;;"EM" POINTER GOES HERE
2692 010146 032777 000100 170764  BIT      $SW06,@SWR    ;;TYPE THE REGISTERS INSTEAD OF THE NORMAL
2693 ;;'DH' & 'DT' LINES ?
2694 010154 001402          BEQ     3$            ;;BR IF NOT
2695 010156 012700 023502    MOV      $ETSP,R0     ;;CHANGE MESSAGE BLOCK
2696 010162 012037 010176    3$:     MOV      (R0)+,4$    ;;PICK UP DATA HEADER (DH) POINTER
2697 010166 001404          BEQ     5$            ;;BRANCH IF NONE
2698 010170 104401 001217    TYPE    ,SCRLF        ;;CARRIAGE RETURN-LINE FEED
2699 010174 104401
2700 010176 000000          4$:     .WORD   0        ;;"DH" POINTER GOES HERE
2701 010200 012001 5$:     MOV      (R0)+,R1     ;;PICKUP DATA TABLE (DT) POINTER
2702 010202 001450          BEQ     20$           ;;BRANCH IF NONE
2703 010204 005005          CLR     R5            ;;SET INDENT SWITCH
2704 010206 012000          MOV     (R0)+,R0     ;;DATA FORMAT (DF) POINTER
2705 010210 012002          MOV     (R0)+,R2     ;;NUMBER OF DH'S TO TYPE
2706 010212 001441          BEQ     17$           ;;BRANCH IF DH NUMBER IS 0
2707 010214 005105          COM    R5            ;;NO INDENT
2708 010216 104401 001217    TYPE    ,SCRLF        ;;CARRIAGE RETURN-LINE FEED
2709 010222 112003 10$:     MOVB     (R0)+,R3     ;;NUMBER OF DATA WORDS TO TYPE
2710 010224 112004          MOVB     (R0)+,R4     ;;AND HOW TO TYPE THEM
2711 010226 006004 11$:     ROR     R4            ;;OCTAL OR DECIMAL?
2712 010230 103403          BCS     12$           ;;DECIMAL--BRANCH
2713 010232 013146          MOV     @R1+,-(SP)   ;;SAVE @R1)+ FOR TYPEOUT
2714 010234 104402          TYPCC
2715 010236 000402          BR      13$
2716 010240 12$:
2717 010240 013146          MOV     @R1+,-(SP)   ;;SAVE @R1)+ FOR TYPEOUT

```



2718	010242	104405			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
2719	010244	005303		13S:	DEC	R3	::MORE NUMBERS TO TYPE?
2720	010246	001403			BEQ	14S	::NO--BRANCH
2721	010250	104401	002043		TYPE	BLNKS2	::YES--TYPE SEPARATORS
2722	010254	000764			BR	11S	::LOOP
2723	010256	005302		14S:	DEC	R2	::MORE DH'S?
2724	010260	003421			BLE	20S	::NO--BRANCH
2725	010262	104401	001217		TYPE	\$CRLF	::YES--START A NEW LINE
2726	010266	005105			COM	R5	::INDENT?
2727	010270	001002			BNE	15S	::NO--BRANCH
2728	010272	104401	002043		TYPE	BLNKS2	::YES--TYPE SPACES
2729	010276	012037	010304	15S:	MOV	(R0)+,16S	::GET NEXT DH
2730	010302	104401			TYPE		::AND TYPE IT
2731	010304	000000		16S:	WORD	0	::DH POINTER GOES HERE
2732	010306	104401	001217		TYPE	\$CRLF	::CARRIAGE RETURN-LINE FEED
2733	010312	005705			TST	R5	::INDENT?
2734	010314	001342			BNE	10S	::NO--BRANCH
2735	010316	104401	002043	17S:	TYPE	BLNKS2	::YES--TYPE SPACES
2736	010322	000737			BR	10S	::LOOP
2737	010324	104413		20S:	RESREG		::RESTORE R0-R5
2738	010326	000207			RTS	PC	::RETURN

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

2759	010330	105737	001157	\$TYPE:	TSTB	\$TPFLG	::IS THERE A TERMINAL?
2760	010334	100002			BPL	1S	::BR IF YES
2761	010336	000000			HALT		::HALT HERE IF NO TERMINAL
2762	010340	000407			BR	3S	::LEAVE
2763	010342	010046		1S:	MOV	R0, -(SP)	::SAVE R0
2764	010344	017600	000002		MOV	22(SP), R0	::GET ADDRESS OF ASCIZ STRING
2765	010350	112046		2S:	MOVB	(R0)+, -(SP)	::PUSH CHARACTER TO BE TYPED ONTO STACK
2766	010352	001005			BNE	4S	::BR IF IT ISN'T THE TERMINATOR
2767	010354	005726			TST	(SP)+	::IF TERMINATOR POP IT OFF THE STACK
2768	010356	012600		60S:	MOV	(SP)+, R0	::RESTORE R0
2769	010360	062716	000002	3S:	ADD	#2, (SP)	::ADJUST RETURN PC
2770	010364	000002			RTI		::RETURN
2771	010366	122716	000011	4S:	CMPB	#HT, (SP)	::BRANCH IF <HT>
2772	010372	001430			BEQ	8S	
2773	010374	122716	000200		CMPB	#CRLF, (SP)	::BRANCH IF NOT <CRLF>

```

2774 010400 001006 BNE 55
2775 010402 005726 TST (SP)+ ;:POP <CR><LF> EQUIV
2776 010404 104401 TYPE ;:TYPE A CR AND LF
2777 010406 001217 SCRLF
2778 010410 105037 010544 CLRB $CHARCNT ;:CLEAR CHARACTER COUNT
2779 010414 000755 BR 25 ;:GET NEXT CHARACTER
2780 010416 004737 010500 5$: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
2781 010422 123726 001156 6$: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
2782 010426 001350 BNE 25 ;:IF NO GO GET NEXT CHAR.
2783 010430 013746 001154 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
2784 ;:AND THE NULL CHAR.
2785 010434 105366 000001 7$: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
2786 010440 002770 BLT 65 ;:BR IF NO--GO POP THE NULL OFF OF STACK
2787 010442 004737 010500 JSR PC,$TYPEC ;:GO TYPE A NULL
2788 010446 105337 010544 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
2789 010452 000770 BR 75 ;:LOOP

```

;HORIZONTAL TAB PROCESSOR

```

2791
2792
2793 010454 112716 000040 8$: MOVB #' (SP) ;:REPLACE TAB WITH SPACE
2794 010460 004737 010500 9$: JSR PC,$TYPEC ;:TYPE A SPACE
2795 010464 132737 000007 010544 BITB #',$CHARCNT ;:BRANCH IF NOT AT
2796 010472 001372 BNE 95 ;:TAB STOP
2797 010474 005726 TST (SP)+ ;:POP SPACE OFF STACK
2798 010476 000724 BR 25 ;:GET NEXT CHARACTER
2799 010500 105777 170444 $TYPEC: TSTB $STPB ;:WAIT UNTIL PRINTER IS READY
2800 010504 100375 BPL $TYPEC
2801 010506 116677 000002 170436 MOVB 2(SP),$STPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
2802 010514 122766 000015 000002 CMPB #CR,2(SP) ;:IS CHARACTER A CARRIAGE RETURN?
2803 010522 001003 BNE 15 ;:BRANCH IF NO
2804 010524 105037 010544 CLRB $CHARCNT ;:YES--CLEAR CHARACTER COUNT
2805 010530 000406 BR $TYPEX ;:EXIT
2806 010532 122766 000012 000002 1$: CMPB #LF,2(SP) ;:IS CHARACTER A LINE FEED?
2807 010540 001402 BEQ $TYPEX ;:BRANCH IF YES
2808 010542 105227 INCB (PC)+ ;:COUNT THE CHARACTER
2809 010544 000000 $CHARCNT: .WORD 0 ;:CHARACTER COUNT STORAGE
2810 010546 000207 $TYPEX: RTS PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

2811
2812
2813 ;:*****
2814 ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2815 ;:OCTAL (ASCII) NUMBER AND TYPE IT.
2816 ;:$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2817 ;:$CALL:
2818 ;:
2819 ;:
2820 ;:MOV NUM,-(SP) ;:NUMBER TO BE TYPED
2821 ;:TYPOS ;:CALL FOR TYPEOUT
2822 ;: .BYTE N ;:N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2823 ;: .BYTE M ;:M=1 OR 0
2824 ;: ;:1=TYPE LEADING ZEROS
2825 ;: ;:0=SUPPRESS LEADING ZEROS
2826 ;:
2827 ;:$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2828 ;:$TYPOS OR $TYPOC
2829 ;:$CALL:

```

```

2830          *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2831          *      TYPON                      ;;CALL FOR TYPEOUT
2832          *
2833          *STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2834          *CALL:
2835          *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2836          *      TYPOC                      ;;CALL FOR TYPEOUT
2837
2838 010550 017646 000000          STYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
2839 010554 116637 000001 010773  MOVB     1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
2840 010562 112637 010775          MOVB     (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
2841 010566 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
2842 010572 000406
2843 010574 112737 000001 010773  STYPOC: MOVB     #1,SOFILL      ;;SET THE ZERO FILL SWITCH
2844 010602 112737 000006 010775  MOVB     #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
2845 010610 112737 000005 010772  STYPON: MOVB     #5,SOCNT      ;;SET THE ITERATION COUNT
2846 010616 010346          MOV      R3,-(SP)          ;;SAVE R3
2847 010620 010446          MOV      R4,-(SP)          ;;SAVE R4
2848 010622 010546          MOV      R5,-(SP)          ;;SAVE R5
2849 010624 113704 010775          MOVB     $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2850 010630 005404          NEG      R4
2851 010632 062704 000006          ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
2852 010636 110437 010774          MOVB     R4,SOMODE      ;;SAVE IT FOR USE
2853 010642 113704 010773          MOVB     SOFILL,R4      ;;GET THE ZERO FILL SWITCH
2854 010646 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
2855 010652 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
2856 010654 006105          15:    ROL      R5          ;;ROTATE MSB INTO "C"
2857 010656 000404          BR      35          ;;GO DO MSB
2858 010660 006105          25:    ROL      R5          ;;FORM THIS DIGIT
2859 010662 006105
2860 010664 006105
2861 010666 010503          MOV      R5,R3
2862 010670 006103          35:    ROL      R3          ;;GET LSB OF THIS DIGIT
2863 010672 105337 010774          DECB     $OMODE      ;;TYPE THIS DIGIT?
2864 010676 100016          BPL      75          ;;BR IF NO
2865 010700 042703 177770          BIC      #177770,R3      ;;GET RID OF JUNK
2866 010704 001002          BNE     45          ;;TEST FOR 0
2867 010706 005704          TST     R4          ;;SUPPRESS THIS 0?
2868 010710 001403          BEQ     55          ;;BR IF YES
2869 010712 005204          45:    INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S
2870 010714 052703 000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
2871 010720 052703 000040          55:    BIS     #' ,R3      ;;MAKE ASCII IF NOT ALREADY
2872 010724 110337 010770          MOVB     R3,#5          ;;SAVE FOR TYPING
2873 010730 104401 010770          TYPE     #5          ;;GO TYPE THIS DIGIT
2874 010734 105337 010772          75:    DECB     $OCNT      ;;COUNT BY 1
2875 010740 003347          BGT     25          ;;BR IF MORE TO DO
2876 010742 002402          BLT     65          ;;BR IF DONE
2877 010744 005204          INC     R4          ;;INSURE LAST DIGIT ISN'T A BLANK
2878 010746 000744          BR      25          ;;GO DO THE LAST DIGIT
2879 010750 012605          65:    MOV     (SP)+,R5      ;;RESTORE R5
2880 010752 012604          MOV     (SP)+,R4      ;;RESTORE R4
2881 010754 012603          MOV     (SP)+,R3      ;;RESTORE R3
2882 010756 016666 000002 000004  MOV     2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
2883 010764 012616          MOV     (SP)+,(SP)
2884 010766 000002          RTI
2885 010770          85:    .BYTE 0          ;;RETURN
                ;;STORAGE FOR ASCII DIGIT
    
```

2886 010771 000  
2887 010772 000  
2888 010773 000  
2889 010774 000000

.BYTE 0  
SOCNT: .BYTE 0  
SOFILL: .BYTE 0  
SOMODE: .WORD 0  
: TERMINATOR FOR TYPE ROUTINE  
: OCTAL DIGIT COUNTER  
: ZERO FILL SWITCH  
: NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

\*\*\*\*\*  
: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT  
: SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE  
: NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED  
: BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE  
: REPLACED WITH SPACES.

: CALL:  
: \* MOV NUM, -(SP) : PUT THE BINARY NUMBER ON THE STACK  
: \* TYPDS : GO TO THE ROUTINE

2903 010776  
2904 010776 010046  
2905 011000 010146  
2906 011002 010246  
2907 011004 010346  
2908 011006 010546  
2909 011010 012746 020200  
2910 011014 016605 000020  
2911 011020 100004  
2912 011022 005405  
2913 011024 112766 000055 000001  
2914 011032 005000  
2915 011034 012703 011212  
2916 011040 112723 000040  
2917 011044 005002  
2918 011046 016001 011202  
2919 011052 160105  
2920 011054 002402  
2921 011056 005202  
2922 011060 000774  
2923 011062 060105  
2924 011064 005702  
2925 011066 001002  
2926 011070 105716  
2927 011072 100407  
2928 011074 106316  
2929 011076 103003  
2930 011100 116663 000001 177777  
2931 011106 052702 000060  
2932 011112 052702 000040  
2933 011116 110223  
2934 011120 005720  
2935 011122 020027 000010  
2936 011126 002746  
2937 011130 003002  
2938 011132 010502  
2939 011134 000764  
2940 011136 105726  
2941 011140 100003

STYPDS:  
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 R5, -(SP) : PUSH R5 ON STACK  
MOV #20200, -(SP) : SET BLANK SWITCH AND SIGN  
MOV 20(SP), R5 : GET THE INPUT NUMBER  
BPL 1\$ : BR IF INPUT IS POS.  
NEG R5 : MAKE THE BINARY NUMBER POS.  
MOVB #'-, 1(SP) : MAKE THE ASCII NUMBER NEG.  
1\$: CLR R0 : ZERO THE CONSTANTS INDEX  
MOV #S0BLK, R3 : SETUP THE OUTPUT POINTER  
MOVB #' , (R3)+ : SET THE FIRST CHARACTER TO A BLANK  
2\$: CLR R2 : CLEAR THE BCD NUMBER  
MOV \$DTBL(R0), R1 : GET THE CONSTANT  
3\$: SUB R1, R5 : FORM THIS BCD DIGIT  
BLT 4\$ : BR IF DONE  
INC R2 : INCREASE THE BCD DIGIT BY 1  
BR 3\$  
4\$: ADD R1, R5 : ADD BACK THE CONSTANT  
TST R2 : CHECK IF BCD DIGIT=0  
BNE 5\$ : FALL THROUGH IF 0  
TSTB (SP) : STILL DOING LEADING 0'S?  
BHI 7\$ : BR IF YES  
ASLB (SP) : MSD?  
5\$: BCC 6\$ : BR IF NO  
MOVB 1(SP), -1(R3) : YES--SET THE SIGN  
6\$: BIS #'0, R2 : MAKE THE BCD DIGIT ASCII  
7\$: BIS #' , R2 : MAKE IT A SPACE IF NOT ALREADY A DIGIT  
MOVB R2, (R3)+ : PUT THIS CHARACTER IN THE OUTPUT BUFFER  
TST (R0)+ : JUST INCREMENTING  
CMP R0, #10 : CHECK THE TABLE INDEX  
BLT 2\$ : GO DO THE NEXT DIGIT  
BGT 8\$ : GO TO EXIT  
MOV R5, R2 : GET THE LSD  
BR 6\$ : GO CHANGE TO ASCII  
8\$: TSTB (SP)+ : WAS THE LSD THE FIRST NON-ZERO?  
BPL 9\$ : BR IF NO

```

2942 011142 116663 177777 177776 95: MOVB -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
2943 011150 105013 CLAB (R3) ;; SET THE TERMINATOR
2944 011152 012605 MOV (SP)+,R5 ;; POP STACK INTO R5
2945 011154 012603 MOV (SP)+,R3 ;; POP STACK INTO R3
2946 011156 012602 MOV (SP)+,R2 ;; POP STACK INTO R2
2947 011160 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
2948 011162 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
2949 011164 104401 011212 TYPE SDBLK ;; NOW TYPE THE NUMBER
2950 011170 016666 000002 000004 MOV 2(SP),4(SP) ;; ADJUST THE STACK
2951 011176 012616 MOV (SP)+,(SP)
2952 011200 000002 RTI ;; RETURN TO USER

```

```

SOTBL: 10000.
        1000.
        100.
        10.
SDBLK: .BLKW 4

```

.SBTTL TTY INPUT ROUTINE

\*\*\*\*\*

```

2953 011222 000000 ENABL LSB
2954 011224 000000 STKCNT: .WORD 0 ;; NUMBER OF ITEMS IN QUEUE
2955 011226 000000 STKQIN: .WORD 0 ;; INPUT POINTER
2956 011230 000001 STKQOUT: .WORD 0 ;; OUTPUT POINTER
                011231 STKQSRV: .BLKW 1 ;; TTY KEYBOARD QUEUE
                011232 STKQEND=.
                .EVEN

```

```

;#TK INITIALIZE ROUTINE
;#THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;#SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT

```

```

2973 ;#CALL:
2974 ;# JSR PC,STKINT
2975 ;# RETURN
2976
2977 STKINT: CLR STKCNT ;; CLEAR COUNT OF ITEMS IN QUEUE
2978 MOV #STKQSRV,STKQIN ;; MOVE THE STARTING ADDRESS OF THE
2979 MOV STKQIN,STKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
2980 MOV #STKSRV,STKVEC ;; INITIALIZE THE KEYBOARD VECTOR
2981 MOV #200,STKVEC+2 ;; "BR" LEVEL 4
2982 TST STKB ;; CLEAR DONE FLAG
2983 MOV #100,STKS ;; ENABLE TTY KEYBOARD INTERRUPT
2984 RTS PC ;; RETURN TO CALLER

```

```

;#TK SERVICE ROUTINE
;#THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;#BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;#IT IN THE QUEUE.
;#IF THE CHARACTER IS A "CONTROL-C" (1C) STKINT IS CALLED AND
;#UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (STARTS)

```

```

2994 011302 117746 167640 STKSRV: NOVB STKB,-(SP) ;; PICKUP THE CHARACTER
2995 011306 042716 177600 BIC #1C17,(SP) ;; STRIP THE JUNK
2996 011312 021627 000003 CMP (SP),#3 ;; IS IT A CONTROL C?
2997 011316 001007 BNE IS ;; BRANCH IF NO

```

E06

MD-11-DZRPZ-B, RPIIE DRIVE POSITIONING TEST  
DZRPZB.CMB TTY INPUT ROUTINE

MACY11 27(732) 02-NOV-76 15:10 PAGE 70

2998	011320	104401	012441		TYPE	SCNTLC	:: TYPE A CONTROL-C (1C)
2999	011324	004737	011232		JSR	PC,STKINT	:: INIT THE KEYBOARD
3000	011330	005726			TST	(SP)+	:: CLEAN UP STACK
3001	011332	000137	002710		JMP	START5	:: CONTROL C RESTART
3002	011336	021627	000007	18:	CMP	(SP),#7	:: IS IT A CONTROL G?
3003	011342	001004			BNE	25	:: BRANCH IF NO
3004	011344	022737	000176	001140	CMP	#SWREG,SWR	:: IS SOFT-SWR SELECTED?
3005	011352	001500			BEQ	65	:: GO TO SWR CHANGE
3006							
3007	011354			25:			
3008	011354	022737	000001	011222	CMP	#1,STKCNT	:: IS THE QUEUE FULL?
3009	011362	001004			BNE	35	:: BRANCH IF NO
3010	011364	104401	001212		TYPE	SBELL	:: RING THE TTY BELL
3011	011370	005726			TST	(SP)+	:: CLEAN CHARACTER OFF OF STACK
3012	011372	000451			BR	55	:: EXIT
3013	011374	021627	000023	35:	CMP	(SP),#23	:: IS IT A CONTROL-S?
3014	011400	001021			BNE	325	:: BRANCH IF NO
3015	011402	005077	167536		CLR	2STKS	:: DISABLE TTY KEYBOARD INTERRUPTS
3016	011406	005726			TST	(SP)+	:: CLEAN CHAR OFF STACK
3017	011410	105777	167530	318:	TSTB	2STKS	:: WAIT FOR A CHAR
3018	011414	100375			BPL	315	:: LOOP UNTIL ITS THERE
3019	011416	117746	167524		MOVB	2STKB, -(SP)	:: GET THE CHARACTER
3020	011422	042716	177600		BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
3021	011426	022627	000021		CMP	(SP)+, #21	:: IS IT A CONTROL-0?
3022	011432	001366			BNE	315	:: BRANCH IF NO
3023	011434	012777	000100	167502	MOV	#100, 2STKS	:: REENABLE TTY KEYBOARD INTERRUPTS
3024	011442	000002			RTI		:: RETURN
3025	011444	005237	011222	325:	INC	STKCNT	:: COUNT THIS CHARACTER
3026	011450	021627	000140		CMP	(SP), #140	:: IS IT UPPER CASE?
3027	011454	002405			BLT	45	:: BRANCH IF YES
3028	011456	021627	000175		CMP	(SP), #175	:: IS IT A SPECIAL CHAR?
3029	011462	003002			BGT	45	:: BRANCH IF YES
3030	011464	042716	000040		BIC	#40, (SP)	:: MAKE IT UPPER CASE
3031	011470	112677	177530	45:	MOVB	(SP)+, 2STKQIN	:: AND PUT IT IN QUEUE
3032	011474	005237	011224		INC	STKQIN	:: UPDATE THE POINTER
3033	011500	023727	011224	011231	CMP	STKQIN, #STKQEND	:: GO OFF THE END?
3034	011506	001003			BNE	55	:: BRANCH IF NO
3035	011510	012737	011230	011224	MOV	#STKQSR, STKQIN	:: RESET THE POINTER
3036	011516	000002		55:	RTI		:: RETURN

```

*****
#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 INTERRUPT MODE.

```

3043	011520	022737	000176	001140	\$CKSWR: CMP	#SWREG,SWR	:: IS THE SOFT-SWR SELECTED
3044	011526	001124			BNE	155	:: EXIT IF NOT
3045	011530	105777	167410		TSTB	2STKS	:: IS A CHAR WAITING?
3046	011534	100121			BPL	155	:: IF NOT, EXIT
3047	011536	117746	167404		MOVB	2STKB, -(SP)	:: YES
3048	011542	042716	177600		BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
3049	011546	021627	000007		CMP	(SP), #7	:: IS IT A CONTROL-G?
3050	011552	001300			BNE	25	:: IF NOT, PUT IT IN THE TTY QUEUE
3051							:: AND EXIT
3052							
3053							

