

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43

.REM @

IDENTIFICATION

PRODUCT CODE: AC-T116A-MC  
PRODUCT NAME: CZRNGAO RM80 FUNCTIONAL TEST, PT 4  
PRODUCT DATE: APRIL 1, 1982  
MAINTAINER: CX DIAGNOSTIC GROUP  
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44

- 1. ABSTRACT
- 2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PRELIMINARY PROGRAMS
  - 2.3 MEDIA
- 3. LOADING PROCEDURE
- 4. STARTING PROCEDURE
  - 4.1 STARTING ADDRESSES
  - 4.2 OPERATOR ACTION
  - 4.3 PROGRAM ACTION
    - 4.3.1 CONTROL SWITCH SELECTION
    - 4.3.2 RH CONTROLLER ADDRESS SELECTION
    - 4.3.3 DRIVE AND PARAMETER SELECTION
- 5. OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 CONTROL SWITCH SETTINGS
- 6. ERRORS
  - 6.1 ERROR TYPES
  - 6.2 ERROR RECOVERY
- 7. RESTRICTIONS
- 8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 TIMING TEST (TESTS 11 - 14) PRINTOUTS
  - 8.4 END OF TEST
- 9. PROGRAM DESCRIPTION
- 10. PROGRAM LISTING

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57

1. ABSTRACT

THIS PROGRAM CONTAINS A SERIES OF TESTS THAT WILL VERIFY THAT THE DISK IS CAPABLE OF PERFORMING SEEKS, THAT THE SEEK TIMES ARE WITHIN TOLERANCE AND THAT THE TRACK/SECTOR ADDRESSING CIRCUITRY OPERATES PROPERLY. THIS PROGRAM IS PRIMARILY USED TO DETERMINE IF THE SEEK TIMES ARE WITHIN TOLERANCES AND NOT FOR DATA RELIABILITY. THERE IS ABSOLUTELY NO WRITTING OF DATA INVOLVED IN THIS PROGRAM. HOWEVER, THIS PROGRAM DOES PERFORM READ HEADER AND DATA COMMANDS TO DETERMINE IF THE SEEKS PERFORMED ARE VALID.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR  
12K MEMORY  
TERMINAL  
PROGRAM LOADING DEVICE  
KW11-L OR KW11-P (THE KW11-P IS REQUIRED FOR THE TIMING TESTS)  
RH70 CONTROLLER  
1 TO 8 RM80 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RM80 DISKLESS TEST, PART 1 & 2  
RM80 FUNCTIONAL TEST, PART 1, 2 & 3

2.3 MEDIA

THE PROGRAM REQUIRES THAT EACH DRIVE TO BE TESTED HAS A FORMATTED DISK. THE DISK MAY BE FORMATTED IN EITHER 16-BIT OR 18-BIT MODE, DEPENDING ON THE TESTING REQUIREMENTS. NOTE THAT THE PROGRAM WILL NOT TEST A MIXTURE OF DRIVES WITH BOTH 16 AND 18 BIT MODE PACKS.

3. LOADING PROCEDURE

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCAITED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN.

4. STARTING PROCEDURE

58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114

#### 4.1 STARTING ADDRESSES

200 NORMAL STARTING ADDRESS  
204 SELECT OPERATING PARAMETERS  
210 SELECT RH CONTROLLER ADDRESSES  
214 COMBINATION OF 204 AND 210

NOTE: STARTING ADDRESSES 210 AND 214 ARE AVAILABLE WHEN THE PROGRAM IS INITIALLY STARTED; THESE STARTING ADDRESSES ARE TREATED AS ADDRESSES 200 OR 204 RESPECTIVELY ON RESTARTS.

#### 4.2 OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. BRING DRIVE(S) TO ONLINE STATE, WRITE ENABLED, AND LOCKED ON PORT.
3. LOAD ADDRESS 200.
4. SET SWITCHES (SEE SECTION 5.)
5. PRESS START.
6. THE PROGRAM WILL TYPEOUT THE STATUS OF THE DRIVES ATTACHED TO THE SELECTED MASSBUS SUBSYSTEM. TO INHIBIT THIS TYPEOUT, DO NOT RESTART THE PROGRAM FROM ANY OF THE STARTING ADDRESSES; INSTEAD TYPE A 'CONTROL C' ON THE KEYBOARD TO RETURN THE PROGRAM TO COMMAND ENTRY MODE.

#### 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

THE 'PERIOD PERIOD' TERMINATOR IS TYPED TO INDICATE THE END OF TEST PARAMETER MODIFICATION AND TO SIGNAL THE START OF TEST EXECUTION.

<,><CR> COMMA

THE COMMA IS USED AS A SEPARATOR BETWEEN DRIVE NUMBERS AND TEST NUMBERS.

</> SLASH

115 A MODIFICATION INDICATOR: IF A SLASH FOLLOWS A TEST  
 116 NUMBER, THE PROGRAM WILL OPEN THAT TEST FOR PARAMETER  
 117 MODIFICATION.  
 118  
 119 <^U> CONTROL-U  
 120  
 121 DELETE THE PRESENT INPUT STRING AND START A NEW  
 122 LINE. TYPED BY DEPRESSING THE "CONTROL KEY"  
 123 (CTRL) AND THEN STRIKING THE 'U'.  
 124  
 125 <\> RUBOUT  
 126  
 127 DELETE THE LAST CHARACTER FROM THE INPUT STRING.  
 128 TYPED BY STRIKING THE 'RUBOUT' KEY. WHICH WILL  
 129 BE ECHOED BY A BACKSLASH (\) FOLLOWED BY THE  
 130 CHARACTER DELETED.  
 131  
 132  
 133

#### 4.3.1 CONTROL SWITCH SELECTION

STARTING THE PROGRAM AT ANY OF THE POSSIBLE STARTING ADDRESSES WITH SW<07>=1 WILL RESULT IN ENTERING THE "CONTROL SWITCH SETTING" MODE. THUS, ALLOWING THE OPERATOR TO SPECIFY THE DESIRED STATE OF "C.SWR".

CONTROL SWITCH SELECTION EXAMPLES:

##### EXAMPLE #1

```
SET SW<07>=0
C.SWR=000000 / 400..
```

##### EXAMPLE #2

```
SET SW<07>=0
C.SWR=000000 / 220.
C.SWR=000000 / 220..
```

#### 4.3.2 RH CONTROLLER ADDRESS SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECT OF THE DEFAULT VALUES OF BUS ADDRESS (RMCS1), AND VECTOR ADDRESS OF THE RH CONTROLLER. IF THE DEFAULT VALE OF THE BUS ADDRESS DOES NOT RESPOND (TIMES OUT) WHEN ADDRESSED, AN ERROR IS REPORTED. 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 210 OR 214 ALLOWS THE OPERATOR TO CHANGE THE ADDRESS OF THE RH CONTROLLER AND THE VECTOR ADDRESS.

ADDRESS SELECTION EXAMPLES

115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171

172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228

## EXAMPLE #1

RMCS1=176700 / 177200.

## EXAMPLE #2

RMCS1=176700 / 176300<CR>  
RMVEC=254 / 260<CR>

## EXAMPLE #3

RMCS1=176700<CR>  
RMVEC=254 / 260.

## EXAMPLE #4

RH/RM FAILED TO RESPOND TO ADDRESSING  
RMCS1 ERR PC  
176300 XXXXXX  
RMCS1=176300 / 176700.

## EXAMPLE #5

RMCS1=176700 / 1776\67\6300<CR>  
RMVEC=254<CR>  
RMCS1=176300.

## 4.3.3 DRIVE AND PARAMETER SELECTION

STARTING THE PROGRAM AT 200 OR 210 WILL RESULT IN AUTOMATIC  
SELECTION OF THE DRIVES TO TEST AND THE TESTS TO RUN.

STARTING THE PROGRAM AT 204 OR 214 ALLOWS THE OPERATOR  
TO SELECT THE DRIVE(S) TO BE TESTED, THE TESTS TO BE EXECUTED,  
AND THE PARAMETERS TO USE.

## 4.3.3.1 DRIVE AND PARAMETER SELECTION DESCRIPTION

THE FOLLOWING IS A TABLE OF TERMS USED BY THE PSI.

'R''	REPEATS (ITERATIONS)
'FC''	FIRST CYLINDER ADDRESS
'LC''	LAST CYLINDER ADDRESS
'IC''	INCREMENT CYLINDER
'FT''	FIRST TRACK ADDRESS
'LT''	LAST TRACK ADDRESS
'IT''	INCREMENT TRACK
'FS''	FIRST SECTOR ADDRESS
'LS''	LAST SECTOR ADDRESS
'PAT''	PATTERN (USED FOR DATA TEST) (NOT USED)
'WDX''	WORD OF PATTERN 0 WHERE X IS 1 TO 16 (NOT USED)

\*'S'' ALL SEEK TESTS (TESTS 0 - 7)  
\*'T'' ALL TIMING TESTS (TESTS 11 - 14)

\* USED BY THE OPERATOR TO SELECT TEST GROUPS

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

## 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 MODIFIED, THE REMAINING TESTS WILL BE UNCHANGED.

WHEN THE PROGRAM IS STARTED FROM LOCATION 200 OR 210, TESTS 0-7 AND 11, 12 & 14 WILL BE RUN USING ALL AVAILABLE, ONLINE DRIVES. IF THE OPERATOR WISHES TO SELECT THE DRIVES TO BE TESTED, THE TESTS TO BE PERFORMED, OR THE PARAMETERS TO BE USED, THE CONVERSATION MODE MAY BE ENTERED BY TYPING A 'CONTROL C' OR BY STARTING THE PROGRAM FROM EITHER LOCATION 204 OR 214.

THE PROGRAM WILL THEN RESPOND WITH:

DRIVE(S)=

THE FOLLOWING EXAMPLES ASSUME THAT THE OPERATOR IS 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<CR>' WHICH SAYS 'THIS IS THE END OF DRIVE ENTRY'. THE PROGRAM WILL THEN REQUEST TEST NUMBERS.

THE TRANSACTION APPEARS AS FOLLOWS:

DRIVE(S)=3<CR>  
TEST=

THE OPERATOR MAY NOW ENTER DESIRED TEST NUMBERS. IN THE EXAMPLE, HE WANTS TESTS 2 THRU 7 AND TEST 11 SO HE TYPES 2-7<,> (THE 'COMMA' SEPARATES ENTRIES), 11<.><CR> ('PERIOD' 'CARRIAGE RETURN' - END OF CHANGES, START TEST EXECUTION.)

IT NOW LOOKS LIKE THIS

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

IN THE NEXT EXAMPLE, IT IS ASSUMED THAT THE OPERATOR WISHES TO TEST DRIVE 4 AND TO RUN TESTS 1 AND 3 THRU 11, MODIFYING THE PARAMETERS FOR TESTS 3 AND 10.

THE TRANSACTION WOULD BE AS FOLLOWS:

DRIVE(S)=4<CR>  
TEST=

THE OPERATOR NOW ENTERS THE TEST NUMBERS. THE TRANSACTION IS GIVEN BELOW:

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

NOTICE THIS SAYS SELECT TEST 1, CONTINUE<,>; SELECT TEST 3, OPEN</>;

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

SELECT TESTS 4-7, CONTINUE<, >; SELECT TEST 10, OPEN</>; SELECT TEST 11, END OF INPUT <.>.

THE PROGRAM SCANS THE TEST NUMBER INPUT AND DETERMINES THAT THE PARAMETERS FOR TEST 3 AND TEST 10 ARE TO BE CHANGED. THE OTHER TESTS WILL NOT BE ALTERED.

(THE ENTIRE TRANSACTION IS REPEATED FOR CLARITY)

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

THE NEW VALUE FOR 'R' MAY BE ENTERED. TERMINATING THE ENTRY WITH A <.> (PERIOD) WILL TERMINATE THE CHANGES FOR THIS TEST; TYPING A <CR> OR TERMINATING THE ENTRY WITH A <CR> WILL CAUSE THE PROGRAM TO MOVE TO 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
```

IF THE OPERATOR DOES NOT WISH TO CHANGE 'FC', THE FOLLOWING OCCURS:

```
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=560 /
```

THE PROGRAM RESPONDS WITH THE PREVIOUSLY ASSIGNED PARAMETER FOR LAST CYLINDER ADDRESS IN THIS CASE USING 560 AS THE EXAMPLE. THIS IS WHAT THE OPERATOR INTENDED TO MODIFY AND IS WHY TEST 3 WAS OPENED. TO CHANGE THE VALUE TO '20', THE NEW VALUE IS TYPED FOLLOWED BY A 'PERIOD' TERMINATOR (<.><CR>).

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= 560 / 20.<CR>
TEST 10
R=1 /
```

THE PROGRAM HAS LOADED TEST 3 WITH ITS NEW PARAMETERS AND THE PROGRAM IS WAITING FOR CHANGES TO TEST 10'S PARAMETERS.

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



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

```
R=1 / <CR>
FC=0 / <CR>
LC= 560 / 20.<CR>
TEST 10
R=1 / 10.<CR>
```

THE OPERATOR TYPES THE NEW VALUE (10) AND TERMINATES THE ENTRY WITH A 'PERIOD' 'CARRIAGE RETURN'.

THE PROGRAM NOW LOADS TEST 10 WITH THE NEW PARAMETERS (TEST 11 RETAINS THE PREVIOUSLY ASSIGNED PARAMETERS) AND RESPONDS WITH:

DRIVE(S)=

SINCE THE USER DID NOT END THE CONVERSATION MODE WITH A 'PERIOD PERIOD', 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 PROGRAM, A 'PERIOD PERIOD' MUST BE TYPED.

IF A SINGLE 'PERIOD' IS TYPED WHILE DRIVE OR TEST NUMBERS ARE BEING ENTERED, THE PROGRAM WILL START EXECUTION IMMEDIATELY. A 'PERIOD PERIOD' MUST BE TYPED BEFORE THE PROGRAM WILL EXIT TEST PARAMETER CHANGE MODE TO GO TO EXECUTION.

#### 4.3.3.2 DRIVE 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>          ;SELECT DRIVE #0 AND MAKE CHANGES ""
TEST=1-5.<CR>        ;RUN TEST 1 THRU 5 ONLY, USE DEFAULT
                      ;PARAMETERS AND TERMINATE AND EXECUTE."
```

##### EXAMPLE #3

```
DRIVE=2<CR>          ;SELECT DRIVE #2 AND MAKE CHANGES ""
TEST=1-5,6/7/10/<CR> ;RUN TEST 1-5 WITH DEFAULT PARAMETERS, OPEN
TEST 6               ;TEST 6,7 AND 10 FOR CHANGES
R=1 / <CR>           ;LEAVE 'R' AS IS AND MOVE TO NEXT PARAMETER
FC=0 / 10.<CR>       ;SET 'FC' CYLINDER ADDRESS TO 10, END CHANGES
                      ;TO TEST 6.
```

##### TEST 7

```
R=1 / 50<CR>        ;50 ITERATIONS, MOVE TO NEXT PARAMETER
FC=0 / <CR>         ;DO NOT CHANGE 'FC' CYLINDER ADDRESS BUT CONTINUE
LC=560 / 50.<CR>   ;TEST 10 IS STILL PENDING AND WILL BE
```

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  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456

;RETAIN ITS PRESENT PARAMETERS.

EXAMPLE #4  
-----

DRIVE=0<CR>  
TEST=S.<CR>

;SELECT DRIVE #0 AND MAKE CHANGES  
;RUN ALL SEEK TESTS

EXAMPLE #5  
-----

DRIVE=1<CR>  
TEST=S/T<CR>  
TEST 0  
R=10 / <CR>  
FC=0 / 10..<CR>

;RUN ALL SEEK TESTS (OPEN FOR CHANGES) AND  
;THE TIMING TESTS (WITH DEFAULT PARAMETERS).  
;RUN WITH 10 ITERATIONS  
;CHANGE FIRST CYLINDER ADDRESS  
;AND START EXECUTION  
;TESTS 1 - 10 WILL RETAIN THEIR PREVIOUSLY  
;ASSIGNED PARAMETERS.

EXAMPLE #6  
-----

DRIVE=1<CR>  
TEST=S/<CR>  
TEST 0  
R=10 / 100.<CR>  
TEST 1  
R=100 / 1000.<CR>  
TEST 2  
R=1 / 10<CR>  
FC=0 / 50<CR>  
LC=560 / 51.<CR>  
TEST 3  
R=1.<CR>  
TEST 4  
R=1..<CR>

;OPEN THE SEEK TESTS (TESTS 0-7)  
;CHANGE TO 100 ITERATIONS, TO TO THE NEXT TEST  
;CHANGE 'R' TO 1000 ITERATIONS, MOVE TO NEXT TEST  
;CHANGE 'R' TO 10 ITERATIONS, GO TO NEXT PARAMETER  
;CHANGE 'FC' TO 50, GO TO NEXT PARAMETER  
;CHANGE 'LC' TO 51, GO TO THE NEXT TEST  
;MOVE TO NEXT TEST  
;USE TEST 4'S PARAMETERS AND START PROGRAM EXECUTION

EXAMPLE #7  
-----

DRIVE=0,1,4<CR>  
TEST=0-5/<CR>  
TEST 0  
R=10 / <CR>  
FC=0 / <CR>  
LC=560 / 1..<CR>

;TEST DRIVES 0,1, AND 4 IN SEQUENCE  
;CHANGE TEST 5  
;CHANGE LAST CYLINDER FROM 560 TO 1  
;START PROGRAM EXECUTION.

5. SWITCH SETTINGS  
-----

5.1 OPERATIONAL SWITCH SETTINGS

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  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN DRIVE INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTINGS ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED, IF THE PROGRAM FINDS ALL 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

THE SWITCH SETTINGS ARE:

SW<15>=1	HALT ON ERROR
SW<14>=1	LOOP ON TEST
SW<13>=1	INHIBIT ERROR TYPEOUTS
SW<12>=1	TYPE TEST NUMBER
SW<11>=1	INHIBIT ITERATIONS
SW<10>=1	RING BELL ON ERROR
SW<09>=1	LOOP ON ERROR
SW<08>=1	PRINT ERROR MESSAGE ON LINE PRINTER
SW<07>=1	READ 'C.SWR' SETTINGS FROM TTY
SW<06>=1	INHIBIT TIME REPORTS (TESTS 11-15)

## 5.2 CONTROL SWITCH SETTINGS

THE CONTROL SWITCH SETTINGS ARE ENTERED THROUGH THE KEYBOARD.

TO ENTER THE CONTROL SWITCH SETTING MODE PLACE SW<07>=1 BEFORE PRESSING START. THEN UPON STARTING THE PROGRAM IT WILL TYPE THE PRESENT CONTENTS OF THE CONTROL SWITCH REGISTER (C.SWR) AND WAIT FOR THE NEW SETTING TO BE INPUT. THE INPUT STRING MUST CONSIST OF 1 TO 6 OCTAL DIGITS, TWO PERIODS (..), AND A CARRIAGE RETURN.

THE C.SWR SETTINGS ARE:

C.SWR<14>=0	NO STALL BETWEEN DRIVE FUNCTIONS
=1	STALL AFTER EVERY DRIVE FUNCTION
C.SWR<13>=0	USE SPECIFIC STALL TIMES
=1	USE RANDOM STALL TIMES
C.SWR<12>=0	NO INCREMENTING STALLS IN TEST 4
=1	PERFORM INCREMENTING STALLS IN TEST 4

514 C.SWR<08>=0 DO IMPLIED SEEKS WITH DATA TRANSFERS  
 515 =1 DO EXPLICIT SEEKS BEFORE DATA TRANSFERS  
 516 C.SWR<07>=0 DO READ HEADER AND DATA COMMANDS IN TESTS 0-7  
 517 =1 DO EXPLICIT SEEK COMMANDS IN TESTS 0-7  
 518 C.SWR<06>=0 60 HZ POWER SOURCE  
 519 =1 50 HZ POWER SOURCE  
 520 C.SWR<05>=0 ALLOW SOFTWARE TIMEOUTS(ENABLE WATCHDOG TIMER)  
 521 =1 INHIBIT SOFTWARE TIMEOUTS(DISABLE WATCHDOG TIMER)  
 522 C.SWR<00>=0 OPERATE IN 32. SECTOR (16 BIT) MODE  
 523 =1 OPERATE IN 30. SECTOR (18 BIT) MODE

524 THE DEFAULT CONDITION OF C.SWR<15:00>=0.

525 REFER TO 4.3.1 FOR C.SWR SELECTION

## 526 6. ERRORS

527 -----  
 528 THERE ARE ANUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM.  
 529 WHEN AN ERROR IS ENCOUNTERED, THE CALL TO THE ERROR ROUTINE  
 530 IS MADE AND IF SW<13> IS NOT SET, AN ERROR MESSAGE PERTAINING  
 531 TO THE ERROR WILL BE TYPED. EACH ERROR TYPEOUT WILL CONTAIN  
 532 THE FOLLOWING:

- 533 1. AN ERROR MESSAGE
- 534 2. A DATA HEADER
- 535 3. A DATA STRING

536 REFER TO THE FOLLOWING SECTION FOR THE DIFFERENT ERRORS  
 537 THAT CAN OCCUR.

### 538 6.1 ERROR TYPES

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

#### 541 6.1.1 DRIVER ERROR

542 THESE ERRORS WILL BE DETECTED BY THE RH/RM DRIVER.  
 543 THERE ARE TWO CLASSES OF DRIVER ERRORS; THOSE THAT  
 544 CAN NOT BE IDENTIFIED IN A MANNER THAT ALLOWS THE  
 545 INFORMATION TO BE RETURNED TO A 'DATA PARAMETER BLOCK'  
 546 (DPB) AND THOSE THAT CAN. THE FIRST CLASS WILL BE  
 547 REPORTED BY ERROR CALLS (EMT'S) 1-5 WITHIN THE DRIVER.  
 548 THE SECOND CLASS WILL PASS THE ERROR CODES TO THE  
 549 STATUS/ERROR WORD (DPB+16) OF THE PROPER DPB.

#### 550 6.1.2 NON-FATAL ERRORS

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

#### 554 6.1.3 FATAL ERRORS

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  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627

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 PRETEST ERROR

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

### 6.2.2 NON-FATAL ERROR

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

### 6.2.3 FATAL ERROR

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

## 7. RESTRICTIONS

THE PROGRAM WILL TEST THE DRIVES IN EITHER 16 BIT MODE OR IN 18 BIT MODE DEPENDING ON THE SETTING OF 'S.SWR<00>'. IF 'C.SWR<00>' IS 0, ALL OF THE DRIVES WILL BE TESTED IN 16 BIT MODE; IF 'C.SWR<00>' IS 1, ALL OF THE DRIVES WILL BE TESTED IN 18 BIT MODE. THE PROGRAM HAS NO PROVISIONS FOR TESTING DRIVES WITH INTERMIXED PACKS OR TESTING BOTH 16 BIT MODE AND 18 BIT MODE DRIVES ON THE SAME SYSTEM. ACT11 AUTOMATIC MODE ASSUMES 16 BIT MODE.

BEFORE THE PROGRAM IS STARTED, PROPERLY FORMATTED PACKS MUST BE MOUNTED ON THE DRIVES WHICH WILL BE TESTED. THE PROGRAM ASSUMES A PROPERLY FORMATTED PACK. THE FORMAT OF THE PACK IS NOT ALTERED BY THE PROGRAM.

## 8. MISCELLANEOUS

### 8.1 EXECUTION TIME

THE PROGRAM REQUIRES APPROXIMATELY 10 MINUTES TO MAKE ONE PASS WITH RM80 DRIVES. THIS ASSUMES THE DEFAULT TEST SEQUENCE (TESTS 0-7 AND 11, 12 & 14) AND DEFAULT TEST PARAMETERS.

### 8.2 STACK POINTER

628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684

THE STACK POINTER IS INITIALLY SET TO 1100.

### 8.3 TIMING TESTS (TESTS 11-15) PRINTOUTS

AT THE COMPLETION OF EACH OF THE TIMING TESTS THE TIME OF THE MINIMUM SEEK, MAXIMUM SEEK, AND THE AVERAGE OF ALL OF TH<sup>E</sup> SEEKS PERFORMED ARE TYPED ON THE TTY. THE NUMBER OF SEEKS THAT HAD TIMES BELOW THE MINIMUM TIME ALLOWED WILL BE TYPED ON THE SAME LINE AS THE MINIMUM TIME. THE NUMBER ABOVE THE MAXIMUM WILL BE TYPED ON THE SAME LINE AS THE MAXIMUM TIME, AND THE TOTAL NUMBER OF SEEKS PERFORMED WILL BE ON THE SAME LINE AS THE AVERAGE.

#### 8.3.1 TIMING TOLERANCES

##### 1. TEST 11 -- ROTATIONAL SPEED TIMES

60 HZ  
MINIMUM=15970 US  
MAXIMUM=17270 US  
NOMINAL=16670 US

50 HZ  
MINIMUM=15970 US  
MAXIMUM=17270 US  
NOMINAL=16670 US

##### 2. TEST 12 -- ONE CYLINDER SEEK TIMES

MAXIMUM=6000 US

##### 3. TEST 13 -- AVERAGE SEEK TIME TEST (NOT DEFAULT)

MAXIMUM=25000 US

\*\* THERE ARE NO SPECIFICATIONS GIVEN FOR AN AVERAGE SEEK TIME \*\*  
\*\* ON THIS PARTICULAR DRIVE. THEREFORE, THIS TEST SHOULD BE \*\*  
\*\* USED FOR REFERENCE ONLY. \*\*

##### 4. TEST 14 -- MAXIMUM SEEK TIMES

MAXIMUM (FORWARD)=50000 US

\*\* THERE IS NO SPECIFICATION GIVEN FOR THE MAXIMUM REVERSE \*\*  
\*\* SEEK TIME ON THIS PARTICULAR DRIVE. THEREFORE, ANY REVERSE \*\*  
\*\* SEEK TIMES ABOVE THE MAXIMUM TIME OF 50.0 MS WILL NOT BE \*\*  
\*\* TYPED IN THE TIMING REPORT. HOWEVER, THE TIMING REPORT \*\*  
\*\* WILL STILL TYPE THE MINIMUM, MAXIMUM AND AVERAGE TIMES \*\*  
\*\* FOR THE REVERSE SEEKS. (SEE SECTION 8.3.2, EX. 2) \*\*

##### 5. TEST 15 -- AVERAGE SEEK TIME MEASUREMENT (NOT DEFAULT)

MAXIMUM=25000 US

\*\* WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN \*\*

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

## 8.3.2. TIMING TESTS PRINTOUT EXAMPLES

## EXAMPLE #1

-----  
ROTATIONAL SPEED TIMES  
MIN=16670 US  
MAX=16690 US  
AVG=16680 US 10 SEARCHES TIMED

ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80  
MIN=15970 US  
MAX=17270 US

ONE CYLINDER SEEK TIMES  
\* FORWARD  
MIN=5350 US  
MAX=5920 US  
AVG=5550 US 559 SEEKS TIMED  
\* REVERSE  
MIN=5040 US  
MAX=5970 US  
AVG=5330 US 560 SEEKS TIMED

ALLOWABLE ONE CYLINDER SEEK LIMIT  
MAX=6000 US

AVERAGE SEEK TIMES  
\* FORWARD  
MIN=23770 US  
MAX=24640 US  
AVG=24230 US 128 SEEKS TIMED  
\* REVERSE  
MIN=23990 US  
MAX=24550 US  
AVG=24220 US 128 SEEKS TIMED

ALLOWABLE AVERAGE TIME LIMIT  
MAX=25000 US

MAXIMUM SEEK TIMES  
\* FORWARD  
MIN=47990 US  
MAX=49980 US  
AVG=48010 US 128 SEEKS TIMED  
\* REVERSE  
MIN=47120 US  
MAX=49650 US  
AVG=48340 US 128 SEEKS TIMED

ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT  
MAX=50000 US

EXAMPLE #2

742  
743  
744  
745  
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

-----  
ROTATIONAL SPEED TIMES

MIN=16670 US  
MAX=16690 US  
AVG=16680 US 10 SEARCHES TIMED

## ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80

MIN=15970 US  
MAX=17270 US

## ONE CYLINDER SEEK TIMES

\* FORWARD  
MIN=5470 US  
MAX=7940 US 3 ABOVE THE MAXIMUM OF 6000 US  
AVG=5830 US 559 SEEKS TIMED  
\* REVERSE  
MIN=5040 US  
MAX=5970 US  
AVG=5330 US 560 SEEKS TIMED

## ALLOWABLE ONE CYLINDER SEEK LIMIT

MAX=6000 US

## AVERAGE SEEK TIMES

\* FORWARD  
MIN=24730 US  
MAX=32620 US 73 ABOVE THE MAXIMUM OF 25000 US  
AVG=30320 US 128 SEEKS TIMED  
\* REVERSE  
MIN=25620  
MAX=32230 US 128 ABOVE THE MAXIMUM OF 25000 US  
AVG=32800 US 128 SEEKS TIMED

## ALLOWABLE AVERAGE TIME LIMIT

MAX=25000 US

## MAXIMUM SEEK TIMES

\* FORWARD  
MIN=50510 US  
MAX=57240 US 128 ABOVE THE MAXIMUM OF 50000 US  
AVG=56020 US 128 SEEKS TIMED  
\* REVERSE  
MIN=50050 US  
MAX=57550 US  
AVG=56210 US 128 SEEKS TIMED

## ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT

MAX=50000 US

## 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. ALSO, THE  
END OF TEST COULD OCCUR ON A DRIVE, IF THE MAXIMUM ERROR LIMIT



IN LOCATION 'ERMAX' IS EXCEEDED.

9. PROGRAM DESCRIPTION  
-----

THIS PROGRAM CONTAINS 16 TESTS, NUMBERED 0-15 IN OCTAL. TESTS 0-7 WILL READ THE CYLINDER, TRACK, AND SECTOR INFORMATION FROM THE HEADER, USING A 'READ HEADER AND DATA' COMMAND, AND THEN CHECK THE INFORMATION FOR VALIDITY, THUS, INSURING THE SEEK OPERATION FUNCTIONS PROPERLY. TESTS 11-14 WILL MEASURE THE ROTATIONAL SPEED, THE ONE CYLINDER SEEK, THE AVERAGE SEEK TIME, AND THE MAXIMUM SEEK TIMES TO ENSURE THEY ARE ALL WITHIN THE TOLERANCES ALLOWED. TEST 15 MEASURES THE AVERAGE SEEK TIME FOR UNIQUE NUMBER OF SEEKS, BUT DOES NOT HAVE ANY PASS/FAIL LIMITS.

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 AND 11, 12 & 14) WILL BE PERFORMED ON EACH DRIVE BEFORE MOVING TO THE NEXT DRIVE IN SEQUENCE. 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 SECTIONS FOR DETAILED DESCRIPTIONS OF EACH TEST.

9.1 TEST 0 - RECAL/RANDOM SEEK TEST

THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE COMMAND AND THEN SEEK TO A RANDOM CYLINDER BETWEEN 'FC' AND 'LC'. AT THE COMPLETION OF BOTH COMMANDS, STATUS INDICATORS ARE CHECKED TO ENSURE THAT NO ERRORS OCCURRED.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	200
FC	-	0
LC	-	560
FT	-	0
FS	-	0

9.2 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 ENSURE PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	100
FC	-	0

799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855

856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912

LC	-	256
IC	-	0
FT	-	0
LT	-	0
FS	-	0
LS	-	0

### 9.3 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 ENSURE PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	1
FC	-	0
LC	-	560
IC	-	1
FT	-	0
FS	-	0

### 9.4 TEST 3 - STEPPING SEEK TEST

THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0, 1, 2, 4, 8, 16, 32, 64, 128, 256 AND 512. AT THE COMPLETION OF EACH SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO VERIFY PROPER OPERATION.

THE PARAMETERS USED BY THE TEST ARE GIVEN BELOW:

R	-	8
FC	-	0
LC	-	512
IC	-	1
FT	-	0
FS	-	0

### 9.5 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 ENSURE PROPER OPERATION.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
LC	-	560
IC	-	1

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  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969

FT - 0  
FS - 0

#### 9.6 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 ENSURE PROPER OPERATION. 'NC1' AND 'NC2' DEFAULT TO 'FC' AND 'LC' RESPECTIVELY.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R - 1  
FC - 0  
LC - 560  
IC - 1  
FT - 0  
FS - 0

#### 9.7 TEST 6 - SERVO ADDRESSING LOGIC NOISE GENERATOR TEST

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 ENSURE PROPER OPERATION.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R - 1  
FC - 0  
LC - 560  
IC - 1  
FT - 0  
FS - 0

#### 9.8 TEST 7 - RANDOM SEEK TEST (NOT DEFAULT)

THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC' 'LC'. AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK. THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION OF POSITIONING OCCURS USING EACH HEAD. TRACK ADDRESSES ARE INCREMENTED BETWEEN PARAMETERS 'FT' AND 'LT'.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R - 5000  
FC - 0  
LC - 560  
FT - 0  
LT - 4

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  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026

## 9.9 TEST 10 - ALL SEEKS TEST

THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER TO ALL OTHER CYLINDERS.

BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER BETWEEN 'FC' AND 'LC' FROM CYLINDER 'FC'. THE BEGINNING CYLINDER ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'. THE SEQUENCE CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.

THE FOLLOWING PARAMETERS ARE USED BY THIS TEST:

R	-	1
FC	-	0
LC	-	560
IC	-	1
FT	-	0
FS	-	0

## 9.10 TEST 11 - ROTATIONAL SPEED TIMING TEST

THIS TEST WILL START A SEARCH TO CYLINDER 'FC', TRACK 'FT', SECTOR 'FS'. AS SOON AS THE INTERRUPT OCCURS, THE GO BIT IS SET AGAIN AND THE OPERATION IS TIMED. THIS PROCEDURE IS REPEATED 10 TIMES THEN THE AVERAGE TIME IS CALCULATED AND CHECKED TO ENSURE IT IS WITHIN TOLERANCE:

LO LIMIT (16.67MS - 4.3%) = 15.97MS IF 50/60 HZ.  
HI LIMIT (16.67MS + 3.5%) = 17.27MS IF 50/60 HZ.

THE FOLLOWING PARAMETERS ARE USED BY THE TEST:

R	-	1
FC	-	0
FT	-	0
FS	-	0

## 9.11 TEST 12 - ONE CYLINDER SEEK TIMING TEST

THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK. MAXIMUM TIME IS 6.0 MS

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	560

## 9.12 TEST 13 - AVERAGE SEEK TIMING TEST (NOT DEFAULT)

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO

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  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082

CYLINDER 'FC'. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE AVERAGE SEEK TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS). MAXIMUM TIME IS 25.0 MS

\*\* THERE ARE NO SPECIFICATIONS GIVEN FOR AN AVERAGE SEEK TIME \*\*  
\*\* ON THIS PARTICULAR DRIVE. THEREFORE, THIS TEST SHOULD BE \*\*  
\*\* USED FOR REFERENCE ONLY. \*\*

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	155

### 9.13 TEST 14 - MAXIMUM SEEK TIMING TEST

THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO CYLINDER 'FC'. BOTH SEEKS ARE TIMED, BUT ONLY THE FORWARD SEEKS ARE CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE MAXIMUM SEEK TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256. SEEKS). MAXIMUM (FORWARD) TIME IS 50.0 MS

\*\* THERE IS NO SPECIFICATION GIVEN FOR THE MAXIMUM REVERSE \*\*  
\*\* SEEK TIME ON THIS PARTICULAR DRIVE. THEREFORE, ANY REVERSE \*\*  
\*\* SEEK TIMES ABOVE THE MAXIMUM TIME OF 50.0 MS WILL NOT BE \*\*  
\*\* TYPED IN THE TIMING REPORT. HOWEVER, THE TIMING REPORT \*\*  
\*\* WILL STILL TYPE THE MINIMUM, MAXIMUM AND AVERAGE TIMES \*\*  
\*\* FOR THE REVERSE SEEKS. \*\*

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	560

### 9.14 TEST 15 - AVERAGE SEEK TIME MEASUREMENT TEST (NOT DEFAULT)

THIS TEST MEASURES AVERAGE SEEK TIME BY DOING ALL UNIQUE SEEKS BETWEEN 'FC' AND 'LC' AND MEASURING THE SEEK TIMES. THE TOTAL SEEK TIME FOR ALL SEEKS IS THEN DIVIDED BY THE NUMBER OF SEEKS PERFORMED TO PROVIDE AN AVERAGE SEEK TIME VALUE FOR THE RANGE OF SEEKS TIMED. MAXIMUM TIME IS 25.0 MS

\*\* WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN \*\*

THE TEST USES THE FOLLOWING PARAMETERS:

R	-	1
FC	-	0
LC	-	560

## 10. PROGRAM LISTING

a

1  
39  
40  
  
41  
  
42  
43  
52  
53  
54  
55  
56  
61  
62  
63  
64  
65  
66  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80

;\*LAST REVISION 24-SEP-81

.TITLE CZRNGAO RM80 FCTNL PT4

\*COPYRIGHT (C) 1982  
\*DIGITAL EQUIPMENT CORPORATION  
\*COLORADO SPGS., CO. 80919

\*PROGRAM BY MIKE LEAVITT

\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
\*PACKAGE (MAINDEC-11-D20AC-C5), 18-MAR-81

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	TYPE TEST NUMBER
10	BELL ON ERROR
9	LOOP ON ERROR
8	PRINT ERROR MESSAGE ON LINE PRINTER
7	READ 'C.SWR' SETTINGS FROM TTY
6	INHIBIT TIME REPORTS (TESTS 11-14)

.SBTTL CONTROL SWITCH SETTINGS

SWITCH	STATE	USE
14	0	NO STALL BETWEEN DRIVE FUNCTIONS
	1	STALL AFTER EVERY DRIVE FUNCTION
13	0	USE SPECIFIC STALL TIME
	1	USE RANDOM STALL
12	0	NO INCREMENTING STALL IN TEST 4
	1	DO INCREMENTING STALL IN TEST 4
7	0	DO 'READ HEADER AND DATA' IN TESTS 0-10
	1	DO EXPLICIT SEEKS IN TESTS 0-10
6	0	60 HZ
	1	50 HZ
5	0	RUN WATCHDOG TIMER
	1	INHIBIT WATCHDOG TIMER
0	0	TEST DRIVE(S) IN 31. SECTOR (16 BIT) MODE
	1	TEST DRIVE(S) IN 30. SECTOR (18 BIT) MODE

.SBTTL BASIC DEFINITIONS

	001100	STACK = 1100	*** INITIAL ADDRESS OF THE STACK POINTER ***
	104000	ERROR = EMT	::BASIC DEFINITION OF ERROR CALL
	000004	SCOPE = IOT	::BASIC DEFINITION OF SCOPE CALL

\*MISCELLANEOUS DEFINITIONS

	000011	HT = 11	::CODE FOR HORIZONTAL TAB
	000012	LF = 12	::CODE FOR LINE FEED
	000015	CR = 15	::CODE FOR CARRIAGE RETURN
	000200	CRLF = 200	::CODE FOR CARRIAGE RETURN-LINE FEED

```

177776 PS = 177776 ;;PROCESSOR STATUS WORD
177776 PSW=PS
177774 STKLMT = 177774 ;;STACK LIMIT REGISTER
177772 PIRQ = 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR = 177570 ;;HARDWARE SWITCH REGISTER
177570 DDISP = 177570 ;;HARDWARE DISPLAY REGISTER
    
```

;\*GENERAL PURPOSE REGISTER DEFINITIONS

```

000000 R0 = X0 ;;GENERAL REGISTER
000001 R1 = X1 ;;GENERAL REGISTER
000002 R2 = X2 ;;GENERAL REGISTER
000003 R3 = X3 ;;GENERAL REGISTER
000004 R4 = X4 ;;GENERAL REGISTER
000005 R5 = X5 ;;GENERAL REGISTER
000006 R6 = X6 ;;GENERAL REGISTER
000007 R7 = X7 ;;GENERAL REGISTER
000006 SP = X6 ;;STACK POINTER
000007 PC = X7 ;;PROGRAM COUNTER
    
```

;\*PRIORITY LEVEL DEFINITIONS

```

000000 PR0 = 0 ;;PRIORITY LEVEL 0
000040 PR1 = 40 ;;PRIORITY LEVEL 1
000100 PR2 = 100 ;;PRIORITY LEVEL 2
000140 PR3 = 140 ;;PRIORITY LEVEL 3
000200 PR4 = 200 ;;PRIORITY LEVEL 4
000240 PR5 = 240 ;;PRIORITY LEVEL 5
000300 PR6 = 300 ;;PRIORITY LEVEL 6
000340 PR7 = 340 ;;PRIORITY LEVEL 7
    