```

*****

```

```

3054                                     :#CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
3055                                     :#ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
3056                                     :#CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
3057 011554 123727 001134 000001 6S:   CMPB   SAUTOB, #1      :ARE WE RUNNING IN AUTO-MODE?
3058 011562 001674                BEQ     2S           :BRANCH IF YES
3059 011564 005726                TST    (SP)+        :CLEAR CONTROL-G OFF STACK
3060 011566 004737 011232        JSR    PC, STKINT   :FLUSH THE TTY INPUT QUEUE
3061 011572 005077 167346        CLR    2STKS        :DISABLE TTY KEYBOARD INTERRUPTS
3062 011576 112737 000001 001135  MOVB   #1, SINTAG   :SET INTERRUPT MODE INDICATOR
3063
3064 011604 104401 012453        TYPE   , SCNTLG     :ECHO THE CONTROL-G (↑G)
3065 011610 104401 012460  SGTSWR: TYPE   , SMSWR     :TYPE CURRENT CONTENTS
3066 011614 013746 000176        MOV    SWREG, -(SP) :SAVE SWREG FOR TYPEOUT
3067 011620 104402                TYPOC                :GO TYPE--OCTAL ASCII(ALL DIGITS)
3068 011622 104401 012471        TYPE   , SMNEW      :PROMPT FOR NEW SWR
3069 011626 005046                CLR    -(SP)         :CLEAR COUNTER
3070 011630 005046                CLR    -(SP)         :THE NEW SWR
3071 011632 105777 167306  7S:   TSTB   2STKS        :CHAR THERE?
3072 011636 100375                BPL    7S           :IF NOT TRY AGAIN
3073
3074 011640 117746 167302        MOVB   2STKB, -(SP) :PICK UP CHAR
3075 011644 042716 177600        BIC    #1C177, (SP) :MAKE IT 7-BIT ASCII
3076
3077 011650 021627 000003        CMP    (SP), #3     :IS IT A CONTROL-C?
3078 011654 001015                BNE    9S           :BRANCH IF NOT
3079 011656 104401 012441        TYPE   , SCNTLC     :YES, ECHO CONTROL-C (↑C)
3080 011662 062706 000006        ADD    #6, SP        :CLEAN UP STACK
3081 011666 123727 001135 000001  CMPB   SINTAG, #1   :REENABLE TTY KEYBOARD INTERRUPTS?
3082 011674 001003                BNE    8S           :BRANCH IF NO
3083 011676 012777 000100 167240  MOV    #100, 2STKS  :ALLOW TTY KEYBOARD INTERRUPTS
3084 011704 000137 002710  8S:   JMP     STARTS      :CONTROL-C RESTART
3085
3086
3087 011710 021627 000025  9S:   CMP    (SP), #25   :IS IT A CONTROL-U?
3088 011714 001005                BNE    10S          :BRANCH IF NOT
3089 011716 104401 012446        TYPE   , SCNTLU     :YES, ECHO CONTROL-U (↑U)
3090 011722 062706 000006  20S:  ADD    #6, SP        :IGNORE PREVIOUS INPUT
3091 011726 000737                BR     19S          :LET'S TRY IT AGAIN
3092
3093
3094 011732 021627 000015  10S:  CMP    (SP), #15   :IS IT A <CR>?
3095 011734 001022                BNE    16S          :BRANCH IF NO
3096 011736 005766 000004        TST    4(SP)        :YES, IS IT THE FIRST CHAR?
3097 011742 001403                BEQ    11S          :BRANCH IF YES
3098 011744 016677 000002 167166  MOV    2(SP), 2SMR  :SAVE NEW SWR
3099 011752 062706 000006  11S:  ADD    #6, SP        :CLEAR UP STACK
3100 011756 104401 001217  14S:  TYPE   , SCRLF      :ECHO <CR> AND <LF>
3101 011762 123727 001135 000001  CMPB   SINTAG, #1   :RE-ENABLE TTY KBD INTERRUPTS?
3102 011770 001003                BNE    15S          :BRANCH IF NOT
3103 011772 012777 000100 167144  MOV    #100, 2STKS  :RE-ENABLE TTY KBD INTERRUPTS
3104 012000 000002                RTI                    :RETURN
3105 012002 004737 010500  16S:  JSR    PC, STYPEC   :ECHO CHAR
3106 012006 021627 000060        CMP    (SP), #60    :CHAR < 0?
3107 012012 002420                BLT    18S          :BRANCH IF YES
3108 012014 021627 000067        CMP    (SP), #67    :CHAR > 7?
3109 012020 003015                BGT    18S          :BRANCH IF YES

```

```

3110 012022 042726 000060      BIC      #60,(SP)+      ;;STRIP-OFF ASCII
3111 012026 005766 000002      TST      2(SP)         ;;IS THIS THE FIRST CHAR
3112 012032 001403              BEQ      17$          ;;BRANCH IF YES
3113 012034 006316              ASL      (SP)         ;;NO, SHIFT PRESENT
3114 012036 006316              ASL      (SP)         ;;CHAR OVER TO MAKE
3115 012040 006316              ASL      (SP)         ;;ROOM FOR NEW ONE.
3116 012042 005266 000002      17$: INC      2(SP)         ;;KEEP COUNT OF CHAR
3117 012046 056616 177776      BIS      -2(SP),(SP)  ;;SET IN NEW CHAR
3118 012052 000667              BR       7$          ;;GET THE NEXT ONE
3119 012054 104401 001216      18$: TYPE  $QUES      ;;TYPE ?(CR)(LF)
3120 012060 000720              BR       20$        ;;SIMULATE CONTROL-U
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132 012062 011646 000004 000002  SRDCHR: MOV      (SP),-(SP)  ;;PUSH DOWN THE PC AND
3133 012064 016666 000004 000002      MOV      4(SP),2(SP)  ;;THE PS
3134 012072 005066 000004              CLR      4(SP)        ;;GET READY FOR A CHARACTER
3135 012076 005046              CLR      -(SP)        ;;PUT NEW PS ON STACK
3136 012100 012746 012106      MOV      #64$,-(SP)   ;;PUT NEW PC ON STACK
3137 012104 000002              RTI                    ;;POP NEW PC AND PS
3138 012106
3139 012106 005737 011222      64$: TST      STKCNT    ;;WAIT ON A CHARACTER
3140 012112 001775              BEQ      1$          ;;
3141 012114 005337 011222      1$: DEC      STKCNT    ;;DECREMENT THE COUNTER
3142 012120 117766 177102 000004      MOVB    2$STKQOUT,4(SP) ;;GET ONE CHARACTER
3143 012126 005237 011226      INC      STKQOUT      ;;UPDATE THE POINTER
3144 012132 023727 011226 011231      CMP      STKQOUT,#$STKQEND ;;DID IT GO OFF OF THE END?
3145 012140 001003              BNE      2$          ;;BRANCH IF NO
3146 012142 012737 011230 011226      MOV      #$STKQRT,STKQOUT ;;RESET THE POINTER
3147 012150 000002              RTI                    ;;RETURN
3148
3149
3150
3151
3152
3153
3154
3155 012152 010346 000004 000002  SRDLIN: MOV      R3,-(SP)  ;;SAVE R3
3156 012154 005046              CLR      -(SP)        ;;CLEAR THE RUBOUT KEY
3157 012156 012703 012406      1$: MOV      #$TTYIN,R3  ;;GET ADDRESS
3158 012162 022703 012441      2$: CMP      #$TTYIN+27.,R3 ;;BUFFER FULL?
3159 012166 101456              BLOS    4$          ;;BR IF YES
3160 012170 104410              RDRCHR  ;;GO READ ONE CHARACTER FROM THE TTY
3161 012172 112613              MOVB    (SP)+,(R3)    ;;GET CHARACTER
3162 012174 122713 000177      10$: CMPB    #177,(R3)   ;;IS IT A RUBOUT
3163 012200 001022              BNE      5$          ;;BR IF NO
3164 012202 005716              TST      (SP)        ;;IS THIS THE FIRST RUBOUT?
3165 012204 001007              BNE      6$          ;;BR IF NO

```

```

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

```

```

*****
*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
*CALL:
*   RDLIN
*   RETURN HERE
*
*   INPUT A STRING FROM THE TTY
*   ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
*   TERMINATOR WILL BE A BYTE OF ALL 0'S

```



```

3166 012206 112737 000134 012404      MOVB      #' \, 9S      ;; TYPE A BACK SLASH
3167 012214 104401 012404              TYPE      9S
3168 012220 012716 177777              MOV      #-1, (SP)    ;; SET THE RUBOUT KEY
3169 012224 005303              DEC      R3           ;; BACKUP BY ONE
3170 012226 020327 012406      6S:      CMP      R3, #STTYIN  ;; STACK EMPTY?
3171 012232 103434              BLO      4S          ;; BR IF YES
3172 012234 111337 012404      MOVB      (R3), 9S    ;; SETUP TO TYPEOUT THE DELETED CHAR.
3173 012240 104401 012404      TYPE      9S        ;; GO TYPE
3174 012244 000746              BR      2S          ;; GO READ ANOTHER CHAR.
3175 012246 005716      5S:      TST      (SP)        ;; RUBOUT KEY SET?
3176 012250 001406              BEQ      7S          ;; BR IF NO
3177 012252 112737 000134 012404      MOVB      #' \, 9S    ;; TYPE A BACK SLASH
3178 012260 104401 012404      TYPE      9S
3179 012264 005016              CLR      (SP)        ;; CLEAR THE RUBOUT KEY
3180 012266 122713 000025      7S:      CNPB      #25, (R3)  ;; IS CHARACTER A CTRL U?
3181 012272 001003              BNE      8S          ;; BR IF NO
3182 012274 104401 012446      TYPE      , SCNTLU   ;; TYPE A CONTROL "U"
3183 012300 000726              BR      1S          ;; GO START OVER
3184 012302 122713 000022      8S:      CNPB      #22, (R3)  ;; IS CHARACTER A "r"?
3185 012306 001011              BNE      3S          ;; BRANCH IF NO
3186 012310 105013              CLRB      (R3)       ;; CLEAR THE CHARACTER
3187 012312 104401 001217      TYPE      , SCRLF   ;; TYPE A "CR" & "LF"
3188 012316 104401 012406      TYPE      , STTYIN  ;; TYPE THE INPUT STRING
3189 012322 000717              BR      2S          ;; GO PICKUP ANOTHER CHACTER
3190 012324 104401 001216      4S:      TYPE      , SQUES   ;; TYPE A '?'
3191 012330 000712              BR      1S          ;; CLEAR THE BUFFER AND LOOP
3192 012332 111337 012404      3S:      MOVB      (R3), 9S   ;; ECHO THE CHARACTER
3193 012336 104401 012404      TYPE      9S
3194 012342 122723 000015      CNPB      #15, (R3)+  ;; CHECK FOR RETURN
3195 012346 001305              BNE      2S          ;; LOOP IF NOT RETURN
3196 012350 105063 177777      CLRB      -(R3)     ;; CLEAR RETURN (THE 15)
3197 012354 104401 001220      TYPE      , SLF     ;; TYPE A LINE FEED
3198 012360 005726              TST      (SP)+      ;; CLEAN RUBOUT KEY FROM THE STACK
3199 012362 012603              MOV      (SP)+, R3  ;; RESTORE R3
3200 012364 011646              MOV      (SP), -(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3201 012366 016666 000004 000002      MOV      4(SP), 2(SP) ;; FIRST ASCII CHARACTER ON IT
3202 012374 012766 012406 000004      MOV      #STTYIN, 4(SP)
3203 012402 000002              RTI
3204 012404              9S:      .BYTE      0      ;; STORAGE FOR ASCII CHAR. TO TYPE
3205 012405              .BYTE      0      ;; TERMINATOR
3206 012406 000033      STTYIN:  .BLKB     27.  ;; RESERVE 27. BYTES FOR TTY INPUT
3207 012441              136 006503 000012  SCNTLC:  .ASCIZ  /tC/<15><12>  ;; CONTROL "C"
3208 012446 052536 005015 000  SCNTLU:  .ASCIZ  /tU/<15><12>  ;; CONTROL "U"
3209 012453              136 006507 000012  SCNTLG:  .ASCIZ  /tG/<15><12>  ;; CONTROL "G"
3210 012460 005015 053523 020122  SMSWR:  .ASCIZ  <15><12>/SWR = /
3211 012466 020075              000
3212 012471              040 047040 053505  SMNEW:  .ASCIZ  / NEW = /
3213 012476 036440 000040

```

.SBTTL SCOPE HANDLER ROUTINE

```

3214
3215
3216
3217
3218      ;; *****
3219      ;; THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3220      ;; AND LOAD THE TEST NUMBER(STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3221      ;; AND LOAD THE ERROR FLAG (SERFLG) INTO DISPLAY<15:08>
3222      ;; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:

```

3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239  
3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260  
3261  
3262  
3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277

```

;#SW14=1      LOOP ON TEST
;#SW11=1      INHIBIT ITERATIONS
;#SW09=1      LOOP ON ERROR
;CALL
;#           SCOPE           ;;SCOPE=IOT

SSCOPE:
012502 104407 040000 166426 1S: CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
012502 032777 040000 166426 1S: BIT #BIT14,SWR  ;;LOOP ON PRESENT TEST?
012512 001076 040000 166426 1S: BNE $OVER      ;;YES IF SW14=1
;#####START OF CODE FOR THE XOR TESTER#####
3233 012514 000416 040000 166426 1S: SXTSTR: BR 6S  ;;IF RUNNING ON THE "XOR" TESTER CHANGE
; THIS INSTRUCTION TO A "NOP" (NOP=240)
3235 012516 013746 000004 000004 1S: MOV @ERRVEC, -(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3236 012522 012737 012542 000004 1S: MOV #SS, @ERRVEC  ;;SET FOR TIMEOUT
3237 012530 005737 177060 000004 1S: TST @177060      ;;TIME OUT ON XOR?
3238 012534 012637 000004 000004 1S: MOV (SP)+, @ERRVEC ;;RESTORE THE ERROR VECTOR
3239 012540 000450 000004 000004 1S: BR $SVLAD        ;;GO TO THE NEXT TEST
3240 012542 022626 000004 000004 1S: 5S: CMP (SP)+, (SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
3241 012544 012637 000004 000004 1S: MOV (SP)+, @ERRVEC ;;RESTORE THE ERROR VECTOR
3242 012550 000413 000004 000004 1S: BR 7S           ;;LOOP ON THE PRESENT TEST
3243 012552 000413 000004 000004 1S: 6S: ;#####END OF CODE FOR THE XOR TESTER#####
3244 012552 105737 001103 001103 2S: TSTB $ERFLG     ;;HAS AN ERROR OCCURRED?
3245 012556 001421 001103 001103 2S: BEQ 3S          ;;BR IF NO
3246 012560 123737 001115 001103 2S: CMPB $ERMAX, $ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
3247 012566 101015 001103 001103 2S: BHI 3S          ;;BR IF NO
3248 012570 032777 001000 166342 2S: BIT #BIT09, SWR ;;LOOP ON ERROR?
3249 012576 001404 001000 166342 2S: BEQ 4S          ;;BR IF NO
3250 012600 013737 001110 001106 7S: MOV $LPERR, $LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
3251 012606 000440 001110 001106 7S: BR $OVER
3252 012610 105037 001103 001106 4S: CLRB $ERFLG     ;;ZERO THE ERROR FLAG
3253 012614 005037 001206 001106 4S: CLR $TIMES      ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
3254 012620 000412 001206 001106 4S: BR 1S          ;;ESCAPE TO THE NEXT TEST
3255 012622 032777 004000 166310 3S: BIT #BIT11, SWR ;;INHIBIT ITERATIONS?
3256 012630 001006 004000 166310 3S: BNE 1S          ;;BR IF YES
3257 012632 005237 001104 001104 3S: INC $ICNT       ;;INCREMENT ITERATION COUNT
3258 012636 023737 001206 001104 3S: CMP $TIMES, $ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
3259 012644 002021 001206 001104 3S: BGE $OVER      ;;BR IF MORE ITERATION REQUIRED
3260 012646 012737 000001 001104 1S: MOV #1, $ICNT   ;;REINITIALIZE THE ITERATION COUNTER
3261 012654 013737 012724 001206 1S: MOV $SMXCNT, $TIMES ;;SET NUMBER OF ITERATIONS TO DO
3262 012662 105237 001102 001106 1S: SSVLAD: INCB $STSTM ;;COUNT TEST NUMBERS
3263 012666 011637 001106 001106 1S: MOV (SP), $LPADR ;;SAVE SCOPE LOOP ADDRESS
3264 012672 011637 001110 001106 1S: MOV (SP), $LPERR ;;SAVE ERROR LOOP ADDRESS
3265 012676 005037 001210 001106 1S: CLR $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3266 012702 112737 000001 001115 1S: MOVB #1, $ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3267 012710 013777 001102 166224 1S: $OVER: MOV $STSTM, @DISPLAY ;;DISPLAY TEST NUMBER
3268 012716 013716 001106 166224 1S: MOV $LPADR, (SP) ;;FUDGE RETURN ADDRESS
3269 012722 000002 001106 166224 1S: RTI            ;;FIXES PS
3270 012724 000001 001106 166224 1S: $SMXCNT: 1     ;;MAX. NUMBER OF ITERATIONS

.SBTTL SAVE AND RESTORE RO-RS ROUTINES

;#####
;#SAVE RO-RS
;CALL:
;# SAVREG

```

```

3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289 012726
3290 012726 010046
3291 012730 010146
3292 012732 010246
3293 012734 010346
3294 012736 010446
3295 012740 010546
3296 012742 016646 000022
3297 012746 016646 000022
3298 012752 016646 000022
3299 012756 016646 000022
3300 012762 000002

```

```

: #UPON RETURN FROM SSAVREG THE STACK WILL LOOK LIKE:
: *
: *TOP---(+16)
: * +2---(+18)
: * +4---R5
: * +6---R4
: * +8---R3
: * +10---R2
: * +12---R1
: * +14---R0

```

```

SSAVREG:
MOV RO,-(SP) ;; PUSH RO ON STACK
MOV R1,-(SP) ;; PUSH R1 ON STACK
MOV R2,-(SP) ;; PUSH R2 ON STACK
MOV R3,-(SP) ;; PUSH R3 ON STACK
MOV R4,-(SP) ;; PUSH R4 ON STACK
MOV R5,-(SP) ;; PUSH R5 ON STACK
MOV 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

```

```

3301
3302
3303
3304
3305 012764
3306 012764 012666 000022
3307 012770 012666 000022
3308 012774 012666 000022
3309 013000 012666 000022
3310 013004 012605
3311 013006 012604
3312 013010 012603
3313 013012 012602
3314 013014 012601
3315 013016 012600
3316 013020 000002

```

```

: #RESTORE RO-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

```

.SBTTL INTEGER MULTIPLY ROUTINE

```

3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332 013022
3333 013022 010046

```

```

: *****
: #CALL
: * MOV MULTIPLIER,-(SP)
: * MOV MULTIPLICAND,-(SP)
: * JSR PC,#SMULT
: * RETURN ;; PRODUCT IS ON THE STACK
: *
: * STACK PRODUCT
: * -----
: * TOP LSB'S
: * +2 MSB'S
SMULT:
MOV RO,-(SP) ;; PUSH RO ON STACK

```

```

3334 013024 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
3335 013026 010246      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
3336 013030 005046      CLR      -(SP)         ;; CLEAR THE SIGN KEY
3337 013032 016601 000012  MOV      12(SP),R1     ;; GET THE MULTIPLICAND
3338 013036 100002      BPL      1$           ;; BR IF PLUS
3339 013040 005216      INC      (SP)         ;; SET THE SIGN KEY
3340 013042 005401      NEG      R1           ;; MAKE THE MULTIPLICAND POSTIVE
3341 013044 016602 000014  1$:      MOV      14(SP),R2     ;; GET THE MULTIPLIER
3342 013050 100002      BPL      2$           ;; BR IF PLUS
3343 013052 005316      DEC      (SP)         ;; UPDATE THE SIGN KEY
3344 013054 005402      NEG      R2           ;; MAKE THE MULTIPLIER POSTIVE
3345 013056 012746 000021  2$:      MOV      #17,-(SP)     ;; SET THE LOOP COUNT
3346 013062 005000      CLR      R0           ;; SETUP FOR THE MULTIPLY LOOP
3347 013064 103001 3$:      BCC      4$           ;; DON'T ADD IF MULTIPLICAND = 0
3348 013066 060200      ADD      R2,R0
3349 013070 006000 4$:      ROR      R0           ;; POSITION THE PARITIAL PRODUCT AND
3350 013072 006001      ROR      R1           ;; THE MULTIPLICAND
3351 013074 005316      DEC      (SP)         ;; HAS ALL BITS OF THE MULTIPLICAND BEEN DONE?
3352 013076 001372      BNE      3$           ;; BR IF NO
3353 013100 022616      CMP      (SP)+,(SP)   ;; SHOULD PRODUCT BE NEGATIVE?
3354 013102 001403      BEQ      5$           ;; GO TO EXIT IF NO
3355 013104 005400      NEG      R0           ;; YES--SO MAKE IT SO
3356 013106 005401      NEG      R1
3357 013110 005600      SOC      R0
3358 013112 005726 5$:      TST      (SP)+       ;; CLEAR SIGN INFO. OFF OF STACK
3359 013114 010066 000012  MOV      R0,12(SP)     ;; PUT THE PRODUCT ON THE STACK (MSB'S)
3360 013120 010166 000010  MOV      R1,10(SP)     ;; LSB'S
3361 013124 012602      MOV      (SP)+,R2     ;; POP STACK INTO R2
3362 013126 012601      MOV      (SP)+,R1     ;; POP STACK INTO R1
3363 013130 012600      MOV      (SP)+,R0     ;; POP STACK INTO R0
3364 013132 000207      RTS      PC

```

.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.

```

```

3374 013134 010046      STRAP:  MOV      R0,-(SP)  ;; SAVE R0
3375 013136 016600 000002  MOV      2(SP),R0      ;; GET TRAP ADDRESS
3376 013142 005740      TST      -(R0)         ;; BACKUP BY 2
3377 013144 111000      MOV      (R0),R0      ;; GET RIGHT BYTE OF TRAP
3378 013146 006300      ASL      R0           ;; POSITION FOR INDEXING
3379 013150 016000 013170  MOV      STRPAD(R0),R0  ;; INDEX TO TABLE
3380 013154 000200      RTS      R0           ;; GO TO ROUTINE

```

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

```

3385 013156 011646      STRAP2: MOV      (SP),-(SP)  ;; MOVE THE PC DOWN
3386 013160 016666 000004 000002  MOV      4(SP),2(SP)   ;; MOVE THE PSW DOWN
3387 013166 000002      RTI                    ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

3388  
3389

3390  
3391  
3392  
3393  
3394  
3395  
3396  
3397  
3398  
3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445

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

```

ROUTINE
-----
$TRPAD: .WORD $STRAP2
$TYPE    ;;CALL=TYPE    TRAP+1(104401)  TTY TYPEOUT ROUTINE
$TYPOC   ;;CALL=TYPOC   TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS   ;;CALL=TYPOS   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON   ;;CALL=TYPON   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS   ;;CALL=TYPDS   TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

$GTSWR   ;;CALL=GTSWR   TRAP+6(104406)  GET SOFT-SWR SETTING

$CKSWR   ;;CALL=CKSWR   TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
$RDCHR   ;;CALL=RDCHR   TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
$RDLIN   ;;CALL=RDLIN   TRAP+11(104411) TTY TYPEIN STRING ROUTINE
$SAVREG  ;;CALL=SAVREG  TRAP+12(104412) SAVE RO-RS ROUTINE
$RESREG  ;;CALL=RESREG  TRAP+13(104413) RESTORE RO-RS ROUTINE

```

.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

\*\*\*\*\*  
\*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN  
\*UNSIGNED DECIMAL ASCIZ NUMBER.

```

*CALL
*   MOV    NUMBER, -(SP)    ;;PUT BINARY NUMBER ON THE STACK
*   JSR    PC, @#$SB2D     ;;CALL
*   RETURN                               ;;ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK

```

```

1$20:  MOV    2(SP), 1$      ;;SAVE BINARY NUMBER
1$21:  MOV    #1$, -(SP)      ;;SET POINTER
1$22:  JSR    PC, @#$DB2D    ;;CALL DOUBLE LENGTH CONVERT
1$23:  ADD    #5, (SP)       ;;ONLY ALLOW FIVE CHARACTERS
1$24:  MOV    (SP)+, 2(SP)   ;;PICKUP POINTER
1$25:  RTS    PC           ;;RETURN
1$:   .WORD  0,0

```

.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 ASCIZ
*                                       ;;IS ON THE STACK

```

```

SDB2D: SAVREG  ;;SAVE REGISTERS
MOV    2(SP), R2    ;;PICKUP THE DATA POINTER
MOV    #$DECVL, R0  ;;GET ADDRESS OF "SDECVL" STRING

```

```

013170 013156
013172 010330
013174 010574
013176 010550
013200 010610
013202 010776
013204 011610
013206 011520
013210 012062
013212 012152
013214 012726
013216 012764
013220 016637 000002 013250
013226 012746 013250
013232 004737 013254
013236 062716 000005
013242 012666 000002
013246 000207
013250 000000 000000
013254 104412
013256 016602 000002
013262 012700 013434

```

```

3446 013266 010066 000002      MOV      RD,2(SP)      ;;PUT ADDRESS OF ASCIZ STRING ON STACK
3447 013272 012201      MOV      (R2)+,R1     ;;PICKUP THE BINARY NUMBER
3448 013274 012202      MOV      (R2)+,R2
3449 013276 012737 000012 013352      MOV      #10.,4$     ;;SET UP TO DO 10 CONVERSIONS
3450 013304 012704 013364      MOV      #STNPWR,R4   ;;ADDRESS OF TEN POWER
3451 013310 012705 013366      MOV      #STNPWR+2,R5
3452 013314 005003      1$: CLR      R3        ;;CLEAR PARTIAL
3453 013316 161401      2$: SUB      (R4),R1   ;;SUBTRACT TEN POWER
3454 013320 005602      SBC      R2
3455 013322 161502      SUB      (R5),R2
3456 013324 002402      BLT      3$          ;;BR IF TEN POWER TOO LARGE
3457 013326 005203      INC      R3          ;;ADD 1 TO PARTIAL
3458 013330 000772      BR       2$          ;;LOOP
3459 013332 062401      3$: ADD      (R4)+,R1  ;;RESTORE SUBTRACTED VALUE
3460 013334 005502      ADC      R2
3461 013336 062402      ADD      (R4)+,R2
3462 013340 022525      CMP      (R5)+,(R5)+ ;;MOVE TO NEXT TEN POWER
3463 013342 052703 000060      BIS      #'0,R3      ;;CHANGE PARTIAL TO ASCII
3464 013346 110320      MOVB     R3,(RD)+    ;;SAVE IT
3465 013350 005327      DEC      (PC)+      ;;DONE?
3466 013352 000000      4$: .WORD    0
3467 013354 001357      BNE      1$          ;;BR IF NO
3468 013356 105020      CLRB     (RD)+      ;;TERMINATOR
3469 013360 104413      RESREG   ;;RESTORE REGISTERS
3470 013362 000207      RTS      PC         ;;RETURN
3471 013364 145000      STNPWR: 145000      ;;1.0E09
3472 013366 035632      35632
3473 013370 160400      160400      ;;1.0E08
3474 013372 002765      2765
3475 013374 113200      113200      ;;1.0E07
3476 013376 000230      230
3477 013400 041100      041100      ;;1.0E06
3478 013402 000017      17
3479 013404 103240      103240      ;;1.0E05
3480 013406 000001      1
3481 013410 023420      23420      ;;1.0E04
3482 013412 000000      0
3483 013414 001750      1750      ;;1.0E03
3484 013416 000000      0
3485 013420 000144      144      ;;1.0E02
3486 013422 000000      0
3487 013424 000012      12      ;;1.0E01
3488 013426 000000      0
3489 013430 000001      1      ;;1.0E00
3490 013432 000000      0
3491 013434 000014      SDECVL: .BLKB 12.  ;;RESERVE STORAGE FOR ASCIZ STRING
3492
3493      .SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
3494
3495      ;*****
3496      ;THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
3497      ;LEADING NUMBERS.
3498      ;CALL
3499      ;# MOV      #NUMADR,-(SP) ;;FIRST ADDRESS OF ASCIZ STRING
3500      ;# JSR      PC,2#$$SUPRS
3501

```

3502  
 3503 013450 010046  
 3504 013452 016600 000004  
 3505 013456 105710  
 3506 013460 001403  
 3507 013462 122720 000060  
 3508 013466 001773  
 3509 013470 005300  
 3510 013472 010037 013500  
 3511 013476 104401  
 3512 013500 000000  
 3513 013502 012600  
 3514 013504 012616  
 3515 013506 000207

```

SSUPRS: MOV RO, -(SP)          ;; SAVE RO
          MOV 4(SP), RO        ;; PICKUP THE POINTER
1$:      TSTB (RO)             ;; TERMINATE OR?
          BEQ 2$               ;; BR IF YES
          CMPB #'0, (RO)+      ;; IS THIS AN ASCII "0" ?
          BEQ 1$               ;; BR IF YES
2$:      DEC RO                ;; BACKUP BY "1"
          MOV RO, 3$           ;; SAVE FOR TYPING
          TYPE                  ;; GO TYPE
3$:      .WORD 0                ;; ASCIZ POINTER GOES HERE
          MOV (SP)+, RO         ;; RESTORE RO
          MOV (SP)+, (SP)       ;; RESTORE THE STACK
          RTS PC                ;; RETURN
  
```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

;*****
;THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
;WITH A RANGE OF 0 TO 2(+33)-1.
;CALL:
;      JSR PC, $RAND          ;; CALL THE ROUTINE
;      RETURN                 ;; RETURN HERE THE RANDOM
;                               ;; NUMBER WILL BE IN
;                               ;; $HINUM, $LONUM
;
  
```

3528 013510  
 3529 013510 010046  
 3530 013512 010146  
 3531 013514 010246  
 3532 013516 013700 013610  
 3533 013522 013701 013606  
 3534 013526 012702 177771  
 3535 013532 006300  
 3536 013534 006101  
 3537 013536 005202  
 3538 013540 001374  
 3539 013542 063700 013610  
 3540 013546 005501  
 3541 013550 063701 013606  
 3542 013554 062700 001057  
 3543 013560 005501  
 3544 013562 062701 047401  
 3545 013566 010037 013610  
 3546 013572 010137 013606  
 3547 013576 012602  
 3548 013600 012601  
 3549 013602 012600  
 3550 013604 000207  
 3551 013606 176543  
 3552 013610 123456

```

$RAND:  MOV RO, -(SP)          ;; PUSH RO ON STACK
          MOV R1, -(SP)        ;; PUSH R1 ON STACK
          MOV R2, -(SP)        ;; PUSH R2 ON STACK
          MOV $LONUM, RO       ;; SET RO WITH LOW
          MOV $HINUM, R1       ;; SET R1 WITH HIGH
          MOV #7, R2           ;; SET SHIFT COUNT
1$:      ASL RO                 ;; SHIFT RO LEFT AND
          ROL R1                ;; ROTATE CARRY INTO R1 AND
          INC R2                ;; CHECK FOR DONE
          BNE 1$               ;; CONTINUE SHIFT LOOP
          ADD $LONUM, RO        ;; ADD NUMBER TO MAKE X 129
          ADC R1                ;; PROPOGATE CARRY
          ADD $HINUM, R1       ;; ADD NUMBER TO MAKE X 129
          ADD #1057, RO        ;; ADD LOW CONSTANT
          ADC R1                ;; PROPOGATE CARRY
          ADD #47401, R1        ;; ADD HIGH CONSTANT
          MOV RO, $LONUM       ;; SAVE RO
          MOV R1, $HINUM       ;; SAVE R1
          MOV (SP)+, R2        ;; POP STACK INTO R2
          MOV (SP)+, R1        ;; POP STACK INTO R1
          MOV (SP)+, RO        ;; POP STACK INTO RO
          RTS PC                ;; RETURN
$HINUM: .WORD 176543
$LONUM: .WORD 123456
  
```

```

;*****
.SBTTL RPINIT - INITIALIZE THE RP11 & SEE WHICH DRIVES ARE AVAILABLE
;THIS ROUTINE CLEARS THE RP11 AND DETERMINES WHICH DRIVES ARE AVAILABLE.
;THE TABLE 'DRVSTA' IS LOADED TO REFLECT THE SYSTEM STATUS.
  
```

3553  
 3554  
 3555  
 3556  
 3557

```

3558          :CALL
3559          :      JSR      PC, @RPINIT
3560          :      RETURN
3561
3562          RPINIT: SAVREG          ;SAVE RO-R5
3563          MOV      RPVEC, R1      ;VECTOR ADDRESS
3564          CLR      (R1)+          ;SET INTERRUPT ADDRESS TO ZERO
3565          MOV      RPPRIO, (R1)   ;RP11 PRIORITY
3566          CLR      DRVSTYP        ;CLEAR DRIVE TYPE STORAGE
3567          CLR      DRVSTA        ;SET DRIVE STATUS TO OFFLINE
3568          CLR      DRVSTA+2      ;SET DRIVE STATUS TO OFFLINE
3569          CLR      DRVSTA+4      ;SET DRIVE STATUS TO OFFLINE
3570          CLR      DRVSTA+6      ;SET DRIVE STATUS TO OFFLINE
3571          MOV      RPAOR, RO      ;PUT RP11 ADDRESS INTO RO
3572          MOV      @CLEAR, RPCS(RO) ;CLEAR THE RP11
3573          MOV      @CLEAR, RPCS(RO) ;CLEAR THE RP11 AGAIN
3574          MOV      #7, R1         ;'DRVSTA' TABLE INDEX
3575          1$: MOVB      R1, RPCS+1(RO) ;SELECT DRIVE
3576          BIT      @SUFU, RPDS(RO) ;SEE IF DRIVE UNSAFE
3577          BEQ      2$,           ;BR IF NOT UNSAFE
3578          MOVB      #-1, DRVSTA(R1) ;SET INDICATOR TO 'UNSAFE'
3579          BR       4$,           ;EXIT
3580          2$: BIT      @SUOL, RPDS(RO) ;IS DRIVE ONLINE ?
3581          BEQ      5$,           ;BR IF NOT
3582          BIT      @SUSI, RPDS(RO) ;SEEK INCOMPLETE SET ?
3583          BNE      3$,           ;BR IF SET
3584          BIT      @SUDY, RPDS(RO) ;IS DRIVE READY ?
3585          BEQ      5$,           ;BR IF NOT
3586          3$: MOVB      #1, DRVSTA(R1) ;SET DRIVE INDICATOR TO 'ONLINE'
3587          4$: BIT      @SURP03, RPDS(RO) ;IS THE DRIVE AN RP03 ?
3588          BEQ      5$,           ;BR IF NOT
3589          BISB      ATABIT(R1), DRVSTYP ;SET RP03 INDICATOR
3590          5$: DEC      R1         ;DECREMENT THE TABLE INDEX
3591          BPL      1$,           ;BR IF NOT FINISHED
3592          RESREG          ;RESTORE RO-R5
3593          RTS      PC           ;RETURN

```

```

; *****
; SBTTL SEEKS - INITIATE A DIRECT SEEK (OR HOME SEEK) OPERATION
; THIS ROUTINE INITIATES A SEEK OPERATION ON THE DRIVE ADDRESSED IN
; THE SECOND BYTE OF 'DPB'. 1 SECOND IS ALLOWED FOR THE SEEK
; TO COMPLETE; IF THE DRIVE HAS NOT INTERRUPTED ('AIE' IS SET)
; WITHIN THAT TIME, THE SUBSYSTEM IS RESET AND THE TESTING FOR THE
; DRIVE IS TERMINATED.

```

```

3602          :CALL
3603          :      JSR      PC, @SEEKS
3604          :      RETURN
3605
3606          SEEKS: SAVREG          ;SAVE RO-R5
3607          MOV      RPPRIO, PSW    ;RP11 PRIORITY
3608          MOV      RPAOR, RO      ;RP11 BUS ADDRESS
3609          MOV      $LPERR, -(SP)  ;SAVE OLD LOOP ON ERROR ADDRESS
3610          MOV      @SEEKS0, $LPERR ;CHANGE LOOP ON ERROR ADDRESS
3611          1$: MOV      @CLEAR, RPCS(RO) ;CLEAR THE CONTROLLER
3612          MOV      @CLEAR, RPCS(RO) ;CLEAR THE CONTROLLER
3613          MOVB      DPB+1, RPCS+1(RO) ;SELECT THE DRIVE

```



3614	014056	013737	001366	001262		MOV	DPB+2,CYL,DS	::DESIRED CYLINDER
3615	014064	013760	001366	000012		MOV	DPB+2,RPCA(RO)	::LOAD CYLINDER ADDRESS
3616	014072	013760	001370	000014		MOV	DPB+4,RPDA(RO)	::TRACK ADDRESS
3617	014100	012777	014264	165162		MOV	#SEEKS1,3RPVEC	::INTERRUPT RETURN
3618	014106	013760	001364	000004		MOV	DPB,RPCS(RO)	::START THE COMMAND
3619	014114	152760	000040	000005		BISB	#40,RPCS+1(RO)	::SET ATTENTION INTERRUPT ENABLE
3620	014122	005037	177776			CLR	PS	::ALLOW RP11 INTERRUPTS
3621	014126	010637	001242			MOV	SP,SPSAV	::SAVE THE CURRENT STACK POINTER
3622	014132	012737	004704	001300		MOV	#2500,STALLG	::SET STALL VALUE TO 2500MS
3623	014140	004037	015542			JSR	RO,STALLA	::WAIT THE SPECIFIED VALUE
3624	014144	001300				.WORD	STALLG	::STALL VALUE STORAGE
3625	014146	004737	015134			JSR	PC,SAVRP	::OPERATION TIMED OUT, SAVE THE REGISTERS
3626	014152	032737	001000	002004		BIT	#SUFU,SRPDS	::DRIVE UNSAFE ?
3627	014160	001403				BEQ	15	::BR IF NOT
3628	014162	104002				ERROR	2	::DRIVE UNSAFE
3629	014164	000137	015242			JMP	DSELECT	::DESELECT THE DRIVE
3630	014170	032737	040000	002004	15:	BIT	#SUOL,SRPDS	::DRIVE STILL ONLINE ?
3631	014176	001003				BNE	25	::BR IF IT IS
3632	014200	104010				ERROR	10	::DRIVE OFFLINE AFTER POSITIONING
3633	014202	000137	015242			JMP	DSELECT	::DESELECT THE DRIVE
3634	014206	032737	100000	002004	25:	BIT	#SUDY,SRPDS	::IS THE DRIVE READY ?
3635	014214	001003				BNE	35	::BR IF THE DRIVE IS READY
3636	014216	104016				ERROR	16	::DRIVE NOT READY AFTER POSITIONING
3637	014220	000137	015242			JMP	DSELECT	::DESELECT THE DRIVE
3638	014224	104005			35:	ERROR	5	::NO INTERRUPT FROM POSITIONING AFTER 1 SECOND
3639	014226	032737	000200	002010		BIT	#RDY,SRPCS	::IS THE CONTROLLER READY ?
3640	014234	001011				BNE	45	::BR IF IT IS
3641	014236	012760	000001	000004		MOV	#CLEAR,RPCS(RO)	::CLEAR THE CONTROLLER
3642	014244	012760	000001	000004		MOV	#CLEAR,RPCS(RO)	::CLEAR THE CONTROLLER
3643	014252	012777	000100	164664		MOV	#BIT06,3STKS	::RESTORE THE KEYBOARD INTERRUPT ENABLE
3644	014260	000177	164760		45:	JMP	38YPASS	::TERMINATE THE PRESENT TEST
3645	014264	005037	177776		SEEKS1:	CLR	PSW	::GO BACK TO PRIORITY 0
3646	014270	013706	001242			MOV	SPSAV,SP	::RESTORE THE STACK POINTER
3647	014274	013700	001266			MOV	RPADR,RO	::RP11 ADDRESS
3648	014300	004737	015134			JSR	PC,SAVRP	::SAVE THE RP11 REGISTERS
3649	014304	032737	004000	002004		BIT	#SUSI,SRPDS	::SEEK INCOMPLETE ?
3650	014312	001416				BEQ	SEEKS2	::BR IF NOT
3651	014314	012737	014324	001210		MOV	#15,SESCAPE	::ERROR 'ESCAPE' ADDRESS
3652	014322	104015				ERROR	15	::SEEK INCOMPLETE
3653	014324	005037	001210		15:	CLR	SESCAPE	::CLEAR THE ESCAPE ADDRESS
3654	014330	004737	015264			JSR	PC,RESTOR	::DO A HOME SEEK TO CLEAR THE DRIVE
3655	014334	032777	001000	164576		BIT	#SM09,3SMR	::LOOPING ON ERROR ?
3656	014342	001452				BEQ	SEEKS4	::BR IF NOT
3657	014344	000177	164540			JMP	35LPERR	::LOOP ON THE OPERATION
3658	014350	032737	100000	002010	SEEKS2:	BIT	#ERR,SRPCS	::'ERROR' SET ?
3659	014356	001410				BEQ	15	::BR IF NOT
3660	014360	104006				ERROR	6	::DRIVE ERROR AFTER POSITIONING
3661	014362	012760	000001	000004		MOV	#CLEAR,RPCS(RO)	::CLEAR THE CONTROLLER
3662	014370	012760	000001	000004		MOV	#CLEAR,RPCS(RO)	::CLEAR THE CONTROLLER
3663	014376	000415				BR	SEEKS3	::CHECK THE DISK'S POSITION
3664	014400	005001			15:	CLR	R1	::CLEAR R1
3665	014402	113701	001365			MOVB	DPB+1,R1	::DRIVE ADDRESS
3666	014406	136137	001314	002004		BITB	ATABIT(R1),SRPDS	::ATTENTION BIT SET FOR THE DRIVE
3667	014414	001001				BNE	25	::BR IF IT IS
3668	014416	104007				ERROR	7	::ATTENTION BIT NOT SET AFTER POSITIONING
3669	014420	023737	001366	002024	25:	CHP	DPB+2,\$SUCA	::'SUCA' CORRECT ?