```

;'SWITCH REGISTER' SWITCH DEFINITIONS

```

100000 SW15 = 100000
040000 SW14 = 40000
020000 SW13 = 20000
010000 SW12 = 10000
004000 SW11 = 4000
002000 SW10 = 2000
001000 SW09 = 1000
000400 SW08 = 400
000200 SW07 = 200
000100 SW06 = 100
000040 SW05 = 40
000020 SW04 = 20
000010 SW03 = 10
000004 SW02 = 4
000002 SW01 = 2
000001 SW00 = 1
001000 SW9=SW09
000400 SW8=SW08
000200 SW7=SW07
000100 SW6=SW06
000040 SW5=SW05
000020 SW4=SW04
000010 SW3=SW03
000004 SW2=SW02
000002 SW1=SW01
000001 SW0=SW00
    
```

FASIC DEFINITIONS

```

100000      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
040000      BIT15 = 100000
020000      BIT14 = 40000
010000      BIT13 = 20000
004000      BIT12 = 10000
002000      BIT11 = 4000
001000      BIT10 = 2000
000400      BIT09 = 1000
000200      BIT08 = 400
000100      BIT07 = 200
000040      BIT06 = 100
000020      BIT05 = 40
000010      BIT04 = 20
000004      BIT03 = 10
000002      BIT02 = 4
000001      BIT01 = 2
001000      BIT00 = 1
000400      BIT9=BIT09
000200      BIT8=BIT08
000100      BIT7=BIT07
000040      BIT6=BIT06
000020      BIT5=BIT05
000010      BIT4=BIT04
000004      BIT3=BIT03
000002      BIT2=BIT02
000001      BIT1=BIT01
000001      BIT0=BIT00
    
```

```

000004      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000010      ERRVEC = 4          ;; TIME OUT AND OTHER ERRORS
000014      RESVEC = 10        ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014      TBITVEC = 14       ;; 'T' BIT
000014      TRTVEC = 14        ;; TRACE TRAP
000014      BPTVEC = 14        ;; BREAKPOINT TRAP (BPT)
000020      IOTVEC = 20        ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024      PWRVEC = 24        ;; POWER FAIL
000030      EMTVEC = 30        ;; EMULATOR TRAP (EMT) **ERROR**
000034      TRAPVEC = 34       ;; 'TRAP' TRAP
000060      TKVEC = 60         ;; TTY KEYBOARD VECTOR
000064      TPVEC = 64         ;; TTY PRINTER VECTOR
000240      PIRQVEC = 240      ;; PROGRAM INTERRUPT REQUEST VECTOR
    
```

;CONTROL AND STATUS REGISTER 1 (RMCS1)

```

000100      IE = 100           ; INTERRUPT ENABLE (BIT #6)
000200      RDY = 200          ; READY (BIT #7)
000400      A16 = 400          ; HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000      A17 = 1000         ; HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000      PSEL = 2000        ; PORT SELECT (BIT #10)
020000      MCPE = 20000       ; MASSBUSS PARITY ERROR (BIT #13)
040000      TRE = 40000        ; TRANSFER ERROR (BIT #14)
;SC = 100000                  ; SPECIAL CONDITION (BIT #15)
    
```

;WORD COUNT REGISTER (RMWC)  
;(EACH BIT IS CALLED BY BIT NUMBER)

;BUS ADDRESS REGISTER (RMBA)

81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96



```

97          ;(EACH BIT IS CALLED BY BIT NUMBER)
98
99          ;CONTROL AND STATUS REGISTER 2 (RMCS2)
100
101          000001      US1      = 1          ;UNIT SELECT (BIT #0)
102          000002      US2      = 2          ;UNIT SELECT (BIT #1)
103          000004      US4      = 4          ;UNIT SELECT (BIT #2)
104          000010      BAI      = 10         ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
108          000040      CLR      = 40         ;CLEAR (BIT #5)
109          000100      IR       = 100        ;INPUT READY (BIT #6)
110          000200      OR       = 200        ;OUTPUT READY (BIT #7)
111          000400      MPE      = 400        ;MASS BUS PARITY ERROR (BIT #8)
112          001000      MXF      = 1000       ;MISSED TRANSFER ERROR (BIT #9)
113          002000      PGE      = 2000       ;PROGRAM ERROR (BIT #10)
114          004000      NEM      = 4000       ;NON EXISTENT MEMORY (BIT #11)
115          010000      NED      = 10000      ;NON EXISTENT DRIVE (BIT #12)
116          020000      UPE      = 20000     ;UNIBUS PARITY ERROR (BIT #13)
117          040000      WCE      = 40000     ;WRITE CHECK ERROR (BIT #14)
118          100000      DLT      = 100000    ;DATA LATE (BIT #15)
119
120          ;DATA BUFFER REGISTER (RMDB)
121          ;(EACH BIT IS CALLED BY BIT NUMBER)
122
123          .SBTTL  RM80 REGISTERS
124
125          ;CONTROL AND STATUS 1 REGISTER. (#00)
126
127          000001      GO       = 1          ;GO BIT (BIT #0)
128          000002      F1       = 2          ;FUNCTION CODE BIT #1
129          000004      F2       = 4          ;FUNCTION CODE BIT #2
130          000010      F3       = 10         ;FUNCTION CODE BIT #3
131          000020      F4       = 20         ;FUNCTION CODE BIT #4
132          000040      F5       = 40         ;FUNCTION CODE BIT #5
133          004000      DVA      = 4000       ;DEVICE AVAILABLE (BIT #11)
134
135          ;DRIVE STATUS REGISTER (RMD5) (#01)
136
137          000001      OM       = 1          ;OFFSET MODE
138          000100      VV       = 100        ;VOLUME VALID (BIT #6)
139          000200      DRY      = 200        ;DRIVE READY (BIT #7)
140          000400      DPR      = 400        ;DRIVE PRESENT (BIT #8)
141          001000      PGM      = 1000       ;PROGRAMABLE (BIT #9)
142          002000      LST      = 2000       ;LAST SECTOR TRANSFERRED (BIT #10)
143          004000      WRL      = 4000       ;WRITE LOCK (BIT #11)
144          010000      MOL      = 10000      ;MEDIUM ON-LINE (BIT #12)
145          020000      PIP      = 20000     ;POSITIONING OPERATION IN PROGRESS (BIT #13)
146          040000      ERR      = 40000     ;COMPOSITE ERROR (BIT #14)
147          100000      ATA      = 100000    ;ATTENTION ACTIVE (BIT #15)
148
149          ;ERROR REGISTER #01 (RMR1) (#02)
150
151          000001      ILF      = 1          ;ILLEGAL FUNCTION (BIT #0)
152          000002      ILR      = 2          ;ILLEGAL REGISTER (BIT #1)
153          000004      RMR      = 4          ;REGISTER MODIFICATION REFUSED (BIT #2)
154          000010      PAR      = 10         ;PARITY ERROR (BIT #3)
155          000020      FER      = 20         ;FORMAT ERROR (BIT #4)
156          000040      WCF      = 40         ;WRITE CLOCK FAIL (BIT #5)
    
```

157	000100	ECH	= 100	:ECC HARD ERROR (BIT #6)
158	000200	HCE	= 200	:HEADER COMPARE ERROR (BIT #7)
159	000400	HCRC	= 400	:HEADER CRC ERROR (BIT #8)
160	001000	AOE	= 1000	:ADDRESS OVERFLOW ERROR (BIT #9)
161	002000	IAE	= 2000	:INVALID ADDRESS ERROR (BIT #10)
162	004000	WLE	= 4000	:WRITE LOCK ERROR (BIT #11)
163	010000	DTE	= 10000	:DRIVE TIMING ERROR (BIT #12)
164	020000	OPI	= 20000	:OPERATION INCOMPLETE (BIT #13)
165	040000	UNS	= 40000	:DRIVE UNSAFE (BIT #14)
166	100000	DCK	= 100000	:DATA CHECK ERROR (BIT 15)

:MAINTAINABILITY REGISTER #01 (RMMR1)(#03) - READ ONLY BITS

167				
168				
169				
170	000001	DMD	= 1	:DIAGNOSTIC MODE
171	000002	LSIT	= 2	
175	000010	WD	= 10	
176	000020	EECC	= 20	
177	000040	WC	= 40	
178	000100	CONT	= 100	
179	000200	PHA	= 200	
180	000400	PDA	= 400	
181	001000	ECRC	= 1000	
182	002000	PLFS	= 2000	
183	004000	ESRC	= 4000	
184	010000	REX	= 10000	
185	020000	EBL	= 20000	
189	100000	OCC	= 100000	

:MAINTAINABILITY REGISTER #01 (RMMR1) (#03) - WRITE ONLY BITS

190				
191				
192				
193	000001	DMD	= 1	:DIAGNOSTIC MODE BIT
194	000002	MSC	= 2	
195	000004	MJ	= 4	
196	000010	MWP	= 10	
197	000020	DTG	= 20	
198	000040	MS	= 40	
199	000100	MDF	= 100	
200	000200	MSER	= 200	
201	000400	MOC	= 400	
202	001000	MUR	= 1000	
203	002000	MRD	= 2000	
204	004000	MCLK	= 4000	
205	010000	MSEN	= 10000	
206	020000	DTO	= 20000	
207	040000	OBEN	= 40000	
208	100000	OBCK	= 100000	

:ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)

209				
210				
211				
212	000001	AT0	= 1	:DEVICE 0 (BIT #0)
213	000002	AT1	= 2	:DEVICE 1 (BIT #1)
214	000004	AT2	= 4	:DEVICE 2 (BIT #2)
215	000010	AT3	= 10	:DEVICE 3 (BIT #3)
216	000020	AT4	= 20	:DEVICE 4 (BIT #4)
217	000040	AT5	= 40	:DEVICE 5 (BIT #5)
218	000100	AT6	= 100	:DEVICE 6 (BIT #6)
219	000200	AT7	= 200	:DEVICE 7 (BIT #7)

```

220
221      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
222      ;(EACH BIT IS CALLED BY BIT NUMBER)
223
224      ;DRIVE TYPE REGISTER (RMDT) (#06)
225
226      000001      DT00      = 1      ;DRIVE TYPE NUMBER BIT 1
227      000002      DT01      = 2      ;DRIVE TYPE NUMBER BIT 2
228      000004      DT02      = 4      ;DRIVE TYPE NUMBER BIT 3
229      000010      DT03      = 10     ;DRIVE TYPE NUMBER BIT 4
230      000020      DT04      = 20     ;DRIVE TYPE NUMBER BIT 5
231      000040      DT05      = 40     ;DRIVE TYPE NUMBER BIT 6
232      000100      DT06      = 100    ;DRIVE TYPE NUMBER BIT 7
233      000200      DT07      = 200    ;DRIVE TYPE NUMBER BIT 8
234      000400      DT08      = 400    ;DRIVE TYPE NUMBER BIT 9
235      004000      DRQ       = 4000   ;DRIVE REQUEST REQUIRED (BIT #11)
236      020000      MOH       = 20000  ;MOVING HEAD (BIT #13)
237      040000      TAP       = 40000  ;TAPE DRIVE (BIT #14)
238      100000      NBA       = 100000 ;NOT BLOCK ADDRESSED (BIT #15)
239
240      ;LOOK-AHEAD REGISTER (RMLA) (#07)
241
242      000100      SC0       = 100     ;SECTOR COUNT FIELD 0 (BIT #6)
243      000200      SC1       = 200     ;SECTOR COUNT FIELD 1 (BIT #7)
244      000400      SC2       = 400     ;SECTOR COUNT FIELD 2 (BIT #8)
249
250      ;RM MAINTAINABILITY REGISTER #2 (RMMR2) (#10)
251
252      ;RM ERROR REGISTER #02 (RMMR2) (#10)
253
254      ;OFFSET REGISTER (RMOF) (#11)
255
256      000200      OFD       = 200     ;OFFSET DIRECTION (BIT #07)
257      001000      SSEI      = 1000    ;SKIP SECTOR INHIBIT (BIT #09)
258      002000      HCI       = 2000    ;HEADER COMPARE INHIBIT (BIT #10)
259      004000      ECI       = 4000    ;ERROR CORRECTION CODE INHIBIT (BIT #11)
260      010000      FMT16     = 10000   ;FORMAT BIT (BIT #12)
261
262      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
263      ;(EACH BIT IS CALLED BY BIT NUMBER)
264
265      ;CURRENT CYLINDER ADDRESS (RMR) (#13)
266      ;(EACH BIT IS CALLED BY BIT NUMBER)
267
268      ;SERIAL NUMBER REGISTER (RMSN) (#14)
269      ;(EACH IS CALLED BY BIT NUMBER)
270
271      ;RM ERROR REGISTER #02 (RMR2) (#15)
272
273      000010      DPE       = 10      ;DATA PARITY ERROR (BIT #3)
274      000040      SSE       = 40      ;SKIP SECTOR ERROR (BIT #5)
275      000200      DVC       = 200     ;DEVICE CHECK (BIT #7)
276      002000      LBC       = 2000    ;LOSS OF BIT CLOCK (BIT #10)
277      004000      LSC       = 4000    ;LOSS OF SYSTEM CLOCK (BIT #11)
278      010000      IVC       = 10000   ;INVLAID COMMAND ERROR (BIT #12)
279      020000      OPE       = 20000  ;OPERATOR PLUG ERROR (BIT #13)
280      040000      SKI       = 40000  ;SEEK INCOMPLETE (BIT #14)
  
```

```
281          100000          BSE      = 100000          :BAD SECTOR ERROR (BIT #15)
282
283          :ECC POSITION REGISTER (RMEC1) (#16)
284          :(EACH BIT IS CALLED BY BIT NUMBER)
285
286          :ECC PATTERN REGISTER (RMEC2) (#17)
287          :(EACH BIT IS CALLED BY BIT NUMBER)
288
289          :*****
290
291          :DRIVER OPERATION CODE DEFINITIONS
292          SEEK      = 105
293          RECAL     = 107
294          DRVCLR    = 111
295          RELEASE   = 113
296          OFFSET    = 115
297          RTC       = 117
298          READIN    = 121
299          PACK      = 123
300          SEARCH    = 131
301          WRCKD     = 151
302          WRCKHD    = 153
303          WRITE     = 161
304          WRTHD     = 163
305          READ      = 171
306          READHD    = 173
307          GETREG    = 141
308          SETFORM   = 143
309          SELDRV    = 145
310
311          :OTHER EQUATES
312
313          177400          SCTRWC = -256.          :WORD COUNT FOR SECTOR
314
```

```

1          .SBTTL TRAP CATCHER
          .=0
          000000
          ;*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
          000174 000174
          000176 000000
          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
          SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER

          .SBTTL STARTING ADDRESS(ES)
          000200 000137 003270      JMP      @WSTART1          ;;JUMP TO STARTING ADDRESS OF PROGRAM
          2
          3 000204 000137 003312      JMP      @WSTART2          ;SELECT OPERATING PARAMETERS
          4
          5 000210 000137 003260      JMP      @WSTART3          ;SELECT RH/RM ADDRESSES
          6
          7 000211 000137 003302      JMP      @WSTART4          ;COMBINATION OF 204 AND 210
          8
          9          .SBTTL ACT11 HOOKS
          ;*****
          ;HOOKS REQUIRED BY ACT11
          000220      $SVPC=.          ;SAVE PC
          000046 000046      .=46
          000046 015366      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
          000052 000052      .=52
          000052 020000      .WORD 20000          ;;2)SET LOC.52 TO 20000
          000220 000220      .=$SVPC          ;; RESTORE PC

          10          .=1100
          11          .SBTTL APT PARAMETER BLOCK
          12          ;*****
          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
          ;*****
          000024 001100      .$X=.          ;;SAVE CURRENT LOCATION
          000024 000024      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          000024 000200      200          ;;FOR APT START UP
          000044 000044      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
          000044 001100      $APTHDR      ;;POINT TO APT HEADER BLOCK
          001100 001100      .=.$X          ;;RESET LOCATION COUNTER
          ;*****
          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
          ;INTERFACE SPEC.

          001100      $APTHD:
          001100 000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
          001102 001234      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
          001104 001034      $STM:   .WORD 540.      ;;RUN TIM OF LONGEST TEST
          001106 001034      $PASTM: .WORD 540.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
          001110 001034      $UNITM: .WORD 540.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
          001112 000030      .WORD $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
          13          TAB.XY=.          ;COMMON TAG STARTING ADDRESS
          14
  
```

0

.SBTTL COMMON TAGS

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

Address	Value	Label	Format	Description
001114	001114	SCMTAG:	.=TAB.XY	::START OF COMMON TAGS
001114	000000		.WORD 0	
001116	000	\$STNUM:	.BYTE 0	::CONTAINS THE TEST NUMBER
001117	000	\$ERFLG:	.BYTE 0	::CONTAINS ERROR FLAG
001120	000000	\$ICNT:	.WORD 0	::CONTAINS SUBTEST ITERATION COUNT
001122	000000	\$LPADR:	.WORD 0	::CONTAINS SCOPE LOOP ADDRESS
001124	000000	\$LPERR:	.WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
001126	000000	\$ERTTL:	.WORD 0	::CONTAINS TOTAL ERRORS DETECTED
001130	000	\$ITEMB:	.BYTE 0	::CONTAINS ITEM CONTROL BYTE
001131	001	\$ERMAX:	.BYTE 1	::CONTAINS MAX. ERRORS PER TEST
001132	000000	\$ERRPC:	.WORD 0	::CONTAINS PC OF LAST ERROR INSTRUCTION
001134	000000	\$GDADR:	.WORD 0	::CONTAINS ADDRESS OF 'GOOD' DATA
001136	000000	\$BDADR:	.WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
001140	000000	\$GGDAT:	.WORD 0	::CONTAINS 'GOOD' DATA
001142	000000	\$BDDAT:	.WORD 0	::CONTAINS 'BAD' DATA
001144	000000		.WORD 0	::RESERVED--NOT TO BE USED
001146	000000		.WORD 0	
001150	000	\$AUTOB:	.BYTE 0	::AUTOMATIC MODE INDICATOR
001151	000	\$INTAG:	.BYTE 0	::INTERRUPT MODE INDICATOR
001152	000000		.WORD 0	
001154	177570	\$WR:	.WORD DSWR	::ADDRESS OF SWITCH REGISTER
001156	177570	\$DISPLAY:	.WORD DDISP	::ADDRESS OF DISPLAY REGISTER
001160	177560	\$TKS:	177560	::TTY KBD STATUS
001162	177562	\$TKB:	177562	::TTY KBD BUFFER
001164	177564	\$TPS:	177564	::TTY PRINTER STATUS REG. ADDRESS
001166	177566	\$TPB:	177566	::TTY PRINTER BUFFER REG. ADDRESS
001170	000	\$NULL:	.BYTE 0	::CONTAINS NULL CHARACTER FOR FILLS
001171	002	\$FILLS:	.BYTE 2	::CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012	\$FILLC:	.BYTE 12	::INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000	\$TPFLG:	.BYTE 0	::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000	\$REGAD:	.WORD 0	::CONTAINS THE ADDRESS FROM WHICH (\$REGO) WAS OBTAINED
001176	000000	\$REG0:	.WORD 0	::CONTAINS ((\$REGAD)+0)
001200	000000	\$REG1:	.WORD 0	::CONTAINS ((\$REGAD)+2)
001202	000000	\$REG2:	.WORD 0	::CONTAINS ((\$REGAD)+4)
001204	000000	\$REG3:	.WORD 0	::CONTAINS ((\$REGAD)+6)
001206	000000	\$REG4:	.WORD 0	::CONTAINS ((\$REGAD)+10)
001210	000000	\$REG5:	.WORD 0	::CONTAINS ((\$REGAD)+12)
001212	000000	\$TMP0:	.WORD 0	::USER DEFINED
001214	000000	\$TMP1:	.WORD 0	::USER DEFINED
001216	000000	\$TMP2:	.WORD 0	::USER DEFINED
001220	000000	\$TIMES:	0	::MAX. NUMBER OF ITERATIONS
001222	000000	\$ESCAPE:	0	::ESCAPE ON ERROR ADDRESS
001224	207	\$BELL:	.ASCIZ <207><377><377>	::CODE FOR BELL
001230	077	\$QUES:	.ASCII /?/	::QUESTION MARK
001231	015	\$CRLF:	.ASCII <15>	::CARRIAGE RETURN
001232	012	\$LF:	.ASCIZ <12>	::LINE FEED

\*\*\*\*\*  
.SBTTL APT MAILBOX-ETABLE

```

*****
.EVEN
001234          000000 $MAIL:          :: APT MAILBOX
001234 000000 $MSGTY: .WORD  AMSGTY  :: MESSAGE TYPE CODE
001236 000000 $FATAL: .WORD  AFATAL  :: FATAL ERROR NUMBER
001240 000000 $TESTN: .WORD  ATESTN  :: TEST NUMBER
001242 000000 $PASS:  .WORD  APASS   :: PASS COUNT
001244 000000 $DEVCT: .WORD  ADEVCT  :: DEVICE COUNT
001246 000000 $UNIT:  .WORD  AUNIT   :: I/O UNIT NUMBER
001250 000000 $MSGAD: .WORD  AMSGAD  :: MESSAGE ADDRESS
001252 000000 $MSGLG: .WORD  AMSGLG  :: MESSAGE LENGTH
001254          $ETABLE:          :: APT ENVIRONMENT TABLE
001254          $ENV:           .BYTE  AENV    :: ENVIRONMENT BYTE
001255          $ENVM:          .BYTE  AENVM   :: ENVIRONMENT MODE BITS
001256 000000 $$SWREG: .WORD  ASWREG  :: APT SWITCH REGISTER
001260 000000 $USWR:  .WORD  AUSWR   :: USER SWITCHES
001262 000000 $CPUOP: .WORD  ACPUOP  :: CPU TYPE, OPTIONS
                          BIT 15=11=CPU TYPE
                          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                          11/70=06,PDQ=07,Q=10
                          BIT 10=REAL TIME CLOCK
                          BIT 9=FLOATING POINT PROCESSOR
                          BIT 8=MEMORY MANAGEMENT
001264          000    $MAMS1: .BYTE  AMAMS1  :: HIGH ADDRESS, M.S. BYTE
001265          000    $MTYP1: .BYTE  AMTYP1  :: MEM. TYPE, BLK#1
                          MEM. TYPE BYTE -- (HIGH BYTE)
                          900 NSEC CORE=001
                          300 NSEC BIPOLAR=002
                          500 NSEC MOS=003
001266 000000 $MADR1: .WORD  AMADR1  :: HIGH ADDRESS, BLK#1
                          MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF 'TYPE' ABOVE
001270          000    $MAMS2: .BYTE  AMAMS2  :: HIGH ADDRESS, M.S. BYTE
001271          000    $MTYP2: .BYTE  AMTYP2  :: MEM. TYPE, BLK#2
001272 000000 $MADR2: .WORD  AMADR2  :: MEM. LAST ADDRESS, BLK#2
001274          000    $MAMS3: .BYTE  AMAMS3  :: HIGH ADDRESS, M.S. BYTE
001275          000    $MTYP3: .BYTE  AMTYP3  :: MEM. TYPE, BLK#3
001276 000000 $MADR3: .WORD  AMADR3  :: MEM. LAST ADDRESS, BLK#3
001300          000    $MAMS4: .BYTE  AMAMS4  :: HIGH ADDRESS, M.S. BYTE
001301          000    $MTYP4: .BYTE  AMTYP4  :: MEM. TYPE, BLK#4
001302 000000 $MADR4: .WORD  AMADR4  :: MEM. LAST ADDRESS, BLK#4
001304 000000 $VECT1: .WORD  AVECT1  :: INTERRUPT VECTOR#1, BUS PRIORITY#1
001306 000000 $VECT2: .WORD  AVECT2  :: INTERRUPT VECTOR#2, BUS PRIORITY#2
001310 000000 $BASE:  .WORD  ABASE   :: BASE ADDRESS OF EQUIPMENT UNDER TEST
001312 000000 $DEVN:  .WORD  ADEVN   :: DEVICE MAP
001314          .MEXIT
  
```

.SBTTL USER DEFINED TAGS

001314	000000	C.SWR:	.WORD	0	:CONTROL SWITCHES
001316	000031	ERMAX:	.WORD	25.	:MAXIMUM NUMBER OF ERRORS ALLOWED PER DRIVE
001320	000000	SAVCSW:	.WORD	0	:PREVIOUS CONTENTS OF 'C.SWR'
001322	000000	CNTRLC:	.WORD	0	:CONTROL 'C' FLAG
001324	000000	BUSADR:	.WORD	0	:GET RM ADDRESSES FLAG (0=NO, -1=YES)
001326	000000	LPTAVL:	.WORD	0	:LPT AVAILABLE STATUS (0=NO, 1=YES)
001330	000000	DRVSEL:	.WORD	0	:DRIVES SELECTED FOR TESTING
001332	013377	TSTNMS:	.WORD	13377,0	:DEFAULT IS RUN TESTS 0-7 & 11,12 & 14
001336	000000	OPNFLG:	.WORD	0,0	:MODIFY TEST PARAMETER FLAGS
001342	000000	CLKSTA:	.WORD	0	:CLOCK STATUS (0=NO CLOCK, +1=KW11-P, AND -1=KW11-L)
001344	000020	TICKMS:	.WORD	16.	:16 MILLISECONDS PER CLOCK TICK
001346	040432	TICKUS:	.WORD	16666.	:16666 MIRCSECONDS PER CLOCK TICK
001350	000000	BYPASS:	.WORD	0	
001352	000000	CHKDRV:	.WORD	0	:DRIVE UNDER TEST
001354	000000	DRVMSK:	.WORD	0	:DRIVE MASK BIT
001356	000000	SVSTAT:	.WORD	0	:STATUS/ERROR INDICATOR IS SAVED HERE ON AN ERROR
001360	000000	CYL.RD:	.WORD	0	:CYLINDER READ
001362	000000	TRK.RD:	.WORD	0	:TRACK READ
001364	000000	SEC.RD:	.WORD	0	:SECTOR READ
001366	000000	CYL.DS:	.WORD	0	:CYLINDER DESIRED
001370	000000	SEC.DS:	.WORD	0	:SECTOR DESIRED
001372	000000	TRK.DS:	.WORD	0	:TRACK DESIRED
001374	000000	LSTRK:	.WORD	0	:CONTAINS THE LAST TRACK OF THE UNIT UNDER TEST. RM80 = 13.
001376	000000	TIM.UP:	.WORD	0	:MINIMUM TIME
001400	000000		.WORD	0	:NUMBER OF COUNTS BELOW MIN. LIMIT
001402	000000		.WORD	0	:MAXIMUM TIME
001404	000000		.WORD	0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
001406	000000		.WORD	0,0	:TOTAL TIME OF ALL SEEKS
001412	000000		.WORD	0	:NUMBER OF SEEKS PERFORMED
001414	000000	TIM.DN:	.WORD	0	:MINIMUM TIME
001416	000000		.WORD	0	:NUMBER OF COUNTS BELOW MIN. LIMIT
001420	000000		.WORD	0	:MAXIMUM TIME
001422	000000		.WORD	0	:NUMBER OF COUNTS ABOVE MAX. LIMIT
001424	000000		.WORD	0,0	:TOTAL TIME OF ALL SEEKS
001430	000000		.WORD	0	:NUMBER OF SEEKS PERFORMED
001432	000000	TIM.PT:	.WORD	0	:POINTS TO TABLE OF TIMES
001434	000000	WCEFLG:	.WORD	0	:FATAL WRITE CHECK ERROR FLAG
001436	000000	STALLO:	.WORD	0	:VARIABLE STALL (TEST 4)
001440	000000	SVADR:	.WORD	0,0	:SAVE DISK ADDRESS
001444	000000	SEKTMR:	.WORD	0	:SEEK TIMER
001446	000000	SEKCNT:	.WORD	0	:SEEK COUNTER
001450	000000	DELTA:	.WORD	0	:TESTING RANGE FOR SERVO SETTLE DOWN TEST
001452	160400	TRCKWC:	.WORD	-<256.*31.>	:WORD COUNT FOR A FULL TRACK IN 16 BIT MODE
001454	000012	STALL1:	.WORD	10.	:10 MILLISECONDS STALL (TEST 0-10)
001456	000012	STALL2:	.WORD	10.	:10 MILLISECONDS STALL
001460	011610	STALL3:	.WORD	5000.	:5 SEC STALL
001462	000031	MXSTAL:	.WORD	25.	:MAX. INCREMENTING STALL ALLOWED IN TEST 4
001464	020	ERR.CT:	.BYTE	16.	:NUMBER OF ERRORS ALLOWED IN TESTS BEFORE GOING TO THE NEXT TEST
001465	000		.BYTE	0	:RESERVED
001466	000000	BASFLG:	.WORD	0	:FLAG FOR DETECTING BAD SECTOR
001470	000000	XXDP:	.WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH



:THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE  
 : 'XXDP' DEVICE CODE FOR THE RM80.

:THE FOLLOWING 8. LOCATIONS ARE USED IN THE AVERAGE SEEK TIME  
 : MEASUREMENT TEST ONLY. (TEST 15)

001472	000000	000000	000000	SKTJM: .WORD	0,0,0,0	:ACCUMULATED SEEK TIME
001502	000000	000000		SKCNT: .WORD	0,0	:NUMBER OF SEEKS
001506	000000			SKSIZ: .WORD	0	:SIZE OF CURRENT SEEK CYCLE
001510	000000			CYINC: .WORD	0	:CYLINDER OFFSET VALUE
001512				ERRCN: .BLKB	8.	:TOTAL ERROR COUNT FOR DRIVES 0-7.
001522	176700			RH.ADR: .WORD	176700	:RH/RM UNIBUS ADDRESS
001524	000254	000240		RHVEC: .WORD	254,240	:VECTOR ADDRESS AND PRIORITY
001530	000104	000106		PKV: .WORD	104,106	:KW11-P VECTOR ADDRESS
001534	172540			PKCS: .WORD	172540	:KW11-P CONTROL AND STATUS REG.
001536	172542			PKB: .WORD	172542	:KW11-P COUNT SET BUFFER
001540	172544			PKC: .WORD	172544	:KW11-P COUNTER
001542	000100	000102		LKV: .WORD	100,102	:KW11-L VECTOR ADDRESS
001546	177546			LKS: .WORD	177546	:KW11-L STATUS REGISTER
001550	177564			TPS: .WORD	177564	:TTY PRINTER STATUS
001552	177566			TPB: .WORD	177566	:TTY PRINTER BUFFER
001554	177514			LPS: .WORD	177514	:LINE PRINTER STATUS
001556	177516			LPB: .WORD	177516	:LINE PRINTER BUFFER

:BIT TABLE

001560	000001			BITS: .WORD	BIT00
001562	000002			.WORD	BIT01
001564	000004			.WORD	BIT02
001566	000010			.WORD	BIT03
001570	000020			.WORD	BIT04
001572	000040			.WORD	BIT05
001574	000100			.WORD	BIT06
001576	000200			.WORD	BIT07
001600	000400			.WORD	BIT08
001602	001000			.WORD	BIT09
001604	002000			.WORD	BIT10
001606	004000			.WORD	BIT11
001610	010000			.WORD	BIT12
001612	020000			.WORD	BIT13
001614	040000			.WORD	BIT14
001616	100000			.WORD	BIT15
001620	000001			.WORD	BIT00
001622	000002			.WORD	BIT01
001624	000004			.WORD	BIT02
001626	000010			.WORD	BIT03
001630	000020			.WORD	BIT04
001632	000040			.WORD	BIT05
001634	000100			.WORD	BIT06
001636	000200			.WORD	BIT07

.SBTTL TIMING LIMITS

:ROTATIONAL TEST TABLES

:60HZ TABLE

001640	037422			T11A: .WORD	ROTATE
001642	000000			.WORD	0

001644 003075 .WORD 1597. ;LO LIMIT (16.67MS - 4.3%)  
001646 003277 .WORD 1727. ;HI LIMIT (16.67MS + 3.5%)

:50HZ TABLE

T11B: .WORD ROTATE  
.WORD 0  
.WORD 1597. ;LO LIMIT (16.67MS - 4.3%)  
.WORD 1727. ;HI LIMIT (16.67MS + 3.5%)

:SEEK TEST TABLES

T12: .WORD ONECYL ;FORWARD  
.WORD REV ;REVERSE  
.WORD 0 ;NO LOWER LIMIT  
.WORD 600. ;HI LIMIT (6.0MS)

T13: .WORD AVERAGE ;FORWARD  
.WORD REV ;REVERSE  
.WORD 0 ;NO LOWER LIMIT  
.WORD 2500. ;HI LIMIT (25.0MS)

T14: .WORD MXSEEK ;FORWARD  
.WORD REV ;REVERSE  
.WORD 0 ;NO LOWER LIMIT  
.WORD 5000. ;HI LIMIT (50.0MS)

:SPECS. MESSAGE TABLES FOR ROTATIONAL AND TIMING TESTS

:ROTATIONAL MESSAGE AND LO/HI LIMITS

:60HZ TABLE

SP11A: .WORD MSG11X  
.WORD 1597. ;LO LIMIT (16.67MS - 4.3%)  
.WORD 1727. ;HI LIMIT (16.67MS + 3.5%)

:50HZ TABLE

SP11B: .WORD MSG11X  
.WORD 1597. ;LO LIMIT (16.67MS - 4.3%)  
.WORD 1727. ;HI LIMIT (16.67MS + 3.5%)

:TIMING TEST MESSAGES AND LO/HI LIMITS

SP12: .WORD MSG12X  
.WORD 0 ;NO LOWER LIMIT  
.WORD 600. ;HI LIMIT (6.0MS)

SP13: .WORD MSG13X  
.WORD 0 ;NO LOWER LIMIT  
.WORD 2500. ;HI LIMIT (25.0MS)

SP14: .WORD MSG14X  
.WORD 0 ;NO LOWER LIMIT  
.WORD 5000. ;HI LIMIT (50.0MS)

:STATUS/ERROR MESSAGE POINTER TABLE

STATBL: .WORD MSGB14 ;OFFLINE OR UNSAFE DRIVE REQUESTED  
000000 000000 .WORD C 0 0 0 ;FILLER WORDS  
001760 041140 .WORD MSGB09 ;SOFTWARE TIMEOUT ON THIS DRIVE  
001762 041177 .WORD MSGB08 ;SOFTWARE TIMEOUT ON ANOTHER DRIVE

001764 041241  
001766 041305  
001770 041355  
001772 000000  
001774 041375  
001776 041445

.WORD MSGB06  
.WORD MSGB05  
.WORD MSGB04  
.WORD 0  
.WORD MSGB02  
.WORD MSGB01

:ERROR OCCURRED DURING I/O OPERATION  
:ERROR OCCURRED DURING NON-I/O OPERATION  
:UNSAFE OCCURRED  
:FILLER WORD  
:DRIVE HAS NOT RESPONDED TO PORT REQUEST  
:DRIVE HAS BECOME NONEXISTENT

.SBTIL ERROR POINTER TABLE

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

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

002000

\$ERRTB:

4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45

002000 040311  
 002002 041503  
 002004 042754  
 002006 043364  
  
 002010 040363  
 002012 041520  
 002014 042760  
 002016 043370  
  
 002020 000000  
 002022 000000  
 002024 000000  
 002026 000000  
  
 002030 000000  
 002032 000000  
 002034 000000  
 002036 000000

:\*ERROR 1

:\* RH CONTROLLER INTERRUPT OCCURED (RMAS=0)  
 :\* ERR PC RMAS  
 :\* \$ERRPC \$REG3

EM1  
 DH1  
 DT1  
 DF1

:\*ERROR 2

:\* UNEXPECTED ATTENTION OCCURRED  
 :\* ERR PC DRIVE RMAS RMDS RMER1 RMMR2 RMER2  
 :\* \$ERRPC \$REG1 \$REG3 RMERRS RMERRS+2 RMERRS+4 RMERRS+4

EM2  
 DH2  
 DT2  
 DF2

:\*ERROR 3 -- NOT USED

0  
 0  
 0  
 0

:\*ERROR 4 -- NOT USED

0  
 0  
 0  
 0

:\*ERROR 5

:\* ADDRESS PLUG CHANGE BIT SET  
 :\* ERR PC DRIVE RMAS RMDS RMER1 RMMR2 RMER2  
 :\* \$ERRPC \$REG1 \$REG3 RMERRS RMERRS+2 RMERRS+4 RMERRS+6

46			
47	002040	040421	EM5
48	002042	041520	DH2
49	002044	042760	DT2
50	002046	043370	DF2
51			
52			;*ERROR 6 -- NOT USED
53			
54	002050	000000	0
55	002052	000000	0
56	002054	000000	0
57	002056	000000	0
58			
59			;*ERROR 7 -- NOT USED
60			
61	002060	000000	0
62	002062	000000	0
63	002064	000000	0
64	002066	000000	0
65			
66			;*ERROR 10
67			
68			;* RH CONTROLLER FAILED TO RESPOND TO ADDRESSING
69			;* RMCS1 ERR PC
70			;* RH.ADR \$ERRPC
71			
72	002070	040455	EM10
73	002072	041605	DH10
74	002074	043016	DT10
75	002076	043404	DF10
76			
77			;*ERROR 11
78			
79			;* DRIVE SELECTED IS NOT ONLINE
80			;* DRIVE ERR PC
81			;* \$REG2 \$ERRPC
82			
83	002100	040533	EM11
84	002102	041624	DH11
85	002104	043022	DT11
86	002106	043410	DF11
87			
88			;*ERROR 12
89			
90			;* IMPROPER HEADER DATA
91			;* TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR
92			;* \$TMP0 \$ERRPC \$REG0 \$CHKDRV CYL.DS TRK.DS SEC.DS
93			;* GDCYL GDTRK GDSCTR BDCYL BDTRK BDSCTR
94			;* CYL.DS TRK.DS SEC.DS CYL.RD TRK.RD SEC.RD
95			;* CYLNDR, TRACK, AND SECTOR ARE DECIMAL
96			
97	002110	040570	EM12
98	002112	041643	DH12
99	002114	043026	DT12
100	002116	043414	DF12
101			
102			;*ERROR 13

```

103
104
105      :*      DATA COMPARE FAILURE
106      :*      TEST      ERR PC  TST PC  DRIVE  CYLNR  TRACK  SECTOR
107      :*      $TMP0    $ERRPC $REG0  CHKDRV  CYL.DS  TRK.DS  SEC.DS
108      :*      GDDAT  BDDAT  WRDCNT  GDADR  BDADR
109      :*      $GDDAT $BDDAT $REG4  $GDADR $BDADR
110      :*      CYLNR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
111      EM13
112      DH12
113      DT13
114      DF13
115
116      :*ERROR 14 -- FOLLOWS #13
117
118      :*      $GDDAT $BDDAT $REG4  $GDADR $BDADR
119
120      0
121      0
122      DT13A
123      DF14
124
125      :*ERROR 15
126
127      :*      DATA COMPARE FAILURE
128      :*      TEST      ERR PC  TST PC  DRIVE  CYLNR  TRACK  SECTOR
129      :*      $TMP0    $ERRPC $REG0  CHKDRV  CYL.DS  TRK.DS  SEC.DS
130      :*      GDDAT  BDDAT  WRDCNT  GDADR  BDADR
131      :*      $GDDAT $BDDAT $REG4  $GDADR $BDADR
132      :*      CYLNR, TRACK, SECTOR, AND WRDCNT ARE DECIMAL
133
134      EM13
135      DH12
136      DT13
137      DF13
138
139      :*ERROR 16 -- FOLLOWS #15
140
141      :*      $GDDAT $BDDAT $REG4  $GDADR $BDADR
142
143      0
144      0
145      DT13A
146      DF14
147
148      :*ERROR 17
149
150      :*      DISK ERROR IN TIMING TEST
151      :*      TEST      ERR PC  DRIVE  RMCS1  RMD5  RMER1  RMMR2  RMER2
152      :*      $TMP0    $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
153
154      EM17
155      DH17
156      DT17
157      DF17
158
159      :*ERROR 20
    