```

3670 014426 001401 BEQ SEEKS3 ;BR IF IT IS
3671 014430 104011 ERROR 11 ;'SUCA' NOT CORRECT
3672 014432 004737 014524 SEEKS3: JSR PC,VERIFY ;VERIFY THE DISK'S POSITION
3673 014436 000414 BR SEEKS4 ;AT THE CORRECT CYLINDER
3674 014440 122737 000015 001364 CMPB #HOMSEK,DPB ;DOING A HOME SEEK ?
3675 014446 001410 BEQ SEEKS4 ;BR IF YES, NO SENSE DOING A HOME SEEK AGAIN
3676 014450 004737 015264 JSR PC,RESTOR ;DO A HOME SEEK
3677 014454 032777 001000 164456 BIT #SM09,JSWR ;LOOP ON THE OPERATION ?
3678 014462 001402 BEQ SEEKS4 ;BR IF NOT
3679 014464 000177 164420 JMP JSLPERR ;LOOP
3680 014470 032777 000010 164442 SEEKS4: BIT #SM03,JSWR ;STALL ?
3681 014476 001403 BEQ 15 ;BR IF NOT
3682 014500 004037 015546 JSR RD,STALL ;STALL
3683 014504 001276 .WORD STALL1 ;2SMS STALL
3684 014506 016037 000024 001254 15: MOV SUCA(RD),CYL.CR ;CURRENT CYLINDER ADDRESS
3685 014514 012637 001110 MOV (SP)+,SLPERR ;RESTORE THE LOOP ON ERROR ADDRESS
3686 014520 104413 RESREG ;RESTORE RD-RS
3687 014522 000207 RTS PC ;RETURN

```

```

*****
:SBTTL VERIFY - ROUTINE TO VERIFY THE POSITION AFTER A SEEK
:THIS ROUTINE VERIFIES THE POSITION OF THE DRIVE AFTER A SEEK
:BY READING A HEADER AT THE DESTINATION CYLINDER. THE HEADER
:READ IS FROM THE SECTOR WHICH IS 2 SECTORS FROM THE ADDRESS IN
:THE 'SOT' COUNTER. THE CYLINDER FIELD FROM 'DPB+2' IS COMPARED
:WITH THE CYLINDER FIELD FROM THE HEADER.
:CALL

```

```

3697 JSR PC,VERIFY
3698 RETURN
3699 ERROR RETURN ;NORMAL RETURN
3700 ;CYLINDER FIELDS DID NOT COMPARE
3701 014524 104412 VERIFY: SAVREG ;SAVE RD-RS
3702 014526 013746 001110 MOV SLPERR,-(SP) ;SAVE THE LOOP ON ERROR ADDRESS
3703 014532 032777 000001 164400 BIT #SM0,JSWR ;CHECK POSITION ?
3704 014540 001171 BNE VERIFY1 ;BR IF NOT
3705 014542 012737 014550 001110 MOV #VERIFY0,SLPERR ;LOOP ON ERROR ADDRESS
3706 014550 012737 177777 027142 VERIFY0: MOV #-1,BUFFER+2 ;CLEAR CYLINDER STORAGE
3707 014556 013700 001266 MOV RPAOR,RD ;LOAD RPII ADDRESS
3708 014562 016046 000014 MOV RPDA(RD),-(SP) ;GET THE 'SOT'
3709 014566 042716 177417 BIC #C360,(SP) ;SAVE THE 'SOT' BITS
3710 014572 006216 ASR (SP) ;RIGHT JUSTIFY THE BITS
3711 014574 006216 ASR (SP) ;RIGHT JUSTIFY THE BITS
3712 014576 006216 ASR (SP) ;RIGHT JUSTIFY THE BITS
3713 014600 006216 ASR (SP) ;RIGHT JUSTIFY THE BITS
3714 014602 062716 000002 ADD #2,(SP) ;FORM SECTOR ADDRESS FOR CHECK
3715 014606 022716 000012 CMP #10.,(SP) ;SEE IF THE ADDRESS OVERFLOWED
3716 014612 101005 BHI 25 ;BR IF NOT
3717 014614 103402 BLO 15 ;BR IF ADDR IS 11
3718 014616 005016 CLR (SP) ;SET SECTOR ADDRESS TO ZERO
3719 014620 000402 BR 25 ;CONTINUE
3720 014622 012716 000001 15: MOV #1,(SP) ;SET SECTOR ADDRESS TO ONE
3721 014626 012777 014776 164434 25: MOV #45,3RPVEC ;INTERRUPT ADDRESS
3722 014634 112637 001370 MOVB (SP)+,DPB+4 ;PUT VALUE INTO THE DPB
3723 014640 113737 001370 001260 MOVB DPB+4,SEC.RD ;SELECTED SECTOR FOR ERROR MESSAGE
3724 014646 113737 001371 001256 MOVB DPB+5,TRK.RD ;TRACK FOR ERROR MESSAGE
3725 014654 013746 001364 MOV DPB,-(SP) ;SETUP RPCS LOAD VALUE

```

# E07

```

3726 014660 112716 000017                    MOV#    #READ,(SP)            ;READ COMMAND CODE
3727 014664 052716 014100                    BIS    #MODE!HDR!IDE,(SP)    ;SET BITS FOR HEADER READ
3728 014670 012760 177775    000006                    MOV    #3,RPMC(RO)        ;LOAD WORD COUNT
3729 014676 012760 027140    000010                    MOV    #BUFFER,RPBA(RO)    ;BUFFER ADDRESS
3730 014704 013760 001370    000014                    MOV    DPB+4,RPDA(RO)      ;SECTOR ADDRESS
3731 014712 012660 000004                    MOV    (SP)+,RPCS(RO)      ;START THE ORDER
3732 014716 005037 177776                    CLR    PSW                ;SET PRIORITY BACK TO ZERO
3733 014722 010637 001242                    MOV    SP,SPSAV            ;SAVE THE CURRENT STACK POINTER
3734 014726 012737 004704    001300                    MOV    #2500,STALLG        ;SET STALL VALUE TO 2500MS
3735 014734 004037 015542                    JSR    RO,STALLA          ;WAIT THE SPECIFIED VALUE
3736 014740 001300                    .WORD    STALLG            ;STALL VALUE STORAGE
3737 014742 004737 015134                    JSR    PC,SAVRP            ;STORE THE RP11 REGISTERS
3738 014746 104014                    ERROR    14                    ;NO INTERRUPT WITHIN 1 SECOND
3739 014750 012760 000001    000004                    MOV    #CLEAR,RPCS(RO)    ;CLEAR THE CONTROLLER
3740 014756 012760 000001    000004                    MOV    #CLEAR,RPCS(RO)    ;CLEAR THE CONTROLLER
3741 014764 012777 000100    164152                    MOV    #BIT6,#STKS        ;SET TTY KEYBOARD INTERRUPT ENABLE
3742 014772 000177 164246                    JMP    #BYPASS            ;TERMINATE THE PRESENT TEST
3743 014776 005037 177776                    CLR    PSW                ;SET PRIORITY TO ZERO
3744 015002 013706 001242                    MOV    SPSAV,SP          ;RESTORE STACK POINTER
3745 015006 004737 015134                    JSR    PC,SAVRP            ;STORE THE RP11 REGISTERS
3746 015012 013700 001266                    MOV    RPADR,RO          ;PUT RP11 ADDRESS INTO RO
3747 015016 032737 100000    002010                    BIT    #ERR,#RPCS        ;DID THE ORDER TERMINTTE WITH AN ERROR ?
3748 015024 001410                    BEQ    6S                ;BR IF NOT
3749 015026 104012                    ERROR    12                    ;ERROR DURING POSITIONING VERIFICATION
3750 015030 012760 000001    000004                    MOV    #CLEAR,RPCS(RO)    ;CLEAR THE ERROR
3751 015036 012760 000001    000004                    MOV    #CLEAR,RPCS(RO)    ;CLEAR THE CONTROLLER
3752 015044 000427                    BR      VERIFY1          ;BYPASS POSITIONING CHECK
3753 015046                    6S:
3754 015046 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3755 015052 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3756 015056 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3757 015062 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3758 015066 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3759 015072 006237 027142                    ASR    BUFFER+2          ;ALIGN CYLINDER ADDRESS FOR COMPARE
3760 015076 023737 001366    027142                    CMP    DPB+2,BUFFER+2     ;SEE IF CYLINDER ADDR FROM HEADER IS CORRECT
3761 015104 001407                    BEQ    VERIFY1          ;BR IF OK
3762 015106 013737 027142    001126                    MOV    BUFFER+2,#BD0AT    ;CYLINDER ADDRESS FROM HEADER
3763 015114 104013                    ERROR    13                    ;CYLINDER ADDRESS DOES NOT COMPARE
3764 015116 062766 000002    000002                    ADD    #2,2(SP)          ;RETURN+2
3765 015124 012637 001110                    VERIFY1: MOV    (SP)+,#LPER        ;RESTORE LOOP ON ERROR ADDRESS
3766 015130 104413                    RESREG                    ;RESTORE RO-R5
3767 015132 000207                    RTS      PC                ;RETURN
3768
3769
3770                    ;*****
3771                    ;SBTTL SAVRP - STORE THE RP11E REGISTERS
3772                    ;THIS ROUTINE STORES THE RP11E REGISTERS FOR USE BY THE PROGRAM.
3773                    ;THE PROGRAM DOES NOT USE THE 'LIVE' REGISTERS FOR ERROR DETERMINATION.
3774                    ;CALL
3775                    ;      JSR      PC,SAVRP
3776                    ;      RETURN
3777 015134 010046                    SAVRP: MOV    RO,-(SP)        ;SAVE RO
3778 015136 013700 001266                    MOV    RPADR,RO          ;RP11 ADDRESS
3779 015142 016037 000000    002004                    MOV    RPDS(RO),#RPDS    ;STORE RPDS
3780 015150 016037 000002    002006                    MOV    RPER(RO),#RPER    ;STORE RPER
3781 015156 016037 000004    002010                    MOV    RPCS(RO),#RPCS    ;STORE RPCS

```

```

3782 015164 016037 000006 002012      MOV      RPWC(RO),SRPWC      :STORE RPWC
3783 015172 016037 000010 002014      MOV      RPBA(RO),SRPBA      :STORE RPBA
3784 015200 016037 000012 002016      MOV      RPCA(RO),SRPCA      :STORE RPCA
3785 015206 016037 000014 002020      MOV      RPD(RO),SRPD        :STORE RPD
3786 015214 016037 000016 002022      MOV      RPM1(RO),SRPM1      :STORE RPM1
3787 015222 016037 000024 002024      MOV      SUCA(RO),SSUCA      :STORE SUCA
3788 015230 016037 000026 002026      MOV      SILO(RO),SSILO      :STORE SILO
3789 015236 012600      MOV      (SP)+,RO           :RESTORE RO
3790 015240 000207      RTS      PC                 :RETURN
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800 015242 013701 001246      DSELCT: MOV      CHKDRV,R1      :GET DRIVE NUMBER
3801 015246 156137 001314 001252      BISB     ATABIT(R1),DRVBAD    ;USE AS INDEX TO LOAD ATA BIT FOR DRIVE
3802
3803 015254 005037 177776      CLR      PSW                ;IN DESELECT LOCATION.
3804 015260 000137 007464      JMP      @#SEOP              ;SET PRIORITY BACK TO ZERO
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815 015264 104412      RESTOR: SAVREG              ;SAVE RO-R5
3816 015266 013746 001110      MOV      SLPERR,-(SP)        ;SAVE LOOP ON ERROR ADDRESS
3817 015272 012737 015304 001110      MOV      @S,SLPERR          ;LOOP ON ERROR ADDRESS
3818 015300 013700 001266      MOV      RPAOR,RO           ;RP11 ADDRESS
3819 015304 012760 000001 000004 15:      MOV      @CLEAR,RPCS(RO)    ;CLEAR THE RP11
3820 015312 012760 000001 000004      MOV      @CLEAR,RPCS(RO)    ;CLEAR THE CONTROLLER
3821 015320 012777 015500 163742      MOV      @RESTRI,@RPVEC     ;INTERRUPT RETURN
3822 015326 013746 001364      MOV      DPB,-(SP)          ;SETUP RPCS CONTENTS
3823 015332 112716 000015      MOV      @HOMSEK,(SP)       ;COMMAND CODE
3824 015336 052716 020000      BIS      @AIE,(SP)          ;ATTENTION INTERRUPT ENABLE
3825 015342 012660 000004      MOV      (SP)+,RPCS(RO)     ;START THE HOME SEEK
3826 015346 005037 177776      CLR      PSW                ;SET PRIORITY TO ZERO
3827 015352 010637 001242      MOV      SP,SPSAV           ;SAVE THE STACK POINTER
3828 015356 012737 004704 001300      MOV      @2500,@STALLG     ;SET STALL VALUE TO 2500MS
3829 015364 004037 015542      JSR      RO,STALLA          ;WAIT THE SPECIFIED VALUE
3830 015370 001300      .WORD     STALLG            ;STALL VALUE STORAGE
3831 015372 004737 015134      JSR      PC,SAVRP          ;GET THE REGISTERS
3832 015376 032737 001000 002004      BIT      @SUFU,SRPDS        ;DRIVE UNSAFE ?
3833 015404 001403      BEQ      @#                ;BR IF NOT
3834 015406 104002      ERROR     @#                ;DRIVE UNSAFE
3835 015410 000137 015242      JMP      DSELCT            ;DESELECT THE DRIVE
3836 015414 032737 040000 002004 25:      BIT      @SUOL,SRPDS        ;IS THE DRIVE STILL ONLINE ?
3837 015422 001003      BNE     @#                ;BR IF IT IS
    
```

```

3838 015424 104010          ERROR 10          ;DRIVE OFFLINE AFTER POSITIONING
3839 015426 000137 015242  JMP     DSELECT ;DESELECT THE DRIVE
3840 015432 032737 100000 002004 3S:  BIT     #SUDY,SRPDS ;IS THE DRIVE READY ?
3841 015440 001003          BNE     45        ;BR IF IT IS
3842 015442 104016          ERROR 16        ;DRIVE NOT READY AFTER POSITIONING
3843 015444 000137 015242  JMP     DSELECT ;DESELECT THE DRIVE
3844 015450 104005          ERROR 5         ;DRIVE HAS NOT INTERRUPTED WITHIN 1 SEC
3845 015452 012760 000001 000004 4S:  MOV     #CLEAR,RPCS(RO) ;CLEAR THE CONTROLLER
3846 015460 012760 000001 000004  MOV     #CLEAR,RPCS(RO) ;CLEAR THE CONTROLLER
3847 015466 012777 000100 163450  MOV     #BIT6,STKS ;SET INTERRUPT ENABLE FOR TTY KEYBOARD
3848 015474 000177 163544          JMP     JBYPASS  ;TERMINATE THE PRESENT TEST
3849 015500 013706 001242          MOV     SPSAV,SP  ;RESTORE THE STACK POINTER
3850 015504 005037 177776          CLR     PSM      ;RESET THE PSM
3851 015510 004737 015134          JSR     PC,SAVRP ;SAVE THE REGISTERS
3852 015514 032737 004000 002004  BIT     #SUSI,SRPDS ;SEEK INCOMPLETE ?
3853 015522 001403          BEQ     25        ;BR IF NOT
3854 015524 104015          ERROR 15        ;SEEK INCOMPLETE
3855 015526 000137 015242  JMP     DSELECT ;DESELECT THE DRIVE
3856 015532 012637 001110 2S:  MOV     (SP)+,SLPERR ;RESTORE THE LOOP ON ERROR ADDRESS
3857 015536 104413          RESREG ;RESTORE RO-RS
3858 015540 000207          RTS     PC      ;RETURN
3859
3860
3861 ;*****
3862 ;SBTTL STALL - USED TO STALL AFTER A DRIVE FUNCTION IS COMPLETE
3863 ;THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
3864 ;AMOUNT OF TIME IF SWITCH 02 IS NOT SET OR A RANDOM AMOUNT OF TIME IF
3865 ;SWITCH 02 IS SET.
3866 ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0-6.
3867 ;CALL
3868 ;       JSR     RO,#STALL
3869 ;       TIME POINTER ;WHERE TO FIND THE STALL TIME
3870 ;
3871 ;       JSR     RO,#STALLA
3872 ;       TIME POINTER ;ENTER HERE TO BYPASS RANDOM STALL SELECTION
3873 ;       ;WHERE TO FIND THE STALL TIME
3874
3873 015542 013046          STALLA: MOV     2(RO)+,-(SP) ;PICKUP STALL TIME
3874 015544 000413          BR     STALLB  ;BYPASS RANDOM STALL CHECK
3875 015546 013046          STALL: MOV     2(RO)+,-(SP) ;PICKUP STALL TIME
3876 015550 032777 000004 163362  BIT     #SW2,SWR  ;USE A RANDOM TIME?
3877 015556 001406          BEQ     STALLB  ;NO--BRANCH
3878 015560 004737 013510          JSR     PC,#SRAND ;YES--FORM RANDOM NUMBER
3879 015564 013716 013610          MOV     2#SLONUM,(SP) ;AND USE IT FOR THE STALL TIME
3880 015570 042716 177700          BIC     #1C77,(SP) ;BUT NEVER > 64 MILLISECONDS
3881 015574 005046          STALLB: CLR     -(SP) ;CLEAR TEMP. LOCATION
3882 015576 162766 000001 000002 2S:  SUB     #1,2(SP) ;MORE STALL REQUIRED?
3883 015604 103407          BLO     45        ;NO--BRANCH
3884 015606 012716 000620          MOV     #400.,(SP) ;STALL TIME CONSTANT
3885 015612 005700          3S:  TST     RO      ;NOP TO KILL TIME
3886 015614 005366 000000          DEC     0(SP)   ;COUNT
3887 015620 001374          BNE     3S      ;LOOP IF MORE COUNTS NEEDED
3888 015622 000765          BR     2S
3889 015624 022626          4S:  CMP     (SP)+,(SP)+ ;CLEAN OFF THE STACK
3890 015626 000200          RTS     RO     ;EXIT
3891
3892
3893 ;*****

```

```

3894 .SBTTL SORT - ROUTINE TO SORT & PRINT SEEK TIMES
3895 ;*THIS ROUTINE DETERMINES THE NUMBER OF TIMES A PARTICULAR FORWARD OR REVERSE
3896 ;*SEEK TIME WAS OBTAINED. THE SEEK TIMES ARE SORTED IN DECENDING SEQUENCE
3897 ;*AND THE NUMBER OF OCCURENCES OF A PARTICULAR SEEK TIME ARE TOTALED. THE
3898 ;*THE ROUTINE WILL TYPEOUT ONLY THE LONGEST 5 SEEK TIMES. FORWARD SEEK TIMES
3899 ;*ARE STORED IN A TABLE STARTING AT 'FWSTR'; REVERSE SEEK TIMES ARE STORED
3900 ;*IN A TABLE STARTING AT 'RVSTR'.
3901 ;CALL
3902 ;
3903 ; JSR RD SORT ; SORT AND TYPE THE SEEK TIMES
3904 ; ; NUMBER OF ITEMS IN THE TABLE (405 MAX)
3905 ;
3906 ; SORT: MOV (R0)+,7$ ; GET THE ENTRY COUNT
3907 ; SAVREG ; SAVE R0-R5
3908 ; MOV #2,R5 ; COUNT FOR FORWARD & REVERSE
3909 ; MOV #FWSTR,SREG0 ; INITIALIZE POINTER TO FORWARD SEEK TIMES
3910 ; MOV #FWSTR+2,SREG1 ; SECOND FORWARD SEEK TIME POINTER
3911 ; MOV #BUFR4,SREG2
3912 ; MOV #BUFR5,SREG3 ; INITIALIZE POINTER TO '# OF TIMES'
3913 ; 1$: MOV SREG0,R0 ; TYPE 'SEEK TIME'
3914 ; MOV SREG1,R1
3915 ; MOV 7$,R2 ; NUMBER OF ITEMS IN THE TABLE
3916 ; CLR R3
3917 ; 2$: CMP (R0),(R1) ; SORT THE ITEMS & PUT THEM
3918 ; BLE 3$ ; IN DECENDING ORDER
3919 ; MOV (R0),R4 ; LARGER ITEMS AT TOP OF LIST,
3920 ; MOV (R1),(R0) ; SMALLER AT THE BOTTOM
3921 ; MOV R4,(R1)
3922 ; 3$: INC R3
3923 ; TST (R0)+
3924 ; TST (R1)+
3925 ; DEC R2
3926 ; BNE 2$
3927 ; TST R3 ; SORTED ALL ITEMS?
3928 ; BNE 1$ ; IF NOT LOOP BACK
3929 ; MOV SREG0,R0 ; PTR TO 'SEEK TIME'
3930 ; MOV SREG2,R1 ; SAVE 'SEEK TIME' HERE
3931 ; MOV SREG3,R2 ; SAVE '# OF TIMES' HERE
3932 ; MOV R2,R4
3933 ; CLR (R4)+ ; CLR OUT 5 WORDS OF
3934 ; CLR (R4)+ ; '# OF TIMES'BUFR
3935 ; CLR (R4)+
3936 ; CLR (R4)+
3937 ; CLR (R4)+
3938 ; MOV #5,R3 ; SAVE ONLY 5 DIFFERENT 'SEEK TIMES'
3939 ; MOV (R0),(R1) ; FIND OUT THE '# OF TIMES'
3940 ; MOV 7$,R3 ; EACH 'SEEK TIME' WAS
3941 ; 4$: CMP (R0)+,(R1) ; OBTAINED
3942 ; BEQ 5$
3943 ; TST (R1)+
3944 ; MOV -2(R0),(R1) ; SAVE 'SEEK TIME'
3945 ; TST (R2)+
3946 ; MOV #1,(R2) ; KEEP '# OF TIMES'
3947 ; DEC R3
3948 ; BEQ 6$
3949 ; BR 4$
3949 ; 5$: INC (R2) ; INCREMENT '# OF TIMES'

```

```

3950 016032 005303          DEC      R3          ;ALL DONE?
3951 016034 001362          BNE      45          ;IF NOT, GO BACK
3952 016036 005205          65:     INC      R5          ;SORTED BOTH FORWARD & REVERSE ?
3953 016040 001417          BEQ      GOTYPE      ;IF YES, TYPE THE VALUES
3954 016042 012737 025464 001162  MOV      #RVSTR, $REG0 ;IF NOT, INITIALIZE POINTER TO REVERSE SEEK TIMES
3955 016050 012737 025466 001164  MOV      #RVSTR+2, $REG1 ;SECOND REVERSE SEEK TIME POINTER
3956 016056 012737 023650 001166  MOV      #BUFR6, $REG2 ;SAVE 'SEEK TIME'
3957 016064 012737 023730 001170  MOV      #BUFR7, $REG3 ;SAVE '# OF TIMES' HERE
3958 016072 000137 015672          JMP      15          ;GO BACK & DO SORTING FOR REVERSE SEEK TIMES
3959
3960 016076 000000          75:     .WORD    0          ;ITEMS IN TABLE COUNT HERE
3961
3962          ;TYPEOUT THE HEADER FOR THE SEEK TIME RESULTS
3963
3964 016100          GOTYPE:
3965 016100 104401 016106          TYPE     655          ;;TYPE ASCIZ STRING
3966 016104 000421          BR       645          ;;GET OVER THE ASCIZ
3967          ;;655: .ASCIZ <15><12>/((SEEK TIME IS IN MICRO SECONDS)/
3968 016150          645:
3969 016150 104401 016156          TYPE     675          ;;TYPE ASCIZ STRING
3970 016154 000420          BR       665          ;;GET OVER THE ASCIZ
3971          ;;675: .ASCIZ <15><12><12>/ FORWARD REVERSE/
3972 016216          665:
3973 016216 104401 016224          TYPE     695          ;;TYPE ASCIZ STRING
3974 016222 000420          BR       685          ;;GET OVER THE ASCIZ
3975          ;;695: .ASCIZ <15><12>/ # OF SEEK # OF SEEK/
3976 016264          685:
3977 016264 104401 016272          TYPE     715          ;;TYPE ASCIZ STRING
3978 016270 000421          BR       705          ;;GET OVER THE ASCIZ
3979          ;;715: .ASCIZ <15><12>/ SEEKS TIME SEEKS TIME/<15><12>
3980 016334          705:
3981          ;TYPEOUT THE '# OF SEEKS' & 'SEEK TIME' OBTAINED FOR EACH OF THOSE SEEKS
3982
3983
3984 016334 005000          TYPTIM: CLR      R0
3985 016336 005005          CLR      R5
3986 016340 104401 001217          15:     TYPE     ,SCRLF
3987 016344 016046 023570          MOV      BUFR5(R0),-(SP) ;GET '# OF SEEKS', IF NONE (0)
3988 016350 001436          BEQ      35          ;SKIP TYPING (FRWRD SEEK)
3989 016352 104405          TYPDS
3990 016354 104401          TYPE
3991 016356 002043          BLNKS2
3992 016360 016046 023510          MOV      BUFR4(R0),-(SP) ;GET 'SEEK TIME' FOR EACH OF
3993 016364 104405          TYPDS          ;OF THAT '# OF SEEKS'. 'GO
3994          ;TYPE OUT IN DECIMAL
3995 016366 104401 016374          TYPE     655          ;;TYPE ASCIZ STRING
3996 016372 000401          BR       645          ;;GET OVER THE ASCIZ
3997          ;;655: .ASCIZ /0/
3998 016376          645:
3999 016376 016046 023730          25:     MOV      BUFR7(R0),-(SP) ;GET '# OF SEEKS', IF NONE - 0
4000 016402 001424          BEQ      45          ;SKIP TYPING (REVRSE SEEK)
4001 016404 005705          TST     R5          ;'FORWARD' VALUE FOR THIS LINE ?
4002 016406 001404          BEQ      65          ;BR IF YES
4003 016410 104401 002035          TYPE     ,BLNKS8
4004 016414 104401 002036          TYPE     ,BLNKS7
4005 016420 104405          65:     TYPDS          ;TYPE OUT IN DECIMAL
    