```

```

160
161
162      :*      CLOCK (KW11-P) OVERFLOW IN TIMING TEST
163      :*      TEST      ERR PC  DRIVE  RMCS1  RMDS      RMER1      RMMR2      RMER2
164      :*      STMPO      $ERRPC  CHKDRV  RM.REG  RM.REG+12 RM.REG+14 RM.REG+40 RM.REG+42
165      EM20
166      002170 040674      DH17
167      002172 042057      DT17
168      002174 043110      DF17
169      002176 043440
170      :*ERROR 21
171
172      :*      DATA COMPARE FAILURE
173      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK
174      :*      STMPO      $ERRPC  $REGO  CHKDRV  CYL.DS  TRK.DS
175      :*      GDDAT  BDDAT  WRDCNT  SECTOR
176      :*      $REG1  $BDDAT $REG4  $REG1
177      :*      CYLINDR, TRACK, WRDCNT, AND SECTOR ARE DECIMAL
178
179      EM13
180      002200 040615      DH21
181      002202 042154      DT21
182      002204 043130      DF21
183      002206 043444
184      :*ERROR 22--FOLLOWS #21
185
186      :*      $REG1  $BDDAT  $REG4  $REG1
187
188      0
189      0
190      002210 000000      DT21A
191      002212 000000      DF22
192      002214 043144
193      002216 043454
194      :*ERROR 23
195
196      :*      DISK ERROR DURING SEEK
197      :*      TEST      ERR PC  DRIVE  CYLNDR  RMCS1  RMCS2  RMDS
198      :*      STMPO      $ERRPC  CHKDRV  CYL.DS  RM.REG  RM.REG+10 RM.REG+12
199      :*      RMER1  RMMR2  RMER2  RMDC  RMR
200      :*      RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
201      EM23
202      002220 040743      DH23
203      002222 042271      DT23
204      002224 043154      DF23
205      002226 043460
206      :*ERROR 24
207
208      :*      SEEK NOT COMPLETE WITHIN 120 MS
209      :*      TEST      ERR PC  DRIVE  CYLNDR  RMCS1  RMCS2  RMDS
210      :*      STMPO      $ERRPC  CHKDRV  CYL.DS  RM.REG  RM.REG+10 RM.REG+12
211      :*      RMER1  RMMR2  RMER2  RMDC  RMR
212      :*      RM.REG+14 RM.REG+40 RM.REG+42 RM.REG+34 RM.REG+36
213
214      EM24
215      002230 040772      DH23
216      002232 042271      DT23
217      002234 043154
  
```

217 002236 043460  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234 002240  
235  
236  
237  
238  
239  
240  
241  
242 002240 041032  
243 002242 042423  
244 002244 043204  
245 002246 043470  
246  
247  
248  
249  
250  
251  
252  
253 002250 041032  
254 002252 042461  
255 002254 043214  
256 002256 043474  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266 002260 041032  
267 002262 042461  
268 002264 043232  
269 002266 043500  
270  
271  
272  
273

DF23  
:\*\*\*\*\*  
:\* ERRORS 23-40 NOT USED  
:\* ERRORS 41-46 WILL HAVE AN EM THAT  
:\* VARIES DEPENDING ON THE ERROR, IT WILL BE IN THE FORM:  
:\* RH/RM ERROR (MESSAGE)  
:\* WHERE MESSAGE WILL BE ONE OR MORE OF THE FOLLOWING:  
:\* 1) OFFLINE DRIVE REQUESTED  
:\* 3) SOFTWARE TIMEOUT ON THIS DRIVE  
:\* 4) SOFTWARE TIMEOUT ON ANOTHER DRIVE  
:\* 5) ERROR OCCURRED DURING I/O OPERATION  
:\* 6) ERROR OCCURRED DURING NON-I/O OPERATION  
:\* 7) UNSAFE OCCURED  
:\*\*\*\*\*  
ITEM41:  
:\*ERROR 41  
:\* RH/RM ERROR (MESSAGE)  
:\* TEST ERR PC TST PC DRIVE  
:\* \$TMP0 \$ERRPC \$REGO CHKDRV  
EM41  
DH41  
DT41  
DF41  
:\*ERROR 42  
:\* RH/RM ERROR (MESSAGE)  
:\* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS  
:\* \$TMP0 \$ERRPC \$REGO CHKDRV RM.REG RM.REG+10 RM.REG+12  
EM41  
DH42  
DT42  
DF42  
:\*ERROR 43  
:\* RH/RM ERROR (MESSAGE)  
:\* TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS  
:\* \$TMP0 \$ERRPC \$REGO CHKDRV RM.REG RM.REG+10 RM.REG+12  
:\* RMER1 RMMR2 RMER2  
:\* RM.REG+14 RM.REG+40 RM.REG+42  
EM41  
DH42  
DT43  
DF43  
:\*ERROR 44  
:\* RH/RM ERROR (MESSAGE)



```

274      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK  SECTOR
275      :*      $TMP0    $ERRPC  $REGO   CHKDRV  CYL.DS  TRK.DS  SEC.DS
276      :*              RMCS1    RMCS2    RMD5    RMHR    RMDC    RMDA
277      :*              RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
278      :*      RMER1    RMR2    RMER2
279      :*      RM.REG+14  RM.REG+40  RM.REG+42
280      :*      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
    
```

```

281
282 002270 041032
283 002272 041643
284 002274 043256
285 002276 043510
    
```

```

EM41
DH12
DT44
DF44
    
```

:\*ERROR 45

```

288      :*      RH/RM ERROR (MESSAGE)
289      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK  SECTOR
290      :*      $TMP0    $ERRPC  $REGO   CHKDRV  CYL.DS  TRK.DS  SEC.DS
291      :*              RMCS1    RMCS2    RMD5    RMHR    RMDC    RMDA
292      :*              RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
293      :*      RMER1    RMR2    RMER2    RMWC    RMBA    RMDB
294      :*      RM.REG+14  RM.REG+40  RM.REG+42  RM.REG+2  RM.REG+4  RM.REG+22
295      :*      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
    
```

```

296
297
298 002300 041032
299 002302 041643
300 002304 043316
301 002306 043524
    
```

```

EM41
DH12
DT45
DF45
    
```

:\*ERROR 46

```

302
303      :*      FATAL WRITE CHECK ERROR (MESSAGE)
304      :*      TEST      ERR PC  TST PC  DRIVE  CYLNDR  TRACK  SECTOR
305      :*      $TMP0    $ERRPC  $REGO   CHKDRV  CYL.DS  TRK.DS  SEC.DS
306      :*              RMCS1    RMCS2    RMD5    RMHR    RMDC    RMDA
307      :*              RM.REG    RM.REG+10 RM.REG+12 RM.REG+36 RM.REG+34 RM.REG+06
308      :*      RMER1    RMR2    RMER2    RMWC    RMBA    RMDB
309      :*      RM.REG+14  RM.REG+40  RM.REG+42  RM.REG+2  RM.REG+4  RM.REG+22
310      :*      CYLNDR, TRACK, AND SECTOR ARE DECIMAL
    
```

```

311
312
313
314 002310 041046
315 002312 041643
316 002314 043316
317 002316 043524
    
```

```

EM46
DH12
DT45
DF45
    
```

1  
 2  
 3  
 4 002320 000000  
 5  
 6 002322 000000  
 7 002324 000000  
 8 002326 000000  
 9 002330 000000  
 10 002332 000000  
 11 002334 000000  
 12 002336 000000  
 13 002340 000000  
 14 002342 000000  
 15 002344 000000  
 16  
 17 002346 000000  
 18 002350 000000  
 19  
 20  
 21 002352 002726  
 24 002354 002742  
 002356 002764  
 002360 003002  
 002362 003020  
 002364 003036  
 002366 003054  
 002370 003072  
 002372 003106  
 002374 003124  
 002376 003136  
 002400 003146  
 002402 003156  
 002404 003166  
 25 002406 000000  
 26  
 27  
 28 002410 077777  
 29 002412 001060  
 30 002414 001060  
 31 002416 001060  
 32 002420 000015  
 33 002422 000015  
 34 002424 000015  
 35 002426 000036  
 36 002430 000036  
 37 002432 177777  
 38  
 39  
 40 002434 036710  
 41 002436 036712  
 42 002440 036715  
 43 002442 036720  
 44 002444 036723  
 45 002446 036726  
 46 002450 036731  
 47 002452 036734

.SBTTL TEST PARAMETER POINTERS AND TABLES

:COMMON STORAGE FOR TEST PARAMETER

PRM: .WORD 0 :THIS WORD TELLS WHICH OF THE  
 :FOLLOWING PARAMETERS ARE TO BE USED  
 RPT: .WORD 0 :REPEAT COUNTS FOR ALL TESTS  
 FC: .WORD 0 :FIRST CYLINDER  
 LC: .WORD 0 :LAST CYLINDER  
 IC: .WORD 0 :INCREMENT CYLINDER  
 FT: .WORD 0 :FIRST TRACK  
 LT: .WORD 0 :LAST TRACK  
 IT: .WORD 0 :INCREMENT TRACK  
 FS: .WORD 0 :FIRST SECTOR  
 LS: .WORD 0 :LAST SECTOR  
 PAT: .WORD 0 :PATTERN CODE  
 NC1: .WORD 0 :NEW CYLINDER ADDRESS  
 NC2: .WORD 0 :NEW CYLINDER 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  
 .WORD PRM13  
 .WORD PRM14  
 .WORD PRM15  
 .WORD 0 :TERMINATOR

:TABLE OF PARAMETER UPPER LIMITS

PRMLMT: .WORD 32767. :'R'  
 .WORD 560. :'FC'  
 .WORD 560. :'LC'  
 .WORD 560. :'IC'  
 .WORD 13. :'FT'  
 .WORD 13. :'LT'  
 .WORD 13. :'IT'  
 .WORD 30. :'FS'  
 .WORD 30. :'LS'  
 .WORD 177777 :'PAT'

:TABLE OF MESSAGE POINTERS

PRMSG: .WORD MSG.R  
 .WORD MSG.FC  
 .WORD MSG.LC  
 .WORD MSG.IC  
 .WORD MSG.FT  
 .WORD MSG.LT  
 .WORD MSG.IT  
 .WORD MSG.FS

```

48 002454 036737          .WORD  MSG.LS
49
50          ;STATUS/ERROR INDICATOR MESSAGES POINTER TABLE
51          ;DEFAULT VALUES OF TEST PARAMETERS
52
53 002456 000227 000310 000000 DFLT:  .WORD  227,200.,0.560.,0.0          :RECAL/RANDOM SEEK (T0)
54 002472 000677 000144 000000          .WORD  677,100.,0.256.,0.0,0.0,0.0          :SEEK/SEEK (T1)
55 002514 000237 000001 000000          .WORD  237,1,0,560.,1,0,0          :INCREMENT SEEK (T2)
56 002532 000237 000010 000000          .WORD  237,10,0,512.,1,0,0          :STEPPING SEEK (T3)
57 002550 000237 000001 000000          .WORD  237,1,0,560.,1,0,0          :OSCILLATING SEEK (T4)
58 002566 000237 000001 000000          .WORD  237,1,0,560.,1,0,0          :CONVERGING/DIVERGING SEEK (T5)
59 002604 000237 000001 000000          .WORD  237,1,0,560.,1,0,0          :SERVO ADDRESSING LOGIC NOISE (T6)
60 002622 000067 011610 000000          .WORD  67,5000.,0,560.,0,13.          :RANDOM SEEK TEST (T7)
61 002636 000237 000001 000000          .WORD  237,1,0,560.,1,0,0          :ALL SEEKS TEST (T10)
62 002654 000223 000001 000000          .WORD  223,1,0,0,0          :ROTATIONAL SPEED TIMING TEST (T11)
63 002666 000007 000001 000000          .WORD  7,1,0,560.          :ONE CYLINDER SEEK TIMING TEST (T12)
64 002676 000007 000001 000000          .WORD  7,1,0,155.          :AVERAGE SEEK TIMING TEST (T13)
65 002706 000007 000001 000000          .WORD  7,1,0,560.          :MAXIMUM SEEK TIMING TEST (T14)
66 002716 000007 000001 000000          .WORD  7,1,0,560.          :AVG. SEEK TIME MEASUREMENT TEST (T15)
67
68          ;PARAMETER TABLES
69
70          ;RECAL/RANDOM SEEK (T0)
71 002726 000227          PRM0:  .WORD  227
72 002730 000310          .WORD  200.
73 002732 000000          .WORD  0
74 002734 001060          .WORD  560.
75 002736 000000          .WORD  0
76 002740 000000          .WORD  0
77
78          ;SEEK/SEEK (T1)
79 002742 000677          PRM1:  .WORD  677
80 002744 000144          .WORD  100.
81 002746 000000          .WORD  0
82 002750 000400          .WORD  256.
83 002752 000000          .WORD  0
84 002754 000000          .WORD  0
85 002756 000000          .WORD  0
86 002760 000000          .WORD  0
87 002762 000000          .WORD  0
88
89          ;INCREMENT SEEK (T2)
90 002764 000237          PRM2:  .WORD  237
91 002766 000001          .WORD  1
92 002770 000000          .WORD  0
93 002772 001060          .WORD  560.
94 002774 000001          .WORD  1
95 002776 000000          .WORD  0
96 003000 000000          .WORD  0
97
98          ;STEPPING SEEK (T3)
99 003002 000237          PRM3:  .WORD  237
100 003004 000001          .WORD  1
101 003006 000000          .WORD  0
102 003010 001000          .WORD  512.
103 003012 000001          .WORD  1
104 003014 000000          .WORD  0
    
```

105	003016	000000	.WORD	0
106				
107			:OSCILLATING SEEK (T4)	
108	003020	000237	PRM4: .WORD	237
109	003022	000001	.WORD	1
110	003024	000000	.WORD	0
111	003026	001060	.WORD	560.
112	003030	000001	.WORD	1
113	003032	000000	.WORD	0
114	003034	000000	.WORD	0
115				
116			:CONVERGING/DIVERGING SEEK (T5)	
117	003036	000237	PRM5: .WORD	237
118	003040	000001	.WORD	1
119	003042	000000	.WORD	0
120	003044	001060	.WORD	560.
121	003046	000001	.WORD	1
122	003050	000000	.WORD	0
123	003052	000000	.WORD	0
124				
125			:SERVO ADDRESSING LOGIC NOISE GENERATOR (T6)	
126	003054	000237	PRM6: .WORD	237
127	003056	000001	.WORD	1
128	003060	000000	.WORD	0
129	003062	001060	.WORD	560.
130	003064	000001	.WORD	1
131	003066	000000	.WORD	0
132	003070	000000	.WORD	0
133				
134			:RANDOM SEEK TEST (T7)	
135	003072	000067	PRM7: .WORD	67
136	003074	011610	.WORD	5000.
137	003076	000000	.WORD	0
138	003100	001060	.WORD	560.
139	003102	000000	.WORD	0
140	003104	000015	.WORD	13.
141				
142			:ALL SEEKS TEST (T10)	
143	003106	000237	PRM10: .WORD	237
144	003110	000001	.WORD	1
145	003112	000000	.WORD	0
146	003114	001060	.WORD	560.
147	003116	000001	.WORD	1
148	003120	000000	.WORD	0
149	003122	000000	.WORD	0
150				
151			:ROTATIONAL SPEED TIMING TEST (T11)	
152	003124	000223	PRM11: .WORD	223
153	003126	000001	.WORD	1
154	003130	000000	.WORD	0
155	003132	000000	.WORD	0
156	003134	000000	.WORD	0
157				
158			:ONE CYLINDER SEEK TIMING TEST (T12)	
159	003136	000007	PRM12: .WORD	7
160	003140	000001	.WORD	1
161	003142	000000	.WORD	0

162 003144 001060  
163  
164  
165 003146 000007  
166 003150 000001  
167 003152 000000  
168 003154 000233  
169  
170  
171 003156 000007  
172 003160 000001  
173 003162 000000  
174 003164 001060  
175  
176  
177 003166 000007  
178 003170 000001  
179 003172 000000  
180 003174 001060  
181  
465  
473  
481  
486  
511  
529  
536  
540  
549  
553  
572  
581  
587  
588  
598  
608

.WORD 560.

;AVERAGE SEEK TIMING TEST (T13)

PRM13: .WORD 7  
.WORD 1  
.WORD 0  
.WORD 155.

;MAXIMUM SEEK TIMING TEST (T14)

PRM14: .WORD 7  
.WORD 1  
.WORD 0  
.WORD 560.

;AVERAGE SEEK TIME MEASUREMENT TEST (T15)

PRM15: .WORD 7  
.WORD 1  
.WORD 0  
.WORD 560.

```

1          ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 003176 011600      BADTMO: MOV      (SP),R0          ;SAVE PC WHERE THE TIME OUT OCCURED
4 003200 005740      TST      -(R0)             ;ADJUST PC -2
5 003202 022626      CMP      (SP)+,(SP)+         ;RESTORE STACK POINTER
6 003204 104401 003212  TYPE     .65$           ;:TYPE ASCIZ STRING
   003210 000417      BR       64$             ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 003250 010046      MOV      R0,-(SP)          ;SETUP FOR TYPING OUT PC
8 003252 104402      TYPOC
9 003254 000240      NOP
   ;:PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;:TO STOP ON UNEXPECTED TIMEOUT.
10
11 003256 000404      BR       START1          ;BRANCH TO START1
12
13          .SBTTL  START OF PROGRAM
14
15 003260 012737 177777 001324  START3: MOV      #-1,BUSADR      ;GET BUSADR FLAG
16 003266 000402      BR       STRT1A
17
18 003270 005037 001324      START1: CLR     BUSADR          ;CLR BUSADR FLAG
19 003274 005037 001322  STRT1A: CLR     CNTRLC         ;NO CONTROL 'C'
20 003300 000411      BR       START
21
22 003302 012737 177777 001324  START4: MOV      #-1,BUSADR      ;SET BUSADR FLAG
23 003310 000402      BR       STRT2A
24
25 003312 005037 001324      START2: CLR     BUSADR          ;CLR BUSADR FLAG
26 003316 012737 177777 001322  STRT2A: MOV      #-1,CNTRLC      ;SET CONTROL 'C' FLAG
27
28 003324 005227 000000      START:  INC     #0             ;TTY LOOP, WAIT FOR INCREMENT
29 003330 001375      BNE     -.4             ;OF WORD
30 003332 000005      RESET
   ;CLEAR THE WORLD
31
32          .SBTTL  INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV      #SCMTAG,R6          ;:FIRST LOCATION TO BE CLEARED
   CLR     (R6)+               ;:CLEAR MEMORY LOCATION
   CMP     #SWR,R6             ;:DONE?
   BNE     -.6                 ;:LOOP BACK IF NO
   MOV     #STACK,SP          ;:SETUP THE STACK POINTER
   ;:INITIALIZE A FEW VECTORS
   MOV     #SCOPE,@IOTVEC      ;:IOT VECTOR FOR SCOPE ROUTINE
   MOV     #340,@IOTVEC+2      ;:LEVEL 7
   MOV     #ERROR,@EMTVEC     ;:EMT VECTOR FOR ERROR ROUTINE
   MOV     #340,@EMTVEC+2     ;:LEVEL 7
   MOV     #STRAP,@TRAPVEC    ;:TRAP VECTOR FOR TRAP CALLS
   MOV     #340,@TRAPVEC+2    ;:LEVEL 7
   MOV     SENDCT,$EOPCT      ;:SETUP END-OF-PROGRAM COUNTER
   MOV     #176543,$SHINLM    ;:PRIME THE RANDOM NUMBER GENERATOR
   MOV     #123456,$LONUM     ;:BOTH HIGH AND LOW WORDS
   CLR     $TIMES              ;:INITIALIZE NUMBER OF ITERATIONS
   CLR     $ESCAPE            ;:CLEAR THE ESCAPE ON ERROR ADDRESS
   MOVB   #1,$SERMAX          ;:ALLOW ONE ERROR PER TEST
   MOV     #.,$SLPADR         ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
   MOV     #.,$SLPERR        ;:SETUP THE ERROR LOOP ADDRESS
   ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS

```

```

        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
003474 013746 000004      MOV    @#ERRVEC, -(SP)    ;;SAVE ERROR VECTOR
003500 012737 003534 000004  MOV    #64$, @#ERRVEC    ;;SET UP ERROR VECTOR
003506 012737 177570 001154  MOV    #DSWR, SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
003514 012737 177570 001156  MOV    #DDISP, DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
003522 022777 177777 175424  CMP    #-1, @SWR        ;;TRY TO REFERENCE HARDWARE SWR
003530 001012                BNE    66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;;AND THE HARDWARE SWR IS NOT = -1
                                ;;BRANCH IF NO TIMEOUT
003532 006403                BR     65$              ;;BRANCH IF NO TIMEOUT
003534 012716 003542 64$:   MOV    #65$, (SP)        ;;SET UP FOR TRAP RETURN
003540 000002                RTI
003542 012737 000176 001154 65$:   MOV    #SWREG, SWR     ;;POINT TO SOFTWARE SWR
003550 012737 000174 001156  MOV    #DISPREG, DISPLAY
003556 012637 000004 66$:   MOV    (SP)+, @#ERRVEC  ;;RESTORE ERROR VECTOR

003562 005037 001242      CLR    $PASS            ;;CLEAR PASS COUNT
003566 132737 000200 001255  BITB   #APTSIZE, $ENVM  ;;TEST USER SIZE UNDER APT
003574 001403                BEQ    67$              ;;YES, USE NON-APT SWITCH
003576 012737 001256 001154  MOV    #SSWREG, SWR     ;;NO, USE APT SWITCH REGISTER
003604
33      67$:
34      ;;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
35      MOV    #BADTMU, ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
36      MOV    #PR6, ERRVEC+2  ;;LEVEL 6
37

.SBTTL  TYPE PROGRAM NAME
        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003620 005227 177777      INC    #-1              ;;FIRST TIME?
003624 001033                BNE    68$              ;;BRANCH IF NO
003626 022737 015366 000042  CMP    #SENDAD, @#42    ;;ACT-11?
003634 001427                BEQ    68$              ;;BRANCH IF YES
003636 104401 003644      TYPE   ,69$            ;;TYPE ASCIZ STRING
003642 000424                BR     68$              ;;GET OVER THE ASCIZ
        ;;69$:
        .ASCIZ <CRLF>@CZRNGAO - RM80 FUNCTIONAL TEST, PT 4@<CRLF>
003714
        68$:
.SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
003714 005737 000042      TST    @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
003720 001012                BNE    70$              ;;BRANCH IF YES
003722 123727 001254 000001  CMPB   $ENV, #1         ;;ARE WE RUNNING UNDER APT?
003730 001406                BEQ    70$              ;;BRANCH IF YES
003732 023727 001154 000176  CMP    SWR, #SWREG     ;;SOFTWARE SWITCH REG SELECTED?
003740 001005                BNE    71$              ;;BRANCH IF NO
003742 104406                GTSWR   ;;GET SOFT-SWR SETTINGS
003744 000403                BR     71$
003746 112737 000001 001150 70$:   MOVB   #1, $AUTOB      ;;SET AUTO-MODE INDICATOR
003754                71$:

38
39 003754 012700 001174      MOV    #SREGAD, RO      ;;FIRST ADDRESS
40 003760 005020                CLR    (RO)+           ;;CLEAR VARIABLE STORAGE
41 003762 022700 001224      CMP    #SBELL, RO      ;;DONE?
42 003766 001374                BNE    1$              ;;NO--BRANCH
43 003770 013737 001550 001164  MOV    TPS, $TPS       ;;SETUP THE STATUS AND BUFFER REG'S
44 003776 013737 001552 001166  MOV    TPB, $TPB       ;;FOR THE TYPE ROUTINE
45
46      ;;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
47      ;;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
48
49 004004 005037 001470      CLR    XXDP            ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
    
```

50	004010	122737	000016	000041	CMPB	#16,2W41	:LOADED FROM AN RM80 ?
51	004016	001121			BNE	4S	:BR IF NOT
52	004020	013737	000040	001470	MOV	2W40,XXDP	:GET DEVICE INDICATOR AND NUMBER
53	004026	122737	000007	001470	CMPB	#7,XXDP	:IS IT A VALID NUMBER ?
54	004034	103002			BHIS	2S	:YES
55	004036	105037	001470		CLRB	XXDP	:NO, DEFAULT TO DRIVE 0
56	004042	005737	000042		TST	2W42	:CHAIN MODE OR ACT11 AUTO ACCEPT ?
57	004046	001425			BEQ	3S	:BR IF NEITHER
58	004050	104401	004056		TYPE	73S	::TYPE ASCIZ STRING
	004054	000412			BR	72S	::GET OVER THE ASCIZ
					::73S:	.ASCIZ	<CRLF>/NOT TESTING DRIVE /
					::72S:		
59	004102	005046			CLR	-(SP)	:CLEAR WORD ON STACK
60	004104	113716	001470		MOVW	XXDP,(SP)	:GET DRIVE ADDRESS
61	004110	104403			TYPOS		:TYPE THE ADDRESS
62	004112	001			.BYTE	1	:ONLY 1 CHARACTER
63	004113	000			.BYT:	0	:SUPPRESS LEADING ZEROS
64	004114	104401	001231		TYPE	,SCRLF	:CR-LF
65	004120	000460			BR	4S	:GET NUMBER OF DRIVES
66							
67	004122	005227	177777		3S:	INC	#-1
68	004126	001055			BNE	4S	:NO
69	004130	104401	004136		TYPE	75S	::TYPE ASCIZ STRING
	004134	000410			BR	74S	::GET OVER THE ASCIZ
					::75S:	.ASCIZ	<CRLF>/TO TEST DRIVE /
					::74S:		
70	004156	005046			CLR	-(SP)	:CLEAR WORD ON STACK
71	004160	113716	001470		MOVW	XXDP,(SP)	:GET DRIVE ADDRESS
72	004164	104403			TYPOS		:TYPE DRIVE ADDRESS
73	004166	001			.BYTE	1	:ONLY 1 CHARACTER
74	004167	000			.BYTE	0	:SUPPRESS LEADING ZEROS
75	004170	104401	004176		TYPE	76S	::TYPE ASCIZ STRING
	004174	000432			BR	4S	::GET OVER THE ASCIZ
					::76S:	.ASCIZ	/, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
					::4S:		
79	004262	004737	020134		JSR	PC,\$TKINT	:TURN ON THE TTY KEYBOARD INTERRUPT
80	004266	005227	177777		INC	#-1	:SEE IF FIRST START
81	004272	001002			BNE	SRTINT	:BR IF NOT
82	004274	004737	036276		JSR	PC,GETADR	:GET OR CHECK THE RH/RM ADDRESS
83							
84	004300	105037	001512		SRTINT:	CLRB	ERRCN
87	004304	105037	001513		CLRB	ERRCN+1	:CLEAR DRV 1 ERROR COUNT
	004310	105037	001514		CLRB	ERRCN+2	:CLEAR DRV 2 ERROR COUNT
	004314	105037	001515		CLRB	ERRCN+3	:CLEAR DRV 3 ERROR COUNT
	004320	105037	001516		CLRB	ERRCN+4	:CLEAR DRV 4 ERROR COUNT
	004324	105037	001517		CLRB	ERRCN+5	:CLEAR DRV 5 ERROR COUNT
	004330	105037	001520		CLRB	ERRCN+6	:CLEAR DRV 6 ERROR COUNT
	004334	105037	001521		CLRB	ERRCN+7	:CLEAR DRV 7 ERROR COUNT
88	004340	004737	023176		JSR	PC,LP.AVL	:CHECK FOR A LINE PRINTER
89	004344	012737	000001	001120	MOV	#1,\$ICNT	:SET ITERATION COUNT TO 1
90	004352	004737	026656		JSR	PC,GETSWR	:GO CHECK FOR CONTROL SWITCHES
91							
92	004356	004737	023240		SETVEC:	JSR	PC,ST.CLK
93	004362	004737	031376		JSR	PC,RMINIT	:CHECK THE DRIVE STATUS
94	004366	012737	177777	031330	MOV	#-1,SAVEFG	:SET THE SAVE REGISTERS FLAG
95	004374	005037	177776		CLR	PS	:ENSURE THE PRIORITY = 0
96	004400	005227	177777		INC	#-1	:FIRST TIME THRU HERE ?



97	004404	001403		BEQ	1\$		:BR IF NO
98	004406	005737	001322	TST	CNTRLC		:CONTROL 'C' SWITCH SET ?
99	004412	001066		BNE	SRTDRV		:CONTINUE IF YES
100	004414	005004		1\$:	CLR	R4	:DRIVE TABLE POINTER
101	004416	104401	036777	TYPE	,UNSTAT		:TYPE 'UNIT STATUS'
102	004422	104401	001231	2\$:	,\$CRLF		:CR-LF
103	004426	010446		MOV	R4,-(SP)		::SAVE R4 FOR TYPEOUT
	004430	104403		TYPOS			::GO TYPE--OCTAL ASCII
	004432	002		.BYTE	2		::TYPE 2 DIGIT(S)
	004433	000		.BYTE	0		::SUPPRESS LEADING ZEROS
104	004434	104401	040051	TYPE	,BLNKS4		:TYPE 4 BLANKS
105	004440	105764	031242	TSTB	DRVSTA(R4)		:CHECK DRIVE'S STATUS
106	004444	100416		BMI	5\$		:BR IF UNSAFE
107	004446	001020		BNE	6\$		:BR IF ONLINE
108	004450	105764	031252	TSTB	DRVTYP(R4)		:SEE IF OFFLINE OR NONEXISTENT
109	004454	001404		BEQ	3\$		:BR IF NONEXISTENT
110	004456	100006		BPL	4\$		:BR IF OFFLINE
111	004460	104401	037063	TYPE	,NOTRM		:DRIVE NOT AN RM80
112	004464	000433		BR	11\$		:CHECK NEXT DRIVE
113							
114	004466	104401	037036	3\$:	TYPE	,NOTPRS	:DRIVE NOT PRESENT
115	004472	000430		BR	11\$		:CHECK NEXT DRIVE
116							
117	004474	104401	037015	4\$:	TYPE	,UNTOFF	:DRIVE OFFLINE
118	004500	000416		BR	8\$		:PRINT DRIVE TYPE
119							
120	004502	104401	037053	5\$:	TYPE	,NOTSAF	:DRIVE UNSAFE
121	004506	000413		BR	8\$		:PRINT DRIVE TYPE
122							
123	004510	005737	001470	6\$:	TST	XXDP	:LOADED FROM THIS DEVICE ?
124	004514	001406		BEQ	7\$		:BR IF NO
125	004516	123704	00147C	CMPB	XXDP,R4		:LOADED FROM THIS DRIVE ?
126	004522	001003		BNE	7\$		:BR IF NO
127	004524	104401	037100	TYPE	,LODEV		:DRIVE IS LOAD DEVICE
128	004530	000411		BR	11\$		
129	004532	104401	037026	7\$:	TYPE	,UNTON	:DRIVE ONLINE
130	004536	104401	040053	8\$:	TYPE	,BLNKS2	:TYPE 2 BLANKS
135	004542	012737	037115 004552	MOV	#\$RM80,10\$		:ASSUME ADDRESS OF RM80 MESSAGE
140							
141	004550	104401		9\$:	TYPE		:TYPE THE DRIVE TYPE MESSAGE
142	004552	000000		10\$:	.WORD	0	:MESSAGE ADDRESS HERE
143							
144	004554	005204		11\$:	INC	R4	:INCREMENT DRIVE NUMBER/TABLE POINTER
145	004556	020427	000010	CMP	R4,#8.		:FINISHED ?
146	004562	001317		BNE	2\$		:BR IF NOT
147	004564	104401	001231	TYPE	,\$CRLF		:CR-LF

```

1          .SBTTL  GET UNIT STATUS
2
3 004570 004737 023146      SRTDRV: JSR    PC,CNTCLR      ;GO CLEAR MASSBUS CONTROLLER
4 004574 005737 001322      TST    CNTRLC          ;CONTROL 'C' START/RESTART?
5 004600 001427              BEQ    2$              ;NO--BRANCH
6 004602 013746 001320      MOV    SAVCSW,-(SP)    ;GET THE PREVIOUS 'C.SWR' CONTENTS
7 004506 063716 001314      ADD    C.SWR,(SP)      ;SET UP TO SEE IF 'BIT00' IS DIFFERENT
8 004612 032726 000001      BIT    #BIT00,(SP)+    ;IS 'BIT00' DIFFERENT ?
9 004616 001405              BEQ    1$              ;BR IF NOT
10 004620 013737 001314 001320  MOV    C.SWR,SAVCSW    ;STORE PRESENT 'C.SWR' VALUE
11 004626 004737 023516      JSR    PC,LODFLT      ;RESET DEFAULT PARAMETERS
12 004632 004737 027106      1$:   JSR    PC,GT.PRM    ;GET PARAMETERS
13 004636 005737 001332      TST    TSTNMS         ;ANY TEST SELECTED THIS CYCLE?
14 004642 001034              BNE    6$              ;BR IF YES
15 004644 005737 001334      TST    TSTNMS+2       ;ANY TEST SELECTED THIS CYCLE ?
16 004650 001031              BNE    6$              ;BR IF YES
17 004652 104401 037375      TYPE   ,NOTEST        ;TYPE 'NO TESTS SPECIFIED'
18 004656 000765              BR     1$
19
20 004660 004737 023516      2$:   JSR    PC,LODFLT    ;SETUP DEFAULT PARAMETERS
21 004664 005037 001330      CLR    DRVSEL         ;NO DRIVES SELECTED
22 004670 005000              CLR    R0              ;DETERMINE THE DRIVES THAT
23 004672 012701 000001      MOV    #1,R1          ;ARE AVAILABLE FOR TESTING
24
25 004676 105760 031242      3$:   TSTB   DRVSTA(R0)    ;IS DRIVE ON-LINE ?
26 004702 003411              BLE    5$              ;BR IF NO
27 004704 005737 001470      TST    XXDP           ;LOADED FROM THIS DEVICE ?
28 004710 001403              BEQ    4$              ;BR IF NO
29 004712 123700 001470      CMPB   XXDP,R0        ;LOADED FROM THIS DRIVE ?
30 004716 001403              BEQ    5$              ;BR IF YES
31 004720 156037 031356 001330  4$:   BISB   ATABIT(R0),DRVSEL ;YES, SELECT DRIVE FOR TESTING
32 004726 005200      5$:   INC    R0              ;TRY NEXT DRIVE
33 004730 106301              ASLB   R1              ;ANY MORE DRIVES TO CHECK ?
34 004732 001361              BNE    3$              ;BR IF YES
35 004734      6$:
43 004734 104401 037122      7$:   TYPE   ,DRIVES      ;'DRIVES(S) TO BE TESTED'
44 004740 005037 015322      CLR    $ENDCT         ;DETERMINE PASSES TO MAKE AND
45 004744 005000              CLR    R0              ;THE DRIVES TO BE TESTED
46 004746 013701 001330      MOV    DRVSEL,R1      ;ANY DRIVES SELECTED?
47 004752 001017              BNE    9$              ;YES--BRANCH
48 004754 104401 037153      TYPE   ,NONE          ;'NONE'
49 004760 104401 001231      TYPE   ,$CRLF         ;CR-LF
50 004764 005737 000042      TST    #42            ;ANY MONITOR PRESENT ?
51 004770 001002              BNE    8$              ;BR IF YES
52 004772 000137 003312      JMP    START2         ;RETURN TO '^C' INPUT
53 004776 005300      8$:   DEC    R0              ;THESE TWO LOOPS ARE ADDED TO
54 005000 001376              BNE    -2              ;WAIT FOR TTY
55 005002 005300              DEC    R0
56 005004 001376              BNE    -2
57 005006 000137 015356      JMP    $GET42         ;RETURN CONTROL TO MONITOR
58
59 005012 006201      9$:   ASR    R1              ;REPORT THE DRIVES TO BE TESTED
60 005014 103011              BCC    10$
61 005016 005237 015322      INC    $ENDCT         ;GIVE THIS DRIVE A PASS
62 005022 010046      MOV    R0,-(SP)       ;;SAVE R0 FOR TYPEOUT
    005024 104403              TYPOS  ;;GO TYPE--OCTAL ASCII
    005026 001              .BYTE 1              ;;TYPE 1 DIGIT(S)
    
```

005027	000				.BYTE	0		::SUPPRESS LEADING ZEROS
63 005030	005701				TST	R1		:MORE DRIVES?
64 005032	001404				BEQ	11\$		:NO--BRANCH
65 005034	104401	037160			TYPE	,COMMA		:
66 005040	005200			10\$:	INC	R0		:FORM DRIVE NUMBER
67 005042	000763				BR	9\$		
68								
69 005044	104401	001231		11\$:	TYPE	,SCLF		:CR-LF
70 005050	013737	015322	015314		MOV	\$ENDCT,\$EOPCT		
71 005056	005737	001342			TST	CLKSTA		:IS KW11-P AVAILABLE ?
72 005062	003006				BGT	12\$		:BR IF YES
73 005064	032737	017000	001332		BIT	#17000,\$TSTNMS		:ANY TIMING TESTS TO BE PERFORMED ?
74 005072	001402				BEQ	12\$		:BR IF NO
75 005074	104401	037166			TYPE	,NOCLOK		:TYPE NO KW11-P CLOCK MESSAGE
76 005100	000414			12\$:	BR	RSTRT1		

```

1          .SBTTL PROGRAM RESTARTS HERE
2
3          .ENABL LSB
4
5 005102 005737 001330 RSTART: TST   DRVSEL      ;ANY DRIVES SELECTED ?
6 005106 001022          BNE    3$      ;BR IF YES
7 005110 005737 000042 RSTART: TST   @#42      ;ANY MONITOR PRESENT ?
8 005114 001402          BEQ    1$      ;BR IF NO
9 005116 000137 015356 RSTART: JMP   $GET42     ;RETURN CONTROL TO MONITOR
10 005122 104401 037351 1$:   TYPE  ,NODRVS     ;TYPE 'NO DRIVES TO TEST'
11 005126 000137 003312 RSTART: JMP   START2    ;NO--GET DRIVE ENTRY & RESTART AT BEGINNING
12
13 005132 005037 001352 RSTR1: CLR   CHKDRV     ;START WITH DRIVE 0 AGAIN
14 005136 012737 000001 001354 MOV   #1,DRVMSK
15 005144 033737 001354 001330 2$:   BIT   DRVMSK,DRVSEL ;IS THIS DRIVE SELECTED?
16 005152 001006          BNE    4$      ;YES--GO CHECK IF DRIVE IS READY FOR TESTING
17 005154 005237 001352 3$:   INC   CHKDRV     ;MOVE TO NEXT DRIVE NUMBER
18 005160 106337 001354 ASLB  DRVMSK     ;DONE TESTING ALL DRIVES ?
19 005164 103762          BCS   RSTR1    ;BR IF YES
20 005166 000766          BR    2$      ;NO--CHECK DRIVE SELECT
21
22 005170 013702 001352 4$:   MOV   CHKDRV,R2     ;PICKUP THE DRIVE NUMBER
23 005174 105762 031242 TSTB  DRVSTA(R2)   ;IS DESIRED DRIVE ON-LINE?
24 005200 003007          BGT   5$      ;YES, BRANCH
25 005202 104011          EMT   11
26 005204 043737 001354 001330 BIC   DRVMSK,DRVSEL ;DESELECT DRIVE FROM TEST
27 005212 005337 015314 DEC   $EOPCT     ;ADJUST 'EOP' COUNT
28 005216 000731          BR    RSTART   ;RETURN
29
30 005220 004737 023146 5$:   JSR   PC,CNTCLR   ;GO CLEAR MASSBUS CONTROLLER
31 005224 005037 177776 CLR   PS        ;ENSURE THE PRIORITY = 0
32 005230 010237 036506 MOV   R2,DPB.A  ;SET THE DRIVE NUMBER INTO THE DPB'S
33 005234 010237 036530 MOV   R2,DPB.B
34 005240 010237 036552 MOV   R2,DPB.C
35 005244 010237 036574 MOV   R2,DTADPB
36 005250 004737 024032 JSR   PC,LDCMD  ;LOAD COMMAND INTO DPB.B AND DPB.C
37 005254 012737 015142 001350 MOV   #EOP,BYPASS ;IF ERROR GO TO END OF PROGRAM
38 005262 112737 000020 036507 MOVB  #20,DPB.A+1 ;ASSUME 16 BIT FORMAT
39 005270 032737 000001 001314 BIT   #BIT00,C.SWR ;16 BIT FORMAT REQUESTED ?
40 005276 001402          BEQ   6$      ;BR IF YES
41 005300 105037 036507 CLRB  DPB.A+1    ;CLEAR THE 'FMT16' BIT
42 005304 112737 000143 036510 6$:   MOVB  #SETFORM,DPB.A+2 ;SET THE FORMAT BIT PER DPB.A+1
43 005312 004037 024076 JSR   RO,CALL.A ;GO EXECUTE THE COMMAND
44 005316 112737 000107 036510 MOVB  #RECAL,DPB.A+2 ;RECAL:COMMAND
45 005324 004037 024076 JSR   RO,CALL.A ;GO EXECUTE THE COMMAND
46 005330 104401 001231 TYPE  ,SCLF     ;CR-LF
47 005334 104401 037273 TYPE  ,MSDRIV   ;TYPE 'DRIVE'
48 005340 010246          MOV   R2,-(SP)  ;SAVE R2 FOR TYPEOUT
   005342 104403          TYPOS ;GO TYPE--OCTAL ASCII
   005344 002          .BYTE 2 ;TYPE 2 DIGIT(S)
   005345 000          .BYTE 0 ;SUPPRESS LEADING ZEROS
49 005346 104401 037160 TYPE  ,COMMA   ;TYPE ','
50 005352 104401 037253 TYPE  ,SERIAL  ;TYPE 'SERIAL'
51 005356 012700 000004 MOV   #4,R0     ;FOUR DIGITS TO TYPE
52 005362 013701 036670 MOV   RM,REG+30,R1 ;SERIAL NUMBER
53 005366 005002 7$:   CLR   R2      ;ZERO
54 005370 006101          ROL   R1      ;PUT THE NEXT DIGIT

```