```

```

4006 016422 104401 002043          TYPE      BLNKS2          ;2 BLANKS
4007 016426 016046 023650          MOV      BUFR6(RO),-(SP) ;GET 'SEEK TIME' & TYPE IT
4008 016432 104405                   TYPDS                   ;OUT IN DECIMAL
4009 016434 104401 016442          TYPE      67$          ;TYPE ASCIZ STRING
4010 016440 000401                   BR      66$            ;GET OVER THE ASCIZ
4011                                     ;:67$:
4012                                     ;66$:
4013 016444 000406                   BR      5$            ;
4014 016446 005726          3$:      TST      (SP)+          ;POP STACK
4015 016450 005205                   INC      R5
4016 016452 000751                   BR      2$            ;
4017 016454 005726          4$:      TST      (SP)+          ;POP STACK
4018 016456 005705                   TST      R5
4019 016460 001004                   BNE     7$            ;
4020 016462 005720                   TST      (RO)+          ;INCREMENT PTR TO TABLES
4021 016464 020027 000012          5$:      CMP      RO,#10.        ;ALL DONE?
4022 016470 001323                   BNE     1$            ;IF NOT GO BACK
4023 016472 104413          7$:      RESREG  1$            ;RESTORE RO-R5
4024 016474 000200                   RTS      RO            ;RETURN
4025
4026
4027                                     ;*****
4028                                     ;SBTTL  SEEK TIME MEASUREMENT ROUTINE
4029                                     ;*THIS ROUTINE MEASURES THE TIME REQUIRED TO SEEK TO THE SPECIFIED CYLINDER.
4030                                     ;*THE ROUTINE USES THE 'SOT' COUNTER IN THE CONTROLLER TO MEASURE THE SEEK
4031                                     ;*TIME.  THE SEEK TIME MEASUREMENT IS BASED ON THE NUMBER OF SECTORS
4032                                     ;*OCCURRING BETWEEN SEEK START AND SEEK COMPLETE.  THE RESOLUTION OF THIS
4033                                     ;*MEASUREMENT METHOD IS + OR - 2.5 MS (ONE SECTOR TIME).
4034                                     ;CALL
4035                                     ;      JSR      PC,TIMSEK          ;TIME THE SEEK TO THE CYLINDER IN DPB+2
4036                                     ;      RETURN                     ;SEEK TIME IS RETURNED IN R3
4037
4038 016476 010046          TIMSEK: MOV      RO,-(SP)          ;SAVE RO
4039 016500 010246          MOV      R2,-(SP)          ;SAVE R2
4040 016502 013746 001110          MOV      SLPERR,-(SP)      ;SAVE CURRENT LOOP ON ERROR ADDRESS
4041 016506 012737 016520 001110          MOV      #TIMSK1,SLPERR   ;NEW LOOP ON ERROR ADDRESS
4042 016514 013700 001266          MOV      RPADR,RO          ;RPI1 BASE ADDRESS
4043 016520 005003          TIMSK1: CLR      R3          ;R3 WILL BE USED TO COUNT THE DISK REVOLUTIONS
4044 016522 012737 000062 001300          MOV      #50,STALLG       ;SET STALL VALUE TO 50MS
4045 016530 004037 015542          JSR      RO,STALLA        ;WAIT THE SPECIFIED VALUE
4046 016534 001300          .WORD   STALLG           ;STALL VALUE STORAGE
4047 016536 016002 000014          1$:      MOV      RPDAR(RO),R2     ;READ THE SECTOR/TRACK REGISTER
4048 016542 032702 000200          BIT      #BIT7,R2         ;WAIT FOR SECTOR 0, THEN START LOOKING
4049 016546 001773          BEQ     1$               ;FOR SECTOR 0.
4050 016550 016002 000014          2$:      MOV      RPDAR(RO),R2     ;READ THE SECTOR REGISTER
4051 016554 026002 000014          CMP      RPDAR(RO),R2     ;WAS IT CHANGING ?
4052 016560 001373          BNE     2$               ;BR IF IT WAS
4053 016562 032702 000360          BIT      #360,R2         ;WAIT FOR SEC 0, INDEX MARK
4054 016566 001370          SNE     2$               ;AS SOON AS IT IS AT SECTOR 0, START
4055                                     ;A SEEK & BEGIN TIMING
4056 016570 012777 017036 162472          MOV      #TIMSK2,RPVEC    ;INTERRUPT ADDRESS
4057 016576 012760 000377 000000          MOV      #377,RPDS(RO)   ;CLEAR ANY ATTENTION BITS
4058 016604 013760 001366 000012          MOV      DPB+2,RPCA(RO)  ;LOAD CYLINDER ADDRESS
4059 016612 013746 001364          MOV      DPB,-(SP)       ;SETUP COMMAND
4060 016616 052716 020000          BIS      #AIE,(SP)       ;SET ATTENTION INTERRUPT ENABLE
4061 016622 012660 000004          MOV      (SP)+,RPCS(RO)  ;ISSUE A SEEK, START TIMING

```



4062	016626	005037	177776			CLR	PSW	:SET PRIORITY TO ZERO
4063	016632	016002	000014	3\$:		MOV	RPDA(R0),R2	:READ THE SECTOR REGISTER
4064	016636	026002	000014			CMP	RPDA(R0),R2	:WAS IT CHANGING ?
4065	016642	001373				BNE	3\$	:BR IF IT WAS
4066	016644	032702	000360			BIT	#360,R2	:WAIT FOR SEC CNTR TO MOVE
4067	016650	001770				BEQ	3\$	:FROM 0 TO 1
4068	016652	016002	000014	4\$:		MOV	RPDA(R0),R2	:READ THE SECTOR REGISTER
4069	016656	026002	000014			CMP	RPDA(R0),R2	:WAS IT CHANGING ?
4070	016662	001373				BNE	4\$	:BR IF IT WAS
4071	016664	032702	000360			BIT	#360,R2	:WAIT FOR SECTOR 0
4072	016670	001370				BNE	4\$	:BR IF NOT AT 0
4073	016672	005203				INC	R3	:COUNT A DISK REVOLUTION (25MS)
4074	016674	022703	000050			CMP	#40.,R3	:1 SECOND ELAPSED ?
4075	016700	003354				BGT	3\$	:BR IF NOT
4076	016702	004737	015134			JSR	PC,SAVRP	:SAVE THE REGISTERS
4077	016706	032737	001000	002004		BIT	#SUFU,\$RPDS	:FILE UNSAFE ?
4078	016714	001403				BEQ	5\$	:BR IF NOT
4079	016716	104002				ERROR	2	:REPORT DRIVE UNSAFE
4080	016720	000137	015242			JMP	DSELECT	:SETUP TO DESELECT THE DRIVE
4081	016724			5\$:				
4082	016724	013737	002004	001126		MOV	\$RPDS,\$BDDAT	:GET CONTENTS OF RPDS
4083	016732	012737	000000	001122		MOV	#RPDS,\$BDDADR	:FORM REGISTER ADDRESS FOR ERROR MESSAGE
4084	016740	063737	001266	001122		ADD	RPADR,\$BDDADR	:ADD RPII BASE ADDRESS
4085	016746	012737	140000	001124		MOV	#SUDY!SUOL,\$GDDAT	:WHAT REGISTER SHOULD BE
4086	016754	013737	001126	001204		MOV	\$BDDAT,\$TMP0	:MOVE REGISTER CONTENTS TO 'STMP0'
4087	016762	042737	037777	001204		BIC	#1C140000,\$TMP0	:SAVE THE SPECIFIED BITS
4088	016770	023737	001124	001204		CMP	\$GDDAT,\$TMP0	:COMPARE THE BITS
4089	016776	001403				BEQ	6\$	:BR IF OK
4090	017000	104017				ERROR	17	:NOT READY/OFFLINE DURING TIMING TEST
4091	017002	000137	015242			JMP	DSELECT	:DESELECT THE DRIVE
4092	017006	104005			6\$:	ERROR	5	:NO INTERRUPT AFTER 1 SECOND
4093	017010	012760	000001	000004		MOV	#CLEAR,RPCS(R0)	:CLEAR THE CONTROLLER
4094	017016	012760	000001	000004		MOV	#CLEAR,RPCS(R0)	:CLEAR THE CONTROLLER
4095	017024	012777	000100	162112		MOV	#BIT06,\$STKS	:SET KEYBOARD INTERRUPT ENABLE
4096	017032	000177	162206			JMP	\$BYPASS	:TERMINATE THE TEST
4097	017036	012626			TIMSK2:	MOV	(SP)+,(SP)+	:RESTORE THE STACK POINTER
4098	017040	016002	000014			MOV	RPDA(R0),R2	:SAVE RPDA
4099	017044	005037	177776			CLR	PSW	:SET PRIORITY TO ZERO
4100	017050	032702	000360			BIT	#360,R2	:AT SECTOR 0 ?
4101	017054	001001				BNE	1\$	:BR IF NOT
4102	017056	005203				INC	R3	:INCREMENT THE REVOLUTION COUNTER
4103	017060	004737	015134		1\$:	JSR	PC,SAVRP	:STORE THE REGISTERS
4104	017064	032737	004000	002004		BIT	#SUSI,\$RPDS	:SEEK INCOMPLETE ?
4105	017072	001405				BEQ	2\$	:BR IF NOT
4106	017074	104015				ERROR	15	:SEEK INCOMPLETE
4107	017076	004737	015264			JSR	PC,RESTOR	:DO A HOME SEEK
4108	017102	000177	162136			JMP	\$BYPASS	:TERMINATE THE PRESENT TEST
4109	017106	032737	100000	000004	2\$:	BIT	#ERR,RPCS	:OTHER ERRORS ?
4110	017114	001412				BEQ	SKDON	:BR IF NOT
4111	017116	104004				ERROR	4	:REPORT ERROR DURING TIMING TEST
4112	017120	012760	000001	000004		MOV	#CLEAR,RPCS(R0)	:CLEAR THE CONTROLLER
4113	017126	012760	000001	000004		MOV	#CLEAR,RPCS(R0)	:CLEAR THE CONTROLLER
4114	017134	113760	001365	000005		MOVB	DPB+1,RPCS+1(R0)	:RESELECT THE DRIVE
4115	017142	042702	177417		SKDON:	BIC	#1C360,R2	:CLEAR SECTOR/TRACK BITS - LEAVE THE 'SOT'
4116	017146	006202				ASR	R2	:RIGHT JUSTIFY 'SOT'
4117	017150	006202				ASR	R2	:RIGHT JUSTIFY 'SOT'

```

4118 017152 006202          ASR      R2          ;RIGHT JUSTIFY 'SOT'
4119 017154 006202          ASR      R2          ;RIGHT JUSTIFY 'SOT'
4120 017156 012746 000010  MOV      #10,-(SP)    ;:PUT THE MULTIPLIER ON THE STACK
4121 017162 010346          MOV      R3,-(SP)    ;:PUT THE MULTIPLICAND ON THE STACK
4122 017164 004737 013022  JSR      PC,@#SMULT  ;:CALL THE MULTIPLY ROUTINE
4123 017170 012616          MOV      (SP)+,(SP)  ;:DISREGARD THE MSB'S
4124 017172 012603          MOV      (SP)+,R3    ;:GET THE LSB'S OF THE PRODUCT
4125 017174 060203          ADD      R2,R3      ;ADD PARTIAL SECTOR COUNT
4126                                     ;NOTE THERE IS A SCALE FACTOR
4127 017176 012746 000372  MOV      #250,-(SP)  ;:PUT THE MULTIPLIER ON THE STACK
4128 017202 010346          MOV      R3,-(SP)    ;:PUT THE MULTIPLICAND ON THE STACK
4129 017204 004737 013022  JSR      PC,@#SMULT  ;:CALL THE MULTIPLY ROUTINE
4130 017210 012616          MOV      (SP)+,(SP)  ;:DISREGARD THE MSB'S
4131 017212 012603          MOV      (SP)+,R3    ;:GET THE LSB'S OF THE PRODUCT
4132
4133 ;:*****
4134 ;SEEK TIME (IN US) = [(R3)X(10)+(R2)]X(250)X(10)
4135 ;:*****
4136
4137 017214 012637 001110  MOV      (SP)+,$LPERR ;RESTORE THE LOOP ON ERROR ADDRESS
4138 017220 012602          MOV      (SP)+,R2    ;RESTORE R2
4139 017222 012600          MOV      (SP)+,R0    ;RESTORE R0
4140 017224 000207          RTS       PC         ;RETURN
4141
4142 ;:*****
4143 ;SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS
4144 ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
4145 ;OF THE RP11 IS SETUP TO READ THE PROPER VALUE.
4146 ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
4147 ;REQUIRED.
4148 ;NOTE: THIS ROUTINE DESTROYS R0-R4
4149 ;CALL
4150
4151 ;
4152 ; JSR      PC,@#GETADR
4153 ; RETURN
4154 017226 005737 001224  GETADR: TST      @#BUSADR ;INPUT FROM TTY REQUESTED?
4155 017232 001447          BEQ      7$          ;NO--BRANCH
4156 017234 005037 001224  CLR      @#BUSADR    ;YES--CLEAR THE REQUEST FLAG
4157 017240 012700 001266  1$: MOV      #RPADR,R0   ;FIRST ADDRESS
4158 017244 012703 021410  MOV      #RPPCS,R3   ;"RPADR="
4159 017250 011004          MOV      (R0),R4    ;PRESENT RP11 ADDRESS
4160 017252 004037 017430  JSR      R0,@#GETNUM ;GET NEW RP11 ADDRESS
4161 017256 000402          BR       2$          ;COMMA
4162 017260 000767          BR       1$          ;PERIOD
4163 017262 000430          BR       5$          ;DOUBLE PERIOD
4164 017264 010420 2$: MOV      R4,(R0)+    ;SAVE NEW RHCS1
4165 017266 012703 021420  MOV      #RPVEC,R3   ;"RPVEC="
4166 017272 011004          MOV      (R0),R4    ;PRESENT RP11 VECTOR ADDRESS
4167 017274 004037 017430  JSR      R0,@#GETNUM ;GET NEW RPVEC
4168 017300 000402          BR       3$          ;COMMA
4169 017302 000756          BR       1$          ;PERIOD
4170 017304 000417          BR       5$          ;DOUBLE PERIOD
4171 017306 010420 3$: MOV      R4,(R0)+    ;SAVE NEW RPVEC
4172 017310 012703 021431  MOV      #RPPRI,R3   ;"RPPRI="
4173 017314 011004          MOV      (R0),R4    ;PRESENT RP11 PRIORITY LEVEL

```

```

4174 017316 006304 ASL R4 ;POSITION FOR TYPEOUT
4175 017320 006304 ASL R4
4176 017322 006304 ASL R4
4177 017324 000304 SWAB R4
4178 017326 004037 017430 JSR RO,@GETNUM ;GET NEW RPPRIO
4179 017332 000402 BR 4$ ;COMMA
4180 017334 000401 BR 4$ ;PERIOD
4181 017336 000404 BR 6$ ;DOUBLE PERIOD
4182 017340 010210 4$: MOV R2,(RO) ;SAVE NEW RPPRIO
4183 017342 000736 BR 1$ ;LOOP
4184 017344 010410 5$: MOV R4,(RO) ;SAVE INPUT
4185 017346 000401 BR 7$
4186 017350 010210 6$: MOV R2,(RO) ;SAVE PRIORITY
4187 017352 013701 000004 7$: MOV @ERRVEC,R1 ;SAVE THE ERROR VECTOR
4188 017356 012737 017376 000004 MOV @8$,@ERRVEC ;SETUP FOR TRAP
4189 017364 005777 161676 TST @RPADR ;CHECK FOR RP11
4190 017370 010137 000004 MOV R1,@ERRVEC ;RESTORE ERROR VECTOR
4191 017374 000207 RTS PC ;RETURN
4192 017376 010137 000004 8$: MOV R1,@ERRVEC ;RESTORE ERROR VECTOR
4193 017402 022626 CMP (SP)+,(SP)+ ;CLEAN OFF THE STACK
4194 017404 104001 ERROR 1 ;REPORT THE ERROR
4195 017406 005737 000042 TST @42 ;IS THERE A MONITOR?
4196 017412 001712 BEQ 1$ ;NO--GO ASK FOR ADDRESS
4197 017414 005037 001232 CLR @DRVSEL ;YES--NO DRIVES SELECTED
4198 017420 005037 007640 CLR @SEOPCT ;NO PASSES
4199 017424 000137 007464 JMP @SEOP ;GO TO END OF PROGRAM

```

```

*****
.SBTTL GETNUM - ROUTINE TO GET A NUMBER
*THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
*INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
*IF REQUIRED.
*NOTE: THIS ROUTINE DESTROYS R1
CALL
MOV @ADR,R3 ;ADDRESS OF ASCIZ MESSAGE
MOV @NUM,R4 ;OCTAL NUMBER
JSR RO,@GETNUM
RETURN1 ;INPUT TERMINATED WITH A COMMA
RETURN2 ;WITH A PERIOD
RETURN3 ;WITH A DOUBLE PERIOD
;R4=INPUT NUMBER AND
;R2=R4*32 FOR ALL
;THREE RETURNS

```

```

4218 017430 010337 017436 GETNUM: MOV R3,2$ ;SAVE MESSAGE POINTER
4219 017434 104401 1$: TYPE ;TYPE THE MESSAGE
4220 017436 000000 2$: .WORD 0 ;MESSAGE POINTER GOES HERE
4221 017440 010446 MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
4222 017442 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
4223 017444 104411 RDLIN ;READ AN ASCIZ STRING
4224 017446 012601 MOV (SP)+,R1 ;ADDRESS OF FIRST CHARACTER
4225 017450 004037 021014 JSR RO,@CK.CHR ;CHECK ONE CHARACTER
4226 017454 017434 1$ ;ILLEGAL CHARACTER
4227 017456 017434 1$ ;CARRIAGE RETURN
4228 017460 017470 3$ ;"/"
4229 017462 017514 7$ ;","

```

```

4230 017464 017522      BS      ;" "
4231 017466 017434      IS      ;DIGIT 0-9
4232 017470
4233 017470 004037 021250 3S:      JSR      RD, @#CK.NUM ;CHECK THE NUMBER
4234 017474 017434      IS      ;ILLEGAL INPUT
4235 017476 017510      6S      ;TERMINATED WITH A " "
4236 017500 017506      5S      ;TERMINATED WITH A "' "
4237 017502 017504      4S      ;TERMINATED WITH A ". "
4238 017504 005720      4S:      TST      (R0)+ ;DOUBLE PERIOD
4239 017506 005720      5S:      TST      (R0)+ ;SINGLE PERIOD
4240 017510 010204      6S:      MOV      R2, R4 ;COMMA--SAVE INPUT NUMBER
4241 017512 000414      BR      10S ;GO TO EXIT
4242 017514 105711      7S:      TSTB     (R1) ;TERMINATOR AFTER A COMMA?
4243 017516 001346      BNE     IS ;NO--LOOP
4244 017520 000411      BR      10S ;YES--EXIT
4245 017522 105711      8S:      TSTB     (R1) ;TERMINATOR AFTER A PERIOD?
4246 017524 001406      BEQ     9S ;YES--EXIT
4247 017526 122721 000056  CMPB   #'.,(R1)+ ;NO--DOUBLE PERIOD?
4248 017532 001340      BNE     IS ;NO--LOOP
4249 017534 105711      TSTB     (R1) ;YES--TERMINATOR?
4250 017536 001336      BNE     IS ;NO--LOOP
4251 017540 005720      TST      (R0)+ ;DOUBLE PERIOD
4252 017542 005720      9S:      TST      (R0)+ ;PERIOD
4253 017544 010402      10S:     MOV      R4, R2 ;COMMA--POSITION THE
4254 017546 000302      SWAB    R2 ;NUMBER IN CASE IT
4255 017550 006202      ASR     R2 ;IS THE PRIORITY LEVEL
4256 017552 006202      ASR     R2
4257 017554 006202      ASR     R2
4258 017556 000200      RTS     R0 ;EXIT

```

```

*****
;SMTL GT.PRM - GET PARAMETERS
;THIS ROUTINE IS USED TO CHANGE OR MODIFY
;THE TEST PARAMETERS. IT GIVES THE OPERATOR
;THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
;TESTS TO RUN AND HOW MANY TIMES TO
;REPEAT EACH TEST

```