```
55 005372 006102      ROL      R2      :INTO R2
56 005374 006101      ROL      R1
57 005376 006102      ROL      R2
58 005400 006101      ROL      R1
59 005402 006102      ROL      R2
60 005404 006101      ROL      R1
61 005406 006102      ROL      R2
62 005410 062702 000060  ADD      #'0,R2      :MAKE IT ASCII
63 005414 010227      MOV      R2,(PC)+   :SAVE IT
64 005416 000000      8$:      .WORD      0
65 005420 104401 005416  TYPE      ,8$      :TYPE
66 005424 005300      DEC      R0      :ALL DIGITS TYPED?
67 005426 003357      BGT      7$      :NO -- BRANCH
68 005430 104401 001231  TYPE      ,$CRLF   :CR-LF
69 005434 113737 001464 001131  MOVB     ERR.CT,$ERMAX :SETUP MAX ERROR COUNT
70
71      .DSABL  LSB
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
43  
44

```

:////////////////////
*IN THE DESCRIPTIONS OF THE BELOW TESTS THE VARIABLES USED
*AND THEIR DEFAULT VALUES (UNLESS SPECIFIED OTHERWISE) ARE:
*
*MNEMONIC      VALUE      VARIABLE
*-----
*R              1          ITERATIONS (REPEATS)
*FC             0          FIRST CYLINDER ADDRESS
*LC             560.       LAST CYLINDER ADDRESS
*IC             1          INCREMENT VALUE
*NC OF NC1     FC+IC      NEW OR MODIFIED CYLINDER
                    ADDRESS
*NC2           LC-IC      NEW OR MODIFIED CYLINDER
                    ADDRESS
*
*FT             0          FIRST TRACK ADDRESS
*LT             13.        LAST TRACK ADDRESS
*IT             1          INCREMENT VALUE
*NT             FT+IT      NEW OR MODIFIED TRACK ADDRESS
*
*FS             0          FIRST SECTOR ADDRESS
*LS             30.        LAST SECTOR ADDRESS
*
:////////////////////

```

.SBTTL SEEK TESTS

```

:////////////////////
*THE SEEK TESTS WILL BE EXECUTED USING IMPLIED SEEKS. THESE
*IMPLIED SEEKS WILL BE PERFORMED BY 'READ HEADER AND
*DATA' COMMANDS TO TRACK 'FT' SECTOR 'FS' OF THE DESIRED CYLINDER.
*THE WORD COUNT WILL BE SET SUCH THAT ONLY THE CYLINDER AND
*TRACK/SECTOR WORDS OF THE HEADER ARE READ.
:////////////////////

```

```

:*****
*TEST 0      RECAL/RANDOM SEEK TEST
*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A RECALIBRATE COMMAND
*AND THEN SEEK TO A RANDOM CYLINDER BETWEEN 'FC' AND 'LC'. AT
*THE COMPLETION OF BOTH COMMANDS, STATUS INDICATORS ARE CHECKED
*TO ENSURE THAT NO ERRORS OCCURRED.
:*****

```

```

005442
005442 009240
005444 033737 001560 001332
005452 001002
005454 000137 0057C4

005460 012737 000000 001116
005466 004737 023734
005472 012737 005612 001124
005500 013737 002322 001220
005506 112737 000031 001131
005514 012737 000000 001240

```

```

TST0:
NOP
BIT     BITS+<0*2>,TSTNMS      ;DO THIS TEST?
BNE     .+6                     ;BR IF YES
JMP     TST1                     ;NO--JUMP TO TEST1

MOV     #0,$TSTNM                ;SET TEST #0 AND CLEAR ($ERFLG)
JSR     PC,LODPRM                ;LOAD THE PARAMETERS FOR THE TEST
MOV     #TST0,$LPERR             ;SETUP THE LOOP ON ERROR ADDRESS
MOV     RPT,$TIMES               ;GET THE ITERATION COUNT
MOVB    #25,$ERMAX               ;MAX ERRORS ALLOWED FOR TEST
MOV     #0,$TESTN                ;SET TEST NUMBER IN APT MAIL BOX

```

45

```

005522 032777 010000 173424      BIT      #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
005530 001406                      BEQ      .+16          ;BR IF YES
005532 104401 037265              TYPE    .MSGTST       ;TYPE 'TEST'
005536 013746 001240              MOV     $TESTN,-(SP)  ;;SAVE $TESTN FOR TYPEDOUT
005542 104403                      TYPOS   $TESTN,-(SP)  ;;GO TYPE--OCTAL ASCII
005544 002                          .BYTE   2             ;;TYPE 2 DIGIT(S)
005545 000                          .BYTE   0             ;;SUPPRESS LEADING ZEROS

46 005546 112737 000107 036510      MOVB    #RECAL,DPB.A+2 ;RECAL=COMMAND
47 005554 113737 002340 036540      MOVB    FS,DPB.B+10   ;FS
48 005562 113737 002332 036541      MOVB    FT,DPB.B+11   ;FT
49 005570 013737 002326 036542      MOV     LC,DPB.B+12   ;LC
70 005576 012737 005702 001350      MOV     #EXITO,BYPASS ;GO TO EXITO ON ERROR
005604 012737 005612 001122      MOV     #TESTO,$LPADR ;SETUP LGOP ADDRESS
005612                                TESTO:
005612 012706 001100      MOV     #STACK,SP    ;SET UP STACK PCINTER
005616 004037 024076      JSR     RO,CALL.A    ;GO EXECUTE THE COMMAND
005622 013737 002324 036542      MOV     FC,DPB.B+12  ;INITIAL CYLINDER ADDRESS
005630 023737 002324 002326      CMP     FC,LC        ;CYLINDER LIMITS THE SAME ?
005636 001417                      BEQ     1$           ;BR IF THEY ARE
005640 004737 022502      JSR     PC,$RAND     ;CYCLE THE RANDOM NUMBER GENERATOR
005644 013746 022600      MOV     $HINUM,-(SP) ;USE THE HIGH RANDOM NUMBER
005650 005046                      CLR     -(SP)        ;UPPER DIVIDEND
005652 013746 002326      MOV     LC,-(SP)     ;FORM THE DIVISOR
005656 005216                      INC     (SP)         ;INCREMENT
005660 163716 002324      SUB     FC,(SP)      ;SUBTRACT THE LOWER LIMIT
005664 004737 022604      JSR     PC,$DIV      ;DIVIDE
005670 062037 036542      ADD     (SP)+,DPB.B+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER
005674 005726                      TST    (SP)+        ;DISCARD THE QUOTENT
005676                                1$:
005676 004037 024244      JSR     RO,CALL.B    ;GO EXECUTE THE COMMAND
005702 000004      EXITO: SCOPE        ;CALL SCOPE ROUTINE

71
78
79
;*****
;*TEST 1      SEEK/SEEK TEST
;*THIS TEST WILL CAUSE THE DRIVE TO EXECUTE A FORWARD SFEK
;*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 ENSURE PROPER OPERATION.
;*****
TST1:
005704                                NOP
005704 000240      BIT     BITS+<1*2>,$TSTNMS ;DO THIS TEST?
005706 033737 001562 061332      BNE    .+6           ;BR IF YES
005714 001002      JMP     TST2         ;NO--JUMP TO TEST2
005716 000137 006112

005722 012737 000001 001116      MOV     #1,$TSTNM    ;SET TEST #1 AND CLEAR (SERFLG)
005730 004737 023734      JSR     PC,LODPRM   ;LOAD THE PARAMETERS FOR THE TEST
005734 012737 006074 001124      MOV     #TEST1,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
005742 013737 002322 001220      MOV     RPT,$TIMES  ;GET THE ITERATION COUNT
005750 112737 000031 001131      MOVB   #25,$ERMAX   ;MAX ERRORS ALLOWED FOR TEST
005756 012737 060001 001240      MOV     #1,$TESTN   ;;SET TEST NUMBER IN APT MAIL BOX

80
005764 032777 010000 173162      BIT     #SW12,@SWR   ;INHIBIT TYPING TEST NUMBER ?
005772 001406                      BEQ     .+16         ;BR IF YES
005774 104401 037265              TYPE    .MSGTST     ;TYPE 'TEST'
    
```

T1

```

006000 013746 001240      MOV      $TESTN,-(SP)      ;;SAVE $TESTN FOR TYPEOUT
006004 104403              TYPCLS                    ;;GO TYPE--OCTAL ASCII
006006 002                .BYTE 2                    ;;TYPE 2 DIGIT(S)
006007 000                .BYTE 0                    ;;SUPPRESS LEADING ZEROS

81 006010 005037 001466      CLR      BASFLG           ;CLEAR BAD SECTOR ENCOUNTER FOR THE DRIVE
82 006014 113737 002340 036540  MOVB    FS,DPB.B+10      ;FS
83 006022 113737 002342 036562  MOVB    LS,DPB.C+10      ;LS
84 006030 113737 002332 036541  MOVB    FT,DPB.B+11      ;FT
85 006036 113737 002334 036563  MOVB    LT,DPB.C+11      ;LT
86 006044 013737 002324 036542  MOV     FC,DPB.B+12      ;FC
87 006052 013737 002326 036564  MOV     LC,DPB.C+12      ;LC
92 006060 012737 006110 001350  MOV     #EXIT1,BYPASS    ;GO TO EXIT1 ON ERROR
    006066 012737 006074 001122  MOV     #TEST1,$LPADR    ;SETUP LOOP ADDRESS
    006074

93 006074 012706 001100      MOV     #STACK,SP       ;SET THE STACK PCINTER
94 006100 004037 024466      JSR     RO,CALL.C       ;GO EXECUTE THE COMMAND
95 006104 004037 024244      JSR     RO,CALL.B       ;GO EXECUTE THE COMMAND
96 006110 000004      EXIT1: SCOPE           ;CALL SCOPE ROUTINE
97
108
109

```

```

*****
;*TEST 2      INCREMENT/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
;*ENSURE PROPER OPERATION.
*****
TST2:

```

```

006112 000240
006112 033737 001564 001332  NOP
006114 001002 006354 001332  BIT     BITS+<2*2>,$TSTNMS ;DO THIS TEST?
006122 000137 006354 001332  BNE    +6                ;BR IF YES
006124 000137 006354 001332  JMP     TST3              ;NO--JUMP TO TEST3

006130 012737 000002 001116  MOV     #2,$TSTNM        ;SET TEST #2 AND CLEAR ($ERFLG)
006136 004737 023734 001116  JSR     PC,LODPRM        ;LOAD THE PARMETERS FOR THE TEST
006142 012737 006246 001124  MOV     #TEST2,$LPERR    ;SETUP THE LOOP ON ERROR ADDRESS
006150 013737 002322 001220  MOV     RPT,$TIMES       ;GET THE ITERATION COUNT
006156 112737 000031 001131  MOVB   #25,$ERMAX        ;MAX ERRORS ALLOWED FOR TEST
006164 012737 000002 001240  MOV     #2,$TESTN        ;SET TEST NUMBER IN APT MAIL BOX

110
006172 032777 010000 172754  BIT     #SW12,$SWR        ;INHIBIT TYPING TEST NUMBER ?
006200 001406 001000 172754  BEQ    +16                ;BR IF YES
006202 104401 037265 001000  TYPE   ,MSGTST           ;TYPE 'TEST'
006206 013746 001240 001000  MOV     $TESTN,-(SP)     ;;SAVE $TESTN FOR TYPEOUT
006212 104403 001000 001000  TYPCLS                    ;;GO TYPE--OCTAL ASCII
006214 002                .BYTE 2                    ;;TYPE 2 DIGIT(S)
006215 000                .BYTE 0                    ;;SUPPRESS LEADING ZEROS

111 006216 012737 006224 001122  MOV     #1,$,$LPADR      ;SETUP LOOP ADDRESS
112 006224 113737 002340 036540 1$:  MOVB   FS,DPB.B+10      ;FS
113 006232 113737 002332 036541  MOVB   FT,DPB.B+11      ;FT
117 006240 012737 006352 001350  MOV     #EXIT2,BYPASS    ;GO TO EXIT2 ON ERROR

```



```

T2 INCREMENT/SEEK TEST
006246
118 006246 013737 002324 036542
119 006254 012737 006254 001124
006262 012706 001100
120 006266
121 006266 004037 024244
122 006272 063737 002330 036542
123 006300 023737 002326 036542
124 006306 002367
125 006310 013737 002326 036542
126 006316 012737 006316 001124
006324 012706 001100
127 006330
128 006330 004037 024244
129 006334 163737 002330 036542
130 006342 023737 002324 036542
131 006350 003767
132 006352 000004
133
140
141

```

```

TEST2:
MOV FC,DPB.B+12 ;FC
MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,SP ;LOAD THE STACK POINTER

INCSK:
JSR RO,CALL.B ;GO EXECUTE THE COMM'ND
ADD IC,DPB.B+12 ;MOVE TO NEXT CYLINDER
CMP LC,DPB.B+12 ;OUT OF CYLINDERS?
BGE INCSK ;NO--BRANCH
MOV LC,DPB.B+12
MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,SP ;LOAD THE STACK POINTER

DECSK:
JSR RO,CALL.B ;GO EXECUTE THE COMMAND
SUB IC,DPB.B+12
CMP FC,DPB.B+12
BLE DECSK

EXIT2: SCOPE ;CALL SCOPE ROUTINE

```

```

:*****
:*TEST 3 STEPPING SEEK TEST
:*THIS TEST WILL COMMAND SEEK CYCLES TO CYLINDER 0, 1, 2, 4,
:*8, 16, 32, 64, 128, 256 AND 512. AT THE COMPLETION OF EACH
:*SEEK COMMAND THE PROPER INDICATORS ARE EXAMINED TO ENSURE
:*PROPER OPERATION.
:*****

```

```

TST3:
NOP
BIT BITS+<3*2>,TSTNMS ;DO THIS TEST?
BNE .+6 ;BR IF YES
JMP TST4 ;NO--JUMP TO TEST4

006354
006354 000240
006356 033737 001566 001332
006364 001002
006366 000137 006574

006372 012737 000003 001116
006400 004737 023734
006404 012737 006510 001124
006412 013737 002322 001220
006420 112737 000031 001131
006426 012737 000003 001240

```

```

142 006434 032777 010000 172512
006442 001406
006444 104401 037265
006450 013746 001240
006454 104403
006456 002
006457 000

BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
BEQ .+16 ;BR IF YES
TYPE MSGTST ;TYPE 'TEST'
MOV $TESTN,-(SP) ;SAVE $TESTN FOR TYPEOUT
TYPOS ;GO TYPE--OCTAL ASCII
.BYTE 2 ;TYPE 2 DIGIT(S)
.BYTE 0 ;SUPPRESS LEADING ZEROS

```

```

143 006460 012737 006466 001122
144 006466 113737 002340 036540
145 006474 113737 002332 036541
149 006502 012737 006572 001350
006510

1$:
MOV #1,$LPADR ;SETUP TEST LOOP ADDRESS
MOVB FS,DPB.B+10 ;FS
MOVB FT,DPB.B+11 ;FT
MOV #EXIT3,BYPASS ;GO TO BYPASS ON ERROR

TEST3:
MOV FC,DPB.B+12 ;FC
MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,SP ;LOAD THE STACK POINTER
JSR RO,CALL.B ;GO EXECUTE THE COMMAND

```

```

153 006534 013701 002330      MOV      IC,R1          :CYLINDER 1
154 006540 012737 006540 001124  MOV      #,$SLPERR     :SETUP THE ERROR LOOP ADDRESS
      006546 012706 001100      MOV      #STACK,SP    :LOAD THE STACK POINTER
155 006552 010137 036542      1$:     MOV      R1,DPB.B+12 :DESIRED CYLINDER
156 006556 004037 024244      JSR      R0,CALL.B     :GO EXECUTE THE COMMAND
157 006562 006301          ASL      R1            :MOVE TO NEXT CYLINDER
158 006564 020137 002326      CMP      R1,LC        :DONE?
159 006570 003770          BLE     1$           :NO--LOOP
160 006572 000004      EXIT3:  SCOPE        :CALL SCOPE ROUTINE
161
169
170
    
```

```

*****
*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 ENSURE PROPER OPERATION.
*****
TST4:
    
```

```

006574 000240
006574 033737 001570 001332      NOP
006576 033737 001570 001332      BIT      BITS+<4*2>,TSTNMS :DO THIS TEST?
006604 001002          BNE     +6           :BR IF YES
006606 000137 007206          JMP     TST5        :NO--JUMP TO TESTS

006612 012737 000004 001116      MOV      #4,$TSTNM     :SET TEST #4 AND CLEAR ($ERFLG)
006620 004737 023734          JSR     PC,LODPRM    :LOAD THE PARAMETERS FOR THE TEST
006624 012737 006744 001124      MOV      #TEST4,$LPERR :SETUP THE LOOP ON ERROR ADDRESS
006632 013737 002322 001220      MOV      RPT,$TIMES   :GET THE ITERATION COUNT
006640 112737 000031 001131      MOV     #25,$SERMAX   :MAX ERRORS ALLOWED FOR TEST
006646 012737 000004 001240      MOV     #4,$TESTN    :SET TEST NUMBER IN APT MAIL BOX

171 006654 032777 010000 172272      BIT     #SW12,$SWR    :INHIBIT TYPING TEST NUMBER ?
006662 001406          BEQ     +16         :BR IF YES
006664 104401 037265          TYPE   ,MSGTST      :TYPE 'TEST'
006670 013746 001240          MOV     $TESTN,-(SP) :SAVE $TESTN FOR TYPEOUT
006674 104403          TYPOS :GO TYPE--OCTAL ASCII
006676 002          .BYTE 2           :TYPE 2 DIGIT(S)
006677 000          .BYTE 0           :SUPPRESS LEADING ZEROS

172 006700 012737 006706 001122      MOV     #1,$SLPADR   :SETUP LOOP ADDRESS
173 006706 113737 002340 036540 1$:     MOV     FS,DPB.B+10  :FS
174 006714 113737 002332 036541      MOV     FT,DPB.B+11  :FT
182 006722 012737 007204 001350      MOV     #EXIT4,BYPASS :GO TO EXIT4 ON ERROR
006730 005002          CLR     R2          :CLEAR STALL SWITCH (NO STALL)
006732 032737 010000 001314      BIT     #SW12,C.SWR  :STALL REQUIRED?
006740 001401          BEQ     TEST4      :NO--BRANCH
006742 005102          COM     R2        :YES--SET SWITCH
006744          TEST4:

183 006744 013701 002324          MOV     FC,R1       :SET NC TO FC
184 006750 005037 001436          CLR     STALLO     :START AT ZERO IF STALLS REQUIRED
185 006754 012737 006754 001124      MOV     #,$SLPFR    :SETUP THE ERROR LOOP ADDRESS
      006762 012706 001100      MOV     #STACK,SP   :LOAD THE STACK POINTER
186 006766 010137 036542      1$:     MOV     R1,DPB.B+12 :NC
187 006772 004037 024244      JSR     R0,CALL.B   :GO EXECUTE THE COMMAND
188 006776 005702          TST     R2         :STALL?
189 007000 001403          BEQ     2$         :NO--BRANCH
    
```

```

190 007002 004037 025220 JSR R0,STALL ;YES-GO TO STALL ROUTINE
191 007006 001436 .WORD STALLO ;TIME POINTER
192 007010 013737 002324 036542 2$: MOV FC,DPB.B+12 ;FC
193 007016 004037 024244 JSR R0,CALL.B ;GO EXECUTE THE COMMAND
194 007022 005702 TST R2 ;STALL?
195 007024 001413 BEQ 3$ ;NO--BRANCH
196 007026 004037 025220 JSR R0,STALL ;YES--GO TO STALL ROUTINE
197 007032 001436 .WORD STALLO ;TIME POINTER
198 007034 005237 001436 INC STALLO ;UPDATE THE TIME
199 007040 023737 001462 001436 CMP MXSTAL,STALLO ;TIME TO BIG?
200 007046 003347 BGT 1$ ;NO--BRANCH
201 007050 005037 001436 CLR STALLO ;YES--START OVER AT ZERO
202 007054 063701 002330 3$: ADD IC,R1 ;MOVE TO NEXT CYLINDER
203 007060 020137 002326 CMP R1,LC ;LAST CYLINDER COMPLETED?
204 007064 003740 BLE 1$ ;NO--BRANCH
205 007066 013701 002326 MOV LC,R1 ;SET NC TO LC
206 007072 012737 007072 001124 MOV #,$LPERR ;SETUP THE ERROR LOOP ADDRESS
MOV #STACK,SP ;LOAD THE STACK POINTER
207 007104 010137 036542 4$: MOV R1,DPB.B+12 ;NC
208 007110 004037 024244 JSR R0,CALL.B ;GO EXECUTE THE COMMAND
209 007114 005702 TST R2 ;STALL?
210 007116 001403 BEQ 5$ ;NO--BRANCH
211 007120 004037 025220 JSR R0,STALL ;YES--GO TO STALL ROUTINE
212 007124 001436 .WORD STALLO ;TIME POINTER
213 007126 013737 002326 036542 5$: MOV LC,DPB.B+12 ;LC
214 007134 004037 024244 JSR R0,CALL.B ;GO EXECUTE THE COMMAND
215 007140 005702 TST R2 ;STALL?
216 007142 001413 BEQ 6$ ;NO--BRANCH
217 007144 004037 025220 JSR R0,STALL ;YES--GO TO STALL ROUTINE
218 007150 001436 .WORD STALLO ;TIME POINTER
219 007152 005237 001436 INC STALLO ;UPDATE STALL TIME
220 007156 023737 001462 001436 CMP MXSTAL,STALLO ;TIME TOO BIG?
221 007164 003347 BGT 4$ ;NO--BRANCH
222 007166 005037 001436 CLR STALLO ;YES--SET STALL TIME BACK TO ZERO
223 007172 163701 002330 6$: SUB IC,R1 ;NEXT CYLINDER
224 007176 020137 002324 CMP R1,FC ;DONE?
225 007202 002340 BGE 4$ ;NO--BRANCH
226 007204 000004 EXIT4: SCOPE ;CALL SCOPE ROUTINE
227
228
229

```

```

*****
*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
*ENSURE PROPER OPERATION. 'NC1' AND 'NC2' DEFAULT TO
*'FC' AND 'LC' RESPECTIVELY.
*****
TST5:

```

```

007206
007206 000240 NOP
007210 033737 001572 001332 BIT BITS+<5*2>,TSTNMS ;DO THIS TEST?
007216 001002 BNE +6 ;BR IF YES
007220 000137 007432 JMP TST6 ;NO--JUMP TO TEST6

```

```

007224 012737 000005 001116      MOV      #5,$STSTNM      ;SET TEST #5 AND CLEAR ($ERFLG)
007232 004737 023734                JSR      PC,LODPRM      ;LOAD THE PARAMETERS FOR THE TEST
007236 012737 007342 001124      MOV      #TEST5,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
007244 013737 002322 001220      MOV      RPT,$TIMES     ;GET THE ITERATION COUNT
007252 112737 000031 001131      MOV      #25,$SERMAX    ;MAX ERRORS ALLOWED FOR TEST
007260 012737 000005 001240      MOV      #5,$TESTN      ;:SET TEST NUMBER IN APT MAIL BOX

240
007266 032777 010000 171660      BIT      #SW12,@SWR     ;INHIBIT TYPING TEST NUMBER ?
007274 001406                BEQ      .+16           ;BR IF YES
007276 104401 037265                TYPE    ,MSGTST        ;TYPE 'TEST'
007302 013746 001240                MOV      $TESTN,-(SP)   ;:SAVE $TESTN FOR TYPEOUT
007306 104403                TYPOS                    ;:GO TYPE--OCTAL ASCII
007310 002                          .BYTE   2              ;:TYPE 2 DIGIT(S)
007311 000                          .BYTE   0              ;:SUPPRESS LEADING ZEROS

241 007312 012737 007320 001122      MOV      #1$, $LPADR    ;SETUP LOOP ADDRESS
242 007320 113737 002340 036540 1$:  MOV      FS,DPB.B+10    ;FS
243 007326 113737 002332 036541      MOV      FT,DPB.B+11    ;FT
247 007334 012737 007430 001350      MOV      #EXIT5,BYPASS ;GO TO EXITS ON ERROR

TEST5:
248 007342 013701 002324                MOV      FC,R1          ;START NC1 AT FC
249 007346 013702 002326                MOV      LC,R2          ;START NC2 AT LC
250 007352 012737 007352 001124      MOV      #,$SLPERR      ;SETUP THE ERROR LOOP ADDRESS
007360 012706 001100                MOV      #STACK,SP      ;LOAD THE STACK POINTER
251 007364 010137 036542 1$:  MOV      R1,DPB.B+12    ;NC1
252 007370 004037 024244                JSR      R0,CALL.B       ;GO EXECUTE THE COMMAND
253 007374 010237 036542                MOV      R2,DPB.B+12    ;NC2
254 007400 004037 024244                JSR      R0,CALL.B       ;GO EXECUTE THE COMMAND
255 007404 063701 002330                ADD      IC,R1          ;NEXT NC1
256 007410 163702 002330                SUB      IC,R2          ;NEXT NC2
257 007414 020137 002326                CMP      R1,LC          ;DONE?
260 007420 003003                BGT      EXIT5         ;YES--BRANCH
261 007422 020237 002324                CMP      R2,FC          ;?
262 007426 002356                BGE     1$             ;NO--BRANCH
263 007430 000004                EXIT5: SCOPE          ;CALL SCOPE ROUTINE
264
273
274
    
```

```

:*****
:*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 ENSURE PROPER OPERATION.
:*****
TST6:
    
```

```

007432
007432 000240
007434 033737 001574 001332      NOP
007442 001002                BIT      BITS+<6*2>,$TSTNMS ;DO THIS TEST?
007444 000137 007722                BNE     .+6           ;BR IF YES
                                JMP      TST7          ;NO--JUMP TO TEST7

007450 012737 000006 001116      MOV      #6,$STSTNM    ;SET TEST #6 AND CLEAR ($ERFLG)
007456 004737 023734                JSR      PC,LODPRM      ;LOAD THE PARAMETERS FOR THE TEST
007462 012737 007566 001124      MOV      #TEST6,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
007470 013737 002322 001220      MOV      RPT,$TIMES     ;GET THE ITERATION COUNT
007476 112737 000031 001131      MOV      #25,$SERMAX    ;MAX ERRORS ALLOWED FOR TEST
    
```

```

275 007504 012737 000006 001240      MOV      #6,$TESTH      ;;SET TEST NUMBER IN APT MAIL BOX
007512 032777 010000 171434      BIT      #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
007520 001406                      BEQ      .+16            ;BR IF YES
007522 104401 037265                      TYPE     'MSGTST'        ;TYPE 'TEST'
007526 013746 001240      MOV      $TESTN,-(SP)    ;;SAVE $TESTN FOR TYPECUT
007532 104403                      TYPOS    ;GO TYPE--OCTAL ASC: I
007534 002                      .BYTE   2              ;;TYPE 2 DIGIT(S)
007535 000                      .BYTE   0              ;;SUPPRESS LEADING ZEROS

276 007536 012737 007544 001122      MOV      #1$, $LPADR    ;SETUP LOOP ADDRESS
277 007544 113737 002340 036540      1$:     MOV      FS,DPB.B+10 ;FS
278 007552 113737 002332 036541      MOV      FT,DPB.B+11    ;FT
282 007560 012737 007720 001350      MOV      #EXIT6,BYPASS  ;GO TO EXIT6 ON ERROR

TEST6:
283 007566 013701 002324      MOV      FC,R1          ;PICKUP 'FC'
284 007572 013702 002326      MOV      LC,R2          ;FORM LAST CYLINDER THAT
285 007576 162702 000005                      SUB      #5,R2          ;IS AVAILABLE FOR TESTING
286 007602 012737 007602 001124      MOV      #,$LPERR       ;SETUP THE ERROR LOOP ADDRESS
007610 012706 001100      MOV      #STACK,SP      ;LOAD THE STACK POINTER
287 007614 020102      1$:     CMP      R1,R2        ;LAST CYLINDER
290 007616 003040                      BGT      EXIT6         ;YES--BRANCH
291 007620 010137 036542      MOV      R1,DPB.B+12    ;NC
292 007624 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
293 007630 062737 000004 036542      ADD      #4,DPB.B+12    ;NC+4
294 007636 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
295 007642 162737 000003 036542      SUB      #3,DPB.B+12    ;NC+1
296 007650 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
297 007654 062737 000002 036542      ADD      #2,DPB.B+12    ;NC+3
298 007662 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
299 007666 162737 000001 036542      SUB      #1,DPB.B+12    ;NC+2
300 007674 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
301 007700 062737 000003 036542      ADD      #3,DPB.B+12    ;NC+5
302 007706 004037 024244      JSR      R0,CALL.B      ;GO EXECUTE THE COMMAND
303 007712 063701 002330      ADD      IC,R1
304 007716 000736                      BR       1$
305 007720 000004      EXIT6:  SCOPE          ;CALL SCOPE ROUTINE
306
315
316

```

```

*****
;*TEST 7      RANDOM SEEK TEST
;*THIS TEST PERFORMS RANDOM SEEK OPERATIONS BETWEEN CYLINDERS 'FC'
;*AND 'LC'.  AFTER EACH SEEK, THE POSITION OF THE DRIVE IS VERIFIED BY
;*READING A SECTOR FROM THE CURRENTLY ADDRESSED CYLINDER AND TRACK.
;*THE TRACK ADDRESS IS INCREMENTED FOR EACH SEEK SO THAT VERIFICATION
;*OF POSITIONING OCCURS USING EACH HEAD.  TRACK ADDRESSES ARE INCREMENTED
;*BETWEEN PARAMETERS 'FT' AND 'LT'.
*****

```

```

007722 000240
007722 033737 001576 001332      NOP
007724 001002                      BIT      BITS+<7*2>,$TSTNMS ;DO THIS TEST?
007732 000137 010320                      BNE     .+6            ;BR IF YES
007734 000137 010320                      JMP     TST10          ;NO--JUMP TO TEST10

007740 012737 000007 001116      MOV      #7,$TSTNM      ;SET TEST #7 AND CLEAR ($ERFLG)
007746 004737 023734                      JSR      PC,LODPRM      ;LOAD THE PARAMETERS FOR THE TEST
007752 012737 010056 001124      MOV      #TEST7,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS

```

```
007760 013737 002322 001220      MOV      RPT,$TIMES      ;GET THE ITERATION COUNT
007766 112737 000031 001131      MOV     #25,$ERMAX      ;MAX ERRORS ALLOWED FOR TEST
007774 012737 000007 001240      MOV     #7,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

317 010002 032777 010000 171144      BIT     #SW12,@SWR      ;INHIBIT TYPING TEST NUMBER ?
010010 001406                      BEQ     .+16            ;BR IF YES
010012 104401 037265                      TYPE   $MSGTST         ;TYPE 'TEST'
010016 013746 001240                      MOV     $TESTN,-(SP)   ;;SAVE $TESTN FOR TYPEOUT
010022 104403                      TYPOS  ;GO TYPE--OCTAL ASCII
010024 002                          .BYTE  2              ;;TYPE 2 DIGIT(S)
010025 000                          .BYTE  0              ;;SUPPRESS LEADING ZEROS

318 010026 113737 002332 036541      MOV     FT,DPB.B+11    ;LOAD STARTING TRACK ADDRESS
319 010034 112737 000105 036510      MOV     #SEEK,DPB.A+2 ;SEEK=COMMAND
324 010042 012737 010316 001350      MOV     #EXIT7,BYPASS ;ERROR TERMINATION ADDRESS
010050 012737 010056 001122      MOV     #TEST7,$LPADR ;SETUP THE LOOP ON TEST ADDRESS
010056 013704 031366                      MOV     RMADR,R4      ;UNIBUS ADDRESS OF THE RH/RM
325 010062 012706 001100      MOV     #STACK,SP     ;SETUP THE STACK POINTER
326 010066 013737 002324 036542      MOV     FC,DPB.B+12   ;INITIAL CYLINDER ADDRESS
327 010074 023737 002324 002326      CMP     FC,LC          ;CYLINDER LIMITS THE SAME ?
328 010102 001422                      BEQ     1$            ;BR IF THEY ARE
329 010104 004737 022502      JSR     PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
330 010110 013746 022600      MOV     $HNUM,-(SP)   ;USE THE HIGH RANDOM NUMBER
331 010114 005046                      CLR     -(SP)         ;UPPER DIVIDEND
332 010116 013746 002326      MOV     LC,-(SP)      ;FORM THE DIVISOR
333 010122 005216                      INC     (SP)          ;INCREMENT
334 010124 163716 002324      SUB     FC,(SP)       ;SUBTRACT THE LOWER LIMIT
335 010130 004737 022604      JSR     PC,$DIV       ;DIVIDE
336 010134 062637 036542      ADD     (SP)+,DPB.B+12 ;ADD THE REMAINDER TO THE INITIAL CYLINDER
337 010140 005726                      TST    (SP)+         ;DISCARD THE QUOTENT
338 010142 013737 036542 036520      MOV     DPB.B+12,DPB.A+12 ;COPY NEW CYLINDER ADDRESS
339 010150                      1$:
010150 012737 010150 001124      MOV     #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
010156 012706 001100      MOV     #STACK,SP     ;LOAD THE STACK POINTER
340 010162 004037 024076      JSR     R0,CALL.A     ;GO EXECUTE THE COMMAND
341 010166 012737 010166 001124      MOV     #,$LPERR      ;SETUP THE ERROR LOOP ADDRESS
010174 012706 001100      MOV     #STACK,SP     ;LOAD THE STACK POINTER
342 010200 113764 036506 000010      MOV     DPB.A,RMCS2(R4) ;SELECT THE DRIVE
343 010206 016446 000020      MOV     RMLA(R4),-(SP) ;GET THE LOOK AHEAD REGISTER
344 010212 006316                      ASL    (SP)           ;ALIGN THE SECTOR ADDRESS
345 010214 006316                      ASL    (SP)           ;ALIGN THE SECTOR ADDRESS
346 010216 000316                      SWAB   'SP)          ;PUT ADDRESS IN LOWER BYTE
347 010220 105766 000001      TST    1(SP)         ;IN THE 1ST 20% OF SECTOR ?
348 010224 001401                      BEQ    2$            ;BR IF YES
349 010226 105216                      INCB   (SP)          ;INCREMENT THE SECTOR ADDRESS
350 010230 105216                      INCB   (SP)          ;INCREMENT THE SECTOR ADDRESS
351 010232 112637 036604      MOV     (SP)+,DTADPB+10 ;LOAD THE DPB
352 010236 013746 002426      MOV     PRMLM+16,-(SP) ;PUT LAST SECTOR ADDRESS ON THE STACK
353 010242 005216                      INC    (SP)          ;INCREMENT IT
354 010244 122637 036604      CMP     (SP)+,DTADPB+10 ;NEW SECTOR ADDRESS TOO LARGE ?
355 010250 103007                      BHS   4$            ;BR IF NOT
356 010252 103403                      BLO   3$            ;BR IF ADDRESS IS 2 GREATER
357 010254 105037 036604      CLRB   DTADPB+10     ;RESET TO SECTOR ADDRESS 0
358 010260 000403                      BR    4$            ;CONTINUE
359 010262 112737 000001 036604 3$:
360 010270 004037 024244 4$:
010270                      MOV     #1,DTADPB+10 ;RESET ADDRESS TO SECTOR 1
010270                      JSR     R0,CALL.B     ;GO EXECUTE THE COMMAND
```



```

361 010274 105237 036541 INCB DPB.B+11 ;INCREMENT THE TRACK ADDRESS
362 010300 123737 036541 002334 CMPB DPB.B+11.LT ;MAXIMUM ?
365 010306 101403 BLOS EXIT7 ;BR IF NOT
366 010310 113737 002332 036541 MOVB FT,DPB.B+11 ;RELOAD STARTING TRACK ADDRESS
367 010316 000004 EXIT7: SCOPE ;CALL SCOPE ROUTINE
368
379
380
  
```

```

:*****
:*TEST 10 ALL SEEKS TEST
:*THIS TEST VERIFIES THAT THE DISK DRIVE CAN SEEK FROM EACH CYLINDER
:*TO ALL OTHER CYLINDERS.
:*
:*BEGINNING WITH CYLINDER 'FC', THE TEST SEEKS TO EACH CYLINDER
:*BETWEEN 'FC' AND 'LC' FROM CYLINDER 'FC'. THE BEGINNING CYLINDER
:*ADDRESS IS INCREMENTED AND THE TEST SEEKS BETWEEN THE NEW CYLINDER
:*ADDRESS AND ALL CYLINDERS BETWEEN 'FC' AND 'LC'. THE SEQUENCE
:*CONTINUES UNTIL ALL CYLINDERS HAVE BEEN CHECKED.
:*****
TST10:
  
```

```

010320
010320 000240 NOP
010322 033737 001600 001332 BIT BITS+<10*2>,TSTNMS ;DO THIS TEST?
010330 001002 BNE +6 ;BR IF YES
010332 000137 010564 JMP TST11 ;NO--JUMP TO TEST11

010336 012737 000010 001116 MOV #10,$TSTNM ;SET TEST #10 AND CLEAR ($ERFLG)
010344 004737 023734 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
010350 012737 010504 001124 MOV #TEST10,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
010356 013737 002322 001220 MOV RPT,$TIMES ;GET THE ITERATION COUNT
010364 012737 000031 001131 MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
010372 012737 000010 001240 MOV #10,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

381 010400 032777 010000 170546 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
010406 001406 BEQ +16 ;BR IF YES
010410 104401 037265 TYPE ,MSGTST ;TYPE 'TEST'
010414 013746 001240 MOV $TESTN,-(SP) ;;SAVE $TESTN FOR TYPEOUT
010420 104403 TYPOS ;GO TYPE--OCTAL ASCII
010422 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
010423 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS

382 010424 012737 010432 001122 MOV #1$, $LPADR ;SETUP THE LOOP ADDRESS
383 010432 113737 002340 036540 1$: MOVB FS,DPB.B+10 ;SECTOR ADDRESS
384 010440 113737 002340 036562 MOVB FS,DPB.C+10 ;SECTOR ADDRESS
385 010446 113737 002332 036541 MOVB FT,DPB.B+11 ;TRACK ADDRESS
386 010454 113737 002332 036563 MOVB FT,DPB.C+11 ;TRACK ADDRESS
387 010462 013737 002324 036542 MOV FC,DPB.B+12 ;STARTING CYLINDER ADDRESS
388 010470 013737 002324 036564 MOV FC,DPB.C+12 ;STARTING CYLINDER ADDRESS
392 010476 012737 010562 001350 MOV #EXIT10,BYPASS ;TEST ABORT EXIT

TEST10:
393 010504 012706 001100 MOV #STACK,SP ;SETUP THE STACK POINTER
394 010510 1$:
010510 004037 024466 JSR RC,CALL.C ;GO EXECUTE THE COMMAND
395 010514 004037 024244 JSR RO,CALL.B ;GO EXECUTE THE COMMAND
396 010520 063737 002330 036564 ADD IC,DPB.C+12 ;INCREMENT THE ENDING CYLINDER ADDRESS
397 010526 023737 002326 036564 CMP LC,DPB.C+12 ;CHECK IF EXCEEDING MAXIMUM
398 010534 002365 BGE 1$ ;BR IF NOT
399 010536 013737 002324 036564 MOV FC,DPB.C+12 ;RESET ENDING CYLINDER ADDRESS
400 010544 063737 002330 036542 ADD IC,DPB.B+12 ;INCREMENT THE STARTING ADDRESS
  
```

401	010552	023737	002326	036542	CMP	LC.DPB.B+12	: EXCEEDING MAXIMUM ?
402	010560	002353			BGE	1\$	: BR IF NOT
403	010562	000004			EXIT10: SCOPE		: CALL SCOPE ROUTINE



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
22  
23

.SBTTL TIMING TESTS

```

://////
:*THE TIMING TESTS WILL ENSURE THAT THOSE FUNCTIONS BEING
:*TIMED ARE WITHIN THE TOLERANCES SPECIFIED IN THE 'RM80
:*ENGINEERING SPECIFICATIONS'.
:*THE SEEK TIMING WILL BE PERFORMED USING EXPLICIT SEEK
:*OPERATIONS. AT THE COMPLETION OF EACH OF THE TIMING
:*TESTS THE MINIMUM, MAXIMUM AND AVERAGE TIMES WILL BE
:*TYPED.
://////
    
```

```

:*****
:*TEST 11 ROTATIONAL SPEED TIMING TEST
:*THIS TEST WILL START A SEARCH TO CYLINDER 'FC', TRACK 'FT',
:*SECTOR 'FS'. AS SOON AS THE INTERRUPT OCCURS, THE GO BIT
:*IS SET AGAIN AND THE OPERATION IS TIMED. THIS PROCEDURE
:*IS REPEATED 10 TIMES THEN THE AVERAGE TIME IS CALCULATED
:*AND CHECKED TO ENSURE IT IS WITHIN TOLERANCE:
:* LO LIMIT (16.67MS - 4.3%) = 15.97MS @ 50/60 HZ.
:* HI LIMIT (16.67MS + 3.5%) = 17.27MS @ 50/60 HZ.
:*****
TST11:
    
```

010564	000240			NOP				
010564	033737	001602	001332	BIT	BITS+<11*2>,TSTNMS	:DO THIS TEST?		
010574	001002			BNE	+6	:BR IF YES		
010576	000137	011426		JMP	TST12	:NO--JUMP TO TEST12		
010602	012737	000011	001116	MOV	#11,\$TSTNM	:SET TEST #11 AND CLEAR (SERFLG)		
010610	004737	023734		JSR	PC,LODPRM	:LOAD THE PARMETERS FOR THE TEST		
010614	012737	011102	001124	MOV	#TEST11,\$LPERR	:SETUP THE LOOP ON ERROR ADDRESS		
010622	013737	002322	001220	MOV	RPT,\$TIMES	:GET THE ITERATION COUNT		
010630	112737	000031	001131	MOVB	#25,\$ERMAX	:MAX ERRORS ALLOWED FOR TEST		
010636	012737	000011	001240	MOV	#11,\$TESTN	::SET TEST NUMBER IN APT MAIL BOX		
24	010644	032777	010000	170302	BIT	#SW12,@SWR	:INHIBIT TYPING TEST NUMBER ?	
	010652	001406			BEQ	+16	:BR IF YES	
	010654	104401	037265		TYPE	.MSGTST	:TYPE 'TEST'	
	010660	013746	001240		MOV	\$TESTN,-(SP)	::SAVE \$TESTN FOR TYPEOUT	
	010664	104403			TYPOS		::GO TYPE--OCTAL ASCII	
	010666	003			.BYTE	3	::TYPE 3 DIGIT(S)	
	010667	000			.BYTE	0	::SUPPRESS LEADING ZEROS	
25	010670	005737	001342		TST	CLKSTA	:KW11-P CLOCK?	
26	010674	003002			BGT	1\$	:YES--START TEST	
29	010676	000137	011424		JMP	EXIT11	:NO--JUMP TO EXIT11	
30	010702	012737	010702	001122	1\$:	MOV	#,\$LPADR	:SETUP LOOP ADDRESS
31	010710	004037	025500		JSR	RO,SRCHOO	:DO A MASSBUS INIT & RECAL	
32	010714	000402			BR	2\$	:RETURN HERE IF NO ERROR	
54	010716	000137	011424		JMP	EXIT11	:RETURN HERE IF ERROR	
	010722	012737	010722	001122	2\$:	MOV	#,\$LPADR	:ERROR LOOP ADDRESS
	010730	112737	000105	036510		MOVB	#SEEK,DPB.A+2	:SEEK=COMMAND
	010736	005037	036516		CLR	DPB.A+10	:USE TRACK 0 & SECTOR 0	
	010742	013737	002324	036520		MOV	FC,DPB.A+12	:STARTING CYLINDER
	010750	012737	011424	001350		MOV	#EXIT11,BYPASS	:GO TO EXIT11 IF ERROR
	010756	004037	024076		JSR	RO,CALL.A	:GO EXECUTE THE COMMAND	

```

010762 013764 002324 000034      MOV      FC,RMDC(R4)      ;FC
010770 013746 002340              MOV      FS,-(SP)        ;FS
010774 113766 002332 000001      MOVVB   FT,1(SP)        ;FT
011002 012664 000006              MOV      (SP)+,RMDA(R4) ;LOAD FT/FS
011006 012737 011424 001222      MOV      #EXIT11,$ESCAPE ;ESCAPE TO EXIT11 ON ERROR
011014 005005              CLR      R5              ;COUNT UP

55                                     ;SETUP PARAMETER TABLE FOR RM80
56 011016 010046              MOV      R0,-(SP)        ;SAVE R0
57 011020 113766 036574 001314      MOVVB   DTADPB,R0       ;DRIVE ADDRESS
58 011024 032757 000100              BIT      #SW06,C.SWR    ;CHECK CONTROL SWR FOR 60 HZ.
59 011032 001411              BEQ      3$              ;BR IF YES
71 011034 012737 001650 011414      MOV      #T11B,R3       ;LOAD 50 HZ. TABLE FOR RM80
011040 012737 001650 011414      MOV      #T11B,TP50
011046 012737 001716 011422      MOV      #SP11B,TPS50
011054 000410              BR       4$              ;EXIT
011056 012737 001640 011376 3$:    MOV      #T11A,TP60     ;LOAD 60 HZ. TABLE FOR RM80
011064 012703 001640              MOV      #T11A,R3
011070 012737 001710 011404      MOV      #SP11A,TPS60
011076 012600 4$:              MOV      (SP)+,R0       ;RESTORE R0
011100 000240              NOP                       ;EXIT
011102                                TEST11:

72 011102 012706 001100              MOV      #STACK,SP      ;SETUP STACK
73 011106 012701 000012              MOV      #10,R1        ;TIME 10 SEARCHES
74 011112 004737 025664              JSR      PC,STRMR       ;INITIALIZE THE TIMERS
75 011116 012777 011304 170404      MOV      #4$,@PKV       ;SETUP VECTOR IN CASE OF OVERFLOW
76 011124 012777 025662 020236      MOV      #DORTI,@RMVEC ;SETUP RM80 VECTOR
77 011132 005077 170400 1$:      CLR      @PKB           ;START COUNTING AT ZERO
78 011136 012777 000131 170370      MOV      #131,@PKCS     ;INT.EN., COUNT UP AT 100K HZ RATE
79 011144 012714 000131              MOV      #SEARCH,(R4)   ;START A SEARCH
80 011150 000601              WAIT                       ;WAIT ON INTERRUPT
81 011152 042777 000101 170354      BIC      #101,@PKCS     ;STOP THE CLOCK
82 011160 032764 040600 000012      BIT      #ERR,RMDS(R4) ;ERROR?
83 011166 001411              BEQ      2$              ;NO--BRANCH
84 011170 104412              SAVREG                      ;SAVE R0-R5
011172 012702 036574              MOV      #DTADPB,R2    ;DPB POINTER
011176 004737 035744              JSR      PC,SVRH70     ;SAVE ALL THE RH/RM REGISTERS
011202 004737 023146              JSR      PC,CNTCLR     ;GO CLEAR CONTROLLER
011206 104413              RESREG                      ;RESTORE R0-R5
85 011210 104017              EMT      17
86 011212 005077 170320 2$:      CLR      @PKB           ;START THE COUNT AT ZERO
87 011216 012714 000131              MOV      #SEARCH,(R4)  ;START A SEARCH
88 011222 012777 000131 170304      MOV      #131,@PKCS    ;START THE CLOCK
89 011230 000001              WAIT                       ;WAIT ON INTERRUPT
90 011232 042777 000101 170274      BIC      #101,@PKCS     ;STOP THE CLOCK
91 011240 032764 040000 000012      BIT      #ERR,RMDS(R4) ;IS 'ERR=1'?
92 011246 001411              BEQ      3$              ;NO--BRANCH
93 011250 104412              SAVREG                      ;SAVE R0-R5
011252 012702 036574              MOV      #DTADPB,R2    ;DPB POINTER
011256 004737 035744              JSR      PC,SVRH70     ;SAVE ALL THE RH/RM REGISTERS
011262 004737 023146              JSR      PC,CNTCLR     ;GO CLEAR CONTROLLER
011266 104413              RESREG                      ;RESTORE R0-R5
94 011270 104017              EMT      17
95 011272 004737 025726 3$:      JSR      PC,COUNT      ;UPDATE THE COUNT
96 011276 005301              DEC      R1              ;DONE?
97 011300 003314              BGT     1$              ;NO--BRANCH
98 011302 000420              BR       5$              ;YES--GO TO THE EXIT
  
```

```

99 011304 042777 000101 170222 4$: BIC #101,@PKCS ;STOP THE CLOCK
100 011312 005037 177776 CLR PS ;DROP THE PRIORITY
101 011316 012600 MOV (SP)+,RO ;PC OF WAIT+2
102 011320 005726 TST (SP)+ ;POP THE PS FROM THE STACK
103 011322 104412 SAVREG ;SAVE R0-R5
011324 012702 036574 MOV #DTADPB,R2 ;DPB POINTER
011330 004737 035744 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
011334 004737 023146 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
011340 104413 RESREG ;RESTORE R0-R5
104 011342 104020 EMT 20
105 011344 004737 023146 5$: JSR PC,CNTCLR ;GO CLEAR MASSBUS CONTROLLER
106
107 011350 004737 023240 JSR PC,ST.CLK ;INITIALIZE THE CLUCK
108 011354 012777 033526 020006 MOV #ISR,@RMVEC ;RESTORE RH/RM INT. VECTOR
109 011362 032737 000100 001314 BIT #SW06,C.SWR ;60 HZ?
128 011370 001007 BNE EXIT.A ;NO-BRANCH
011372 004037 026166 JSR RO,TYPTIM ;TYPE THE TIMING
011376 001640 TP60: T11A ;TABLE ADDRESS
011400 004037 026060 JSR RO,SPTYP ;TYPE THE SPEC
011404 001710 TPS60: SP11A
011406 000406 BR EXIT11 ;EXIT

011410 004037 026166 EXIT.A: JSR RO,TYPTIM ;TYPE THE TIMING
011414 001650 TP50: T11B ;TABLE ADDRESS
011416 004037 026060 JSR RO,SPTYP ;
011422 001716 TPS50: SP11B ;
129 011424 000004 EXIT11: SCOPE ;CALL SCOPE ROUTINE
130
139
140
  
```

```

*****
:*TEST 12 ONE CYLINDER SEEK TIMING TEST
:*THIS TEST WILL COMMAND FORWARD SEEK CYCLES TO ADVANCE THE
:*CYLINDER BY ONE UNTIL THE INCREMENT IS GREATER THAN THE
:*CYLINDER 'LC', THEN REVERSE SEEK TO CYLINDER 'FC'. THE
:*TIME TO PERFORM EACH SEEK IS CHECKED TO ENSURE IT DOES NOT
:*EXCEED THE MAXIMUM TIME PERMITTED FOR A ONE CYLINDER SEEK.
:*MAXIMUM TIME IS 6.0 MS
*****
TST12:
  
```

```

011426
011426 000240
011430 033737 001604 001332 NOP ;DO THIS TEST?
011436 001002 BNE .+6 ;BR IF YES
011440 000137 012130 JMP TST13 ;NO--JUMP TO TEST13

011444 012737 000012 001116 MOV #12,$TSTNM ;SET TEST #12 AND CLEAR (SERFLG)
011452 004737 023734 JSR PC,LODPRM ;LOAD THE PARMETERS FOR THE TEST
011456 012737 011636 001124 MOV #TEST12,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
011464 013737 002322 001220 MOV RPT,$TIMES ;GET THE ITERATION COUNT
011472 112737 000031 001131 MOVB #25,$SERMAX ;MAX ERRORS ALLOWED FOR TEST
141 011500 012737 000012 001240 MOV #12,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX

011506 032777 010000 167440 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
011514 001406 BEQ .+16 ;BR IF YES
011516 104401 037265 TYPE ,MSGTST ;TYPE 'TEST'
011522 013746 001240 MOV $TESTN,-(SP) ;:SAVE $TESTN FOR TYPEOUT
011526 104403 TYPOS ;GO TYPE--ASCII
011530 003 .BYTE 3 ;:TYPE 3 DIGIT(S)
  
```

```

011531      000      .BYTE      0      ;;SUPPRESS LEADING ZEROS

142 011532  05737  001342      TST      CLKSTA      ;KW11-P CLOCK?
143 011536  003002      BGT      1$          ;YES--START TEST
146 011540  000137  012126      JMP      EXIT12      ;NO--JUMP TO EXIT12
147 011544  012737  011544  001122  1$:      MOV      #.,$LPADR    ;SETUP THE LOOP ADDRESS
148 011552  004037  025500      JSR      RO,SRCHOO   ;DO A MASSBUS INIT. AND RECAL
149 011556  000402      BR       2$          ;NO ERROR RETURN
161 011560  000137  012126      JMP      EXIT12      ;ERROR RETURN--SCOPE LOOP CALL
      011564  012737  011564  001122  2$:      MOV      #.,$LPADR    ;ERROR LOOP ADDRESS
      011572  112737  000105  036510      MOV      #SEEK,DPB.A+2 ;SEEK=COMMAND
      011600  005037  036516      CLR      DPB.A+10    ;USE TRACK 0 & SECTOR 0
      011604  013737  002324  036520      MOV      FC,DPB.A+12 ;STARTING CYLINDER
      011612  012737  012126  0C1350      MOV      #EXIT12,BYPASS ;GO TO EXIT12 IF ERROR
      011620  004037  024076      JSR      RO,CALL.A   ;GO EXECUTE THE COMMAND
      011624  012703  001660      MOV      #T12,R3     ;PARAMETER POINTER
      011630  012737  012126  001222      MOV      #EXIT12,$ESCAPE ;;ESCAPE TO EXIT12 ON ERROR
      011636      TEST12:
162 011636  012706  001100      MOV      #STACK,SP   ;SETUP STACK
163 011642  013737  002324  036606      MOV      FC,DTADPB+12 ;START WITH BEGINNING CYLINDER
164 011650  005237  036606      INC      DTADPB+12   ;INCREMENT THE BEGINNING CYLINDER
165 011654  005005      CLR      R5          ;SET THE UP/DOWN SWITCH TO UP
166 011656  004737  025664      JSR      PC,STRMR    ;INITIALIZE THE TIMERS
167 011662  012777  012034  167640      MOV      #5$,@PKV    ;SETUP INCASE OF OVERFLOW
168 011670  012777  025662  017472      MOV      #DORTI,@RMVEC ;SET RM80 VECTOR
169 011676  005077  167634      CLR      @PKB        ;START THE COUNTER AT ZERO
170 011702  013764  036606  000034      MOV      DTADPB+12,RMDC(R4) ;LOAD DESIRED CYLINDER
171 011710  012714  000105      MOV      #SEEK,(R4)  ;START A SEEK
172 011714  012777  000131  167612      MOV      #131,@PKCS ;START THE CLOCK
173 011722  000001      WAIT      ;WAIT ON INTERRUPT
174 011724  042777  000101  167602      BIC      #101,@PKCS  ;STOP THE CLOCK
175 011732  032764  040000  000012      BIT      #ERR,RMDS(R4) ;ANY DISK ERRORS?
176 011740  001411      BEQ      2$          ;NO--BRANCH
177 011742  104412      SAVREG    ;SAVE R0-R5
      011744  012702  036574      MOV      #DTADPB,R2  ;DPB POINTER
      011750  004737  035744      JSR      PC,SVRH70   ;SAVE ALL THE RH/RM REGISTERS
      011754  004737  023146      JSR      PC,CNTCLR   ;GO CLEAR CONTROLLER
      011760  104413      RESREG    ;RESTORE R0-R5
178 011762  104017      EMT      17
179 011764  004737  025726      JSR      PC,COUNT    ;COUNT THIS SEEKS TIME
180 011770  005705      TST      R5          ;UP OR DOWN?
181 011772  001011      BNE      4$          ;DOWN--BRANCH
182 011774  005237  036606      INC      DTADPB+12   ;MOVE TO NEXT CYLINDER
183 012000  023737  036606  002326      CMP      DTADPB+12,LC ;OUT OF CYLINDERS?
184 012006  002733      BLT      1$          ;NO--GO DO THE NEXT SFEK
185 012010  012705  177777      MOV      #-1,R5     ;SET UP/DOWN SWITCH TO DOWN
186 012014  000730      BR       1$          ;GO DO THE NEXT SEEK
187 012016  005337  036606      DEC      DTADPB+12   ;MOVE TO NEXT CYLINDER
188 012022  023737  036606  002324      CMP      DTADPB+12,FC ;OUT OF CYLINDERS?
189 012030  003322      BGT      1$          ;NO--GO DO THE NEXT SEEK
190 012032  000420      BR       6$          ;GO TO THE EXIT
191 012034  042777  000101  167472  5$:      BIC      #101,@PKCS  ;STOP THE CLOCK
192 012042  005037  177776      CLR      PS          ;DROP THE PRIORITY
193 012046  012600      MOV      (SP)+,R0    ;PC OF WAIT+2
194 012050  005726      TST      (SP)+      ;POP THE PS FROM THE STACK
195 012052  104412      SAVREG    ;SAVE R0-R5
      012054  012702  036574      MOV      #DTADPB,R2  ;DPB POINTER
  
```

```

012060 004737 035744 JSR PC,SVRH70 ;SAVE ALL THE RH/RM REGISTERS
012064 004737 023146 JSR PC,CNTCLR ;GO CLEAR CONTROLLER
012070 104413 RESREG ;RESTORE R0-R5
196 012072 104020 EMT 20
197 012074 004737 023146 6$: JSR PC,CNTCLR ;GO CLEAR MASSBUS CONTROLLER
198 012100 004737 023240 JSR PC,ST.CLK ;INITIALIZE THE CLOCK
199 012104 012777 033526 017256 MOV #ISR,ARMVEC ;RESTORE RH/RM INT. VECTOR
204 012112 004037 026166 JSR R0,TYPTIM ;GO TYPE THE TIMES
012116 001660 T12 ;POINTER
012120 004037 026060 JSR R0,SPTYP
012124 001724 SP12
205 012126 000004 EXIT12: SCOPE ;CALL SCOPE ROUTINE
206
220
221
  
```

```

:*****
:*TEST 13 AVERAGE SEEK TIMING TEST
:*THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO
:*CYLINDER 'LC'. THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
:*CYLINDER 'FC'. BOTH SEEKS ARE TIMED AND CHECKED TO ENSURE THEY
:*ARE WITHIN THE TOLERANCE ALLOWED FOR THE AVERAGE SEEK.
:*THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL OF 256 SEEKS).
:*MAXIMUM TIME IS 25.0 MS
:*
:* THERE ARE NO SPECIFICATIONS GIVEN FOR AN AVERAGE SEEK TIME
:* ON THIS PARTICULAR DRIVE. THEREFORE, THIS TEST SHOULD BE
:* USED FOR REFERENCE ONLY.
:*
:*****
  
```

TST13:

```

012130
012130 000240 NOP
012132 033737 001606 001332 BIT BITS+<13*2>,TSTNMS ;DO THIS TEST?
012140 001002 BNE +6 ;BR IF YES
012142 000137 012670 JMP TST14 ;NO--JUMP TO TEST14

012146 012737 000013 001116 MOV #13,$TSTNM ;SET TEST #13 AND CLEAR (SERFLG)
012154 004737 023734 JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
012160 012737 012340 001124 MOV #TEST13,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
012166 013737 002322 001220 MOV RPT,$TIMES ;GET THE ITERATION COUNT
012174 112737 000031 001131 MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
222 012202 012737 000013 001240 MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

012210 032777 010000 166736 BIT #SW12,@SWR ;INHIBIT TYPING TEST NUMBER ?
012216 001406 BEQ +16 ;BR IF YES
012220 104401 037265 TYPE ,MSGTST ;TYPE 'TEST'
012224 013746 001240 MOV $TESTN,-(SP) ;;SAVE $TESTN FOR TYPEOUT
012230 104403 TYPOS ;;GO TYPE--OCTAL ASCII
012232 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
012233 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS

223 012234 005737 001342 TST CLKSTA ;KW11-P CLOCK?
224 012240 003002 BGT 1$ ;YES--START TEST
240 012242 000137 012666 JMP EXIT13 ;NO--JUMP TO EXIT13
012246 012737 012246 001122 1$: MOV #,$SLPADR ;SET THE LOOP ADDRESS
012254 004037 025500 JSR R0,SRCHOO ;DO A MASSBUS INIT & RECAL
012260 000402 BR 2$ ;RETURN HERE IF NO ERROR
0 2262 000137 012666 JMP EXIT13 ;RETURN HERE ON ERROR
012266 012737 012266 001122 2$: MOV #,$SLPADR ;ERROR LOOP ADDRESS
  
```

012274	112737	000105	036510	MOV	#SEEK,DPB.A+2	:SEEK=COMMAND	
012302	005037	036516		CLR	DPB.A+10	:USE TRACK 0 & SECTOR 0	
012306	013737	002324	036520	MOV	FC,DPB.A+12	:STARTING CYLINDER	
012314	012737	012666	001350	MOV	#EXIT13,BYPASS	:GO TO EXIT13 IF ERROR	
012322	004037	024076		JSR	RO,CALL.A	:GO EXECUTE THE COMMAND	
012326	012703	001670		MOV	#T13,R3	:PARAMETER POINTER	
012332	012737	012666	001222	MOV	#EXIT13,\$ESCAPE	:ESCAPE TO EXIT13 ON ERROR	
012340							
241	012340	012706	001100	MOV	#STACK,SP	:SETUP STACK	
242	012344	012701	000200	MOV	#128,R1	:REPEAT "'FC'-'LC'-'FC'" 128 TIMES	
243	012350	004737	025664	JSR	PC,STRMR	:INIT. THE COUNTERS	
244	012354	012777	012574	167146	MOV	#5\$,@PKV	:SET UP VECTOR IN CASE OF OVERFLOW
245	012362	012777	025662	017000	MOV	#DORTI,@RMVEC	:SETUP RM80 VECTOR
246	012370	005077	167142		CLR	@PKB	:START COUNT AT ZERO
247	012374	013764	002326	000034	MOV	LC,RMDC(R4)	: 'MIDDLE' CYLINDER
248	012402	012764	000105	000000	MOV	#SEEK,RMCS1(R4)	:START A SEEK
249	012410	012777	000131	167116	MOV	#131,@PKCS	:START THE CLOCK
250	012416	000001			WAIT		:WAIT ON INTERRUPT
251	012420	042777	000101	167106	BIC	#101,@PKCS	:STOP CLOCK
252	012426	032764	040000	000012	BIT	#ERR,RMDS(R4)	:ERR=1?
253	012434	001411			BEQ	2\$	:NO--BRANCH
254	012436	104412			SAVREG		:SAVE R0-R5
	012440	012702	036574		MOV	#DTADPB,R2	:DPB POINTER
	012444	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS
	012450	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	012454	104413			RESREG		:RESTORE R0-R5
255	012456	104017			EMT	17	
256	012460	005005			CLR	R5	:SET UP/DOWN SWITCH TO UP
257	012462	004737	025726		JSR	PC,COUNT	:UPDATE THE COUNT
258	012466	005077	167044		CLR	@PKB	:START THE COUNT AT ZERO
259	012472	013764	002324	000034	MOV	FC,RMDC(R4)	:BEGINNING CYLINDER
260	012500	012764	000105	000000	MOV	#SEEK,RMCS1(R4)	:START A SEEK
261	012506	012777	000131	167020	MOV	#131,@PKCS	:START THE CLOCK
262	012514	000001			WAIT		:WAIT ON INTERRUPT
263	012516	042777	000101	167010	BIC	#101,@PKCS	:STOP THE CLOCK
264	012524	032764	040000	000012	BIT	#ERR,RMDS(R4)	:ERR=1?
265	012532	001411			BEQ	3\$	:NO--BRANCH
266	012534	104412			SAVREG		:SAVE R0-R5
	012536	012702	036574		MOV	#DTADPB,R2	:DPB POINTER
	012542	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS
	012546	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	012552	104413			RESREG		:RESTORE R0-R5
267	012554	104017			EMT	17	
268	012556	012705	177777		MOV	#-1,R5	:SET UP/DOWN SWITCH TO DOWN
269	012562	004737	025726		JSR	PC,COUNT	:UPDATE THE COUNT
270	012566	005301			DEC	R1	:DONE?
271	012570	003277			BGT	1\$	:NO--BRANCH
272	012572	000420			BR	6\$	:YES--EXIT
273	012574	042777	000101	166732	BIC	#101,@PKCS	:STOP THE CLOCK
274	012602	005037	177776		CLR	PS	:DROP THE PRIORITY
275	012606	012600			MOV	(SP)+,R0	:PC OF WAIT+2
276	012610	005726			TST	(SP)+	:POP THE PS FROM THE STACK
277	012612	104412			SAVREG		:SAVE R0-R5
	012614	012702	036574		MOV	#DTADPB,R2	:DPB POINTER
	012620	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS
	012624	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	012630	104413			RESREG		:RESTORE R0-R5

```

278 012632 104020          EMT      20
279 012634 004737 023146 6$:      JSR      PC,CNTCLR      ;GO CLEAR MASSBUS CONTROLLER
280 012640 004737 023240          JSR      PC,ST.CLK      ;INITIALIZE THE CLOCK
281 012644 012777 033526 016516     MOV      #ISR,@RMVEC    ;RESTORE RH/RM INT. VECTOR
286 012652 004037 026166          JSR      RO,TYPTIM     ;GO TYPE THE TIMES
      012656 001670          T13
      012660 004037 026060          JSR      RO,SPTYP     ;POINTER
      012664 001732          SP13
287 012666 000004          EXIT13: SCOPE        ;CALL SCOPE ROUTINE
288
304
305
  
```

```

:*****
:*TEST 14      MAXIMUM SEEK TIMING TEST
:*THIS TEST WILL COMMAND A FORWARD SEEK FROM CYLINDER 'FC' TO
:*CYLINDER 'LC', THEN A REVERSE SEEK FROM CYLINDER 'LC' TO
:*CYLINDER 'FC'. BOTH SEEKS ARE TIMED, BUT ONLY THE FORWARD SEEKS
:*ARE CHECKED TO ENSURE THEY ARE WITHIN THE TOLERANCE ALLOWED FOR THE
:*MAXIMUM SEEK TIME. THIS SEQUENCE IS REPEATED 128 TIMES (FOR A TOTAL
:*OF 256. SEEKS). MAXIMUM (FORWARD) TIME IS 50.0 MS
:*
:* THERE IS NO SPECIFICATION GIVEN FOR THE MAXIMUM REVERSE SEEK TIME
:* ON THIS PARTICULAR DRIVE. THEREFORE, ANY REVERSE SEEK TIMES ABOVE
:* THE MAXIMUM TIME OF 50.0 MS WILL NOT BE TYPED IN THE TIMING REPORT.
:* HOWEVER, THE TIMING REPORT WILL STILL TYPE THE MINIMUM, MAXIMUM,
:* AND AVERAGE TIMES FOR THE REVERSE SEEKS.
:*
:*****
TST14:
  
```

```

012670 000240
012672 033737 001610 001332      NOP
012700 001002          BIT      BITS+<14*2>,TSTNMS      ;DO THIS TEST?
012702 000137 013440          BNE      +6          ;BR IF YES
      012706 012737 000014 001116     MOV      #14,$TSTNM      ;SET TEST #14 AND CLEAR (SERFLG)
      012714 004737 023734          JSR      PC,LODPRM      ;LOAD THE PARAMETERS FOR THE TEST
      012720 012737 013100 001124     MOV      #TEST14,$LPERR  ;SETUP THE LOOP ON ERROR ADDRESS
      012726 013737 002322 001220     MOV      RPT,$TIMES     ;GET THE ITERATION COUNT
      012734 112737 000031 001131     MOV      #25,$SERMAX    ;MAX ERRORS ALLOWED FOR TEST
      012742 012737 000014 001240     MOV      #14,$TESTN     ;:SET TEST NUMBER IN APT MAIL BOX
306 012750 032777 010000 166176     BIT      #SW12,@SWR     ;INHIBIT TYPING TEST NUMBER ?
      012756 001406          BEQ      +16          ;BR IF YES
      012760 104401 037265          TYPE    ,MSGTST       ;TYPE 'TEST'
      012764 013746 001240          MOV      $TESTN,-(SP)  ;:SAVE $TESTN FOR TYPEOUT
      012770 104403          TYPDS   ;:GO TYPE--OCTAL ASCII
      012772 003          .BYTE   3          ;:TYPE 3 DIGIT(S)
      012773 000          .BYTE   0          ;:SUPPRESS LEADING ZEROS
307 012774 005737 001342          TST      CLKSTA       ;KW11-P CLOCK
308 013000 003002          BGT      1$          ;YES--START TEST
311 013002 000137 013436          JMP      EXIT14       ;NO--JUMP TO EXIT14
312 013006 012737 013006 001122 1$:  MOV      #.,$LPADR     ;SETUP THE LOOP ADDRESS
313 013014 004037 025500          JSR      RO,SRCHOO    ;DO A MASSBUS INIT & RECAL
314 013020 000402          BR       2$          ;RETURN HERE IF NO ERROR
326 013022 000137 013436          JMP      EXIT14       ;RETURN HERE ON ERROR
      013026 012737 013026 001122 2$:  MOV      #.,$LPADR     ;ERROR LOOP ADDRESS
      013034 112737 000105 036510     MOV      #SEEK,DPB.A+2 ;SEEK=COMMAND
  
```



013042	005037	036516		CLR	DPB,A+10	:USE TRACK 0 & SECTOR 0	
013046	013737	002324	036520	MOV	FC,DPB,A+12	:STARTING CYLINDER	
013054	012737	013436	001350	MOV	#EXIT14,BYPASS	:GO TO EXIT14 IF ERROR	
013062	004037	024076		JSR	RO,CALL.A	:GO EXECUTE THE COMMAND	
013066	012703	001700		MOV	#T14,R3	:PARAMETER POINTER	
013072	012737	013436	001222	MOV	#EXIT14,\$ESCAPE	::ESCAPE TO EXIT14 ON ERROR	
013100							
327	013100	012706	001100	TEST14:	MOV	#STACK,SP	:SETUP STACK
328	013104	012701	000200		MOV	#128,R1	:REPEAT "'FC'-'LC'-'FC'" 128 TIMES
329	013110	004737	025664		JSR	PC,STRMR	:INIT. THE TIMERS
330	013114	012777	013344	166406	MOV	#5\$,@PVV	:SETUP VECTOR IN CASE OF OVERFLOW
331	013122	012777	025662	016240	MOV	#DORTI,@RMVEC	:SETUP RM80 VECTOR
332	013130	005077	166402	1\$:	CLR	@PKB	:START COUNTING FROM ZERO
333	013134	013764	002326	000034	MOV	LC,RMDC(R4)	:MAXIMUM CYLINDER
334	013142	012764	000105	000000	MOV	#SEEK,RMCS1(R4)	:START A SEEK
335	013150	012777	000131	166356	MOV	#131,@PKCS	:START THE CLOCK
336	013156	000001			WAIT		:WAIT ON INTERRUPT
337	013160	042777	000101	166346	BIC	#101,@PKCS	:STOP THE CLOCK
338	013166	032764	040000	000012	BIT	#ERR,RMDS(R4)	:ERR=1?
339	013174	001411			BEQ	2\$	:NO--BRANCH
340	013176	104412			SAVREG		:SAVE R0-R5
	013200	012702	036574		MOV	#DTADPB,R2	:DPB POINTER
	013204	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RM/RM REGISTERS
	013210	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	013214	104413			RESREG		:RESTORE R0-R5
341	013216	104017			EMT	17	
342	013220	005005		2\$:	CLR	R5	:SET THE UP/DOWN SWITCH TO UP
343	013222	004737	025726		JSR	PC,COUNT	:UP THE COUNT
344	013226	005077	166304		CLR	@PKB	:START COUNT AT ZERO
345	013232	013764	002324	000034	MOV	FC,RMDC(R4)	:BEGINNING CYLINDER
346	013240	012764	000105	000000	MOV	#SEEK,RMCS1(R4)	:START A SEEK
347	013246	012777	000131	166260	MOV	#131,@PKCS	:START THE CLOCK
348	013254	000001			WAIT		:WAIT ON INTERRUPT
349	013256	042777	000101	166250	BIC	#101,@PKCS	:STOP THE CLOCK
350	013264	032764	040000	000012	BIT	#ERR,RMDS(R4)	:ERR=1?
351	013272	001411			BEQ	3\$	:NO--BRANCH
352	013274	104412			SAVREG		:SAVE R0-R5
	013276	012702	036574		MOV	#DTADPB,R2	:DPB POINTER
	013302	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RM/RM REGISTERS
	013306	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	013312	104413			RESREG		:RESTORE R0-R5
353	013314	104017			EMT	17	
354	013316	012705	177777	3\$:	MOV	#-1,R5	:SET THE UP/DOWN SWITCH TO DOWN
355	013322	004737	025726		JSR	PC,COUNT	:UPDATE THE COUNT
356	013326	005037	001416		CLR	TIM.DN+2	:FORGET ABOUT # OF SEEKS BELOW MINIMUM TIME
357	013332	005037	001422		CLR	TIM.DN+6	:FORGET ABOUT # OF SEEKS ABOVE MAXIMUM TIME
358	013336	005301			DEC	R1	:DONE?
359	013340	003273			BGT	1\$	:NO--BRANCH
360	013342	000420			BR	6\$	:YES--EXIT
361	013344	042777	000101	166162	5\$:	BIC	#101,@PKCS
362	013352	005037	177776		CLR	PS	:STOP THE CLOCK
363	013356	012600			MOV	(SP)+,R0	:DROP THE PRIORITY
364	013360	005726			TST	(SP)+	:PC OF WAIT+2
365	013362	104412			SAVREG		:POP THE PS FROM THE STACK
	013364	012702	036574		MOV	#DTADPB,R2	:SAVE R0-R5
	013370	004737	035744		JSR	PC,SVRH70	:DPB POINTER
	013374	004737	023146		JSR	PC,CNTCLR	:SAVE ALL THE RM/RM REGISTERS
							:GO CLEAR CONTROLLER



```

013400 104413 RESREG ;RESTORE R0-R5
366 013402 104020 EMT 20
367 013404 004737 023146 6$: JSR PC,CNTCLR ;GO CLEAR MASSBUS CONTROLLER
368 013410 004737 023240 JSR PC,ST.CLK ;INITIALIZE THE CLOCK
369 013414 012777 033526 015746 MOV #ISR,ARMVEC ;RESTORE RH/RM INT. VECTOR
374 013422 004037 026166 JSR R0,TYPTIM ;GO TYPE THE TIMES
013426 001700 T14 ;POINTER
013430 004037 026060 JSR R0,SPTYP
013434 001740 SP14
375 013436 000004 EXIT14: SCOPE ;CALL SCOPE ROUTINE
376
387
388
  
```

```

*****
*TEST 15 AVERAGE SEEK TIME MEASUREMENT TEST
*THIS TEST MEASURES THE AVERAGE SEEK TIME BY DOING ALL UNIQUE SEEKS
*BETWEEN 'FC' AND 'LC', AND MEASURING THE SEEK TIMES. THE TOTAL
*SEEK TIME FOR ALL SEEKS IS THEN DIVIDED BY THE NUMBER OF SEEKS
*PERFORMED TO PROVIDE AN AVERAGE SEEK TIME VALUE FOR THE RANGE OF
*SEEKS TIMED. MAXIMUM TIME IS 25.0 MS
*
*WARNING - THIS TEST TAKES APPROXIMATELY 3.0 HOURS TO RUN
*
*****
  
```

```

013440
013440 000240
013442 033737 001612 001332
013450 001002
013452 000137 015142
389
013456 012737 000015 001116
013464 004737 023734
013470 012737 013644 001124
013476 013737 002322 001220
013504 112737 000031 001131
013512 012737 000015 001240
013520 032777 010000 165426
013526 001406
013530 104401 037265
013534 013746 001240
013540 104403
013542 003
013543 000
390 013544 005737 001342
391 013550 003002
394 013552 000137 015140
395 013556 012737 013556 001122 1$:
396 013564 004037 025500
397 013570 000402
408 013572 000137 015140
013576 012737 013576 001122 2$:
013604 112737 000105 036510
013612 005037 036516
013616 013737 002324 036520
013624 012737 015140 001350
013632 004037 024076
  
```

```

TST15:
NOP
BIT BITS+<15*2>,TSTNMS ;DO THIS TEST?
BNE .+5 ;BR IF YES
JMP $EOP ;NO--GO TO THE END OF THE PROGRAM

MOV #15,$TSTNM ;SET TEST #15 AND CLEAR ($ERFLG)
JSR PC,LODPRM ;LOAD THE PARAMETERS FOR THE TEST
MOV #TEST15,$LPERR ;SETUP THE LOOP ON ERROR ADDRESS
MOV RPT,$TIMES ;GET THE ITERATION COUNT
MOVB #25,$ERMAX ;MAX ERRORS ALLOWED FOR TEST
MOV #15,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

BIT #SW12,$SWR ;INHIBIT TYPING TEST NUMBER ?
BEQ .+16 ;BR IF YES
TYPE ,MSGTST ;TYPE 'TEST'
MOV $TESTN,-(SP) ;;SAVE $TESTN FOR TYPEOUT
TYPOS ;;GO TYPE--OCTAL ASCII
.BYTE 3 ;;TYPE 3 DIGIT(S)
.BYTE 0 ;;SUPPRESS LEADING ZEROS

TST CLKSTA ;KW11-P CLOCK?
BGT 1$ ;YES--START TEST
JMP EXIT15 ;NO--JUMP TO EXIT15
MOV #,$LPADR ;SETUP THE LOOP ADDRESS
JSR R0,SRCH00 ;DO A MASSBUS INIT. AND RECAL
BR 2$ ;NO ERROR RETURN
JMP EXIT15 ;ERROR RETURN--SCOPE LOOP CALL
MOV #,$LPADR ;ERROR LOOP ADDRESS
MOVB #SEEK,DPB.A+2 ;SEEK=COMMAND
CLR DPB 4+10 ;USE TRACK 0 & SECTOR 0
MOV FC,DPB.A+12 ;STARTING CYLINDER
MOV #EXIT15,BYPASS ;GO TO EXIT15 IF ERROR
JSR R0,CALL.A ;GO EXECUTE THE COMMAND
  