```

4260
4261
4262
4263
4264
4265
4266
4267
4268 017560 104412      GT.PRM: SAVREG ;SAVE R0-R5
4269 017562 005037 001232  GT.PRI: CLR     DRVSEL ;NO DRIVE SELECTED
4270 017566 104401 017574      TYPE     ,65S ;TYPE ASCIZ STRING
4271 017572 000406      BR      64S ;GET OVER THE ASCIZ
4272      ;65S: .ASCIZ <15><12>/DRIVE(S)=/
4273 017610
4274 017610 104411      64S:      RDLIN ;READ TTY
4275 017612 012601      MOV      (SP)+, R1 ;ADDRESS OF ASCIZ STRING
4276 017614 004037 021014  JSR      RD, @#CK.CHR ;CHECK ONE CHARACTER
4277 017620 017562      GT.PRI ;ILLEGAL CHARACTER
4278 017622 017562      GT.PRI ;CARRIAGE RETURN
4279 017624 017562      GT.PRI ;" /
4280 017626 017562      GT.PRI ;" "
4281 017630 017562      GT.PRI ;" '
4282 017632 017634      IS      ;DIGIT 0-9
4283 017634 005301      1S:      DEC     R1
4284 017636
4285 017636 012702 000007  2S:      MOV      #7, R2 ;UPPER LIMIT OF INPUT

```

4286	017642	004037	021070		JSR	RO, #CK.DIG	:CHECK THE DIGIT(S)
4287	017646	017562			GT.PRI		:ILLEGAL INPUT
4288	017650	017562			GT.PRI		:INPUT TOO LARGE
4289	017652	017660			3\$		:TERMINATED WITH A "-"
4290	017654	017700			4\$		:TERMINATED WITH A "-"
4291	017656	017700			4\$		:TERMINATED WITH A "-"
4292	017660	156237	001314	001232	3\$: BISE	ATABIT(R2), DRVSEL	:SET THE DRIVE SELECTED BIT
4293	017666	105741			TSTB	-(R1)	:WAS THE LINE TERMINATED?
4294	017670	001362			BNE	2\$	:NO-GET THE NEXT DRIVE
4295	017672	005037	001234		CLR	#TSTNMS	:DESELECT ALL TESTS
4296	017676	000405			BR	GTTST1	:YES--SELECT TEST
4297	017700	156237	001314	001232	4\$: BISE	ATABIT(R2), DRVSEL	:SET THE SELECTED DRIVE BITS
4298	017706	104413			GT.PR2:	RESREG	:RESTORE RO-R5
4299	017710	000207			RTS	PC	:EXIT
4300							
4301	017712				GTTST1:		
4302	017712	104401	017720		TYPE	65\$	::TYPE ASCIZ STRING
4303	017716	000403			BR	64\$	::GET OVER THE ASCIZ
4304					65\$:	.ASCIZ	/TEST=
4305	017726				64\$:		
4306	017726	104411			ROLIN		:READ AN ASCIZ STRING
4307	017730	012601			MOV	(SP)+, R1	:POINTER TO R1
4308	017732	122711	000056		CMPB	#'. , (R1)	:DOUBLE PERIOD?
4309	017736	001007			BNE	1\$	:NO--BRANCH
4310	017740	122761	000056	000001	CMPB	#'. , 1(R1)	
4311	017746	001003			BNE	1\$	
4312	017750	105761	000002		TSTB	2(R1)	: "CR"?
4313	017754	001754			BEQ	GT.PR2	:YES--EXIT
4314	017756	005037	001234		1\$: CLR	TSTNMS	:NO TEST SELECTED
4315	017762	005003			CLR	R3	:NO TEST TO BE OPENED
4316	017764	005004			GTTST2:	R4	:NO TEST BITS SET
4317	017766	004037	020740		JSR	RO, #CK.OCT	:OCTAL DIGIT?
4318	017772	000460			BR	GTTST4	:NO--BRANCH
4319	017774	010205			MOV	R2, R5	:YES--SAVE IT
4320	017776	005201			INC	R1	:MOVE TO NEXT CHARACTER
4321	020000	004037	020740		JSR	RO, #CK.OCT	:OCTAL DIGIT
4322	020004	000405			BR	1\$	:NO--BRANCH
4323	020006	005201			INC	R1	:MOVE TO NEXT CHARACTER
4324	020010	006305			ASL	R5	:SCALE HIGH DIGIT
4325	020012	006305			ASL	R5	
4326	020014	006305			ASL	R5	
4327	020016	060502			ADD	R5, R2	:COMBINE HIGH & LOW DIGITS
4328	020020	020227	000012		1\$: CMP	R2, #STN-1	:LEGAL TEST NUMBER?
4329	020024	003332			BGT	GTTST1	:NO--BRANCH
4330	020026	006302			ASL	R2	
4331	020030	016204	001324		MOV	BITS(R2), R4	:SELECT TEST
4332	020034	121127	000055		CMPB	(R1), #'-	:TEST STRING?
4333	020040	101035			BNE	GTTST4	:NO--BRANCH
4334	020042	005201			INC	R1	:YES--MOVE TO NEXT CHARACTER
4335	020044	004037	020740		JSR	RO, #CK.OCT	:OCTAL DIGIT?
4336	020050	000720			BR	GTTST1	:NO--BRANCH
4337	020052	010205			MOV	R2, R5	:YES--SAVE IT
4338	020054	005201			INC	R1	:MOVE TO NEXT CHARACTER
4339	020056	004037	020740		JSR	RO, #CK.OCT	:OCTAL DIGIT?
4340	020062	000405			BR	2\$	:NO--BRANCH
4341	020064	005201			INC	R1	:YES--MOVE TO NEXT CHARACTER

4342	020066	006305			ASL	R5		;SCALE HIGH DIGIT
4343	020070	006305			ASL	R5		
4344	020072	006305			ASL	R5		
4345	020074	060502			ADD	R5,R2		;COMBINE HIGH & LOW DIGIT
4346	020076	020227	000012	25:	CMP	R2,#5TN-1		;LEGAL TEST NUMBER?
4347	020102	003303			BGT	GTTST1		;NO--BRANCH
4348	020104	006302			ASL	R2		
4349	020106	020462	001324		CMP	R4,BITS(R2)		;IS THE FIRST NUMBER OF THE

4350										
4351	020112	002277			BGE	GTTST1				:STRING SMALLER THAN THE LAST?
4352	020114	056204	001324	3S:	BIS	BITS(R2),R4				:NO--BRANCH
4353	020120	005742			TST	-(R2)				:YES--SET TEST SELECT BIT FOR SELECTED TESTS
4354	020122	036204	001324		BIT	BITS(R2),R4				:ADJUST POINTER
4355	020126	001772			BEQ	3S				:DONE?
4356	020130	000401			BR	GTTST4				:NO--BRANCH
4357	020132	005201		GTTST3:	INC	R1				:SKIP THE INCREMENT
4358	020134	050437	001234	GTTST4:	BIS	R4,TSTNMS				:MOVE TO NEXT CHARACTER
4359										:SET SELECTED TEST BITS INTO
4360	020140	121127	000056		CHPB	(R1),8'				:TEST SELECT WORD
4361	020144	001420			BEQ	GTTST5				: "PERIOD"?
4362	020146	005704			TST	R4				:YES--BRANCH
4363	020150	001660			BEQ	GTTST1				:ANY TEST SELECTED THIS CYCLE?
4364	020152	121127	000057		CHPB	(R1),8'/				:NO--BRANCH
4365	020156	001002			BNE	1S				: "OPEN"?
4366	020160	050403			BIS	R4,R3				:NO--BRANCH
										:YES--SET BITS FOR TEST TO OPEN

4367	020162	000403			BR	25		
4368	020164	121127	000054	1S:	CMPB	(R1),#'	:"COMMA"?	
4369	020170	001250			BNE	GTTST1	:NO--BRANCH	
4370	020172	005201		2S:	INC	R1	:MOVE TO NEXT CHARACTER	
4371	020174	105711			TSTB	(R1)	:"CR"?	
4372	020176	001272			BNE	GTTST2	:NO--GO GET NEXT CHARACTER	
4373	020200	005703			TST	R3	:ANY TESTS TO OPEN?	
4374	020202	001026			BNE	OPNTST	:YES--BRANCH	
4375	020204	000642			BR	GTTST1	:NO--START AGAIN	
4376	020206	005201		GTTST5:	INC	R1	:MOVE TO NEXT CHARACTER	
4377	020210	121127	000056		CMPB	(R1),#'	:"PERIOD"?	
4378	020214	001410			BEG	GTTST6	:YES--BRANCH	
4379	020216	105711			TSTB	(R1)	:"CR"?	
4380	020220	001402			BEG	15	:YES--BRANCH	
4381	020222	000137	017712		JMP	GTTST1		
4382	020226	005703		1S:	TST	R3	:ANY TEST TO BE OPENED?	
4383	020230	001013			BNE	OPNTST	:YES--BRANCH	
4384	020232	000137	017706		JMP	GT.PR2	:NO--GO START TESTING	
4385	020236	005201		GTTST6:	INC	R1	:MOVE TO NEXT CHARACTER	
4386	020240	105711			TSTB	(R1)	:"CR"?	
4387	020242	001402			BEG	15	:YES--BRANCH	
4388	020244	000137	017712		JMP	GTTST1	:NO--GO ASK FOR TEST	
4389	020250	005703		1S:	TST	R3	:ANY TEST TO BE OPENED?	
4390	020252	001002			BNE	OPNTST	:YES--BRANCH	
4391	020254	000137	017706		JMP	GT.PR2	:NO--GO START TESTING	
4392								
4393	020260	104412			:OPEN THE TEST FOR CHANGES			
4394	020262	005027		OPNTST:	SAVREG		:SAVE RO-R5	
4395	020264	000000		OPN.CT:	CLR	(PC)+	:START WITH TEST 0	
4396	020266	000411			.WORD	0	:COUNT STORED HERE	
4397	020270	005237	020264		BR	OPN.2	:SKIP THE INCREMENT	
4398	020274	022737	000012	020264	OPN.1:	INC	:MOVE TO THE NEXT TEST	
4399	020302	002003			CMP	#STN-1,OPN.CT	:TEST NUMBER TO BIG?	
4400	020304	104413			BGE	OPN.2	:NO--OPEN THE NEXT TEST	
4401	020306	000137	017712		RESREG		:RESTORE RO-R5	
4402	020312	013705	020264		JMP	GTTST1	:YES--GO ASK FOR MORE TESTS	
4403	020316	006305		OPN.2:	MOV	OPN.CT,R5	:SETUP TO USE THE	
4404	020320	036503	001324		ASL	R5	:TEST NUMBER AS AN INDEX	
4405	020324	001761			BIT	BITS(R5),R3	:OPEN THIS TEST?	
4406	020326	104401	020334		BEG	OPN.1	:NO--MOVE TO NEXT TEST	
4407	020332	000404			TYPE	,655	:TYPE ASCIZ STRING	
4408					BR	645	:GET OVER THE ASCIZ	
4409	020344			::655:	.ASCIZ	/ TEST /		
4410	020344	013746	020264	645:	MOV	OPN.CT,-(SP)	:SAVE OPN.CT FOR TYPEOUT	
4411							:TEST NUMBER	
4412	020350	104403			TYPOS		:GO TYPE--OCTAL ASCII	
4413	020352	002			.BYTE	2	:TYPE 2 DIGIT(S)	
4414	020353	000			.BYTE	0	:SUPPRESS LEADING ZEROS	
4415	020354	104401	001217		TYPE	.SCRLF	:TYPE "CR" & "LF"	
4416	020360	016500	001406		MOV	PRMPT(R5),RO	:PICKUP PARAMETER POINTER	
4417	020364	011046			MOV	(RO),-(SP)	:SAVE THE VARIABLE INDICATOR	
4418	020366	012702	001372		MOV	#PRM,R2	:FIRST ADDRESS OF TABLE	
4419	020372	000405			BR	25		
4420	020374	006216		1S:	ASR	(SP)	:CHECK FOR A VARIABLE	
4421	020376	103403			BCS	25	:GO MOVE THIS ONE	
4422	020400	001404			BEG	OPNPRM	:DONE	



4423	020402	005722			TST	(R2)+	;BUMP THE POINTER
4424	020404	000773			BR	IS	
4425	020406	012022		2S:	MOV	(R0)+,(R2)+	;MOVE THIS VARIABLE INTO THE
4426	020410	000771			BR	IS	;COMMON AREA
4427	020412	013716	001372	OPNPRM:	MOV	2#PRM,(SP)	;GET THE VARIABLE INDICATOR
4428	020416	005004			CLR	R4	;ZERO THE INDEX
4429	020420	006216		1S:	ASR	(SP)	;CHECK FOR A VARIABLE
4430	020422	103403			BCS	3S	;GO GET IT
4431	020424	001772			BEQ	OPNPRM	;OUT OF VARIABLES
4432	020426	005724		2S:	TST	(R4)+	;UPDATE THE INDEX
4433	020430	000773			BR	IS	
4434	020432	104401	002043	3S:	TYPE	BLNKS2	;TYPE SPACES
4435	020436	016437	001446 020446		MOV	PRMSG(R4),4S	;TYPE THE NAME OF THIS VARIABLE
4436	020444	104401			TYPE		
4437	020446	000000		4S:	WORD	0	
4438	020450	104401	021406		TYPE	MSG.EQ	;TYPE "="
4439	020454	016446	001374		MOV	RPT(R4),-(SP)	;PUT RPT(R4) ON THE STACK
4440	020460	004737	013220		JSR	PC,\$SB20	;CONVERT RPT(R4)
4441	020464	004737	013450		JSR	PC,\$SUPRS	;TYPE RPT(R4)
4442	020470	104411			RDLIN		
4443	020472	012601			MOV	(SP)+,R1	;READ AN ASCIZ STRING
4444	020474	004037	021014		JSR	RD,2#CK.CHR	;CHECK ONE CHARACTER
4445	020500	020432			3S		;ILLEGAL CHARACTER
4446	020502	020432			3S		;CARRIAGE RETURN
4447	020504	020550			7S		"/
4448	020506	020514			5S		"/
4449	020510	020522			6S		"/
4450	020512	020432			3S		;DIGIT 0-9
4451	020514	105711		5S:	TSTB	(R1)	"CR"?
4452	020516	001345			BNE	3S	NO--STAY ON THIS VARIABLE
4453	020520	000742			BR	2S	YES--MOVE TO NEXT VARIABLE
4454	020522	105711		6S:	TSTB	(R1)	IS THERE A "CR" AFTER THE PERIOD?
4455	020524	001002			BNE	64S	NO
4456	020526	000137	020604		JMP	OPN.N2	YES--GO CLOSE THIS TEST
4457	020532	122721	000056	64S:	CMPB	#'.,(R1)+	DOUBLE PERIOD?
4458	020536	001335			BNE	3S	NO--GO ASK FOR THIS VARIABLE
4459	020540	105711			TSTB	(R1)	YES--IS A "CR" AFTER THE DOUBLE PERIOD?
4460	020542	001333			BNE	3S	NO--ASK FOR THIS VARIABLE AGAIN
4461	020544	000137	020622		JMP	OPN.X2	;YES--CLOSE ALL TEST
4462	020550			7S:			
4463	020550	016402	001434		MOV	PRMLT(R4),R2	;UPPER LIMIT OF INPUT
4464	020554	004037	021070		JSR	RD,2#CK.DIG	;CHECK THE DIGIT(S)
4465	020560	020432			3S		;ILLEGAL INPUT
4466	020562	020432			3S		;INPUT TOO LARGE
4467	020564	020572			BS		;TERMINATED WITH A " "
4468	020566	020600			OPN.N1		;TERMINATED WITH A " . "
4469	020570	020616			OPN.X1		;TERMINATED WITH A " .. "
4470	020572	010264	001374	8S:	MOV	R2,RPT(R4)	;SAVE THIS VARIABLE
4471	020576	000713			BR	2S	;MOVE TO NEXT VARIABLE
4472	020600	010264	001374	OPN.N1:	MOV	R2,RPT(R4)	;SAVE THIS VARIABLE
4473	020604	005726		OPN.N2:	TST	(SP)+	;CLEAN OFF THE STACK
4474	020606	004737	020660		JSR	PC,CLOSE	;CLOSE THIS TEST
4475	020612	000137	020270		JMP	OPN.I	;GO OPEN THE NEXT TEST
4476	020616	010264	001374	OPN.X1:	MOV	R2,RPT(R4)	;SAVE THIS VARIABLE
4477	020622	005726		OPN.X2:	TST	(SP)+	;CLEAN OFF THE STACK
4478	020624	004737	020660	1S:	JSR	PC,CLOSE	;CLOSE THIS TEST

```

4479 020630 005725          2S:  TST      (R5)+      ;UPDATE THE INDEX
4480 020632 020527 000034    CMP      R5,#16*2    ;INDEX TO BIG?
4481 020636 002404          BLT      3S          ;NO--BRANCH
4482 020640 104413          RESREG   ;YES, RESTORE R0-R5
4483 020642 104413          RESREG   ;YES, RESTORE R0-R5
4484 020644 000137 017706    JMP      GT.PR2     ;GO TO EXIT
4485 020650 036503 001324    3S:  BIT      BITS(R5),R3 ;IS THIS TEST OPEN FOR CHANGE?
4486 020654 001363          BNE     1S          ;YES--GO CLOSE IT
4487 020656 000764          BR      2S          ;NO--MOVE TO NEXT TEST
4488 020660 104412          CLOSE: SAVREG      ;SAVE R0-R5
4489 020662 012700 001372    MOV     #PRM,R0     ;"FROM" ADDRESS
4490 020666 016501 001406    MOV     PRMP(R5),R1 ;"TO" ADDRESS
4491 020672 012002          MOV     (R0)+,R2    ;"FROM" INDICATOR
4492 020674 012103          MOV     (R1)+,R3    ;"TO" INDICATOR
4493 020676 012704 000001    MOV     #1,R4       ;TEST BIT START A "RPT"
4494 020702 030402          1S:  BIT      R4,R2     ;PARAMETER TO BE MOVED?
4495 020704 001403          BEQ     2S          ;NO--BRANCH
4496 020706 030403          BIT      R4,R3     ;A PLACE TO PUT IT?
4497 020710 001404          BEQ     3S          ;NO--BRANCH
4498 020712 011011          MOV     (R0),(R1)   ;YES--MOVE "FROM" TO "TO"
4499 020714 030403          2S:  BIT      R4,R3     ;"TO" PARAMETER?
4500 020716 001401          BEQ     3S          ;NO--BRANCH
4501 020720 005721          TST     (R1)+       ;YES--UPDATE THE POINTER
4502 020722 005720          3S:  TST     (R0)+       ;UPDATE FROM POINTER
4503 020724 006304          ASL     R4          ;POSITION THE TEST BIT
4504 020726 032704 002000    BIT     #BIT10,R4   ;DONE?
4505 020732 001763          BEQ     1S          ;NO--BRANCH
4506 020734 104413          RESREG   ;YES, RESTORE R0-R5
4507 020736 000207          RTS     PC          ;RETURN
4508
4509 ;*****
4510 ;SBTTL CK.OCT -- CHECK FOR OCTAL CHARACTER
4511 ;THIS ROUTINE IS USED TO CHECK IF AN
4512 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
4513 ;CALL
4514 ;       MOV     #ADR,R1 ;ADDRESS OF ASCII CHARACTER
4515 ;       JSR     RD,#CK.OCT ;CHECK THE CHARACTER
4516 ;       RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
4517 ;       RETURN2 ;CHARACTER IS IN R2 AS A
4518 ;               ;OCTAL DIGIT
4519 020740 121127 000060    CK.OCT: CMPB     (R1),#0    ;LESS THAN ZERO?
4520 020744 103407          BLO     1S          ;YES -- BRANCH
4521 020746 121127 000067    CMPB     (R1),#7    ;GREATER THAN SEVEN?
4522 020752 101004          BHI     1S          ;YES -- BRANCH
4523 020754 111102          MOVB    (R1),R2     ;GET THE CHARACTER
4524 020756 042702 177770    BIC     #7,R2       ;STRIP AWAY THE ASCII
4525 020762 005720          TST     (R0)+       ;ADJUST FOR RETURN
4526 020764 000200          1S:  RTS     R0      ;RETURN
4527
4528 ;*****
4529 ;SBTTL CK.DEC -- CHECK FOR DECIMAL CHARACTER
4530 ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
4531 ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
4532 ;CALL
4533 ;       MOV     #ADR,R1 ;ADDRESS OF ASCII CHARACTER
4534 ;       JSR     RD,#CK.DEC ;CHECK THE CHARACTER

```

4535  
4536  
4537  
4538  
4539 020766 121127 000060  
4540 020772 103407  
4541 020774 121127 000071  
4542 021000 101004  
4543 021002 111102  
4544 021004 042702 000060  
4545 021010 005720  
4546 021012 000200  
4547  
4548  
4549  
4550  
4551  
4552  
4553  
4554  
4555  
4556  
4557  
4558  
4559  
4560  
4561  
4562  
4563 021014 105711  
4564 021016 001420  
4565 021020 121127 000057  
4566 021024 001414  
4567 021026 121127 000054  
4568 021032 001410  
4569 021034 121127 000056  
4570 021040 001404  
4571 021042 004037 020766  
4572 021046 000406  
4573 021050 005720  
4574 021052 005720  
4575 021054 005720  
4576 021056 005720  
4577 021060 005720  
4578 021062 005201  
4579 021064 011000  
4580 021066 000200  
4581  
4582  
4583  
4584  
4585  
4586  
4587  
4588  
4589  
4590

RETURN1  
RETURN2  
: NOT BETWEEN 0 AND 9  
: BETWEEN 0 AND 9  
: R2 = DIGIT  
CK.DEC: CMPB (R1), #'0 ; LESS THAN ZERO?  
BLO 1\$ ; YES -- BRANCH  
CMPB (R1), #'9 ; GREATER THAN NINE?  
BHI 1\$ ; YES -- BRANCH  
MOVB (R1), R2 ; GET THE CHARACTER  
BIC #'0, R2 ; STRIP AWAY THE ASCII  
TST (R0)+ ; ADJUST FOR RETURN  
1\$: RTS R0 ; RETURN

\*\*\*\*\*  
: SBTTL CK.CHR -- CHECK CHARACTER  
: \*THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO  
: \*DETERMINE WHAT IT IS.

CALL  
MOV #ADR, R1 ; ADDRESS OF ASCII CHARACTER  
JSR R0, @CK.CHR ; CHECK CHARACTER  
RETURN ADR1 ; UNKNOWN CHARACTER  
RETURN ADR2 ; CARRIAGE RETURN \* (R1)=ADR+1  
RETURN ADR3 ; SLASH \* (R1)=ADR+1  
RETURN ADR4 ; COMMA \* (R1)=ADR+1  
RETURN ADR5 ; PERIOD \* (R1)=ADR+1  
RETURN ADR6 ; DIGIT BETWEEN 0 AND 9.  
: R2 = DIGIT \* (R1)=ADR+1

CK.CHR: TSTB (R1) ; "CARRIAGE RETURN"?  
BEQ 4\$ ; YES -- BRANCH  
CMPB (R1), #'/ ; "SLASH"?  
BEQ 3\$ ; YES -- BRANCH  
CMPB (R1), #', ; "COMMA"?  
BEQ 2\$ ; YES -- BRANCH  
CMPB (R1), #'.' ; "PERIOD"?  
BEQ 1\$ ; YES -- BRANCH  
JSR R0, @CK.DEC ; "DIGIT"?  
BR 5\$ ; NO -- BRANCH  
TST (R0)+ ; DIGIT BETWEEN 0-9  
1\$: TST (R0)+ ; PERIOD  
2\$: TST (R0)+ ; COMMA  
3\$: TST (R0)+ ; SLASH  
4\$: TST (R0)+ ; CARRIAGE RETURN  
INC R1 ; MOVE POINTER TO NEXT CHARACTER  
5\$: MOV (R0), R0 ; UNKNOWN CHARACTER  
RTS R0 ; RETURN

\*\*\*\*\*  
: SBTTL CK.DIG - CHECK DIGIT  
: \*THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL  
: \*CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.

CALL  
MOV #ADR, R1 ; ADDRESS OF ASCII STRING  
MOV #NUM, R2 ; MAX. MAGNITUDE OF INPUT NUMBER  
JSR R0, @CK.DIG ; CHECK DIGITS  
RETURN ADR1 ; ILLEGAL CHARACTER -- R2=?

4591				:	RETURN	ADR2	: INPUT NUMBER TO LARGE -- R2=?
4592				:	RETURN	ADR3	: "COMMA" -- R2 = NUMBER
4593				:	RETURN	ADR4	: "PERIOD" -- R2 = NUMBER
4594				:	RETURN	ADR5	: "PERIOD-PERIOD" -- R2 = NUMBER
4595				:			
4596	021070	010446			CK.DIG: MOV	R4, -(SP)	: SAVE R4
4597	021072	010346			MOV	R3, -(SP)	: SAVE R3
4598	021074	010246			MOV	R2, -(SP)	: SAVE THE MAX. SIZE ON THE STACK
4599	021076	005002			CLR	R2	: START WITH 0
4600	021100	005003			CLR	R3	
4601	021102	005004			CLR	R4	
4602	021104	004037	021014		JSR	RD, @CK.CHR	: CHECK ONE CHARACTER
4603	021110	021234			B\$		: ILLEGAL CHARACTER
4604	021112	021234			B\$		: CARRIAGE RETURN
4605	021114	021234			B\$		: "/"
4606	021116	021234			B\$		: "."
4607	021120	021234			B\$		: "
4608	021122	021124			1\$		: DIGIT 0-9
4609	021124	006303		1\$:	ASL	R3	: *2
4610	021126	010346			MOV	R3, -(SP)	: SAVE *2
4611	021130	006303			ASL	R3	: *4
4612	021132	006303			ASL	R3	: *8
4613	021134	062603			ADD	(SP)+, R3	: (*8)+(*2)=*10.
4614	021136	060203			ADD	R2, R3	: UPDATE THE INPUT NUMBER
4615	021140	004037	021014		JSR	RD, @CK.CHR	: CHECK ONE CHARACTER
4616	021144	021234			B\$		: ILLEGAL CHARACTER
4617	021146	021234			B\$		: CARRIAGE RETURN
4618	021150	021234			B\$		: "/"
4619	021152	021162			3\$		: "."
4620	021154	021160			2\$		: "
4621	021156	021124			1\$		: DIGIT 0-9
4622	021160	005724		2\$:	TST	(R4)+	: "PERIOD"
4623	021162	005724		3\$:	TST	(R4)+	: "COMMA"
4624	021164	004037	021014		JSR	RD, @CK.CHR	: CHECK ONE CHARACTER
4625	021170	021234			B\$		: ILLEGAL CHARACTER
4626	021172	021224			6\$		: CARRIAGE RETURN
4627	021174	021234			B\$		: "/"
4628	021176	021234			B\$		: "."
4629	021200	021204			4\$		: "
4630	021202	021214			5\$		: DIGIT 0-9
4631	021204	005724		4\$:	TST	(R4)+	: "PERIOD-PERIOD"
4632	021206	105711			TSTB	(R1)	: "CR"?
4633	021210	001405			BEQ	6\$	: YES--BRANCH
4634	021212	000410			BR	8\$	
4635	021214	126127	177776 000054	5\$:	CMPB	-2(R1), #'	: WAS CHARACTER BEFORE THE DIGIT A COMMA?
4636	021222	001004			BNE	8\$	: NO--EXIT
4637	021224	020316		6\$:	CMP	R3, (SP)	: INPUT TO LARGE?
4638	021226	101001			BHI	7\$	: YES -- BRANCH
4639	021230	060400			ADD	R4, RD	: ADJUST RETURN ADDRESS
4640	021232	005720		7\$:	TST	(RD)+	
4641	021234	010302		8\$:	MOV	R3, R2	: NUMBER TO R2
4642	021236	005726			TST	(SP)+	: CLEAN MAX. SIZE OFF OF STACK
4643	021240	012603			MOV	(SP)+, R3	: RESTORE R3
4644	021242	012604			MOV	(SP)+, R4	: RESTORE R4
4645	021244	011000			MOV	(RD), RD	: GET RETURN ADDRESS
4646	021246	000200			RTS	RD	: RETURN