```

```

013636 012737 015140 001222 TEST15: MOV #EXIT15,$ESCAPE ;;ESCAPE TO EXIT15 ON ERROR
013644
409 013644 012706 001100 MOV #STACK,SP ;;SETUP STACK
410 013650 005037 001472 CLR SKTIM ;;CLEAR THE SEEK TIMER
413 013654 005037 001474 CLR SKTIM+2
013660 005037 001476 CLR SKTIM+4
013664 005037 001500 CLR SKTIM+6
414 013670 005037 001502 CLR SKCNT ;;CLEAR THE SEEK COUNTER
415 013674 005037 001504 CLR SKCNT+2
416 013700 005037 001506 CLR SKSIZ ;;SET SEEK LENGTH TO ZERO
417 013704 005237 001506 1$: INC SKSIZ ;;INCREMENT THE SEEK LENGTH
418 013710 005037 001510 CLR CYINC ;;CLEAR THE STARTING INCREMENT
419 013714 023737 001510 2$: CMP CYINC,SKSIZ ;;FINISHED WITH THIS PASS ?
420 013722 001025 BNE 3$ ;;IF NE, NO
421
422 ;TYPE CURRENT SEEK SIZE AND NUMBER OF SEEKS, IF SW12=1
423
424 013724 032777 010000 165222 BIT #SW12,@SWR ;;SEE IF SWITCH 12 SET
425 013732 001764 BEQ 1$ ;;IF EQ, NO
426 013734 104401 015046 TYPE ,16$ ;;'SEEK SIZE='
427 013740 013746 001506 MOV SKSIZ,-(SP) ;;MOVE IT TO THE STACK
428 013744 004737 022212 JSR PC,$SB2D ;;CONVERT IT
429 013750 004737 022442 JSR PC,$SUPRS ;;TYPE IT
430 013754 104401 015062 TYPE ,17$ ;;'NUMBER OF SEEKS='
431 013760 012746 001502 MOV #SKCNT,-(SP) ;;SEEK COUNT POINTER
432 013764 004737 022246 JSR PC,$DB2D ;;CONVERT TO ASCII
433 013770 004737 022442 JSR PC,$SUPRS ;;TYPE IT
434 013774 000743 BR 1$ ;;CONTINUE
435
436 013776 013737 001510 002346 3$: MOV CYINC,NC1 ;;SETUP STARTING ADDRESS
437 014004 063737 002324 002346 ADD FC,NC1 ;;ADD THE STARTING ADDRESS
438 014012 013737 002346 002350 MOV NC1,NC2 ;;COPY THE STARTING ADDRESS
439 014020 063737 001506 002350 ADD SKSIZ,NC2 ;;ENDING CYLINDER
440 014026 023737 002350 002326 CMP NC2,LC ;;SEEK IF ENDING CYLINDER TOO LARGE
441 014034 101402 BLOS 4$ ;;IF LOS, NO
442 014036 000137 014650 JMP 14$ ;;FINISHED WITH THIS PASS
443
444 014042 004737 023240 4$: JSR PC,ST.CLK ;;RE-INIT THE CLOCK
445 014046 012777 033526 015314 MOV #ISR,@RMVEC ;;RESTORE THE RM VECTOR
446
447 014054 5$:
014054 112737 000105 036510 MOVB #SEEK,DPB.A+2 ;;SEEK=COMMAND
014062 005037 036516 CLR DPB.A+10 ;;USE TRACK 0 & SECTOR 0
014066 013737 002346 036520 MOV NC1,DPB.A+12 ;;STARTING CYLINDER
014074 012737 015140 001350 MOV #EXIT15,BYPASS ;;GO TO EXIT15 IF ERROR
014102 004037 024076 JSR RO,CALL.A ;;GO EXECUTE THE COMMAND
454
455 014106 005077 165422 6$: CLR @PKCS ;;STOP THE CLCK
456 014112 012777 015102 165410 MOV #18$,@PKV ;;SETUP CLOCK VECTOR IN CASE OF OVERFLOW
457 014120 012777 025662 015242 MOV #DORTI,@RMVEC ;;CHANGE RM80 VECTOR
458
459 014126 005077 165404 7$: CLR @PKB ;;START COUNT AT ZERO
460 014132 013764 002350 000034 MOV NC2,RMDC(R4) ;;LOAD CYLINDER ADDRESS
461 014140 012764 000105 000000 MOV #SEEK,RMCS1(R4) ;;START A SEEK
462 014146 012777 000131 165360 MOV #131,@PKCS ;;START THE CLOCK
463 014154 000001 WAIT ;;WAIT ON INTERRUPT
464 014156 042777 000101 165350 BIC #101,@PKCS ;;STOP CLOCK

```

465	014164	032764	040000	000012	BIT	#BIT14,RMDS(R4)	:ERR=1?	
466	014172	001425			BEQ	8\$	:NO--BRANCH	
467	014174	104412			SAVREG		:SAVE R0-R5	
	014176	012702	036574		MOV	#DTADPB,R2	:DPB POINTER	
	014202	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS	
	014206	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER	
	014212	104413			RESREG		:RESTORE R0-R5	
468	014214	104017			EMT	17	:REPORT THE ERROR	
469	014216	032737	040000	036702	BIT	#SKI,RM.REG+RMER2	:SEE :F SKI SET	
470	014224	001410			BEQ	8\$	:IF EQ, NO	
471	014226	004737	023240		JSR	PC,ST.CLK	:RE-INIT THE CLOCK	
472	014232	012777	033526	015130	MOV	#ISR,@RMVEC	:RESTORE THE RH VECTOR	
473	014240	004037	024710		JSR	RO,CALL.R	:ISSUE A RECAL	
474	014244	000703			BR	5\$	:TRY AGAIN	
475								
476	014246	067737	165266	001472	8\$:	ADD	@PKC,SKTIM	:ADD THE COUNTER VALUE
479	014254	005537	001474		ADC	SKTIM+2	:ADD ANY CARRY	
	014260	005537	001476		ADC	SKTIM+4	:ADD ANY CARRY	
	014264	005537	001500		ADC	SKTIM+6	:ADD ANY CAPRY	
480	014270	062737	000001	001502	ADD	#1,SKCNT	:COUNT THE SEEK	
481	014276	005537	001504		ADC	SKCNT+2	:	
482								
483	014302	013737	002350	002346	MOV	NC2,NC1	:GET NEW STARTING CYLINDER	
484	014310	063737	001506	002350	ADD	SKSIZ,NC2	:NEW ENDING CYLINDER	
485	014316	023737	002350	002326	CMP	NC2,LC	:FINISHED IN THIS DIRECTION ?	
486	014324	101700			BLOS	7\$	:IF LOS, NO	
487	014326	013737	002326	002350	MOV	LC,NC2	:SETUP FOR REVERSE SEEK	
488	014334	163737	001510	002350	SUB	CYINC,NC2	:ADJUST ADDRESS	
489	014342	013737	002350	002346	MOV	NC2,NC1	:COPY ADDRESS	
490	014350	163737	001506	002346	SUB	SKSIZ,NC1	:NEW ENDING ADDRESS	
491								
492	014356	004737	023240		9\$:	JSR	PC,ST.CLK	:RE-INIT THE CLOCK
493	014362	012777	033526	015000	MOV	#ISR,@RMVEC	:RESTORE THE RH VECTOR	
494								
495	014370				10\$:			
	014370	112737	000105	036510	MOVB	#SEEK,DPB.A+2	:SEEK=COMMAND	
496	014376	005037	036516		CLR	DPB.A+10	:USE TRACK 0 & SECTOR 0	
497	014402	013737	002350	036520	MOV	NC2,DPB.A+12	:STARTING CYLINDER	
500	014410	012737	015140	001350	MOV	#EXIT15,BYPASS	:GO TO EXIT15 IF ERROR	
501	014416	004037	024076		JSR	RO,CALL.A	:GO EXECUTE THE COMMAND	
502								
503	014422	005077	165106		11\$:	CLR	@PKCS	:STOP THE CLOCK
504	014426	012777	015102	165074	MOV	#18\$,@PKV	:SETUP CLOCK VECTOR IN CASE OF OVERFLOW	
505	014434	012777	025662	014726	MOV	#DORTI,@RMVEC	:CHANGE RM80 VECTOR	
506								
507	014442	005077	165070		12\$:	CLR	@PKB	:START COUNT AT ZERO
508	014446	013764	002346	000034	MOV	NC1,RMDC(R4)	:LOAD CYLINDER ADDRESS	
509	014454	012764	000105	000000	MOV	#SEEK,RMCS1(R4)	:START A SEEK	
510	014462	012777	000131	165044	MOV	#131,@PKCS	:START THE CLOCK	
511	014470	000001			WAIT		:WAIT ON INTERRUPT	
512	014472	042777	000101	165034	BIC	#101,@PKCS	:STOP CLOCK	
513	014480	032764	040000	000012	BIT	#BIT14,RMDS(R4)	:ERR=1?	
514	014506	001425			BEQ	13\$	:NO--BRANCH	
515	014510	104412			SAVREG		:SAVE R0-R5	
	014512	012702	036574		MOV	#DTADPB,R2	:DPB POINTER	
	014516	004737	035744		JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS	
	014522	004737	023146		JSR	PC,CNTCLR	:GO CLEAR CONTROLLER	

516	014526	104413		RESREG			:RESTORE R0-R5
517	014530	104017		EMT	17		:REPORT THE ERROR
518	014532	032737	040000	BIT	#SKI,RM.REG+RMR2		:SEE IF SKI SET
519	014540	001410		BEQ	13\$		:IF EQ, NO
520	014542	004737	023240	JSR	PC,ST.CLK		:RE-INIT THE CLOCK
521	014546	012777	033526	MOV	#ISR,@RMVEC	014614	:RESTORE THE RM VECTOR
522	014554	004037	024710	JSR	R0,CALL.R		:ISSUE A RECAL
523	014560	000703		BR	10\$		:TRY AGAIN
524	014562	067737	104752	ADD	@PKC,SKTIM	001472	:ADD THE COUNTER VALUE
527	014570	005537	001474	ADC	SKTIM+2		:ADD ANY CARRY
	014574	005537	001476	ADC	SKTIM+4		:ADD ANY CARRY
	014600	005537	001500	ADC	SKTIM+6		:ADD ANY CARRY
528	014604	062737	000001	ADD	#1,SKCNT	001502	:COUNT THE SEEK
529	014612	005537	001504	ADC	SKCNT+2		:.. ..
530							
531	014616	013737	002346	MOV	NC1,NC2	002350	:COPY ENDING ADDRESS
532	014624	163737	001506	SUB	SKSIZ,NC1	002346	:NEW ENDING ADDRESS
533	014632	005737	002346	TST	NC1		:SEE IF THE NEW CYL IS NEGATIVE
534	014636	100404		BMI	14\$		:IF MI, YES - FINISHED
535	014640	023737	002346	CMP	NC1,FC	002324	:FINISHED ?
536	014646	103275		BHIS	12\$		:IF HIS, NO
537	014650	013746	002326	MOV	LC,-(SP)	14\$:	:PUT THE ENDING CYLINDER ON THE STACK
538	014654	163716	002324	SUB	FC,(SP)		:FIND THE DIFFERENCE
539	014660	023726	001506	CMP	SKSIZ,(SP)+		:SEE IF TEST FINISHED
540	014664	103004		BHIS	15\$		:IF HIS, YES
541	014666	005237	001510	INC	CYINC		:INCREMENT THE CYLINDER OFFSET
542	014672	000137	013714	JMP	2\$		:CONTINUE
543							
544	014676	004737	023146	JSR	PC,CNTCLR	15\$:	:GO CLEAR MASSBUS CONTROLLER
545	014702	004737	023240	JSR	PC,ST.CLK		:RE-INIT THE CLOCK
546	014706	012777	033526	MOV	#ISR,@RMVEC	014454	:RESTORE THE RM VECTOR
547							
548	014714	104412		SAVREG			:SAVE R0-R5
583	014716	104401	037634	TYPE	,MEASUR		:TYPE 'AVERAGE SEEK TIME MEASUREMENT'
	014722	013703	001472	MOV	SKTIM,R3		:LOAD THE DIVIDEND
	014726	013702	001474	MOV	SKTIM+2,R2		:.. ..
	014732	013701	001476	MOV	SKTIM+4,R1		:.. ..
	014736	013700	001500	MOV	SKTIM+6,R0		:.. ..
	014742	013705	001502	MOV	SKCNT,R5		:LOAD THE DIVISOR
	014746	013704	001504	MOV	SKCNT+2,R4		:.. ..
	014752	004737	022652	JSR	PC,M.DPID		:FIND THE AVERAGE
	014756	010337	001472	MOV	R3,SKTIM		:LOAD THE QUOTIENT FOR DISPLAY
	014762	010237	001474	MOV	R2,SKTIM+2		:.. ..
	014766	104401	001231	TYPE	,\$CRLF		:CR-LF
	014772	104401	037710	TYPE	,MSGAVG		: 'AVG='
	014776	012746	001472	MOV	#SKTIM,-(SP)		:AVERAGE SEEK TIME POINTER
	015002	004737	022246	JSP	PC,\$DB2D		:CONVERT TO ASCII
	015006	004737	022442	JSR	PC,\$SUPRS		:TYPE IT
	015012	104401	037716	TYPE	,MSGOUS		:TYPE '0 US'
	015016	104401	037160	TYPE	,COMMA		:TYPE ','
	015022	012746	001502	MOV	#SKCNT,-(SP)		:SEEK COUNT POINTER
	015026	004737	022246	JSR	PC,\$DB2D		:CONVERT TO ASCII
	015032	004737	022442	JSR	PC,\$SUPRS		:TYPE IT
	015036	104401	040021	TYPE	,MSGSEK		:TYPE 'SEEKS TIMED'
	015042	104413		RESREG			:RESTORE R0-R5
	015044	000435		BR	EXIT15		:BR TO EXIT15

584					16\$:	.ASCIZ	<CRLF>/SEEK SIZE=/ / , NO. OF SEEKS=/ .EVEN	
585	015046	200	123	105	17\$:	.ASCIZ		
586	015062	054	040	116				
587								
588					18\$:	CLR	@PKCS	:STOP THE CLOCK
589	015102	005077	164426			CLR	PS	:DROP THE PRIORITY
590	015106	005037	177776			MOV	(SP)+,R0	:PC OF WAIT+2
591	015112	012600				TST	(SP)+	:POP THE PS FROM THE STACK
592	015114	005726				SAVREG		:SAVE R0-R5
593	015116	104412				MOV	#DTADPB,R2	:DPB POINTER
	015120	012702	036574			JSR	PC,SVRH70	:SAVE ALL THE RH/RM REGISTERS
	015124	004737	035744			JSR	PC,CNTCLR	:GO CLEAR CONTROLLER
	015130	004737	023146			RESREG		:RESTORE R0-R5
	015134	104413				EMT	20	:CLOCK OVERFLOWED
594	015136	104020			EXIT15:	SCOPE		:CALL SCOPE ROUTINE
595	015140	000004						
596								
601								

1

.SBTTL END OF PASS ROUTINE

```

*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF-PROGRAM AFTER 8. PASSES THRU THE PROGRAM
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO RETURN
  
```

```

015142
015142 104401 015150
015146 000407
SEOP:
TYPE ,65$           ;;TYPE ASCIZ STRING
BR 64$             ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF><LF>/END OF PASS/
64$:
TYPE ,67$           ;;TYPE ASCIZ STRING
BR 66$             ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / ON DRIVE/
66$:
MOV CHKDRV,-(SP)   ;;SAVE CHKDRV FOR TYPEOUT
TYPOS              ;;GO TYPE--OCTAL ASCII
.BYTE 2            ;;TYPE 2 DIGIT(S)
.BYTE 0            ;;SUPPRESS LEADING ZEROS
TST $ERTTL         ;;ANY ERRORS DETECTED ?
BEQ 1$             ;;BR IF NO
TYPE ,69$          ;;TYPE ASCIZ STRING
BR 68$             ;;GET OVER THE ASCIZ
;;69$: .ASCIZ / ERRORS DETECTED=/
68$:
MOV $ERTTL,-(SP)   ;;SAVE $ERTTL FOR TYPEOUT
TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1$: CLR $ERTTL     ;;CLEAR ERROR TOTAL
CLR $STNM          ;;ZERO THE TEST NUMBER
CLR $TIMES         ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS          ;;INCREMENT THE PASS NUMBER
BIC #10000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+          ;;LOOP?
SEOPCT: .WORD 8.
BGT $DOAGN         ;;YES
MOV (PC)+,@(PC)+  ;;RESTORE COUNTER
SENDCT: .WORD 8.
SEOPCT: .WORD 8.
TYPE ,65$          ;;TYPE ASCIZ STRING
BR 64$             ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF>/END OF TEST/<CRLF>
64$:
TYPE $ENULL        ;;TYPE NULL CHARACTER
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN         ;;BRANCH IF NO MONITOR
RESET              ;;CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP                ;;SAVE ROOM
NOP                ;;FOR
NOP                ;;ACT11
$DOAGN: JMP @(PC)+  ;;RETURN
015206
015206 013746 001352
015212 104403
015214 002
015215 000
015216 005737 001126
015222 001420
015224 104401 015232
015230 000412
;;69$: .ASCIZ / ERRORS DETECTED=/
68$:
MOV $ERTTL,-(SP)   ;;SAVE $ERTTL FOR TYPEOUT
TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1$: CLR $ERTTL     ;;CLEAR ERROR TOTAL
CLR $STNM          ;;ZERO THE TEST NUMBER
CLR $TIMES         ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS          ;;INCREMENT THE PASS NUMBER
BIC #10000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+          ;;LOOP?
SEOPCT: .WORD 8.
BGT $DOAGN         ;;YES
MOV (PC)+,@(PC)+  ;;RESTORE COUNTER
SENDCT: .WORD 8.
SEOPCT: .WORD 8.
TYPE ,65$          ;;TYPE ASCIZ STRING
BR 64$             ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF>/END OF TEST/<CRLF>
64$:
TYPE $ENULL        ;;TYPE NULL CHARACTER
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN         ;;BRANCH IF NO MONITOR
RESET              ;;CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP                ;;SAVE ROOM
NOP                ;;FOR
NOP                ;;ACT11
$DOAGN: JMP @(PC)+  ;;RETURN
015256
015256 013746 001126
015262 104402
015264 005037 001126
015270 005037 001116
015274 005037 001220
015300 005237 001242
015304 042737 100000 001242
015312 005327
015314 000010
015316 003027
015320 012737
015322 000010
015324 015314
015326 104401 015334
015332 000407
SEOPCT: .WORD 8.
BGT $DOAGN         ;;YES
MOV (PC)+,@(PC)+  ;;RESTORE COUNTER
SENDCT: .WORD 8.
SEOPCT: .WORD 8.
TYPE ,65$          ;;TYPE ASCIZ STRING
BR 64$             ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <CRLF>/END OF TEST/<CRLF>
64$:
TYPE $ENULL        ;;TYPE NULL CHARACTER
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN         ;;BRANCH IF NO MONITOR
RESET              ;;CLEAR THE WORLD
SENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP                ;;SAVE ROOM
NOP                ;;FOR
NOP                ;;ACT11
$DOAGN: JMP @(PC)+  ;;RETURN
015352
015352 104401 015402
015356 013700 000042
015362 001405
015364 000005
015366 004710
015370 000240
015372 000240
015374 000240
015376
015376 000137
015400 015406
015402 377 377 000
$RTNAD: .WORD RTURN
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
.EVEN
  
```

2						
3	015406	012706	001100	RTURN: MOV	#STACK,SP	;RESTORE STACK
4	015412	004737	020134	JSR	PC,\$TKINT	;MAKE SURE KEYBOARD INTERRUPT AND
5	015416	004737	023240	JSR	PC,ST.CLK	;INITIALIZE THE CLOCK
6	015422	000137	005102	JMP	RSTART	;RETURN TO RESTART

1

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
015426 112737 000001 015672 $ATY1:  MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
015434 112737 000001 015670 $ATY3:  MOVB  #1,$MFLG     ;;TO TYPE A MESSAGE
015442 000403                BR      $ATYC
015444 112737 000001 015672 $ATY4:  MOVB  #1,$FFLG     ;;TO ONLY REPORT FATAL ERROR
015452                SATYC:
015452 010046                MOV    R0,-(SP)      ;;PUSH R0 ON STACK
015454 010146                MOV    R1,-(SP)      ;;PUSH R1 ON STACK
015456 105737 015670                TSTB  $MFLG         ;;SHOULD TYPE A MESSAGE?
015462 001450                BEQ   5$            ;;IF NOT: BR
015464 122737 000001 001254        CMPB  #APTENV,$ENV  ;;OPERATING UNDER APT?
015472 001031                BNE   3$            ;;IF NOT: BR
015474 132737 000100 001255        BITB  #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
015502 001425                BEQ   3$            ;;IF NOT: BR
015504 017600 000004                MOV    @4(SP),R0    ;;GET MESSAGE ADDR.
015510 062766 000002 000004        ADD    #2,4(SP)    ;;BUMP RETURN ADDR.
015516 005737 001234                1$:  TST   $MSGTYPE    ;;SEE IF DONE W/ LAST XMISSION?
015522 001375                BNE   1$            ;;IF NOT: WAIT
015524 010037 001250                MOV    R0,$MSGAD   ;;PUT ADDR IN MAILBOX
015530 105720                2$:  TSTB  (R0)+      ;;FIND END OF MESSAGE
015532 001376                BNE   2$
015534 163700 001250                SUB   $MSGAD,R0    ;;SUB START OF MESSAGE
015540 006200                ASR   R0            ;;GET MESSAGE LGTH IN WORDS
015542 010037 001252                MOV    R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
015546 012737 000004 001234        MOV    #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
015554 000413                BR    5$
015556 017637 000004 015602        3$:  MOV    @4(SP),4$   ;;PUT MSG ADDR IN JSR LINKAGE
015564 062766 000002 000004        ADD    #2,4(SP)    ;;BUMP RETURN ADDRESS
015572 013746 177776                MOV    177776,-(SP) ;;PUSH 177776 ON STACK
015576 004737 017116                JSR   PC,$TYPE     ;;CALL TYPE MACRO
015602 000000                4$:  .WORD  0
015604                5$:
015604 105737 015672                10$: TSTB  $FFLG        ;;SHOULD REPORT FATAL ERROR?
015610 001416                BEQ   12$          ;;IF NOT: BR
015612 005737 001254                TST   $ENV         ;;RUNNING UNDER APT?
015616 001413                BEQ   12$          ;;IF NOT: BR
015620 005737 001234                11$: TST   $MSGTYPE    ;;FINISHED LAST MESSAGE?
015624 001375                BNE   11$         ;;IF NOT: WAIT
015626 017637 000004 001236        MOV    @4(SP),$FATAL ;;GET ERROR #
015634 062766 000002 000004        ADD    #2,4(SP)    ;;BUMP RETURN ADDR.
015642 005237 001234                INC   $MSGTYPE     ;;TELL APT TO TAKE ERROR
015646 105037 015672                12$: CLRB  $FFLG     ;;CLEAR FATAL FLAG
015652 105037 015671                CLRB  $LFLG       ;;CLEAR LOG FLAG
015656 105037 015670                CLRB  $MFLG       ;;CLEAR MESSAGE FLAG
015662 012601                MOV   (SP)+,R1    ;;POP STACK INTO R1
015664 012600                MOV   (SP)+,R0    ;;POP STACK INTO R0
015666 000207                RTS   PC          ;;RETURN
015670 000                $MFLG: .BYTE  0  ;;MESSG. FLAG
015671 000                $LFLG: .BYTE  0  ;;LOG FLAG
015672 000                $FFLG: .BYTE  0  ;;FATAL FLAG
                .EVEN
000200                APTSIZE = 200
000001                APTENV  = 001
000100                APTSPOOL= 100
000040                APTCSUP = 040
  
```



.SETTL ERROR HANDLER ROUTINE

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

015674	105037	016334		\$ERROR:	CLRB	IBSAVE	::CLEAR THE ITEM BYTE SAVE LOCATION
015700	104407				CKSWR		::TEST FOR CHANGE IN SOFT-SWR
015702	032777	000400	163244		BIT	#SW08,@SWR	::SEND ERROR MESSAGE TO TTY?
015710	001411				BEQ	7\$	::YES--BRANCH
015712	005737	001326			TST	LPTAVL	::IS THERE A LINE PRINTER AVAILABLE?
015716	001406				BEQ	7\$	::NO--BRANCH
015720	013737	001554	001164		MOV	LPS,\$STPS	::YES--SETUP STATUS
015726	013737	001556	001166		MOV	LPB,\$TPB	::AND BUFFER REG.'S FOR LINE PRINTER
015734	105237	001117		7\$:	INCB	\$ERFLG	::SET THE ERROR FLAG
015740	001775				BEQ	7\$	::DON'T LET THE FLAG GO TO ZERO
015742	013777	001116	163206		MOV	\$STNM,@DISPLAY	::DISPLAY TEST NUMBER AND ERROR FLAG
015750	032777	002000	163176		BIT	#BIT10,@SWR	::BELL ON ERROR?
015756	001402				BEQ	1\$	::NO - SKIP
015760	104401	001224			TYPE	,\$BELL	::RING BELL
015764	005237	001126		1\$:	INC	\$ERTTL	::COUNT THE NUMBER OF ERRORS
015770	011637	001132			MOV	(SP),\$ERRPC	::GET ADDRESS OF ERROR INSTRUCTION
015774	162737	000002	001132		SUB	#2,\$ERRPC	
016002	117737	163124	001130		MOVB	@ERRPC,\$ITEMB	::STRIP AND SAVE THE ERROR ITEM CODE
016010	032777	001000	163136		BIT	#BIT09,@SWR	::SEE IF LOOP ON ERROR IS SET
016016	001060				BNE	1004\$	::BRANCH AROUND ROUTINE IF SO
016020	122737	000177	001130		CMPB	#177,\$ITEMB	::SEE IF THIS IS THE POWER FAIL CALL
016026	001454				BEQ	1004\$	::BRANCH AROUND ROUTINE IF IT IS
016030	105737	016334			TSTB	IBSAVE	::SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
016034	001047				BNE	1003\$	::BRANCH IF SO
016036	022737	177777	016332		CMF	#-1,CPSAVE	::SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
016044	001445				BEQ	1004\$	::BRANCH IF SO
016046	013746	000004			MOV	ERRVEC,-(SP)	::SAVE CONTENTS OF ERROR VECTOR
016052	012737	016070	000004		MOV	#1000\$,ERRVEC	::SETUP 'TRAP' RETURN ADDRESS
016060	013737	177766	016332		MOV	177766,CPSAVE	::MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
016066	000405				BR	1001\$	
016070	012737	177777	016332	1000\$:	MOV	#-1,CPSAVE	::SET CPU ERROR REGISTER TIMEOUT INDICATOR
016076	012716	016104			MOV	#101\$, (SP)	::SETUP RETURN ADDRESS
016102	000002				RTI		
016104	012637	000004		1001\$:	MOV	(SP)+,ERRVEC	::RESTORE CONTENTS OF ERROR VECTOR
016110	022737	177777	016332	1002\$:	CMF	#-1,CPSAVE	::SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
016116	001420				BEQ	1004\$	::BRANCH IF SO
016120	032737	000001	016332		BIT	#BIT00,CPSAVE	::SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
016126	001414				BEQ	1004\$	::BRANCH IF OK
016130	042737	000001	177766		BIC	#BIT00,177766	::CLEAR THE BIT FOUND SET
016136	113737	001130	016334		MOVB	\$ITEMB,IBSAVE	::MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
016144	112737	000177	001130		MOVB	#177,\$ITEMB	::SET \$ITEMB TO SPECIAL POWER FAIL POINTER
016152	000402				BR	1004\$	::BRANCH OVER IBSAVE CLEARING



```

2
3
4
5
6
7
8
9
10
11 016336 113737 001116 001212 TYPERR: MOVB STSTNM,$TMP0 ;SAVE THE TEST NUMBER
12 016344 104412 SAVREG ;SAVE R0 - R5
13 016346 162700 000004 SUB #4,R0 ;FORM TEST PC
14 016352 010037 001176 MOV R0,$REG0 ;COPY R0-R5 IN $REG0-$REG5
15 016356 010137 001200 MOV R1,$REG1
16 016362 010237 001202 MOV R2,$REG2
17 016366 010337 001204 MOV R3,$REG3
18 016372 010437 001206 MOV R4,$REG4
19 016376 010537 001210 MOV R5,$REG5
20 016402 113700 001130 MOVB $ITEMB,R0 ;PICKUP ERROR ITEM NUMBER
21 016406 122700 000177 CMPB #177,R0 ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
22 016412 001007 BNE 1$ ;BRANCH IF NOT
23 016414 005001 CLR R1
27 016416 013737 001240 017114 MOV $TESTN,PFTSTN ;GET TEST NUMBER
29 016424 012700 016754 MOV #PFECH,R0 ;MOVE POWER FAIL ERROR CALL TABLE TO R0
30 016430 000412 BR 3$
31 016432 010001 1$: MOV R0,R1 ;AND COPY IT INTO R1
32 016434 005300 DEC R0 ;FORM INDEX FOR ERROR TABLE
33 016436 106300 ASLB R0
34 016440 106300 ASLB R0
35 016442 106300 ASLB R0
36 016444 103002 BCC 2$ ;IS ERROR > 37?
37 016446 062700 000240 ADD #ITEM41-$ERRTB,R0 ;YES--FORM OFFSET
38 016452 062700 002000 2$: ADD #$ERRTB,R0 ;FORM ADDRESS
39 016456 012037 016476 3$: MOV (R0)+,4$ ;GET ERROR MESSAGE (EM) POINTER
40 016462 001451 BEQ 9$ ;BRANCH IF THERE ISN'T ONE
41 016464 004737 023030 JSR PC,INCEC ;INCREMENT ERROR COUNT
42 016470 104401 001231 TYPE ,$CRLF ;CARRIAGE RETURN - LINE FEED
43 016474 104401 TYPE
44 016476 000000 4$: .WORD 0 ;'EM' POINTER GOES HERE
45 016500 162701 000041 SUB #41,R1 ;SPECIAL ERROR ITEM NUMBER?
46 016504 100440 BMI 9$ ;NO--BRANCH
47 016506 013701 001356 MOV SVSTAT,R1 ;GET STATUS/ERROR INDICATOR
48 016512 106301 ASLB R1 ;STRIP 'DONE' BIT (BIT07)
49 016514 006301 ASL R1 ;STRIP 'ERROR' BIT (BIT15)
50 016516 012702 001746 MOV #STATBL,R2 ;1ST ADDRESS ON STATUS MESSAGE POINTERS
51 016522 005003 CLR R3 ;CARRIAGE RETURN-LINE FEED SWITCH
52 016524 104401 016532 TYPE ,65$ ;TYPE ASCIZ STRING
016530 000402 BR 64$ ;GET OVER THE ASCIZ
65$: .ASCIZ / (/
64$:
53 016536 012237 016560 5$: MOV (R2)+,7$ ;MESSAGE POINTER
54 016542 006301 ASL R1 ;TYPE THIS MESSAGE?
55 016544 103013 BCC 8$ ;NO--BRANCH
56 016546 005103 COM R3 ;YES--TYPE A 'CR' & 'LF'?
57 016550 001002 BNE 6$ ;NO--BRANCH
58 016552 104401 001231 TYPE ,CRLF ;YES

```

59	016556	104401	6\$:	TYPE			
60	016560	000000	7\$:	.WORD	0		:MESSAGE POINTER GOES HERE
61	016562	005701		TST	R1		:MORE TO TYPE?
62	016564	001403		BEQ	8\$		:NO--BRANCH
63	016566	104401	040053	TYPE	.BLNKS2		:TYPE 2 BLANKS
64	016572	000761		BR	5\$		:LOOP
65	016574	001360	8\$:	BNE	5\$		:BRANCH IF NOT FINISHED
66	016576	104401	016604	TYPE	.67\$		:TYPE ASCIZ STRING
	016602	000401		BR	66\$		:GET OVER THE ASCIZ
			::67\$:	.ASCIZ	/)/		
	016606		66\$:				
67	016606	012037	016622	9\$:	MOV	(R0)+,10\$	:PICK UP DATA HEADER (DH) POINTER
68	016612	001404		BEQ	11\$		:BRANCH IF NONE
69	016614	104401	001231	TYPE	.\$CRLF		:CARRIAGE RETURN-LINE FEED
70	016620	104401		TYPE			
71	016622	000000	10\$:	.WORD	0		: 'DH' POINTER GOES HERE
72	016624	012001	11\$:	MOV	(R0)+,R1		:PICKUP DATA TABLE (DT) POINTER
73	016626	001450		BEQ	20\$		:BRANCH IF NONE
74	016630	005005		CLR	R5		:SET INDENT SWITCH
75	016632	012000		MOV	(R0)+,R0		:DATA FORMAT (DF) POINTER
76	016634	012002		MOV	(R0)+,R2		:NUMBER OF DH'S TO TYPE
77	016636	001441		BEQ	19\$		:BRANCH IF DH NUMBER IS 0
78	016640	005105		COM	R5		:NO INDENT
79	016642	104401	001231	TYPE	.\$CRLF		:CARRIAGE RETURN-LINE FEED
80	016646	112003	12\$:	MOV#	(R0)+,R3		:NUMBER OF DATA WORDS TO TYPE
81	016650	112004		MOV#	(R0)+,R4		:AND HOW TO TYPE THEM
82	016652	006004	13\$:	ROR	R4		:OCTAL OR DECIMAL?
83	016654	103403		BCS	14\$		:DECIMAL--BRANCH
84	016656	013146		MOV	@(R1)+,-(SP)		:SAVE @(R1)+ FOR TYPEOUT
	016660	104402		TYPOC			:GO TYPE--OCTAL ASCII(ALL DIGITS)
85	016662	000402		BR	15\$		
86	016664		14\$:				
	016664	013146		MOV	@(R1)+,-(SP)		:SAVE @(R1)+ FOR TYPEOUT
	016666	104405		TYPDS			:GO TYPE--DECIMAL ASCII WITH SIGN
87	016670	005303	15\$:	DEC	R3		:MORE NUMBERS TO TYPE?
88	016672	001403		BEQ	16\$		:NO--BRANCH
89	016674	104401	040053	TYPE	.BLNKS2		:TYPE 2 BLANKS
90	016700	000764		BR	13\$		:LOOP
91	016702	005302	16\$:	DEC	R2		:MORE DH'S?
92	016704	003421		BLE	20\$		:NO--BRANCH
93	016706	104401	001231	TYPE	.\$CRLF		:YES--START A NEW LINE
94	016712	005105		COM	R5		:INDENT?
95	016714	001002		BNE	17\$		:NO--BRANCH
96	016716	104401	040053	TYPE	.BLNKS2		:TYPE 2 BLANKS
97	016722	012037	016730	17\$:	MOV	(R0)+,18\$	:GET NEXT DH
98	016726	104401		TYPE			:AND TYPE IT
99	016730	000000	18\$:	.WORD	0		:DH POINTER GOES HERE
100	016732	104401	001231	TYPE	.\$CRLF		:CARRIAGE RETURN-LINE FEED
101	016736	005705		TST	R5		:INDENT?
102	016740	001342		BNC	12\$		:NO--BRANCH
103	016742	104401	040053	19\$:	TYPE	.BLNKS2	:TYPE 2 BLANKS
104	016746	000737		BR	12\$		:LOOP
105	016750	104413	20\$:	RESREG			:RESTORE R0 - R5
106	016752	000207		RTS	PC		:RETURN
107	016754	016764	017046 017100	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4		:WORDS DEFINING TABLES BELOW
108	016764	120	117 127	PFECH1:	.ASCIZ ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?		
109	017046	124	105 123	PFECH2:	.ASCIZ ?TESTNO ERR PC CPUERREG?		

110					.EVEN				
111	017100	017114	001132	016332	PFECH3:	.WORD	PFTSTN,	SERRPC,	CPSAVE,0
112	017110	000001			PFECH4:	.WORD	1		:NUMBER OF DATA HEADERS
113	017112	003				.BYTE	3		:NUMBER OF WORDS IN DATA TABLE
114	017113	000				.BYTE	0		:ALL 3 NUMBERS ARE OCTAL
115	017114	000000			PFTSTN:	.WORD	0		:CONTAINS TEST NUMBER FOR PF BIT ERROR

1

.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
*

```

```

017116 105737 001173  $TYPE:  TSTB  $TPFLG  ;; IS THERE A TERMINAL?
017122 100002          BPL  1$      ;; BR IF YES
017124 000000          HALT          ;; HALT HERE IF NO TERMINAL
017126 000430          BR  3$      ;; LEAVE
017130 010046 1$:     MOV  RO,-(SP)  ;; SAVE RO
017132 017600 000002  MOV  @2(SP),RO  ;; GET ADDRESS OF ASCIZ STRING
017136 122737 000001 001254  CMPB #APTENV,$ENV  ;; RUNNING IN APT MODE
017144 001011          BNE  62$     ;; NO, GO CHECK FOR CONSOLE
017146 132737 000100 001255  BITB #APTSPOOL,$ENVM  ;; SPOOL MESSAGE TO APT
017154 001405          BEQ  62$     ;; NO, GO CHECK FOR CONSOLE
017156 010037 017166  MCV  RO,61$  ;; SETUP MESSAGE ADDRESS FOR APT
017162 004737 015434  JSR  PC,$ATY3  ;; SPOOL MESSAGE TO APT
017166 000000 61$:    .WORD  0      ;; MESSAGE ADDRESS
017170 132737 000040 001255  62$:    BITB  #APTC SUP,$ENVM  ;; APT CONSOLE SUPPRESSED
017176 001003          BNE  60$     ;; YES, SKIP TYPE OUT
017200 112046 2$:     MOVB  (RO)+,-(SP)  ;; PUSH CHARACTER TO BE TYPED ONTO STACK
017202 001005          BNE  4$      ;; BR IF IT ISN'T THE TERMINATOR
017204 005726          TST  (SP)+  ;; IF TERMINATOR POP IT OFF THE STACK
017206 012000 60$:    MOV  (SP)+,RO  ;; RESTORE RO
017210 062716 000002 3$:     ADD  #2,(SP)  ;; ADJUST RETURN PC
017214 000002          RTI          ;; RETURN
017216 122716 000011 4$:     CMPB  #HT,(SP)  ;; BRANCH IF <HT>
017222 001430          BEQ  8$      ;;
017224 122716 000200  CMPB  #CRLF,(SP)  ;; BRANCH IF NOT <CRLF>
017230 001006          BNE  5$      ;;
017232 005726          TST  (SP)+  ;; POP <CR><LF> EQUIV
017234 104401          TYPE          ;; TYPE A CR AND LF
017236 001231          $CRLF
017240 105037 017446  CLRB  $CHARCNT  ;; CLEAR CHARACTER COUNT
017244 000755          BR  2$      ;; GET NEXT CHARACTER
017246 004737 017330 5$:     JSR  PC,$TYPEC  ;; GO TYPE THIS CHARACTER
017252 123726 001172 6$:     CMPB  $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
017256 001350          BNE  2$      ;; IF NO GO GET NEXT CHAR.
017260 013746 001170  MOV  $NULL,-(SP)  ;; GET # OF FILLER CHARS. NEEDED
                                ;; AND THE NULL CHAR.
017264 105366 000001 7$:     DECB  1(SP)  ;; DOES A NULL NEED TO BE TYPED?
017270 002770          BLT  6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
017272 004737 017330  JSR  PC,$TYPEC  ;; GO TYPE A NULL
017276 105337 017446  DECB  $CHARCNT  ;; DO NOT COUNT AS A COUNT
017302 000770          BR  7$      ;; LOOP

```

:HORIZONTAL TAB PROCESSOR

```

017304 112716 000040      8$:  MOVB  #' (SP)      ::REPLACE TAB WITH SPACE
017310 004737 017330      9$:  JSR   PC,$TYPEC    ::TYPE A SPACE
017314 132737 000007 017446  BITB  #7,$CHARCNT   ::BRANCH IF NOT AT
017322 001372          BNE   9$           ::TAB STOP
017324 005726          TST   (SP)+        ::POP SPACE OFF STACK
017326 000724          BR    2$           ::GET NEXT CHARACTER
017330
017330 105777 161624      $TYPEC: TSTB  @STKS        ::CHAR IN KYBD BUFFER?
017334 100022          BPL   10$          ::BR IF NOT
017336 017746 161620      MOV   @STKB,-(SP)   ::GET CHAR
017342 042716 177600      BIC   #177600,(SP) ::STRIP EXTRANEIOUS BITS
017346 122716 000023      CMPB  #$XOFF,(SP)  ::WAS CHAR XOFF
017352 001012          BNE   102$         ::BR IF NOT
017354
017354 105777 161600      101$: TSTB  @STKS        ::WAIT FOR CHAR
017360 100375          BPL   101$
017362 117716 161574      MOVB  @STKB,(SP)   ::GET CHAR
017366 042716 177600      BIC   #177600,(SP) ::STRIP IT
017372 122716 000021      CMPB  #$XON,(SP)  ::WAS IT XON?
017376 001366          BNE   101$         ::BR IF NOT
017400
017400 005726          102$: TST   (SP)+        ::FIX STACK
017402
017402 105777 161556      10$:  TSTB  @STPS        ::WAIT UNTIL PRINTER IS READY
017406 100375          BPL   10$
017410 116677 000002 161550  MOVB  2(SP),@STPB  ::LOAD CHAR TO BE TYPED INTO DATA REG.
017416 122766 000015 000002  CMPB  #CR,2(SP)   ::IS CHARACTER A CARRIAGE RETURN?
017424 001003          BNE   1$           ::BRANCH IF NO
017426 105037 017446      CLRB  $CHARCNT    ::YES--CLEAR CHARACTER COUNT
017432 000406          BR    $TYPEX     ::EXIT
017434 122766 000012 000002  1$:  CMPB  #LF,2(SP)   ::IS CHARACTER A LINE FEED?
017442 001402          BEQ   $TYPEX     ::BRANCH IF YES
017444 105227          INCB  (PC)+      ::COUNT THE CHARACTER
017446 000000      $CHARCNT: .WORD 0  ::CHARACTER COUNT STORAGE
017450 000207      $TYPEX: RTS    PC
    
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE   M              ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*
*STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT
*
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT
    
```

017452	017646	000000		STYPOS:	MOV	@(SP),-(SP)	;;PICKUP THE MODE
017456	116637	000001	017675		MOVB	1(SP),%0FILL	;;LOAD ZERO FILL SWITCH
017464	112637	017677			MOVB	(SP)+,%0MODE+1	;;NUMBER OF DIGITS TO TYPE
017470	062716	000002			ADD	#2,(SP)	;;ADJUST RETURN ADDRESS
017474	000406				BR	\$TYPON	
017476	112737	000001	017675	STYPOC:	MOVB	#1,%0FILL	;;SET THE ZERO FILL SWITCH
017504	112737	000006	017677		MOVB	#6,%0MODE+1	;;SET FOR SIX(6) DIGITS
017512	112737	000005	017674	STYPON:	MOVB	#5,%0CNT	;;SET THE ITERATION COUNT
017520	010346				MOV	R3,-(SP)	;;SAVE R3
017522	010446				MOV	R4,-(SP)	;;SAVE R4
017524	010546				MOV	R5,-(SP)	;;SAVE R5
017526	113704	017677			MOVB	%0MODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE
017532	005404				NEG	R4	
017534	062704	000006			ADD	#6,R4	;;SUBTRACT IT FOR MAX. ALLOWED
017540	110437	017676			MOVB	R4,%0MODE	;;SAVE IT FOR USE
017544	113704	017675			MOVB	%0FILL,R4	;;GET THE ZERO FILL SWITCH
017550	016605	000012			MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER
017554	005003				CLR	R3	;;CLEAR THE OUTPUT WORD
017556	006105			1\$:	ROL	R5	;;ROTATE MSB INTO 'C'
017560	000404				BR	3\$	;;GO DO MSB
017562	006105			2\$:	ROL	R5	;;FORM THIS DIGIT
017564	006105				ROL	R5	
017566	006105				ROL	R5	
017570	010503				MOV	R5,R3	
017572	006103			3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
017574	105337	017676			DECB	%0MODE	;;TYPE THIS DIGIT?
017600	100016				BPL	7\$	;;BR IF NO
017602	042703	177770			BIC	#177770,R3	;;GET RID OF JUNK
017606	001002				BNE	4\$	;;TEST FOR 0
017610	005704				TST	R4	;;SUPPRESS THIS 0?
017612	001403				BEQ	5\$	;;BR IF YES
017614	005204			4\$:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S



017616	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
017622	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
017626	110337	017672		MOVB	R3,8\$	::SAVE FOR TYPING
017632	104401	017672		TYPE	8\$	::GO TYPE THIS DIGIT
017636	105337	017674	7\$:	DECB	\$OCNT	::COUNT BY 1
017642	003347			BGT	2\$	::BR IF MORE TO DO
017644	002402			BLT	6\$	::BR IF DONE
017646	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
017650	000744			BR	2\$	::GO DO THE LAST DIGIT
017652	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
017654	012604			MOV	(SP)+,R4	::RESTORE R4
017656	012603			MOV	(SP)+,R3	::RESTORE R3
017660	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
017666	012616			MOV	(SP)+,(SP)	
017670	000002			RTI		::RETURN
017672	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
017673	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
017674	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
017675	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
017676	000000		\$OMODE:	.WORD	0	::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
```

```
017700          $TYPDS:
017700 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
017702 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
017704 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
017706 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
017710 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
017712 012746 020200  MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
017716 016605 000020  MOV      20(SP),R5      ;;GET THE INPUT NUMBER
017722 100004      BPL      1$                ;;BR IF INPUT IS POS.
017724 005405      NEG      R5                ;;MAKE THE BINARY NUMBER POS.
017726 112766 000055 000001  MOVB   #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
017734 005000      CLR      R0                ;;ZERO THE CONSTANTS INDEX
017736 012703 020114      MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER
017742 112723 000040      MOVB   #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
017746 005002      CLR      R2                ;;CLEAR THE BCD NUMBER
017750 016001 020104      MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
017754 160105      SUB      R1,R5            ;;FORM THIS BCD DIGIT
017756 002402      BLT      4$                ;;BR IF DONE
017760 005202      INC      R2                ;;INCREASE THE BCD DIGIT BY 1
017762 000774      BR      3$
017764 060105      ADD      R1,R5            ;;ADD BACK THE CONSTANT
017766 005702      TST      R2                ;;CHECK IF BCD DIGIT=0
017770 001002      BNE      5$                ;;FALL THROUGH IF 0
017772 105716      TSTB   (SP)              ;;STILL DOING LEADING 0'S?
017774 100407      BMI      7$                ;;BR IF YES
017776 106316      ASLB   (SP)              ;;MSD?
020000 103003      BCC      6$                ;;BR IF NO
020002 116663 000001 177777  MOVB   1(SP),-1(R3)      ;;YES--SET THE SIGN
020010 052702 000060      BIS      #'0,R2          ;;MAKE THE BCD DIGIT ASCII
020014 052702 000040      BIS      #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
020020 110223      MOVB   R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
020022 005720      TST      (R0)+          ;;JUST INCREMENTING
020024 020027 000010      CMP      R0,#10         ;;CHECK THE TABLE INDEX
020030 002746      BLT      2$                ;;GO DO THE NEXT DIGIT
020032 003002      BGT      8$                ;;GO TO EXIT
020034 010502      MOV      R5,R2          ;;GET THE LSD
020036 000764      BR      6$                ;;GO CHANGE TO ASCII
020040 105726      TSTB   (SP)+          ;;WAS THE LSD THE FIRST NON-ZERO?
020042 100003      BR      9$                ;;BR IF NO
020044 116663 177777 177776  MOVB   -1(SP),-2(R3)    ;;YES--SET THE SIGN FOR TYPING
020052 105013      CLRB   (R3)              ;;SET THE TERMINATOR
020054 012605      MOV      (SP)+,R5        ;;POP STACK INTO R5
020056 012603      MOV      (SP)+,R3        ;;POP STACK INTO R3
020060 012602      MOV      (SP)+,R2        ;;POP STACK INTO R2
020062 012601      MOV      (SP)+,R1        ;;POP STACK INTO R1
```

020064	012600		MOV	(SP)+,R0	::POP STACK INTO R0
020066	104401	020114	TYPE	\$DBLK	::NOW TYPE THE NUMBER
020072	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
020100	012616		MOV	(SP)+,(SP)	
020102	000002		RTI		::RETURN TO USER
020104	023420		\$DTBL:	10000.	
020106	001750			1000.	
020110	000144			100.	
020112	000012			10.	
020114			\$DBLK:	.BLKW 4	

.SBTTL TTY INPUT ROUTINE

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

020124	000000			.ENABL	LSB		
020126	000000			\$TKCNT:	.WORD	0	::NUMBER OF ITEMS IN QUEUE
020130	000000			\$TKQIN:	.WORD	0	::INPUT POINTER
020132				\$TKQOUT:	.WORD	0	::OUTPUT POINTER
	020133			\$TKQSRV:	.B!KB	1	::TTY KEYBOARD QUEUE
				\$TKQEND=.			
				.EVEN			

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

\*CALL:  
 \* JSR PC,\$TKINT  
 \* RETURN

020134	005037	020124		\$TKINT:	CLR	\$TKCNT	::CLEAR COUNT OF ITEMS IN QUEUE
020140	012737	020132	020126		MOV	#\$TKQSRV,\$TKQIN	::MOVE THE STARTING ADDRESS OF THE
020146	013737	020126	020130		MOV	\$TKQIN,\$TKQOUT	::QUEUE INTO THE INPUT & OUTPUT POINTERS.
020154	012737	020204	000060		MOV	#\$TKSRV,@\$TKVEC	::INITIALIZE THE KEYBOARD VECTOR
020162	012737	000200	000062		MOV	#200,@\$TKVEC+2	::'BR' LEVEL 4
020170	005777	160766			TST	@\$TKB	::CLEAR DONE FLAG
020174	012777	000100	160756		MOV	#100,@\$TKS	::ENABLE TTY KEYBOARD INTERRUPT
020202	000207				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" (^C) \$TKINT IS CALLED AND  
 \*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (SHUT)

020204	117746	160752		\$TKSRV:	MOVB	@\$TKB,-(SP)	::PICKUP THE CHARACTER
020210	042716	177600			BIC	#^C177,(SP)	::STRIP THE JUNK
020214	021627	000021			CMF	(SP),#\$XON	::IS IT A RANDOM XON?
020220	001002				BNE	30\$	::BRANCH IF NO
020222	005726				TST	(SP)+	::CLEAN RANDOM XON OFF STACK
020224	000002				RTI		::RETURN
020226				30\$:			
020226	021627	000003			CMF	(SP),#3	::IS IT A CONTROL C?
020232	001007				BNE	1\$	::BRANCH IF NO
020234	104401	021346			TYPE	,\$CNTLC	::TYPE A CONTROL-C (^C)
020240	004737	020134			JSR	PC,\$TKINT	::INIT THE KEYBOARD
020244	005726				TST	(SP)+	::CLEAN UP STACK
020246	000137	021410			JMP	SHUT	::CONTROL C RESTART
020252	021627	000007		1\$:	CMF	(SP),#7	::IS IT A CONTROL G?
020256	001004				BNE	2\$	::BRANCH IF NO
020260	022737	000176	001154		CMF	,\$SWREG,\$SWR	::IS SOFT-SWR SELECTED?
020266	001500				BEQ	6\$	::GO TO SWR CHANGE
020270				2\$:			
020270	022737	000001	020124		CMF	#1,\$TKCNT	::IS THE QUEUE FULL?
020276	001004				BNE	3\$	::BRANCH IF NO
020300	104401	001224			TYPE	,\$BELL	::RING THE TTY BELL



```

020536 104401 021376          TYPE    .,SMNEW          ;;PROMPT FOR NEW SWR
020542 005046          CLR      -(SP)          ;;CLEAR COUNTER
020544 005046          CLR      -(SP)          ;;THE NEW SWR
020546 105777 160406      7$:    TSTB    @STKS          ;;CHAR THERE?
020552 100375          BPL     7$             ;;IF NOT TRY AGAIN

020554 117746 160402      MOVB    @STKB, -(SP)      ;;PICK UP CHAR
020560 042716 177600      BIC     #'C177,(SP)    ;;MAKE IT 7-BIT ASCII

020564 021627 000003      CMP     (SP),#3        ;;IS IT A CONTROL-C?
020570 0J1015          BNE     9$             ;;BRANCH IF NOT
020572 104401 021346      TYPE    .,SCNTLC      ;;YES, ECHO CONTROL-C (^C)
020576 062706 000006      ADD     #6,SP         ;;CLEAN UP STACK
020602 123727 001151 000001  CMPB    $INTAG,#1      ;;REENABLE TTY KEYBOARD INTERRUPTS?
020610 0C1003          BNE     8$             ;;BRANCH IF NO
020612 012777 000100 160340  MOV     #100,@STKS    ;;ALLOW TTY KEYBOARD INTERRUPTS
020620 000137 021410      8$:    JMP     SHUT        ;;CONTROL-C RESTART

020624 021627 000025      9$:    CMP     (SP),#25    ;;IS IT A CONTROL-U?
020630 001005          BNE     10$           ;;BRANCH IF NOT
020632 104401 021353      TYPE    .,SCNTLU      ;;YES, ECHO CONTROL-U (^U)
020636 062706 000006      20$:   ADD     #6,SP         ;;IGNORE PREVIOUS INPUT
020642 000737          BR      19$           ;;LET'S TRY IT AGAIN

020644 021627 000015      10$:   CMP     (SP),#15    ;;IS IT A <CR>?
020650 001022          BNE     16$           ;;BRANCH IF NO
020652 005766          TST     4(SP)         ;;YES, IS IT THE FIRST CHAR?
020656 001403          BEQ     11$           ;;BRANCH IF YES
020660 016677 000002 160266  MOV     2(SP),@SWR    ;;SAVE NEW SWR
020666 062706 000006      11$:   ADD     #6,SP         ;;CLEAR UP STACK
020672 104401 001231 14$:   TYPE    .,SCRLF      ;;ECHO <CR> AND <LF>
020676 123727 001151 000001  CMPB    $INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
020704 001003          BNE     15$           ;;BRANCH IF NOT
020706 012777 000100 160244  MOV     #100,@STKS    ;;RE-ENABLE TTY KBD INTERRUPTS
020714 000002          RTI     ;;RETURN
020716 004737 017330      16$:   JSR     PC,$TYPEC    ;;ECHO CHAR
020722 021627 000060      CMP     (SP),#60     ;;CHAR < 0?
020726 002420          BLT     18$           ;;BRANCH IF YES
020730 021627 000067      CMP     (SP),#67     ;;CHAR > 7?
020734 003015          BGT     18$           ;;BRANCH IF YES
020736 042726 000060      BIC     #60,(SP)+    ;;STRIP-OFF ASCII
020742 005766 000002      TST     2(SP)        ;;IS THIS THE FIRST CHAR
020746 001403          BEQ     17$           ;;BRANCH IF YES
020750 006316          ASL     (SP)         ;;NO, SHIFT PRESENT
020752 006316          ASL     (SP)         ;;CHAR OVER TO MAKE
020754 006316          ASL     (SP)         ;;ROOM FOR NEW ONE.
020756 005266 000002      17$:   INC     2(SP)        ;;KEEP COUNT OF CHAR
020762 056616 177776      BIS     -2(SP),(SP)  ;;SET IN NEW CHAR
020766 000667          BR      7$           ;;GET THE NEXT ONE
020770 104401 001230      18$:   TYPE    $QUES      ;;TYPE ?<CR><LF>
020774 000720          BR      20$         ;;SIMULATE CONTROL-U

.DSABL  LSB

```

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

```

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

```

```

020776 011646          SRDCHR: MOV      (SP),-(SP)      ::PUSH DOWN THE PC AND
021000 016666 000004 000002      MOV      4(SP),2(SP)      ::THE PS
021006 005066 000004          CLR      4(SP)          ::GET READY FOR A CHARACTER
021012 005046          CLR      -(SP)          ::PUT NEW PS ON STACK
021014 012746 021022          MOV      #64$,-(SP)      ::PUT NEW PC ON STACK
021020 000002          RTI          ::POP NEW PC AND PS
021022
021022 005737 020124          64$:   TST      $TKCNT          ::WAIT ON A CHARACTER
021026 001775          1$:   BEQ      1$
021030 005337 020124          DEC      $TKCNT          ::DECREMENT THE COUNTER
021034 117766 177070 000004      MOVB    @STKQOUT,4(SP)      ::GET ONE CHARACTER
021042 005237 020130          INC      $TKQOUT          ::UPDATE THE POINTER
021046 023727 020130 020133      CMP     $TKQOUT,#$TKQEND  ::DID IT GO OFF OF THE END?
021054 001003          BNE     2$              ::BRANCH IF NO
021056 012737 020132 020130      MOV     #$TKQSRST,$TKQOUT ::RESET THE POINTER
021064 000002          RTI          ::RETURN
2$:

```

```

: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:

```

```

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

```

```

021066 010346          SRDLIN: MOV     R3,-(SP)      ::SAVE R3
021070 005046          CLR     -(SP)          ::CLEAR THE RUBOUT KEY
021072 012703 021322          1$:   MOV     #$TTYIN,R3      ::GET ADDRESS
021076 022703 021346          2$:   CMP     #$TTYIN+20.,R3  ::BUFFER FULL?
021102 101456          BLOS   4$              ::BR IF YES
021104 104410          RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
021106 112613          MOVB   (SP)+,(R3)        ::GET CHARACTER
021110 122713 000177          10$:  CMPB   #177,(R3)        ::IS IT A RUBOUT
021114 001022          BNE   5$              ::BR IF NO
021116 005716          TST   (SP)            ::IS THIS THE FIRST RUBOUT?
021120 001007          BNE   6$              ::BR IF NO
021122 112737 000134 021320      MOVB   #' \,9$          ::TYPE A BACK SLASH
021130 104401 021320          TYPE   ,9$
021134 012716 177777          MOV   #-1,(SP)        ::SET THE RUBOUT KEY
021140 005303          6$:   DEC     R3          ::BACKUP BY ONE
021142 020327 021322          CMP   R3,$$TTYIN      ::STACK EMPTY?
021146 103434          BLO   4$              ::BR IF YES
021150 111337 021320          MOVB   (R3),9$        ::SETUP TO TYPEOUT THE DELETED CHAR.
021154 104401 021320          TYPE   ,9$          ::GO TYPE
021160 000746          BP    2$              ::GO READ ANOTHER CHAR.
021162 005716          5$:   TST   (SP)          ::RUBOUT KEY SET?
021164 001406          BEQ   7$              ::BR IF NO
021166 112737 000134 021320      MOVB   #' \,9$          ::TYPE A BACK SLASH
021174 104401 021320          TYPE   ,9$
021200 005016          CLR   (SP)            ::CLEAR THE RUBOUT KEY
021202 122713 000025          7$:   CMPB   #25,(R3)      ::IS CHARACTER A CTRL U?
021206 001003          BNE   8$              ::BR IF NO
8$:

```

021210	104401	021353		TYPE	,SCNTLU	:::TYPE A CONTROL 'U'
021214	000726			BR	1\$	:::GO START OVER
021216	122713	000022	8\$:	CMPB	#22,(R3)	:::IS CHARACTER A 'R'?
021222	001011			BNE	3\$	:::BRANCH IF NO
021224	105013			CLRB	(R3)	:::CLEAR THE CHARACTER
021226	104401	001231		TYPE	,SCRLF	:::TYPE A 'CR' & 'LF'
021232	104401	021322		TYPE	,STTYIN	:::TYPE THE INPUT STRING
021236	000717			BR	2\$	:::GO PICKUP ANOTHER CHACTER
021240	104401	001230	4\$:	TYPE	,SQUES	:::TYPE A '?'
021244	000712			BR	1\$	:::CLEAR THE BUFFER AND LOOP
021246	111337	021320	3\$:	MOVB	(R3),9\$	:::ECHO THE CHARACTER
021252	104401	021320		TYPE	,9\$	
021256	122723	000015		CMPB	#15,(R3)+	:::CHECK FOR RETURN
021262	001305			BNE	2\$	:::LOOP IF NOT RETURN
021264	105063	177777		CLRB	-1(R3)	:::CLEAR RETURN (THE 15)
021270	104401	001232		TYPE	,SLF	:::TYPE A LINE FEED
021274	005726			TST	(SP)+	:::CLEAN RUBOUT KEY FROM THE STACK
021276	012603			MOV	(SP)+,R3	:::RESTORE R3
021300	011646			MOV	(SP)-(SP)	:::ADJUST THE STACK AND PUT ADDRESS OF THE
021302	016666	000004	000002	MOV	4(SP),2(SP)	::: FIRST ASCII CHARACTER ON IT
021310	012766	021322	000004	MOV	#STTYIN,4(SP)	
021316	000002			RTI		:::RETURN
021320	000		9\$:	.BYTE	0	:::STORAGE FOR ASCII CHAR. TO TYPE
021321	000			.BYTE	0	:::TERMINATOR
021322				STTYIN:	.BLKB	20.
021346	136	103	015	SCNTLC:	.ASCIZ	/'^C/<15><12>
021353	136	125	015	SCNTLU:	.ASCIZ	/'^U/<15><12>
021360	136	107	015	SCNTLG:	.ASCIZ	/'^G/<15><12>
021365	015	012	123	SMSWR:	.ASCIZ	<15><12>/SWR = /
021376	040	040	116	SMNEW:	.ASCIZ	/ NEW = /
				.EVEN		
2						
3	021410	005737	000042	SHUT:	TST	#42
4	021414	001002			BNE	1\$
5	021416	000137	003312		JMP	START2
6	021422	005037	001330	1\$:	CLR	DRVSEL
7	021426	000137	015142		JMP	SEOP
						:::ANY MONITOR PRESENT ?
						:::BR IF YES
						:::GO TO 'START2'
						:::FUDGE NO DRIVES SELECTED
						:::RETURN CONTROL TO MONITOR



1

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*CALL
*          SCOPE          ;;SCOPE=IOT
    
```

```

021432          $SCOPE:
021432 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
021434 032777 040000 157512 1$: BIT #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
021442 001402          BEQ 9$          ;;NO IF SW14=0
021444 000137 022014          JMP $OVER          ;;JUMP OVER SCOPE ROUTINE
021450          9$:
021450 000416          :*****START OF CODE FOR THE XOR TESTER*****
          $XTSTR: BR 6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
          ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
021452 013746 000004          MOV @ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
021456 012737 021476 000004          MOV #5$,@ERRVEC          ;;SET FOR TIMEOUT
021464 005737 177060          TST @177060          ;;TIME OUT ON XOR?
021470 012637 000004          MOV (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
021474 000531          BR $SVLAD          ;;GO TO THE NEXT TEST
021476 022626          5$: CMP (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
021500 012637 000004          MOV (SP)+,@ERRVEC          ;;RESTORE THE ERROR VECTOR
021504 000474          BR 7$          ;;LOOP ON THE PRESENT TEST
021506          6$:*****END OF CODE FOR THE XOR TESTER*****
021506 105737 001117          2$: TSTB $ERFLG          ;;HAS AN ERROR OCCURRED?
021512 001502          BEQ 3$          ;;BR IF NO
021514 022737 177777 016332          CMP #-1,CPSAVE          ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
021522 001455          BEQ 2003$          ;;KICK AROUND ROUTINE IF SO
021524 013746 000004          MOV ERRVEC,-(SP)          ;;SAVE CONTENTS OF ERROR VECTOR
021530 012737 021546 000004          MOV #2000$,ERRVEC          ;;SETUP 'TRAP' RETURN ADDRESS
021536 013737 177766 016332          MOV 177766,CPSAVE          ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
021544 000406          BR 2001$
021546 012737 177777 016332 2000$: MOV #-1,CPSAVE          ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
021554 012716 021562          MOV #2001$,(SP)          ;;SETUP RETURN ADDRESS
021560 000002          RTI
021562 012637 000004          2001$: MOV (SP)+,ERRVEC          ;;RESTORE CONTENTS OF ERROR VECTOR

021566 022737 177777 016332 2002$: CMP #-1,CPSAVE          ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
021574 001430          BEQ 2003$          ;;BRANCH IF SO
021576 032737 000001 016332          BIT #BIT00,CPSAVE          ;;SEE IF THE POWER MONITOR BIT IS ON
021604 001424          BEQ 2003$          ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
021606 042737 000001 177766          BIC #BIT00,177766          ;;CLEAR THE BIT FOUND TO BE SET
021614 013746 001154          MOV SWR,-(SP)          ;;SAVE SWR ADDRESS
021620 017646 000000          MOV @($P),-(SP)          ;;SAVE SWR VALUE
021624 012737 000176 001154          MOV #176,SWR          ;;GET SOFTWARE SWR ADDRESS
021632 011677 157316          MOV (SP),@SWR          ;;GET CURRENT SWR VALUE
021636 042777 001000 157310          BIC #BIT09,@SWR          ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
021644 104177          EMT 177          ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
021646 012676 000000          MOV (SP)+,@(SP)          ;;RESTORE SWR TO ORIGINAL VALUE
021652 012637 001154          MOV (SP)+,SWR          ;;RESTORE SWR ADDRESS
    
```

```

021656          2003$:
021656 123737 001131 001117  CMPB  $SERMAX,$SERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
021664 101015          BHI   3$                ;;BR IF NO
021666 032777 001000 157260  BIT   #BIT09,@SWR      ;;LOOP ON ERROR?
021674 001404          BEQ   4$                ;;BR IF NO
021676 013737 001124 001122  7$:  MOV   $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
021704 000443          BR    $OVER
021706 105037 001117          CLR   $SERFLG          ;;ZERO THE ERROR FLAG
021712 005037 001220          CLR   $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
021716 000412          BR    1$                ;;ESCAPE TO THE NEXT TEST
021720 032777 004000 157226  3$:  BIT   #BIT11,@SWR      ;;INHIBIT ITERATIONS?
021726 001006          BNE   1$                ;;BR IF YES
021730 005237 001120          INC   $ICNT          ;;INCREMENT ITERATION COUNT
021734 023737 001220 001120  CMP   $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
021742 002024          BGE   $OVER          ;;BR IF MORE ITERATION REQUIRED
021744 012737 000001 001120  1$:  MOV   #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
021752 013737 022030 001220  MOV   $SMXCNT,$TIMES  ;;SET NUMBER OF ITERATIONS TO DO
021760 105237 001116          $SVLAD: $STSTM          ;;COUNT TEST NUMBERS
021764 113737 001116 001240  MOVB  $STSTM,$TESTN    ;;SET TEST NUMBER IN APT MAILBOX
021772 011637 001122          MOV   (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
021776 011637 001124          MOV   (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
022002 005037 001222          CLR   $ESCAPE        ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
022006 112737 000001 001131  MOVB  #1,$SERMAX      ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
022014 013777 001116 157134  SOVER:  MOV   $STSTM,@DISPLAY  ;;DISPLAY TEST NUMBER
022022 013716 001122          MOV   $LPADR,(SP)    ;;FUDGE RETURN ADDRESS
022026 000002          RTI          ;;FIXES PS
022030 000001          SMXCNT: 1      ;;MAX. NUMBER OF ITERATIONS
  
```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

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

```

022032
022032 010046
022034 010146
022036 010246
022040 010346
022042 010446
022044 010546
022046 016646 000022
022052 016646 000022
022056 016646 000022
022062 016646 000022
022066 000002
    
```

```

$SAVREG:
MOV R0,-(SP)      ;;PUSH R0 ON STACK
MOV R1,-(SP)      ;;PUSH R1 ON STACK
MOV R2,-(SP)      ;;PUSH R2 ON STACK
MOV R3,-(SP)      ;;PUSH R3 ON STACK
MOV R4,-(SP)      ;;PUSH R4 ON STACK
MOV R5,-(SP)      ;;PUSH R5 ON STACK
MOV 22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP)  ;;SAVE PS OF CALL
MOV 22(SP),-(SP)  ;;SAVE PC OF CALL
RTI
    
```

```

022070
022070 012666 000022
022074 012666 000022
022100 012666 000022
022104 012666 000022
022110 012605
022112 012604
022114 012603
022116 012602
022120 012601
022122 012600
022124 000002
    
```

```

*RESTORE R0-R5
*CALL:
* RESREG
*$RESREG:
MOV (SP)+,22(SP)  ;;RESTORE PC OF CALL
MOV (SP)+,22(SP)  ;;RESTORE PS OF CALL
MOV (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5      ;;POP STACK INTO R5
MOV (SP)+,R4      ;;POP STACK INTO R4
MOV (SP)+,R3      ;;POP STACK INTO R3
MOV (SP)+,R2      ;;POP STACK INTO R2
MOV (SP)+,R1      ;;POP STACK INTO R1
MOV (SP)+,R0      ;;POP STACK INTO R0
RTI
    
```

.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.

022126	010046		\$TRAP: MOV	RO, -(SP)	::SAVE R0
022130	016600	000002	MOV	2(SP), R0	::GET TRAP ADDRESS
022134	005740		TST	-(R0)	::BACKUP BY 2
022136	111000		MOVB	(R0), F0	::GET RIGHT BYTE OF TRAP
022140	006300		ASL	R0	::POSITION FOR INDEXING
022142	016000	022162	MOV	\$TRPAD(R0), R0	::INDEX TO TABLE
022146	000200		RTS	R0	::GO TO ROUTINE

::THIS IS USE TO HANDLE THE 'GETPRI' MACRO

022150	011646		\$TRAP2: MOV	(SP), -(SP)	::MOVE THE PC DOWN
022152	016666	000004	MOV	4(SP), 2(SP)	::MOVE THE PSW DOWN
022160	000002	000002	RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			:	ROUTINE	
			:	-----	
022162	022150		\$TRPAD:	.WORD	\$TRAP2
022164	017116			\$TYPE	::CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
022166	017476			\$TYPOC	::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
022170	017452			\$TYPOS	::CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
022172	017512			\$TYPON	::CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
022174	017700			\$TYPDS	::CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
022176	020524			\$GTSWR	::CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
022200	020434			\$CKSWR	::CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
022202	020776			\$RDCHR	::CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
022204	021066			\$RDLIN	::CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
022206	022032			\$SAVREG	::CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
022210	022070			\$RESREG	::CALL=RESREG TRAP+13(104413) RESTORE R0-R5 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

022212	016637	000002	022242	\$\$SB2D:	MOV	2(SP),1\$	::SAVE BINARY NUMBER
022220	012746	022242			MOV	#1\$,-(SP)	::SET POINTER
022224	004737	022246			JSR	PC,@#\$\$DB2D	::CALL DOUBLE LENGTH CONVERT
022230	062716	000005			ADD	#5,(SP)	::ONLY ALLOW FIVE CHARACTERS
022234	012666	000002			MOV	(SP)+,2(SP)	::PICKUP POINTER
022240	000207				RTS	PC	::RETURN
022242	000000	000000		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,@#SDB2D      ;; THE FIRST ADDRESS OF ASCII
*      RETURN                               ;; IS ON THE STACK

```

```

022246 104412          $DB2D: SAVREG      ;; SAVE REGISTERS
022250 016602 000002  MOV      2(SP),R2      ;; PICKUP THE DATA POINTER
022254 012700 022426  MOV      #$DECVL,R0     ;; GET ADDRESS OF '$DECVL' STRING
022260 010066 000002  MOV      R0,2(SP)      ;; PUT ADDRESS OF ASCII STRING ON STACK
022264 012201        MOV      (R2)+,R1      ;; PICKUP THE BINARY NUMBER
022266 012202        MOV      (R2)+,R2
022270 012737 000012 222344  MOV      #10,4$       ;; SET UP TO DO 10 CONVERSIONS
022276 012704 022356  MOV      #$STNPWR,R4    ;; ADDRESS OF TEN POWER
022302 012705 022360  MOV      #$STNPWR+2,R5
022306 005003        1$: CLR      R3              ;; CLEAR PARTIAL
022310 161401        2$: SUB      (R4),R1      ;; SUBTRACT TEN POWER
022312 005602        SBC      R2
022314 161502        SUB      (R5),R2
022316 002402        BLT      3$              ;; BR IF TEN POWER TO LARGE
022320 005203        INC      R3              ;; ADD 1 TO PARTIAL
022322 000772        BR       2$              ;; LOOP
022324 062401        3$: ADD      (R4)+,R1      ;; RESTORE SUBTRACTED VALUE
022326 005502        ADC      R2
022330 062402        ADD      (R4)+,R2
022332 022525        CMP      (R5)+,(R5)+      ;; MOVE TO NEXT TEN POWER
022334 052703 000060  BIS      #'0,R3        ;; CHANGE PARTIAL TO ASCII
022340 110320        MOVB     R3,(R0)+      ;; SAVE IT
022342 005327        DEC      (PC)+        ;; DONE?
022344 000000        4$: .WORD    0
022346 001357        BNE      1$              ;; BR IF NO
022350 105020        CLRB     (R0)+        ;; TERMINATOR
022352 104413        RESREG
022354 000207        RTS      PC          ;; RESTORE REGISTERS
022356 145000        $STNPWR: 145000      ;; RETURN
022360 035632        35632          ;; 1.0E09
022362 160400        160400          ;; 1.0E08
022364 002765        2765           ;; 1.0E07
022366 113200        113200          ;; 1.0E06
022370 000230        230            ;; 1.0E05
022372 041100        041100          ;; 1.0E04
022374 000017        17             ;; 1.0E03
022376 103240        103240          ;; 1.0E02
022400 000001        1              ;; 1.0E01
022402 023420        23420          ;; 1.0E00
022404 000000        0
022406 001750        1750           ;; 1.0E03
022410 000000        0
022412 000144        144            ;; 1.0E02
022414 000000        0

```

022416 000012  
022420 000000  
022422 000001  
022424 000000  
022426

12  
0  
1  
0  
\$DECVL: .BLKB 12.

:::1.0E01  
:::1.0E00  
:::RESERVE STORAGE FOR ASCII STRING

.SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS

\*\*\*\*\*  
 \*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE  
 \*LEADING NUMBERS.

\*CALL  
 \*     MOV     #NUMADR,-(SP)     ;;FIRST ADDRESS OF ASCIZ STRING  
 \*     JSR     PC,@#\$\$SUPRS

022442	01J046		\$\$SUPRS: MOV	RO,-(SP)	::SAVE RO
022444	016600	000004	MOV	4(SP),RO	::PICKUP THE POINTER
022450	105710		1\$: TSTB	(RO)	::TERMINATEOR?
022452	001403		BEQ	2\$	::BR IF YES
022454	122720	000060	CMPB	#'0,(RO)+	::IS THIS AN ASCII '0' ?
022460	001773		BEQ	1\$	::BR IF YES
022462	005300		2\$: DEC	RO	::BACKUP BY '1'
022464	010037	022472	MOV	RO,3\$	::SAVE FOR TYPING
022470	104401		TYPE		::GO TYPE
022472	000000		3\$: .WORD	0	::ASCIZ POINTER GOES HERE
022474	012600		MOV	(SP)+,RO	::RESTORE RO
022476	012616		MOV	(SP)+,(SP)	::RESTORE THE STACK
022500	000207		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

022502  
 022502 010046  
 022504 010146  
 022506 010246  
 022510 013700 022602  
 022514 013701 022600  
 022520 012702 177771  
 022524 006300  
 022526 006101  
 022530 005202  
 022532 001374  
 022534 063700 022602  
 022540 005501  
 022542 063701 022600  
 022546 062700 001057  
 022552 005501  
 022554 062701 047401  
 022560 010037 022602  
 022564 010137 022600  
 022570 012602  
 022572 012601  
 022574 012600  
 022576 000207  
 022600 176543  
 022602 123456

\$RAND:  
 MOV R0,-(SP) ;:PUSH R0 ON STACK  
 MOV R1,-(SP) ;:PUSH R1 ON STACK  
 MOV R2,-(SP) ;:PUSH R2 ON STACK  
 MOV \$LONUM,R0 ;:SET R0 WITH LOW  
 MOV \$HINUM,R1 ;:SET R1 WITH HIGH  
 MOV #-7,R2 ;:SET SHIFT COUNT  
 1\$: ASL R0 ;:SHIFT R0 LEFT AND  
 ROL R1 ;:ROTATE CARRY INTO R1 AND  
 INC R2 ;:CHECK FOR DONE  
 BNE 1\$ ;:CONTINUE SHIFT LOOP  
 ADD \$LONUM,R0 ;:ADD NUMBER TO MAKE X 129  
 ADC R1 ;:PROPOGATE CARRY  
 ADD \$HINUM,R1 ;:ADD NUMBER TO MAKE Y 129  
 ADD #1057,R0 ;:ADD LOW CONSTANT  
 ADC R1 ;:PROPOGATE CARRY  
 ADD #47401,R1 ;:ADD HIGH CONSTANT  
 MOV R0,\$LONUM ;:SAVE R0  
 MOV R1,\$HINUM ;:SAVE R1  
 MOV (SP)+,R2 ;:POP STACK INTO R2  
 MOV (SP)+,R1 ;:POP STACK INTO R1  
 MOV (SP)+,R0 ;:POP STACK INTO R0  
 RTS PC ;:RETURN  
 \$HINUM: .WORD 176543  
 \$LONUM: .WORD 123456

.SBTTL INTEGER DIVIDE ROUTINE

2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36

```

*****
*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
*SAME SIGN AS THE DIVIDEND.
*CALL:
*   MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
*   MOV     HIGH DIVIDEND,-(SP)   ;;AS LARGE AS THE DIVISOR
*   MOV     DIVISOR,-(SP)
*   JSR    PC,$DIV
*   RETURN                                ;;QUOTIENT & REMAINDER ARE ON THE STACK
*
*   STACK  NO ERROR      OVERFLOW      DIVIDE BY ZERO
*   -----
*   TOP    REMAINDER     ALL ZEROS      ALL ONES
*   +2     QUOTIENT      ALL ZEROS      ALL ONES
*
*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').
    