```

4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660 021250 010346
4661 021252 005003
4662 021254 004037 020740
4663 021260 000440
4664 021262 005201
4665 021264 006303
4666 021266 103435
4667 021270 006303
4668 021272 103433
4669 021274 006303
4670 021276 103431
4671 021300 060203
4672 021302 004037 020740
4673 021306 000401
4674 021310 000764
4675 021312 010302
4676 021314 005003
4677 021316 004037 021014
4678 021322 021362
4679 021324 021362
4680 021326 021362
4681 021330 021352
4682 021332 021336
4683 021334 021362
4684 021336 005723
4685 021340 121127 000056
4686 021344 001002
4687 021346 005201
4688 021350 005723
4689 021352 005723
4690 021354 105711
4691 021356 001001
4692 021360 060300
4693 021362 012603
4694 021364 011000
4695 021366 000200
4696
4697
4698
4699
4700 021370 000122
4701 021372 041506 000
4702 021375 114 000103

```

```

:*****
.SBTTL CK.NUM - CHECK NUMBER (OCTAL)
*THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
*AND FORMS AN OCTAL NUMBER IN R2
CALL
      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
      JSR      RD,R2,CK.NUM ;GO FORM THE NUMBER
      RETURN   ADR1         ;ILLEGAL CHARACTER IN THE INPUT STRING
      RETURN   ADR2         ;"COMMA"--R2=NUMBER
      RETURN   ADR3         ;"PERIOD"--R2=NUMBER
      RETURN   ADR4         ;"PERIOD-PERIOD"--R2=NUMBER
:
CK.NUM: MOV      R3,-(SP)    ;SAVE R3
      CLR      R3          ;START NUMBER AT ZERO
      JSR      RD,R2,CK.OCT ;OCTAL DIGIT?
      BR      6$          ;NO--BRANCH
1$:     INC      R1         ;MOVE TO NEXT CHARACTER
      ASL      R3          ;FOR THE OCTAL NUMBER IN R3
      BCS     6$          ;DON'T LET IT GET TO BIG
      ASL      R3
      BCS     6$
      ASL      R3
      BCS     6$
      ADD     R2,R3
      JSR      RD,R2,CK.OCT ;IS THIS AN OCTAL DIGIT?
      BR      2$          ;NO--FIND OUT WHAT IT IS
      BR      1$          ;YES--MAKE IT PART OF THE NUMBER
2$:     MOV      R3,R2     ;SAVE THE OCTAL NUMBER
      CLR      R3         ;START WITH ZERO INDEX
      JSR      RD,R2,CHR   ;CHECK ONE CHARACTER
      6$          ;ILLEGAL CHARACTER
      6$          ;CARRIAGE RETURN
      6$          ;"/"
      6$          ;" "
      6$          ;"'"
      6$          ;DIGIT 0-9
3$:     TST      (R3)+     ;"PERIOD"
      CMPB     (R1),#'.   ;"PERIOD-PERIOD"?
      BNE     5$          ;NO--BRANCH
      INC     R1         ;YES--ADVANCE THE POINTER
4$:     TST      (R3)+     ;"PERIOD-PERIOD"
5$:     TST      (R3)+     ;"COMMA"
      TSTB     (R1)       ;"CR"?
      BNE     6$          ;NO--BRANCH
6$:     ADD     R3,R0     ;YES--SAVE THE OCTAL NUMBER
      MOV     (SP)+,R3    ;RESTORE R3
      MOV     (R0),R0    ;PICKUP EXIT ADDRESS
      RTS      R0        ;RETURN
:
.SBTTL ASCIZ MESSAGES
MSG.R: .ASCIZ /R/
MSG.FC: .ASCIZ /FC/
MSG.LC: .ASCIZ /LC/

```

4703	021400	041511	000		MSG.IC: .ASCIZ /IC/
4704	021403	124	000113		MSG.TK: .ASCIZ /TK/
4705	021406	000075			MSG.EQ: .ASCIZ /=/
4706	021410	005015	050122	030461	MRPCS: .ASCIZ <15><12>/RP11=/
4707	021416	000075			
4708	021420	005015	050122	042526	MRPVEC: .ASCIZ <15><12>/RPVEC=/
4709	021426	036503	000		
4710	021431	015	051012	050120	MRPPRI: .ASCIZ <15><12>/RPPRI0=/
4711	021436	044522	036517	000	
4712					
4713	021443	040	020040	020040	GRPH1: .ASCII / 0 10 20 30 40 50 60 70 80 90 100/<15><12>
4714	021450	020040	020060	020040	
4715	021456	030061	020040	030062	
4716	021464	020040	030063	020040	
4717	021472	030064	020040	030065	
4718	021500	020040	030066	020040	
4719	021506	030067	020040	030070	
4720	021514	020040	030071	020040	
4721	021522	030061	006460	012	
4722	021527	040	020040	020040	.ASCIZ / I---I---I---I---I---I---I---I---I---I---I/<15><12>
4723	021534	020040	026511	026455	
4724	021542	026511	026455	026511	
4725	021550	026455	026511	026455	
4726	021556	026511	026455	026511	
4727	021564	026455	026511	026455	
4728	021572	026511	026455	026511	
4729	021600	026455	026511	026455	
4730	021606	006511	000012		
4731					.EVEN
4732					
4733					
4734					.SBTTL ERROR HEADER (EM) MESSAGES
4735					
4736	021612	050122	030461	043040	EM1: .ASCIZ @RP11 FAILED TO RESPOND TO ADDRESSING@
4737	021620	044501	042514	020104	
4738	021626	047524	051040	051505	
4739	021634	047520	042116	052040	
4740	021642	020117	042101	051104	
4741	021650	051505	044523	043516	
4742	021656	000			
4743	021657	104	044522	042526	EM2: .ASCIZ @DRIVE UNSAFE@
4744	021664	052440	051516	043101	
4745	021672	000105			
4746	021674	051104	053111	020105	EM3: .ASCIZ @DRIVE OFFLINE@
4747	021702	043117	046106	047111	
4748	021710	000105			
4749	021712	051104	053111	020105	EM4: .ASCIZ @DRIVE ERROR DURING TIMING TEST@
4750	021720	051105	047522	020122	
4751	021726	052504	044522	043516	
4752	021734	052040	046511	047111	
4753	021742	020107	042524	052123	
4754	021750	000			
4755	021751	116	020117	047111	EM5: .ASCIZ @NO INTERRUPT FROM POSITIONING AFTER 1 SECOND@
4756	021756	042524	051122	050125	
4757	021764	020124	051106	046517	
4758	021772	050040	051517	052111	

4759	022000	047511	044516	043516	
4760	022006	040440	052106	051105	
4761	022014	030440	051440	041505	
4762	022022	047117	000104		
4763	022026	044504	045523	042440	EM6: .ASCIZ @DISK ERROR AFTER POSITIONING@
4764	022034	051122	051117	040440	
4765	022042	052106	051105	050040	
4766	022050	051517	052111	047511	
4767	022056	044516	043516	000	
4768	022063	103	051117	042522	EM7: .ASCIZ @CORRECT ATTN BIT NOT SET AFTER POSITIONING@
4769	022070	052103	040440	052124	
4770	022076	020116	044502	020124	
4771	022104	047516	020124	042523	
4772	022112	020124	043101	042524	
4773	022120	020122	047520	044523	
4774	022126	044524	047117	047111	
4775	022134	000107			
4776	022136	051104	053111	020105	EM10: .ASCIZ @DRIVE OFFLINE AFTER POSITIONING@
4777	022144	043117	046106	047111	
4778	022152	020105	043101	042524	
4779	022160	020122	047520	044523	
4780	022166	044524	047117	047111	
4781	022174	000107			
4782	022176	051447	041525	023501	EM11: .ASCIZ @'SUCA' NOT CORRECT AFTER POSITIONING@
4783	022204	047040	052117	041440	
4784	022212	051117	042522	052103	
4785	022220	040440	052106	051105	
4786	022226	050040	051517	052111	
4787	022234	047511	044516	043516	
4788	022242	000			
4789	022243	104	051511	020113	EM12: .ASCIZ @DISK ERROR WHILE VERIFYING POSITION@
4790	022250	051105	047522	020122	
4791	022256	044127	046111	020105	
4792	022264	042526	044522	054506	
4793	022272	047111	020107	047520	
4794	022300	044523	044524	047117	
4795	022306	000			
4796	022307	104	051511	020113	EM13: .ASCIZ @DISK POSITIONED TO WRONG CYLINDER@
4797	022314	047520	044523	044524	
4798	022322	047117	042105	052040	
4799	022330	020117	051127	047117	
4800	022336	020107	054503	044514	
4801	022344	042116	051105	000	
4802	022351	116	020117	047111	EM14: .ASCIZ @NO INTERRUPT FROM I/O AFTER 1 SECOND@
4803	022356	042524	051122	050125	
4804	022364	020124	051106	046517	
4805	022372	044440	047457	040440	
4806	022400	052106	051105	030440	
4807	022406	051440	041505	047117	
4808	022414	000104			
4809	022416	042523	045505	044440	EM15: .ASCIZ @SEEK INCOMPLETE@
4810	022424	041516	046517	046120	
4811	022432	052105	000105		
4812	022436	051104	053111	020105	EM16: .ASCIZ @DRIVE NOT READY AFTER POSITIONING@
4813	022444	047516	020124	042522	
4814	022452	042101	020131	043101	

4815	022460	042524	020122	047520
4816	022466	044523	044524	047117
4817	022474	047111	000107	
4818				
4819	022500	051104	053111	020105
4820	022506	047516	020124	042522
4821	022514	042101	027531	043117
4822	022522	046106	047111	020105
4823	022530	052504	044522	043516
4824	022536	052040	046511	047111
4825	022544	020107	042524	052123
4826	022552	000		

EM17: .ASCIZ @DRIVE NOT READY/OFFLINE DURING TIMING TEST@

.SBTTL DATA HEADER (DH) MESSAGES

4830	022553	105	051122	050040
4831	022560	020103	051040	041520
4832	022566	000123		
4833	022570	042524	052123	021440
4834	022576	020040	051105	020122
4835	022604	041520	020040	051104
4836	022612	053111	020105	020040
4837	022620	050122	051504	020040
4838	022626	020040	050122	051105
4839	022634	020040	020040	050122
4840	022642	051503	000	
4841	022645	124	051505	020124
4842	022652	020043	042440	051122
4843	022660	050040	020103	042040
4844	022666	044522	042526	043040
4845	022674	047522	020115	054503
4846	022702	020114	042504	052123
4847	022710	041440	046131	051440
4848	022716	041525	020101	020040
4849	022724	051040	041520	000101
4850	022732	042524	052123	021440
4851	022740	020040	051105	020122
4852	022746	041520	020040	051104
4853	022754	053111	020105	020040
4854	022762	050122	051504	020040
4855	022770	020040	050122	051105
4856	022776	020040	020040	050122
4857	023004	051503	020040	020040
4858	023012	050122	040503	020040
4859	023020	020040	050122	040504
4860	023026	000		
4861	023027	124	051505	020124
4862	023034	020043	042440	051122
4863	023042	050040	020103	042040
4864	023050	044522	042526	043040
4865	023056	047522	020115	054503
4866	023064	020114	042504	052123
4867	023072	041440	046131	044040
4868	023100	051104	041440	046131
4869	023106	000		
4870	023107	122	041520	020123

DH1: .ASCIZ @ERR PC RPCS@

DH2: .ASCIZ @TEST # ERR PC DRIVE RPDS RPER RPCS@

DH11: .ASCIZ @TEST # ERR PC DRIVE FROM CYL DEST CYL SUCA RPCA@

DH12: .ASCIZ @TEST # ERR PC DRIVE RPDS RPER RPCS RPCA RPDA@

DH13: .ASCIZ @TEST # ERR PC DRIVE FROM CYL DEST CYL HDR CYL@

DH13A: .ASCIZ @RPCS RPCA RPDA SUCA@



4871 023114 020040 051040 041520  
 4872 023122 020101 020040 051040  
 4873 023130 042120 020101 020040  
 4874 023136 051440 041525 000101  
 4875 023144 042524 052123 021440  
 4876 023152 020040 051105 020122  
 4877 023160 041520 020040 051104  
 4878 023166 053111 020105 020040  
 4879 023174 050122 051504 020040  
 4880 023202 020040 050122 051105  
 4881 023210 020040 020040 050122  
 4882 023216 051503 020040 020040  
 4883 023224 050122 041527 000  
 4884 023231 122 041120 020101  
 4885 023236 020040 051040 041520  
 4886 023244 020101 020040 051040  
 4887 023252 042120 020101 020040  
 4888 023260 051040 046520 020061  
 4889 023266 020040 051440 041525  
 4890 023274 020101 020040 051440  
 4891 023302 046111 000117

DHSP: .ASCIZ @TEST # ERR PC DRIVE RPDS RPER RPCS RPWC@

DHSPA: .ASCIZ @RPBA RPCA RPDA RPM1 SUCA SILO@

4892  
 4893  
 4894  
 4895  
 4896  
 4897  
 4898 023306 001116 001266  
 4899 023312 001204 001116 001246  
 4900 023320 002004 002006 002010  
 4901 023326 001204 001116 001246  
 4902 023334 001254 001262 002024  
 4903 023342 002016  
 4904 023344 001204 001116 001246  
 4905 023352 002004 002006 002010  
 4906 023360 002016 002020  
 4907 023364 001204 001116 001246  
 4908 023372 001254 001262 001126  
 4909 023400 002010 002016 002020  
 4910 023406 002024  
 4911 023410 001204 001116 001246  
 4912 023416 002004 002006 002010  
 4913 023424 002012  
 4914 023426 002014 002016 002020  
 4915 023434 002022 002024 002026

.EVEN

.SBTTL DATA TABLE (DT)

DT1: .WORD \$ERRPC,RPADR  
 DT2: .WORD \$TMPD,\$ERRPC,CHKDRV,\$RPDS,\$RPER,\$RPCS  
 DT11: .WORD \$TMPD,\$ERRPC,CHKDRV,CYL.CR,CYL.DS,\$SUCA,\$RPCA  
 DT12: .WORD \$TMPD,\$ERRPC,CHKDRV,\$RPDS,\$RPER,\$RPCS,\$RPCA,\$RPDA  
 DT13: .WORD \$TMPD,\$ERRPC,CHKDRV,CYL.CR,CYL.DS,\$BDDAT  
 DT13A: .WORD \$RPCS,\$RPCA,\$RPDA,\$SUCA  
 DTSP: .WORD \$TMPD,\$ERRPC,CHKDRV,\$RPDS,\$RPER,\$RPCS,\$RPWC  
 DTSPA: .WORD \$RPBA,\$RPCA,\$RPDA,\$RPM1,\$SUCA,\$SILO

.SBTTL DATA FORMAT (DF) TABLE

4916  
 4917  
 4918  
 4919 023442 000001  
 4920 023444 002  
 4921 023445 000  
 4922  
 4923 023446 000001  
 4924 023450 006  
 4925 023451 000  
 4926

DF1: .WORD 1 ;NUMBER OF DATA HEADERS  
 .BYTE 2 ;NUMBER OF WORDS IN DATA TABLE  
 .BYTE 0 ;BOTH NUMBERS ARE OCTAL  
 DF2: .WORD 1  
 .BYTE 6  
 .BYTE 0

4927	023452	000001	DF11:	.WORD	1	
4928	023454	007		.BYTE	7	
4929	023455	000		.BYTE	0	
4930						
4931	023456	000001	DF12:	.WORD	1	
4932	023460	010		.BYTE	8.	
4933	023461	000		.BYTE	0	
4934						
4935	023462	000002	DF13:	.WORD	2	;2 DH'S TO BE TYPED
4936	023464	006		.BYTE	6	
4937	023465	000		.BYTE	0	
4938	023466	023107		.WORD	DH13A	
4939	023470	004		.BYTE	4	
4940	023471	000		.BYTE	0	
4941						
4942	023472	000002	DFSP:	.WORD	2	
4943	023474	007		.BYTE	7	
4944	023475	000		.BYTE	0	
4945	023476	023231		.WORD	DHSPA	
4946	023500	006		.BYTE	6	
4947	023501	000		.BYTE	0	
4948						
4949			.EVEN			
4950						
4951	023502	023144	ETSP:	.WORD	DHSP	;MESSAGE BLOCK FOR SWITCH 6 TYPEOUTS
4952	023504	023410		.WORD	DTSP	
4953	023506	023472		.WORD	DFSP	
4954						
4955	023510	000030	BUFR4:	.BLKH	24.	;WORKING LOCATIONS FOR SEEK TIMER TEST
4956	023570	000030	BUFR5:	.BLKH	24.	
4957	023550	000030	BUFR6:	.BLKH	24.	
4958	023730	000030	BUFR7:	.BLKH	24.	
4959	024010	000626	FRMSTR:	.BLKH	406.	
4960	025464	000626	RVSTR:	.BLKH	406.	
4961		027140	BUFFER=.			
4962						
4963		000001	.END			











RPCA = 000012	1532#	1924	3615*	3784	4058*								
RPCS = 000004	1529#	3572*	3573*	3575*	3611*	3612*	3613*	3618*	3619*	3641*	3642*	3661*	3662*
	3731*	3739*	3740*	3750*	3751*	3781	3819*	3820*	3825*	3845*	3846*	4061*	4093*
	4094*	4109	4112*	4113*	4114*								
RPOA = 000014	1533#	3616*	3708	3730*	3785	4047	4050	4051	4063	4064	4068	4069	4098
RPOS = 000000	1527#	3576	3580	3582	3584	3587	3779	4057*	4083				
RPER = 000002	1528#	3780											
RPINIT 013612	1777	3562#											
RPM1 = 000016	1534#	3786											
RPM2 = 000020	1535#												
RPM3 = 000022	1536#												
RPPRI0 001272	1321#	3565	3607										
RPT 001374	1381#	4439	4470*	4472*	4476*								
RPT0 001634	1439#	1970											
RPT1 001642	1444#	1999											
RPT10 001766	1500#	2315											
RPT11 001772	1504#	2365											
RPT12 002002	1510#	2430											
RPT2 001656	1452#	2034											
RPT3 001672	1460#	2071											
RPT4 001706	1468#	2107											
RPT5 001722	1476#	2183											
RPT6 001736	1484#	2224											
RPT7 001752	1492#	2274											
RPVEC 001270	1320#	3563	3617*	3721*	3821*	4056*							
RPMC = 000006	1530#	3728*	3782										
RSTR1 003674	1902#	1911											
RSTR2 003706	1904#	1908											
RSTR3 003714	1905#												
RVSTR 025464	2336	2378	3954	3955	4960#								
RO =x000000	1094#	1735*	1736*	1737	1756*	1759	1760	1857*	1859	1861	1862*	1871*	1889
	1899#	1923#	1924	1977*	2006*	2041*	2078*	2114*	2129*	2135*	2149*	2155*	2190*
	2231*	2281*	2322*	2346*	2372*	2404*	2437*	2477*	2480	2486	2490	2492*	2494
	2496*	2500	2502*	2504	2508*	2510*	2517*	2525	2537*	2538	2609*	2612	2680*
	2681*	2682*	2683*	2684*	2685*	2686*	2687	2695*	2696	2701	2704*	2705	2709
	2710	2729	2763	2764*	2765	2768*	2904	2914*	2918	2934	2935	2948*	3290
	3315*	3333	3346*	3348*	3349*	3355*	3357*	3359	3363*	3374	3375*	3376	3377*
	3378*	3379*	3380*	3445*	3446	3464*	3468*	3503	3504*	3505	3507	3509*	3510
	3513*	3529	3532*	3535*	3539*	3542*	3545	3549*	3571*	3572*	3573*	3575*	3576
	3580	3582	3584	3587	3608*	3611*	3612*	3613*	3615*	3616*	3618*	3619*	3623*
	3641*	3642*	3647*	3661*	3662*	3682*	3684	3707*	3708	3728*	3729*	3730*	3731*
	3735*	3739*	3740*	3746*	3750*	3751*	3777	3778*	3779	3780	3781	3782	3783
	3784	3785	3786	3787	3788	3789*	3818*	3819*	3820*	3825*	3829*	3845*	3846*
	3873	3875	3885	3890*	3905	3912*	3916	3918	3919*	3922	3928*	3938	3940
	3943	3984*	3987	3992	3999	4007	4020	4021	4024*	4038	4042*	4045*	4047
	4050	4051	4057*	4058*	4061*	4063	4064	4068	4069	4093*	4094*	4098	4112*
	4113*	4114*	4139*	4157*	4159	4160*	4164*	4166	4167*	4171*	4173	4178*	4182*
	4184*	4186*	4225*	4233*	4238	4239	4251	4252	4258*	4276*	4286*	4317*	4321*
	4335*	4339*	4416*	4417	4425	4444*	4464*	4489*	4491	4498	4502	4525	4526*
	4545	4546*	4571*	4573	4574	4575	4576	4577	4579*	4580*	4602*	4615*	4624*
	4639*	4640	4645*	4646*	4662*	4672*	4677*	4692*	4694*	4695*			
R1 =x000001	1095#	1757*	1758	1759*	1808*	1811	1816	1829	1848*	1858*	1863*	1872*	1885*
	1886*	1893	2085*	2086	2088*	2089	2123*	2125	2141*	2142	2144*	2145	2161*
	2162	2195*	2197	2201*	2203	2236*	2239	2241	2253*	2286*	2288	2296*	2297
	2377*	2388*	2443*	2448	2486*	2487*	2488	2518*	2522*	2524*	2701*	2713	2717
	2905	2918*	2919	2923	2947*	3291	3314*	3334	3337*	3340*	3350*	3356*	3360







SW00	=	000001	1131#	1141																	
SW01	=	000002	1130#	1140																	
SW02	=	000004	1129#	1139																	
SW03	=	000010	1128#	1138	3680																
SW04	=	000020	1127#	1137																	
SW05	=	000040	1126#	1136																	
SW06	=	000100	1125#	1135	2692																
SW07	=	000200	1124#	1134	1801																
SW08	=	000400	1123#	1133																	
SW09	=	001000	1122#	1132	3655	3677															
SW1	=	000002	1140#	2119																	
SW10	=	002000	1121#																		
SW11	=	004000	1120#																		
SW12	=	010000	1119#																		
SW13	=	020000	1118#																		
SW14	=	040000	1117#																		
SW15	=	100000	1116#																		
SW2	=	000004	1139#	3876																	
SW3	=	000010	1138#																		
SW4	=	000020	1137#	2470																	
SW5	=	000040	1136#																		
SW6	=	000100	1135#																		
SW7	=	000200	1134#																		
SW8	=	000400	1133#																		
SW9	=	001000	1132#																		
TBITVE	=	000014	1174#																		
TEST0		004134	1971	1982#																	
TEST1		004252	2000	2010#																	
TEST10		006120	2316	2325#																	
TEST11		006374	2366	2375#																	
TEST12		006706	2431	2440#																	
TEST2		004410	2035	2045#																	
TEST3		004602	2072	2082#																	
TEST4		004766	2108	2120	2122#																
TEST5		005314	2184	2194#																	
TEST6		005502	2225	2235#																	
TEST7		005734	2275	2285#																	
TIMEE	=	000020	1216#																		
TIMSEK		016476	2329	2337	2381	2383	2386	2446	2449	4038#											
TIMSK1		016520	4041	4043#																	
TIMSK2		017036	4056	4097#																	
TK		001404	1385#																		
TKVEC	=	000060	1181#	1766#	2981#	2982#															
TK0		001636	1440#	1981																	
TK1		001652	1448#	2009																	
TK2		001666	1456#	2044																	
TK3		001702	1464#	2081																	
TK4		001716	1472#	2117																	
TK5		001732	1480#	2193																	
TK6		001746	1488#	2234																	
TK7		001762	1496#	2284																	
TPVEC	=	000064	1182#																		
TRAPVE	=	000034	1180#	1712#	1713#																
TRK.RD		001256	1313#	3724#																	
TRTVEC	=	000014	1175#																		
TSTNMS		001234	1303#	1755#	1967	1996	2031	2068	2104	2180	2221	2271	2312	2362	2427						







STRPAD	013170	3379	3396#											
STSTNM	001102	1251#	1973#	1975	2002*	2004	2037*	2039	2074*	2076	2110*	2112	2186*	2188
		2227*	2229	2277*	2279	2318*	2320	2368*	2370	2433*	2435	2594*	2641	2666
		2679	3220	3262*	3267	3271								
		3157	3158	3170	3188	3202	3206*							
STTYIN	012406	3402												
STYPBN=	***** U	2903#	3401											
STYPDS	010776	2759#	3389	3397										
STYPE	010330	2780	2787	2794	2799#	2800	3105							
STYPEC	010500	2805	2807	2810#										
STYPEX	010546	2843#	3398											
STYPOC	010574	2842	2845#	3400										
STYPOH	010610	2838#	3399											
STYPOS	010550	3233#												
SXTSTR	012514	2611#												
SSGET4=	000000	2839#	2843*	2853	2888#									
SOFILL	010773	2651	3230											
S4OCAT=	***** U	1044#	1048#	1056	1057#	1059#	1061#	1064#	1248#	1297	1705	1717	1718	1774
	= 027140	1775	1834#	1840#	1846#	1869#	1877#	2344#	2393#	2460#	2468#	2576#	2619	2620#
		2666	2812	2957#	2962	2966#	2967	2968#	3206#	3207	3214	3270	3271	3491#
		3972#	4409#	4955#	4956#	4957#	4958#	4959#	4960#	4961				





.KT11	18					
.SETUP	18	10138	1689			
.SURI	18	10138	1024			
.SURL	10348	1035	1036	1037	1038	1039
.SACT1	18	10138	1052			
.SAPT8	18					
.SAPTH	18					
.SAPTY	18					
.SASTA	18					
.SCATC	18	10138	1042			
.SCHTA	18	10138	1242			
.SDB2D	18	10138	3430			
.SDB20	18					
.SDIV	18					
.SEOP	18	10138	2564			
.SERRO	18	10138	2623			
.SERRT	18					
.SHULT	18	10138	3318			
.SPOWE	18					
.SRAND	18	10138	3517			
.SRDEE	18	10138				
.SRDOC	18	10138				
.SREAO	18	10138	2959			
.SR2AZ	18					
.SSAVE	18	10138	3272			
.SSB2D	18	10138	3411			
.SSB20	18					
.SSCOP	18	10138	3215			
.SSIZE	18					
.SSUPR	18	10138	3493			
.STRAP	18	10138	3366			
.STYP8	18					
.STYPD	18	10138	2891			
.STYPE	18	10138	2742			
.STYPO	18	10138	2813			
.S4OCA	18					
.1170	18					

RDC	3460	3540	3543												
ADD	2048	2141	2201	2243	2247	2251	2253	2292	2296	2496	2502	2508	2510	2686	2769
	2841	2851	2923	3080	3090	3099	3348	3425	3459	3461	3539	3541	3542	3544	3714
ASL	3764	4084	4125	4327	4345	4613	4614	4639	4671	4692					
	2088	2487	2683	2684	2685	3113	3114	3115	3378	3535	4174	4175	4176	4324	4325
	4326	4330	4342	4343	4344	4348	4403	4503	4609	4611	4612	4665	4667	4669	
ASLB	1863	1907	2928												
ASR	1885	1886	3710	3711	3712	3713	3754	3755	3756	3757	3758	3759	4116	4117	4118
	4119	4255	4256	4257	4420	4429									
BCC	1887	1908	2929	3347											
BCC	2712	4421	4430	4666	4668	4670									
BEG	1780	1797	1800	1817	1853	1882	1894	2120	2128	2134	2148	2154	2505	2578	2610
	2640	2643	2659	2662	2688	2694	2697	2702	2706	2720	2772	2807	2868	3005	3058
	3097	3112	3140	3176	3245	3249	3354	3506	3508	3577	3581	3585	3588	3627	3650
	3656	3659	3670	3675	3678	3681	3748	3761	3833	3853	3877	3941	3947	3953	3988
	4000	4002	4049	4067	4078	4089	4105	4110	4155	4196	4246	4313	4355	4361	4363
	4378	4380	4387	4405	4422	4431	4495	4497	4500	4505	4564	4566	4568	4570	4633
BGE	2050	2163	2206	2294	2298	2452	3259	4351	4399						
BGT	2139	2159	2204	2240	2600	2875	2937	3029	3109	4075	4329	4347			
BHI	3247	3716	4522	4542	4638										
BIC	1909	2499	2597	2865	2995	3020	3030	3048	3075	3110	3709	3880	4087	4115	4524
	4544														
BIS	2870	2871	2931	2932	3117	3463	3727	3824	4060	4352	4358	4366			
BISB	1861	3589	3619	3801	4292	4297									
BIT	1801	1904	1967	1996	2031	2068	2104	2119	2180	2221	2271	2312	2362	2427	2470
	2642	2649	2658	2692	3230	3248	3255	3576	3580	3582	3584	3587	3626	3630	3634
	3639	3649	3655	3658	3677	3680	3703	3747	3832	3836	3840	3852	3876	4048	4053
	4066	4071	4077	4100	4104	4109	4354	4404	4485	4494	4496	4499	4504		
BITB	1829	2795	3666												
BLE	1860	2055	2090	2143	2333	2491	2495	2501	2724	3917					
BLO	1761	3171	3717	3883	4520	4540									
BLOS	3159														
BLT	2548	2786	2876	2920	2936	3027	3107	3456	4481						
BMI	1818	2927													
BNE	1705	1726	1738	1740	1748	1750	1785	1802	1830	1850	1864	1873	1880	1884	1905
	1968	1997	2032	2069	2105	2181	2222	2272	2313	2363	2389	2428	2471	2507	2523
	2539	2650	2727	2734	2766	2774	2782	2796	2803	2866	2925	2997	3003	3009	3014
	3022	3034	3044	3050	3078	3082	3088	3095	3102	3145	3163	3165	3181	3185	3195
	3231	3256	3352	3467	3538	3583	3631	3635	3640	3667	3704	3837	3841	3887	3925
	3927	3951	4019	4022	4052	4054	4065	4070	4072	4101	4243	4248	4250	4294	4309
	4311	4333	4365	4369	4372	4374	4383	4390	4452	4455	4458	4460	4486	4636	4686
	4691														
BPL	2340	2552	2655	2760	2800	2864	2911	2941	3018	3046	3072	3338	3342	3591	
BR	1690	1693	1695	1728	1742	1752	1762	1805	1820	1823	1826	1832	1835	1838	1841
	1844	1855	1867	1875	1896	1900	1911	1915	2254	2342	2391	2398	2458	2462	2466
	2483	2493	2497	2503	2509	2511	2521	2527	2531	2555	2574	2580	2588	2605	2715
	2722	2736	2762	2779	2789	2798	2805	2842	2857	2878	2922	2939	3012	3091	3118
	3120	3174	3183	3189	3191	3233	3239	3242	3251	3254	3458	3579	3663	3673	3719
	3752	3874	3888	3948	3966	3970	3974	3978	3995	4010	4013	4016	4161	4162	4163
	4168	4169	4170	4179	4180	4181	4183	4185	4241	4244	4271	4296	4303	4318	4322
	4336	4340	4356	4367	4375	4396	4407	4419	4424	4426	4433	4453	4471	4487	4572
	4634	4663	4673	4674											
CLR	1691	1692	1696	1703	1714	1715	1736	1767	1768	1769	1770	1771	1808	1851	1856
	1857	1870	1871	1902	1979	1980	2008	2043	2080	2116	2118	2124	2140	2160	2192
	2233	2283	2324	2374	2439	2443	2444	2477	2478	2517	2593	2594	2595	2678	2680
	2703	2855	2914	2917	2978	3015	3061	3069	3070	3134	3135	3156	3179	3253	3265

# F10

	3336	3346	3452	3564	3566	3567	3568	3569	3570	3620	3645	3653	3664	3718	3732
	3743	3803	3826	3850	3881	3915	3932	3933	3934	3935	3936	3984	3985	4043	4062
	4099	4156	4197	4198	4269	4295	4314	4315	4316	4394	4428	4599	4600	4601	4661
CLRB	4676														
CMP	2778	2804	2943	3186	3196	3252	3468								
	1704	1725	1737	1749	1760	2049	2054	2089	2138	2142	2158	2162	2203	2205	2239
	2293	2297	2332	2451	2490	2494	2500	2504	2506	2538	2935	2996	3002	3004	3008
	3013	3021	3026	3028	3033	3043	3049	3077	3087	3094	3106	3108	3144	3158	3170
	3240	3258	3353	3462	3669	3715	3760	3889	3916	3940	4021	4051	4064	4069	4074
CMPB	4088	4193	4328	4346	4349	4398	4480	4637							
	1883	2771	2773	2781	2802	2806	3057	3081	3101	3162	3180	3184	3194	3246	3507
	3674	4247	4308	4310	4332	4360	4364	4368	4377	4457	4519	4521	4539	4541	4565
	4567	4569	4635	4685											
COM	2121	2707	2726												
DEC	1849	2335	2339	2388	2551	2598	2682	2719	2723	3141	3169	3343	3351	3465	3509
	3590	3886	3924	3946	3950	4283									
DECB	2785	2788	2863	2874											
ENT	1078														
HALT	1048	2656	2761												
INC	1739	1799	1848	1862	1888	1899	1906	2137	2157	2331	2450	2492	2522	2537	2596
	2645	2869	2877	2921	3025	3032	3116	3143	3257	3339	3457	3537	3921	3949	3952
	4015	4073	4102	4320	4323	4334	4338	4341	4357	4370	4376	4385	4397	4578	4664
	4687														
INCB	2639	2808	3262												
IOT	1079														
JMP	1066	1068	1070	1072	1786	1878	1969	1998	2033	2070	2106	2182	2223	2273	2314
	2364	2429	2512	2540	2617	3001	3084	3629	3633	3637	3644	3657	3679	3742	3804
	3835	3839	3843	3848	3855	3958	4080	4091	4096	4108	4199	4381	4384	4388	4391
	4401	4456	4461	4475	4484										
JSR	1745	1764	1776	1777	1854	1983	2012	2014	2047	2052	2084	2087	2126	2129	2132
	2135	2146	2149	2152	2155	2198	2200	2242	2244	2246	2248	2250	2252	2289	2291
	2326	2329	2337	2346	2376	2381	2383	2386	2395	2396	2402	2403	2404	2441	2446
	2449	2489	2535	2612	2651	2780	2787	2794	2999	3060	3105	3424	3623	3625	3648
	3654	3672	3676	3682	3735	3737	3745	3829	3831	3851	3878	4045	4076	4103	4107
	4122	4129	4160	4167	4178	4225	4233	4276	4286	4317	4321	4335	4339	4440	4441
	4444	4464	4474	4478	4571	4602	4615	4624	4662	4672	4677				
MOV	1689	1694	1697	1698	1702	1706	1708	1709	1710	1711	1712	1713	1717	1718	1721
	1722	1723	1724	1729	1731	1732	1733	1735	1755	1756	1757	1758	1759	1765	1766
	1772	1774	1775	1778	1781	1782	1783	1788	1789	1790	1791	1792	1793	1794	1795
	1798	1809	1811	1858	1872	1889	1901	1903	1912	1918	1923	1924	1970	1971	1972
	1973	1975	1976	1977	1982	1999	2000	2001	2002	2004	2005	2006	2010	2011	2013
	2034	2035	2036	2037	2039	2040	2041	2045	2046	2051	2071	2072	2073	2074	2076
	2077	2078	2082	2083	2085	2086	2107	2108	2109	2110	2112	2113	2114	2122	2123
	2125	2131	2144	2145	2151	2183	2184	2185	2186	2188	2189	2190	2194	2195	2196
	2197	2199	2224	2225	2226	2227	2229	2230	2231	2235	2236	2237	2241	2274	2275
	2276	2277	2279	2280	2281	2285	2286	2287	2288	2290	2295	2315	2316	2317	2318
	2320	2321	2322	2325	2327	2328	2330	2334	2336	2338	2345	2365	2366	2367	2368
	2370	2371	2372	2373	2375	2377	2378	2379	2380	2382	2384	2385	2387	2394	2401
	2430	2431	2432	2433	2435	2436	2437	2440	2442	2445	2447	2448	2480	2486	2488
	2498	2518	2519	2524	2525	2534	2545	2546	2558	2583	2591	2601	2609	2641	2646
	2660	2663	2687	2695	2696	2701	2704	2705	2713	2717	2729	2763	2964	2768	2783
	2838	2846	2847	2848	2854	2861	2879	2880	2881	2882	2883	2904	2905	2906	2907
	2908	2909	2910	2915	2918	2938	2944	2945	2946	2947	2948	2950	2951	2979	2980
	2981	2982	2984	3023	3035	3066	3083	3098	3103	3132	3133	3136	3146	3155	3157
	3168	3199	3200	3201	3202	3235	3236	3238	3241	3250	3260	3261	3263	3264	3267
	3268	3290	3291	3292	3293	3294	3295	3296	3297	3298	3299	3306	3307	3308	3309





	2464	2468	2485	2533	2557	2576	2582	2590	2607	2652	3127	3132	3254	3968	3972
. IFTF	3976	3980	3998	4012	4125	4132	4273	4305	4409						
	1744	1807	1822	1828	1834	1840	1846	1869	1877	1898	1917	2344	2393	2400	2460
	2464	2468	2485	2533	2557	2576	2582	2590	2607	2651	3064	3125	3128	3252	3968
. IIF	3972	3976	3980	3998	4012	4124	4131	4273	4305	4409					
	1013	1018	1023	1028	1029	1030	1031	1034	1035	1036	1037	1038	1039	1048	1297
	1707	1710	1714	1715	1717	1718	1812	1890	1919	2569	2584	2592	2594	2595	2619
	2620	2629	2630	2631	2632	2633	2638	2657	2665	2666	2714	2718	2812	2962	2968
	3007	3008	3067	3198	3207	3214	3221	3222	3223	3224	3225	3229	3253	3254	3267
. IRP	3270	3271	3397	3398	3399	3400	3401	3403	3405	3406	3407	3408	3409	4222	4411
	1298	1689	1787	1956	1986	2017	2058	2093	2166	2209	2257	2301	2350	2409	2573
	2904	2944	3290	3310	3333	3361	3529	3547	3779	3785					
. LIST	1	1013	1048	1184	1279	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290
	1291	1685	1689	1719	1744	1807	1822	1828	1834	1840	1846	1869	1877	1898	1917
	1928	1954	1956	1978	1986	2007	2017	2042	2058	2079	2093	2115	2166	2191	2209
	2232	2257	2282	2301	2323	2344	2350	2373	2393	2400	2409	2438	2460	2464	2468
	2485	2533	2557	2576	2582	2590	2594	2607	2611	2623	2665	3148	3225	3389	3397
	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407	3408	3409	3410	3968	3972
. MACRO	3976	3980	3998	4012	4123	4130	4273	4305	4409						
	1	1034	1242	1685	1956	1986	2017	2058	2093	2166	2209	2257	2301	2350	2409
. MCALL	3389														
. NLIST	1013	1184	1719												
	1	1013	1048	1184	1279	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290
	1291	1685	1689	1719	1744	1807	1822	1828	1834	1840	1846	1869	1877	1898	1917
	1928	1954	1956	1978	1986	2007	2017	2042	2058	2079	2093	2115	2166	2191	2209
	2232	2257	2282	2301	2323	2344	2350	2373	2393	2400	2409	2438	2460	2464	2468
	2485	2533	2557	2576	2582	2590	2594	2607	2611	2623	2665	3148	3225	3389	3397
	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407	3408	3409	3410	3968	3972
. NTYPE	3976	3980	3998	4012	4123	4130	4273	4305	4409						
. PAGE	4123	4130													
. REM	1242	1554	1685	1925											
. REPT	1														
. SBTTL	1048	1281	1290	1929	1950	3710	3753	4116							
	1024	1042	1052	1063	1074	1242	1554	1686	1700	1746	1926	1956	1986	2017	2058
	2093	2166	2209	2257	2301	2350	2409	2564	2622	2623	2668	2742	2813	2891	2959
	3215	3272	3318	3366	3389	3411	3430	3493	3517	3555	3596	3690	3770	3793	3807
	3861	3894	4027	4143	4202	4261	4509	4529	4549	4583	4649	4698	4734	4828	4896
. TITLE	4917														
. WORD	1013														
	1048	1049	1050	1060	1250	1253	1254	1255	1256	1259	1260	1261	1262	1263	1264
	1265	1268	1269	1270	1279	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290
	1298	1299	1300	1301	1302	1303	1304	1305	1307	1308	1309	1310	1311	1312	1313
	1314	1315	1316	1319	1320	1321	1322	1323	1324	1325	1353	1354	1355	1356	1357
	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1375	1380	1381	1382
	1383	1384	1385	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1401
	1402	1403	1404	1405	1408	1409	1410	1411	1412	1415	1416	1418	1420	1422	1424
	1426	1428	1430	1431	1433	1438	1439	1440	1443	1444	1445	1446	1447	1448	1451
	1452	1453	1454	1455	1456	1459	1460	1461	1462	1463	1464	1467	1468	1469	1470
	1471	1472	1475	1476	1477	1478	1479	1480	1483	1484	1485	1486	1487	1488	1491
	1492	1493	1494	1495	1496	1499	1500	1503	1504	1505	1506	1509	1510	1514	1515
	1516	1517	1518	1519	1520	1521	1522	1523	2130	2136	2150	2156	2347	2405	2599
	2602	2618	2691	2700	2731	2809	2889	2963	2964	2965	3396	3428	3466	3512	3551
	3552	3624	3663	3736	3830	3960	4046	4220	4395	4437	4898	4899	4901	4904	4907
	4909	4911	4914	4919	4923	4927	4931	4935	4938	4942	4945	4951	4952	4953	

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

\*,NOW,SEQ/SOL/CRF/PAGNUM/NL:TOC=DZRFZB.SML,DZRPZB.CMB  
RUN-TIME: 44 60 8 SECONDS  
RUN-TIME RATIO: 318/114=2.7  
CORE USED: 36K (71 PAGES)