```

```

24 022604 104412
25 022606 016605 000026
26 022612 005004
27 022614 016602 000030
28 022620 016603 000032
29 022624 005000
30 022626 005001
31 022630 004737 022652
32 022634 010166 000030
33 022640 010366 000032
34 022644 104413
35 022646 012616
36 022650 000207
    
```

```

$DIV: SAVREG      ;STORE R0 - R5
      MOV     26(SP),R5 ;DIVISOR
      CLR    R4        ;OTHER DIVISOR WORD
      MOV    30(SP),R2 ;UPPER DIVIDEND WORD
      MOV    32(SP),R3 ;LOWER DIVIDEND WORD
      CLR    R0        ;CLEAR OTHER DIVIDEND REGISTERS
      CLR    R1
      JSR    PC,M.DPID ;GO TO THE DIVIDE ROUTINE
      MOV    R1,30(SP) ;REMAINDER ON THE STACK
      MOV    R3,32(SP) ;QUOTIENT ON THE STACK
      RESREG ;RESTORE R0 - R5
      MOV    (SP)+,(SP) ;MOVE RETURN UP THE STACK
      RTS    PC
    
```

```

1      .SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE
2
3      :CALL
4      JSR    PC,M.DPID
5
6      DIVIDEND = R0-R1-R2-R3 (R0=MSD)
7      DIVISOR  = R4-R5 (R4=MSD)
8
9      :RETURN
10
11      REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
12      QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
13
14 022652 012746 000040  M.DPID: MOV    #40,-(SP)      ;COUNTER FOR DIVISION CYCLES
15 022656 010446          MOV    R4,-(SP)      ;HIGH ORDER
16 022660 010546          MOV    R5,-(SP)      ;LOW ORDER DIVISOR TO THE STACK
17 022662 005466 000002  NEG    2(SP)          ;FORM NEGATIVE
18 022666 005416          @SP          ;VERSION OF THE DIVISOR
19 022670 005666 000002  SBC    2(SP)
20 022674 061601          ADD    @SP,R1
21 022676 005500          ADC    R0              ;PERFORM THE INITIAL SUBTRACTION
22 022700 066600 000002  ADD    2(SP),R0
23 022704 103445          BCS   M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
24 022706 005046          CLR    -(SP)         ;THIS IS A LONGER LASTING CARRY BIT
25 022710 006103  M.DP40: ROL    R3
26 022712 006102          ROL    R2
27 022714 006101          ROL    R1
28 022716 006100          ROL    R0
29 022720 005716          TST   @SP              ;TEST 'CARRY' INDICATOR
30 022722 001410          BEQ   M.DP41          ;IF NO 'CARRY' THEN ADD ELSE SUBTRACT
31 022724 005016          CLR    @SP              ;CLEAR UP FOR NEXT TIME
32 022726 066601 000002  ADD    2(SP),R1
33 022732 005500          ADC    R0              ;ADD -(DIVISOR)
34 022734 005516          ADC    @SP              ;SET 'CARRY'
35 022736 066600 000004  ADD    4(SP),R0;<-
36 022742 000404          BR    M.DP42
37 022744 060501  M.DP41: ADD    R5,R1
38 022746 005500          ADC    R0              ;ADD +(DIVISOR)
39 022750 005516          ADC    @SP              ;SET 'CARRY'
40 022752 060400          ADD    R4,R0;<-
41 022754 005516  M.DP42: ADC    @SP              ;SET 'CARRY'
42 022756 005716          TST   @SP              ;TEST THE UPDATE INDICATOR
43 022760 001401          BEQ   .+4              ;IF ZERO FORGET IT
44 022762 005203          INC   R3              ;NO CARRY POSSIBLE HERE
45 022764 005366 000006  DEC   6(SP)            ;DECREMENT COUNTER
46 022770 003347          BGT   M.DP40          ;BRANCH IF MORE TO DO
47 022772 006003          ROR   R3
48 022774 103404          BCS   M.DP44
49 022776 060501          ADD    R5,R1
50 023000 005500          ADC    R0
51 023002 060400          ADD    R4,R0
52 023004 000241          CLC
53 023006 006103  M.DP44: ROL    R3
54 023010 062706 000010  ADD    #10,SP          ;ADJUST STACK BY 4 WORDS
55 023014 000242          CLV
56 023016 000207          RTS    PC
57 023020 062706 000006  M.DP50: ADD    #6,SP
    
```

58 023024 000262  
59 023026 000207

SEV  
RTS PC

```

1      ;THIS ROUTINE IS USED TO INCREMENT THE ERROR COUNT FOR EACH DRIVE BEING
2      ;TESTED. IF THE TOTAL ERRORS ON ANY DRIVE EXCEEDS THE MAXIMUM ALLOWABLE
3      ;ERRORS DESIGNATED IN LOCATION 'ERMAX', A MESSAGE WILL BE TYPED AND THE
4      ;DRIVE IN ERROR WILL BE DROPPED FROM THE TEST.
5
6 023030 010046      INCEC:  MOV    RO,-(SP)      ;SAVE RO
7 023032 013700 031366  MOV    RMADR,RO      ;GET RM BASE ADDRESS
8 023036 016000 000010  MOV    RMCS2(R0),RO  ;GET CONTENTS OF RMCS2
9 023042 042700 177770  BIC    #^C7,RO      ;SAVE UNIT SELECT BITS
10 023046 136037 031356 001330 BITB   ATABIT(R0),DRVSEL ;WAS THIS DRIVE SELECTED FOR TEST
11 023054 001432      BEQ    1$           ;BR IF NO
12 023056 105260 001512  INCB   ERRCN(R0)     ;INCREMENT ERROR COUNT
13 023062 126037 001512 001316 CMPB   ERRCN(R0),ERMAX ;EXCEEDED ERROR LIMIT ?
14 023070 103424      BLO    1$           ;BR IF NO
15 023072 104401 001231  TYPE   ,$CRLF       ;CR-LF
16 023076 104401 037273  TYPE   ,MSDRIV      ;TYPE 'DRIVE'
17 023102 010046      MOV    RO,-(SP)     ;SAVE RO FOR TYPEOUT
    023104 104403      TYPOS   ;GO TYPE--OCTAL ASCII
    023106 002        .BYTE   2           ;TYPE 2 DIGIT(S)
    023107 000        .BYTE   0           ;SUPPRESS LEADING ZEROS
18 023110 104401 037302  TYPE   ,DROP        ;TYPE 'DROPPED'
19 023114 104401 037160  TYPE   ,COMMA       ;TYPE ','
20 023120 104401 037313  TYPE   ,EXCEED      ;TYPE 'EXCEEDED MAXIMUM ERROR LIMIT'
21 023124 146037 031356 001330 BICB   ATABIT(R0),DRVSEL ;DESELECT DRIVE FROM TEST
22 023132 005337 015314  DEC    $EOPCT       ;ADJUST 'EOP' COUNT
23 023136 000137 015142  JMP    $EOP         ;RETURN '0 $EOP'
24 023142 012600      1$:    MOV    (SP)+,RO      ;RESTORE RO
25 023144 000207      RTS    PC
    
```

```

26      ;THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTER,
27      ;AND DRIVERS, THEN SELECTS THE DRIVE.
28
29      ;CALL:
30      ;
31      ;
32      ;
    
```

```

33      ;CALL:  JSR    PC,CNTCLR      ;CALL TO ROUTINE
34
35
36
37
    
```

```

38
39      CNTCLR: MOV    RMADR,R4      ;GET RMCS1 BASE ADDRESS
40      MOV    #CLR,RMCS2(R4)     ;ISSUE MASSBUS CLEAR AND
41      BIC    #^C7,CHKDRV        ;SAVE UNIT SELECT BITS
42      MOV    CHKDRV,RMCS2(R4)   ;SELECT THE DRIVE
43      RTS    PC                ;RETURN
    
```

```

44      ;SET 'LPTAVL' TO THE PROPER STATE.
45      ; LPTAVL = 0 IF NO LINE PRINTER AVAILABLE
46      ; LPTAVL = 1 IF LINE PRINTER IS AVAILABLE
47
48      ;CALL
49
50
51
    
```

```

52      ;CALL  JSR    PC,LP.AVL
53
54
55
56      LP.AVL: CLR    LPTAVL      ;START WITH NO PRINTER AVAILABLE
57      MOV    #1$,ERRVEC        ;SETUP THE TIMEOUT VECTOR
58      CLR    ERRVEC+2
59      TST   @LPS              ;IS THERE A LINE PRINTER?
60      INC   LPTAVL           ;YES--SET AVAILABLE SWITCH
61      BR   2$
62      1$:  CMP    (SP)+,(SP)+    ;NO--POP STACK
63      2$:  MOV    #ERRVEC+2,ERRVEC ;RESTORE TIMEOUT VECTOR
64      RTS    PC                ;RETURN
    
```

```

55
56 ;THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
57 ;AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK
58 ;"CLKSTA" WILL INDICATE THE CLOCK TYPE
59 ; 0= NO CLOCK
60 ;+1= KW11-P
61 ;-1= KW11-L
62 ;THIS ROUTINE WILL ALSO SETUP "TICKMS" (TIME
63 ;PER CLOCK TICK IN MILLISECONDS) AND "TICKUS"
64 ;(TIME PER CLOCK TICK IN MICROSECONDS) AS
65 ;PER SW00.
66 ;SW00=0 -- 60HZ
67 ;SW00=1 -- 50HZ
68 ;CALL
69 ;
70 ; JSR PC,ST.CLK
71 ; RETURN
72 023240 010146 ST.CLK: MOV R1,-(SP) ;SAVE R1
73 023242 012701 00C006 MOV #ERRVEC+2,R1 ;SAVE AND SETUP TIMEOUT VECTOR
74 023246 011146 MOV (R1),-(SP)
75 023250 005011 CLR (R1) ;LEVEL 0
76 023252 014146 MOV -(R1),-(SP)
77 023254 012711 023304 MOV #1$, (R1) ;GO TO 1$ ON TIMEOUT
78 023260 005037 001342 CLR CLKSTA ;SET CLOCK STATUS TO NO CLOCK
79 023264 005777 156244 TST @PKCS ;IS THERE A KW11-P?
80 023270 012737 000001 001342 MOV #1,CLKSTA ;YES--SET STATUS TO KW11-P
81 023276 004737 023406 JSR PC,ST.PCLK ;START THE KW11-P
82 023302 000414 BR 3$ ;GO TO EXIT
83 023304 022626 1$: CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
84 023306 012711 023332 MOV #2$, (R1) ;IF TIMEOUT GO TO 2$
85 023312 005777 156230 TST @LKS ;IS THERE A KW11-L?
86 023316 012737 177777 001342 MOV #-1,CLKSTA ;YES-- SET STATUS TO KW11-L
87 023324 004737 023450 JSR PC,ST.LCLK ;START THE KW11-L
88 023330 000401 BR 3$ ;EXIT
89 023332 022626 2$: CMP (SP)+,(SP)+ ;CLEAN UP THE STACK
90 023334 012621 3$: MOV (SP)+,(R1)+ ;RESTORE THE TIMEOUT VECTOR
91 023336 012621 MOV (SP)+,(R1)+
92 023340 012601 MOV (SP)+,R1 ;RESTORE R1
93 023342 032737 000100 001314 BIT #SW06,C.SWR ;50HZ OR 60HZ?
94 023350 001407 BEQ 4$ ;BRANCH IF 60
95 023352 012737 000020 001344 MOV #20,TICKMS ;SETUP TIME PER
96 023360 012737 047040 001346 MOV #20000.,TICKUS ;TICK FOR 50HZ
97 023366 000406 BR 5$
98 023370 012737 000016 001344 4$: MOV #16,TICKMS ;SETUP TIME PER
99 023376 012737 040432 001346 MOV #16666.,TICKUS ;TICK FOR 60HZ
100 023404 000207 5$: RTS PC ;RETURN
101
102 023406 ST.PCLK:
103 023406 032737 000040 001314 BIT #SW05,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
104 023414 001014 BNE 1$ ;NO--BRANCH
105 023416 012777 023504 156104 MOV #SRVCLK,@PKV ;SETUP THE KW11-P VECTOR
106 023424 012777 000300 156100 MOV #300,@PKV+2
107 023432 012777 000001 156076 MOV #1,@PKB ;COUNT ONE TICK
108 023440 012777 000115 156066 MOV #115,@PKCS ;"INT.EN.",COUNT DOWN", "MODE 1 (REPEAT)",
109 ;"LINE FREQ", AND "RUN"
110 023446 000207 1$: RTS PC ;RETURN
111

```

```

112 023450          ST.LCLK:
113 023450 032737 000040 001314 BIT      #SW05,C.SWR ;ALLOW SOFTWARE TIMEOUTS?
114 023456 001011          BNE      1$ ;NO--BRANCH
115 023460 012777 023504 156054 MOV      #SRVCLK,@LKV ;SETUP THE KW11-L VECTOR
116 023466 012777 000300 156050 MOV      #300,@LKV+2
117 023474 012777 000100 156044 MOV      #100,@LKS ;START THE KW11-L
118 023502 000207          1$:      RTS      PC ;RETURN
119
120 023504 013746 001344          SRVCLK: MOV      TICKMS,-(SP) ;TIME PER TICK IN MILLISECONDS
121 023510 004737 035034          JSR      PC,RMTMR ;COUNT THE ELAPSED TIME
122 023514 000002          RTI      ;RETURN AFTER INTERRUPT
123
124          ;THIS ROUTINE SETS UP DEFAULT PARAMETER VALUES WHEN THE PROGRAM IS
125          ;STARTED OR WHEN THE VALUE OF BIT00 IN 'C.SWR' IS CHANGED.
126          ;CALL
127          ;
128          ;      JSR      PC,LODFLT
129          ;      RETURN
130
130 023516          LODFLT:
131 023516 010046          MOV      R0,-(SP) ;:PUSH R0 ON STACK
132 023520 010146          MOV      R1,-(SP) ;:PUSH R1 ON STACK
133 023522 010246          MOV      R2,-(SP) ;:PUSH R2 ON STACK
134 023524 010346          MOV      R3,-(SP) ;:PUSH R3 ON STACK
135 023526 012737 013377 001332 MOV      #13377,TSTNMS ;SELECT TESTS 0-7(SEEK) AND 11,12 & 14(TIMING)
136 023534 012737 000000 001334 MOV      #0,TSTNMS+2 ;SELECT TESTS
137 023542 012700 002456          MOV      #DFLT,R0 ;DEFAULT PARAMETERS POINTER
138 023546 012701 002726          MOV      #PRMO,R1 ;TABLE POINTER
139 023552 010102          MOV      R1,R2 ;STOP ADDRESS
140 023554 012021          1$:      MOV      (R0)+,(R1)+ ;MOVE DEFAULT PARAMETERS INTO
141 023556 020002          CMP      R0,R2 ;RUN TIME TABLES ** DONE?
142 023560 103775          BLO      1$ ;NO--BRANCH
143 023562 032737 000001 001314 BIT      #BIT00,C.SWR ;16 BIT MODE ?
144 023570 001012          BNE      3$ ;BR IF 18 BIT MODE
145 023572 012737 000036 002426 MOV      #30,.PRMLMT+16 ;SET 'FS' LIMIT TO 30.
146 023600 012737 000036 002430 MOV      #30,.PRMLMT+20 ;SET 'LS' LIMIT TO 30.
147 023606 012737 160400 001452 MOV      #-<256.*31.>,TRCKWC ;WORD COUNT FOR A 16 BIT TRACK
148 023614 000411          BR      4$ ;CONTINUE
149 023616 012737 000035 002426 3$:      MOV      #29,.PRMLMT+16 ;SET 'FS' LIMIT TO 29.
150 023624 012737 000035 002430 MOV      #29,.PRMLMT+20 ;SET 'LS' LIMIT TO 29.
151 023632 012737 161000 001452 MOV      #-<256.*30.>,TRCKWC ;WORD COUNT FOR AN 18 BIT TRACK
152 023640 012701 002352          4$:      MOV      #PRMPT,R1 ;ADDRESS OF PARAMETER POINTER TABLE
153 023644 005711          5$:      TST      (R1) ;END OF THE TABLE ?
154 023646 001425          BEQ      8$ ;BR IF END
155 023650 032731 000400          BIT      #BIT08,@(R1)+ ;'LS' SELECTED ?
156 023654 001773          BEQ      5$ ;BR IF NOT
157 023656 016102 177776          MOV      -2(R1),R2 ;PARAMETER TABLE ADDRESS
158 023662 011246          MOV      (R2),-(SP) ;PARAMETER ALLOCATION BITS
159 023664 012703 000011          MOV      #9,R3 ;NUMBER OF PARAMETERS (MAXIMUM) BEFORE 'LS'
160 023670 006216          6$:      ASR      (SP) ;COUNT THE PARAMETER
161 023672 103002          BCC      7$ ;BR IF NOT USED
162 023674 062702 000002          ADD      #2,R2 ;INCREMENT THE PARAMETER TABLE ADDRESS
163 023700 005303          7$:      DEC      R3 ;COUNT THE PARAMETER
164 023702 001372          BNE      6$ ;BR IF NOT THERE YET
165 023704 005726          TST      (SP)+ ;CORRECT THE STACK POINTER
166 023706 021237 002426          CMP      (R2),PRMLMT+16 ;IS 'LS' TOO LARGE FOR THE MODE SELECTED ?
167 023712 101754          BLOS      5$ ;BR IF NOT
168 023714 013712 002426          MOV      PRMLMT+16,(R2) ;RESET VALUE FOR MODE USED
    
```

```

172 023720 000751          BR      5$          ;CONTINUE
173 023722          8$:          MOV      (SP)+,R3      ;;POP STACK INTO R3
    023722 012603          MOV      (SP)+,R2      ;;POP STACK INTO R2
    023724 012607          MOV      (SP)+,R1      ;;POP STACK INTO R1
    023726 012601          MOV      (SP)+,R0      ;;POP STACK INTO R0
    023730 012600          RTS      PC          ;RETURN
174 023732 000207
175
176          ;THIS ROUTINE FILLS THE PARAMETER TABLE THE CURRENT TEST.
177          ;CALL
178          ;          MOV      #TESTNUM,$STNM ;LOAD THE TEST NUMBER
179          ;          JSR      PC,LODPRM
180          ;          RETURN
181
182          LODPRM:
    023734          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
    023734 010146          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
    023736 010246          MOV      R4,-(SP)      ;;PUSH R4 ON STACK
    023740 010446          CLR      R4          ;CLEAR R4
183 023742 005004          MOVB   $STNM,R4      ;GET THE TEST NUMBER
184 023744 113704 001116          ASL      R4          ;SETUP TO ADDRESS WORDS
185 023750 006304          MOV      PRMPT(R4),R1 ;GET THE TEST'S PARAMETER TABLE ADDRESS
186 023752 016401 002352          MOV      #PRM,R2      ;PARAMETER EXECUTION TABLE
187 023756 012702 002320          MOV      CHKDRV,R4    ;GET DRIVE ADDRESS
188 023762 013704 001352          MOV      (R1)+,(R2)+ ;LOAD PARAMETER SPECIFIER
189 023766 012122          1$:      ASR      PRM          ;IS THIS PARAMETER USED IN THE TEST ?
190 023770 006237 002320          BCC     2$          ;BR IF NOT
191 023774 103002          MOV      (R1)+,(R2)+ ;LOAD THE VALUE
192 023776 012122          BR      3$          ;CONTINUE
193 024000 000401          CLR      (R2)+      ;CLEAR THE UNUSED PARAMETER LOCATION
194 024002 005022          2$:      CMP      #PAT+2,R2    ;FINISHED ?
195 024004 022702 002346          BEQ     4$          ;BR IF YES
196 024010 001404          MOV      PRMLMT+12,LSTRK ;GET LAST TRACK FOR AN RM80
197 024012 013737 002422 001374          BR      1$          ;KEEP GOING
198 024020 000763          4$:      MOV      (SP)+,R4      ;;POP STACK INTO R4
199 024022          MOV      (SP)+,R2      ;;POP STACK INTO R2
    024024 012602          MOV      (SP)+,R1      ;;POP STACK INTO R1
    024026 012601          RTS      PC          ;RETURN
    024030 000207
200
201          ;THIS ROUTINE LOADS A READ HEADER AND DATA COMMAND OR A SEEK COMMAND.
202          ;INTO DPB.B+2 AND DPB.C+2, DEPENDING ON THE STATE OF 'CONTROL SWITCH'.
203          ;BIT07.
204          ;CALL
205          ;          JSR      PC,LDCMD
206          ;          RETURN
207
208          LDCMD:
209 024032 032737 000200 001314          BIT      #SW07,C.SWR    ;DO EXPLICIT SEEKS?
210 024040 001007          BNE     1$          ;YES--BRANCH
211 024042 012737 000173 036532          MOV      #READHD,DPB.B+2 ;NO--SET UP FOR READ HEADER AND
212 024050 012737 000173 036554          MOV      #READHD,DPB.C+2 ;DATA COMMAND
213 024056 000406          BR      2$
214 024060 012737 000105 036532          1$:      MOV      #SEEK,DPB.B+2 ;SETUP FOR SEEK COMMAND
215 024066 012737 000105 036554          MOV      #SEEK,DPB.C+2
216 024074 000207          2$:      RTS      PC
217
218          ;THIS ROUTINE WILL CALL THE RM80 DRIVER AND THEN WAIT ON THE FUNCTION
    
```



```

219      ;TO COMPLETE. IF AN ERROR OCCURS IT IS REPORTED.
220      ;CALL
221      ;      FILL 'DPB' WITH COMMAND INFORMATION
222      ;      JSR   RO,CALL.A
223      ;      RETURN
224
225 024076 005037 001222      CALL.A: CLR   $ESCAPE ;NO ESCAPE ADDRESS
226 024102 004037 032054      JSR   RO,RM80      ;CALL RM80 DRIVER
227 024106 036506      DPB.A
228 024110 000772      BR   CALL.A
229 024112 005737 036524      1$:  TST   DPB.A+16      ;DONE?
230 024116 001775      BEQ   1$              ;NO--LOOP
231 024120 100050      BPL   5$              ;BRANCH IF NO ERROR
232 024122 012737 024216 001222      MOV   #3$, $ESCAPE      ;:ESCAPE TO 3$ ON ERROR
233 024130 013737 036520 001366      MOV   DPB.A+12,CYL.DS ;:CYLINDER
234 024136 113737 036517 001372      MOVB  DPB.A+11,TRK.DS ;:TRACK
235 024144 113737 036516 001370      MOVB  DPB.A+10,SEC.DS ;:SECTOR
236 024152 012746 036524      MOV   #DPB.A+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
237 024156 004737 025062      JSR   PC,ERINDX      ;FORM DISPATCH INDEX
238 024162 062607      ADD   (SP)+,PC      ;REPORT PROPER ERROR
239 024164 104041      EMT   41            ;NON-EXIST DRIVE
240 024166 104042      EMT   42            ;PORT TIMEOUT
241 024170 104043      EMT   43            ;UNSAFE ERROR
242 024172 104044      EMT   44            ;NON-I/O ERROR
243 024174 000240      NOP                    ;TO SYNC THE CALLING SEQ OF ERINDX
244 024176 005737 036654      TST   RM.REG+RMER1    ;ANY DRIVE ERROR
245 024202 001004      BNE  2$              ;BRANCH IF SO
246 024204 032737 100000 036702      BIT   #BSE,RM.REG+RMER2 ;BAD SPOT ERROR
247 024212 001007      BNE  4$              ;BRANCH IF SO
248 024214      2$:
249 024214 104045      EMT   45
250 024216 032737 040000 036702      3$:  BIT   #SKI,RM.REG+RMER2 ;SKI ERROR ?
251 024224 001402      BEQ   4$              ;BR IF NO
252 024226 004037 024710      JSR   RO,CALL.R      ;DO RECALIBRATE COMMAND
253 024232 013746 036524      4$:  MOV   DPB.A+16,-(SP) ;STATUS WORD
254 024236 004737 025022      JSR   PC,LOP.CK     ;SEE IF LOOP, ABORT, OR CONTINUE
255 024242 000200      5$:  RTS   RO          ;RETURN
256
257      ;THIS ROUTINE IS THE SAME AS "CALL.A" EXCEPT FOR THE DPB USED AND IF
258      ;THE COMMAND IS A READ HEADER AND DATA THE HEADER (CYLINDER, TRACK,
259      ;AND SECTOR) READ IS CHECKED FOR VALIDITY.
260      ;CALL
261      ;      FILL DPB
262      ;      JSR   RO,CALL.B
263      ;      RETURN
264
265 024244 005037 001222      CALL.B: CLR   $ESCAPE ;NO ESCAPE ADDRESS
266 024250 005037 036550      CLR   DPB.B+$SSENB ;CLEAR SKIP SECTOR ENABLED INDICATOR
267 024254 004037 032054      JSR   RO,RM80      ;CALL RM80 DRIVER
268 024260 036530      DPB.B
269 024262 000770      BR   CALL.B
270 024264 005737 036546      1$:  TST   DPB.B+16      ;DONE?
271 024270 001775      BEQ   1$              ;NO--BRANCH
272 024272 100055      BPL   5$              ;BRANCH IF NO ERROR
273 024274 012737 024400 001222      MOV   #3$, $ESCAPE      ;:ESCAPE TO 3$ ON ERROR
274 024302 013737 036542 001366      MOV   DPB.B+12,CYL.DS ;:CYLINDER
275 024310 113737 036541 001372      MOVB  DPB.B+11,TRK.DS ;:TRACK
    
```

```

024316 113737 036540 001370      MOVB  DPB.B+10,SEC.DS :SECTOR
024324 012746 036546             MOV   #DPB.B+16,-(SP) :STATUS/ERROR INDICATOR ADDRESS
024330 004737 025062             JSR   PC,ERINDX       :FORM DISPATCH INDEX
024334 062607             ADD   (SP)+,PC        :REPORT PROPER ERROR
024336 104041             EMT   41              :NON-EXIST DRIVE
024340 104042             EMT   42              :PORT TIMEOUT
024342 104043             EMT   43              :UNSAFE ERROR
024344 104044             EMT   44              :NON-I/O ERROR
265 024346 000240             NOP                      :TO SYNC THE CALLING SEQ OF ERINDX: RT.
266 024350 005737 036654             TST   RM.REG+RMER1     :DRIVE ERROR ?
267 024354 001404             BEQ   2$              :BR IF NOT
268 024356 022737 000200 036654             CMP   #HCE,RM.REG+RMER1 :SEE IF ONLY 'HCE' SET
269 024364 001420             BEQ   5$              :BR IF YES
270 024366 032737 100000 036702 2$:      BIT   #BSE,RM.REG+RMER2 :BAD SPOT ERROR
271 024374 001033             BNE   7$              :BRANCH IF SO
272 024376 104045             EMT   45
273 024400 032737 040000 036702 3$:      BIT   #SKI,RM.REG+RMER2 :SKI ERROR ?
274 024406 001402             BEQ   4$              :BR IF NO
275 024410 004037 024710             JSR   RO,CALL.R       :DO RECALIBRATE COMMAND
276 024414 013746 036546             MOV   DPB.B+16,-(SP) :STATUS WORD
277 024420 004737 025022             JSR   PC,LOP.CK      :SEE IF LOOP, ABORT, OR CONTINUE
278 024424 000410             BR    6$              :CHECK FOR STALL
279 024426 123727 036532 000173 5$:      CMPB  DPB.B+2,#READHD :DOING IMPLIED SEEKS?
280 024434 001004             BNE   6$              :NO--BRANCH
281 024436 04037 025302             JSR   RO,VERIFY      :YES--GO CHECK THE DATA
282 024442 036540             DPB.B+10
283 024444 000407             BR    7$              :ERROR DURING VERIFY
284 024446 032737 040000 001314 6$:      BIT   #SW14,C.SWR    :STALL?
285 024454 001403             BEQ   7$              :NO--BRANCH
286 024456 004037 025220             JSR   RO,STALL      :YES--CALL STALL ROUTINE
287 024462 001454             .WORD STALL1        :STALL TIME POINTER
288 024464 000200             7$:      RTS    RO           :RETURN
289
290      ;THIS ROUTINE IS THE SAME AS 'CALL.B' EXCEPT FOR THE DPB USED.
291      ;CALL
292      ;
293      ;   FILL DPB
294      ;   JSR   RO,CALL.C
295      ;   RETURN
296 024466 005037 001222      CALL.C: CLR   $ESCAPE   :NO ESCAPE ADDRESS
297 024472 005037 036572      CLR   DPB.C+$SSENB  :CLEAR SKIP SECTOR ENABLED INDICATOR
298 024476 004037 032054      JSR   RO,RM80      :CALL RM80 DRIVER
299 024502 036552      DPB.C
300 024504 000770      BR    CALL.C
301 024506 005737 036570      1$:      TST   DPB.C+16      :DONE?
302 024512 001775      BEQ   1$            :NO--LOOP
303 024514 100055      BPL   5$            :YES--BRANCH IF NO ERROR
304 024516 012737 024622 001222      MOV   #3$,$ESCAPE   :ESCAPE TO 3$ ON ERROR
305 024524 013737 036564 001366      MOV   DPB.C+12,CYL.DS :CYLINDER
024532 113737 036563 001372      MOVB  DPB.C+11,TRK.DS :TRACK
024540 113737 036562 001370      MOVB  DPB.C+10,SEC.DS :SECTOR
024546 012746 036570             MOV   #DPB.C+16,-(SP) :STATUS/ERROR INDICATOR ADDRESS
024552 004737 025062             JSR   PC,ERINDX       :FORM DISPATCH INDEX
024556 062607             ADD   (SP)+,PC        :REPORT PROPER ERROR
024560 104041             EMT   41              :NON-EXIST DRIVE
024562 104042             EMT   42              :PORT TIMEOUT
024564 104043             EMT   43              :UNSAFE ERROR
    
```

```

306 024566 104044      EMT      44      :NON-I/O ERROR
307 024570 000240      NOP
308 024572 005737 036654 TST      RM.REG+RMER1 :TO SYNC THE CALLING SEQ OF ERINDX: RT.
309 024600 022737 000200 036654 BEQ      2$      :DRIVE ERROR ?
310 024606 001420      BEQ      2$      :BR IF NOT
311 024610 032737 100000 036702 2$: CMP      #HCE, RM.REG+RMER1 :SEE IF ONLY 'HCE' SET
312 024616 001033      BEQ      5$      :BR IF YES
313 024620 104045      BIT      #BSE, RM.REG+RMER2 :BAD SPOT ERROR ONLY ?
314 024622 032737 040000 036702 3$: BNE      7$      :BRANCH IF SO
315 024630 001402      EMT      45
316 024632 004037 024710      BIT      #SKI, RM.REG+RMER2 :SKI ERROR ?
317 024636 013746 036570      BEQ      4$      :BR IF NO
318 024642 004737 025022      JSR      RO, CALL.R      :DO RECALIBRATE COMMAND
319 024646 000410      MOV      DPB.C+16, -(SP) :STATUS WORD
320 024650 123727 036554 000173 5$: JSR      PC, LOP.CK      :SEE IF LOOP, ABORT, OR CONTINUE
321 024656 001004      BR      6$
322 024660 004037 025302      CMPB     DPB.C+2, #READHD :DOING IMPLIED SEEK?
323 024664 024664 036562      BNE      6$      :NO--EXIT
324 024666 000407      JSR      RO, VERIFY      :YES--CHECK THE DATA
325 024670 032737 040000 001314 6$: BR      7$      :ERROR DURING VERIFY
326 024676 001403      BIT      #SW14, C.SWR    :STALL?
327 024700 004037 025220      BEQ      7$      :NO--BRANCH
328 024704 001454      JSR      RO, STALL      :YES--CALL STALL ROUTINE
329 024706 000200      .WORD   STALL1        :STALL TIME POINTER
330      RTS      RO
    
```

: THIS ROUTINE WILL ISSUE A RECALIBRATE COMMAND TO THE RM80 DRIVER  
 : AND WAIT FOR THE FUNCTION TO COMPLETE.

```

:CALL
:      JSR      RO, CALL.R
:      RETURN
    
```

```

431
432 024710 005037 001222 CALL.R: CLR      $ESCAPE      :NO ESCAPE ADDRESS
433 024714 004037 032054      JSR      RO, RM80      :CALL RM80 DRIVER
434 024720 036616      DPB.R
435 024722 000772      BR      CALL.R
436 024724 005737 036634 1$: TST      DPB.R+16      :DONE?
437 024730 001775      BEQ      1$      :NO--LOOP
438 024732 100032      BPL      3$      :BRANCH IF NO ERROR
439 024734 012737 025010 001222 MOV      #2$, $ESCAPE    :ESCAPE TO 2$ ON ERROR
440 024742 013737 036630 001366 MOV      DPB.R+12, CYL.DS :CYLINDER
441 024750 113737 036627 001372 MOV      DPB.R+11, TRK.DS :TRACK
442 024756 113737 036626 001370 MOV      DPB.R+10, SEC.DS :SECTOR
443 024764 012746 036634      MOV      #DPB.R+16, -(SP) :STATUS/ERROR INDICATOR ADDRESS
444 024770 004737 025062      JSR      PC, ERINDX    :FORM DISPATCH INDEX
445 024774 062607      ADD      (SP)+, PC      :REPORT PROPER ERROR
446 024776 104041      EMT      41      :NON-EXIST DRIVE
447 025000 104042      EMT      42      :PORT TIMEOUT
448 025002 104043      EMT      43      :UNSAFE ERROR
449 025004 104044      EMT      44      :NON-I/O ERROR
450 025006 000240      NOP
451 025010 013746 036524 2$: MOV      DPB.A+16, -(SP) :TO SYNC THE CALLING SEQ OF ERINDX
452 025014 004737 025022      JSR      PC, LOP.CK    :STATUS WORD
453 025020 000200      JSR      RO          :SEE IF LOOP, ABORT, OR CONTINUE
454      RTS      RO          :RETURN
    
```

: THIS SUBROUTINE CHECK FOR LOOP, ABORT, OR CONTINUE SWITCHES AFTER

```

447 ; ERRORS 41, 42, 43, 44, 45, AND 46.
448 ; CALL
449 ; MOV DTA+16,-(SP) ; STATUS WORD FROM DPB IN USE
450 ; JSR PC,LOP.CK
451 ; RETURN
452
453 025022 032777 001000 154124 LOP.CK: BIT #SW9,@SWR ; LOOP ON ERROR
454 025030 001402 BEQ 1$ ; BR IF NOT
455 025032 000177 154066 JMP @SLPERR ; START AT THE LOOP ADDRESS
456 025036 005037 001222 1$: CLR $ESCAPE ; CLEAR ERROR ESCAPE FLAG
457 025042 032766 072006 000002 BIT #BIT14!BIT13!BIT12!BIT10!BIT02!BIT01,2(SP) ; CHECK ERROR TYPE
458 025050 001402 BEQ 2$ ; BR IF DRIVE NOT OFFLINE, UNLOADED, OR
459 ; PERSISTENT UNSAFE OR FATAL MASSBUS PARITY
460 025052 000137 015142 JMP $EOP ; TERMINATE DRIVE
461 025056 012616 2$: MOV (SP)+,(SP) ; ADJUST RETURN ADDRESS
462 025060 000207 RTS PC
463
464 ; THIS ROUTINE FORMS AN INDEX THAT WILL BE USED TO DISPATCH
465 ; TO THE PROPER ERROR CALL. THE INDEX IS FORMED BY EXAMINING
466 ; THE STATUS/ERROR INDICATOR OF THE APPLICABLE DPB.
467 ; INDEX STATUS/ERROR
468 ;-----
469 ; 0 BIT08!BIT01
470 ; 2 BIT02
471 ; 4 BIT14!BIT04
472 ; 6 BIT05 & <BIT09!COMMAND = NON-I/O>
473 ; 10 BIT06 & <BIT09!COMMAND = I/O>
474 ; CALL
475 ; JSR #DPB+16,-(SP) ; ADDRESS OF STATUS/ERROR INDICATOR
476 ; JSR PC,ERINDX ; FORM INDEX
477 ; RETURN ; INDEX IS ON THE STACK
478
479 025062 010046 ERINDX: MOV R0,-(SP) ; SAVE R0
480 025064 010146 MOV R1,-(SP) ; SAVE R1
481 025066 016600 000006 MOV 6(SP),R0 ; GET STATUS/ERROR INDICATOR POINTER
482 025072 011037 001356 MOV (R0),SVSTAT ; SAVE THE STATUS/ERROR INDICATOR
483 025076 005001 CLR R1 ; START INDEX AT ZERO
484 025100 032710 BIT (PC)+,(R0) ; FORM INDEX OF 0?
485 025102 000402 .WORD BIT08!BIT01
486 025104 001037 BNE 5$ ; YES--BRANCH
487 025106 032710 BIT (PC)+,(R0) ; FORM PARITY ERROR OR PORT REQUEST INDEX (2)?
488 025110 000004 .WORD BIT02
489 025112 001033 BNE 4$ ; YES--BRANCH
490 025114 032710 BIT (PC)+,(R0) ; FORM UNSAFE INDEX (4)?
491 025116 040020 .WORD BIT14!BIT04
492 025120 001027 BNE 3$ ; YES--BRANCH
493 025122 032710 BIT (PC)+,(R0) ; FORM NON-I/O ERROR INDEX (6)?
494 025124 000040 .WORD BIT05
495 025126 001023 BNE 2$ ; YES--BRANCH
496 025130 032710 BIT (PC)+,(R0) ; FORM I/O ERROR INDEX (10)?
497 025132 000100 .WORD BIT06
498 025134 001017 BNE 1$ ; YES--BRANCH
499 025136 032710 BIT (PC)+,(R0) ; SOFTWARE TIMEOUT?
500 025140 001000 .WORD BIT09
501 025142 001420 BEQ 5$ ; NO--FORM INDEX OF 0
502 025144 122760 000150 177762 CMPB #150,-16(R0) ; YES--I/O?
503 025152 003011 BGT 2$ ; NO--BRANCH
    
```

```

504 025154 005737 036654          TST      RM.REG+RMER1      ;ANY DRIVE ERROR ?
505 025160 001005          BNE      1$              ;BRANCH IF SO
506 025162 032737 100000 036702  BIT      #BSE,RM.REG+RMER2 ;BAD SPOT ERROR
507 025170 001401          BEQ      1$              ;BRANCH IF NOT
508 025172 005201          INC      R1              ;SKIP , NOT REPORT BAD SPOT ERROR
509 025174 005201          1$: INC      R1              ;INDEX=10---ERROR=45 OR 46
510 025176 005201          2$: INC      R1              ;INDEX=6---ERROR=44
511 025200 005201          3$: INC      R1              ;INDEX=4---ERROR=43
512 025202 005201          4$: INC      R1              ;INDEX=2---ERROR=42
513 025204 006301          5$: ASL      R1              ;INDEX=0---ERROR=41
514 025206 010166 000006          MOV      R1,6(SP)        ;RETURN INDEX TO USER
515 025212 012601          MOV      (SP)+,R1        ;RESTORE R1
516 025214 012600          MOV      (SP)+,R0        ;RESTORE R0
517 025216 000207          RTS      PC              ;RETURN FROM CALL
518
519          ;THIS ROUTINE WILL PROVIDE A STALL IN MILLISECONDS FOR A SPECIFIC
520          ;AMOUNT OF TIME IF BIT13 OF C.SWR = 0 OR A RANDOM AMOUNT OF TIME
521          ;IF BIT 13 OF C.SWR = 1.
522          ;STALL1 CONTAINS SPECIFIED TIME FOR TESTS 0 - 7, AND STALL2
523          ;CONTAINS THE TIME FOR TESTS 16-21.
524          ;CALL
525          ;
526          ;          JSR      R0,STALL
527          ;          TIME POINTER          ;WHERE TO FIND THE STALL TIME
528 025220 013046          STALL: MOV      @ (R0)+,-(SP) ;PICKUP STALL TIME
529 025222 032737 020000 001314  BIT      #SW13,C.SWR      ;USE A RANDOM TIME?
530 025230 001406          BEQ      1$              ;NO--BRANCH
531 025232 004737 022502          JSR      PC,$RAND        ;YES--FORM RANDOM NUMBER
532 025236 013716 022602          MOV      $LOWUM,(SP)    ;AND USE IT FOR THE STALL TIME
533 025242 042716 177700          BIC      #^C77,(SP)    ;BUT NEVER > 64 MILLISECONDS
534 025246 005046          1$: CLR      -(SP)        ;CLEAR TEMP. LOCATION
535 025250 162766 000001 000002  2$: SUB      #1,2(SP)    ;MORE STALL REQUIRED?
536 025256 103407          BLO      4$              ;NO--BRANCH
537 025260 012716 000144          MOV      #100.,(SP)    ;STALL FOR ABOUT 1 MILLISECOND
538 025264 005700          3$: TST      R0          ;NOP TO KILL TIME
539 025266 005366 000000          DEC      0(SP)        ;COUNT
540 025272 001374          BNE      3$              ;LOOP IF MORE COUNTS NEEDED
541 025274 000765          BR      2$              ;
542 025276 022626          4$: CMP      (SP)+,(SP)+ ;CLEAN OFF THE STACK
543 025300 000200          RTS      R0              ;EXIT
544
545          ;ROUTINE TO SOFTWARE COMPARE HEADER ON IMPLIED SEKS
546          ;CALL
547          ;          JSR      R0,VERIFY
548          ;          ADR POINTER          ;ADDRESS OF DPB+10 (SECTOR NUMBER)
549          ;          BR      ???          ;ERROR RETURN
550          ;          RETURN
551
552 025302 010146          VERIFY: MOV      R1,-(SP) ;SAVE R1
553 025304 012001          MOV      (R0)+,R1      ;GET ADDRESS OF DPB+10
554 025306 042737 170000 043540  BIC      #170000,BUFFER ;STRIP FORMAT AND BAD SECTOR BITS FROM CYLINDER NUMBER
555 025314 111137 025476          MOV      (R1),6$        ;SAVE EXPECTED SECTOR ADDRESS
556 025320 023761 043540 000002  CMP      BUFFER,2(R1)    ;CYLINDER NUMBER OK?
557 025326 001015          BNE      2$              ;NO--BRANCH
558 025330 123761 043543 000001  CMP      BUFFER+3,1(R1) ;IS THE TRACK ADDRESS GOOD ?
559 025336 001011          BNE      2$              ;BR IF NO
560 025340 005761 000010          TST      $$SEN8-10(R1) ;WAS SKIP SECTOR ENABLED ?
    
```

```

561 025344 001402          BEQ      1$          ;BR IF NO
562 025346 105237 025476    INCB     6$          ;INCREMENT EXPECTED SECTOR ADDRESS BY 1
563 025352 123737 043542 025476 1$:    CMPB    BUFFER+2,6$ ;IS THE SECTOR ADDRESS GOOD ?
564 025360 001443          BEQ      4$          ;BR IF YES
565 025362 013737 043540 001360 2$:    MOV     BUFFER,CYL.RD ;GET RECIEVED SECTOR ADDRESS
566 025370 113737 043543 001362        MOVB    BUFFER+3,TRK.RD ;          TRACK
567 025376 113737 043542 001364        MOVB    BUFFER+2,SEC.RD ;          CYLINDER
568 025404 113737 025476 001370        MOVB    6$,SEC.DS      ;GET EXPECTED SECTOR ADDRESS
569 025412 105721          TSTB    (R1)+         ;ADJUST R1 POINTER
570 025414 112137 001372        MOVB    (R1)+,TRK.DS   ;GET EXPECTED TRACK ADDRESS
571 025420 011137 001366        MOV     (R1),CYL.DS   ;          CYLINDER
572 025424 012737 025434 001222        MOV     #3$, $ESCAPE ;:ESCAPE TO 3$ ON ERROR
573 025432 104012          EMT     12
574 025434 012737 000107 036510 3$:    MOV     #RECAL,DPB.A+2 ;LOAD RECALIBRATE ORDER CODE
575 025442 004037 024076        JSR     RO,CALL.A     ;GO EXECUTE THE COMMAND
576 025446 005037 001222        CLR     $ESCAPE      ;CLEAR ERROR ESCAPE FLAG
577 025452 032777 001000 153474        BIT     #SW9,@SWR    ;LOOP ON ERROR ?
578 025460 001403          BEQ     4$          ;BR IF NOT
579 025462 013700 001124        MOV     $LPERR,RO    ;RETURN TO ERROR LOOP ADDRESS
580 025466 000401          BR      5$
581 025470 005720          4$:    TST     (RO)+         ;INCREMENT RETURN ADDRESS
582 025472 012601          5$:    MOV     (SP)+,R1    ;RESTORE R1
583 025474 000200          RTS     RO         ;EXIT
584
585 025476 000000          6$:    .WORD  0         ;TEMPORARY STORAGE FOR EXPECTED SECTOR ADDRESS
586
587          ;THIS ROUTINE WILL PERFORM A 'MASSBUS' INIT. FOLLOWED BY
588          ;A 'RECALIBRATE' ON THE DRIVE UNDER TEST.
589          ;NOTE: THIS ROUTINE DESTROYS R1 AND R4
590          ;CALL
591          ;      JSR     RO,SRCH00      ;DO A MASSBUS INIT. AND RECAL
592          ;      RETURN1      ;RETURN HERE IF NO ERROR
593          ;      RETURN2      ;RETURN HERE ON ERROR
594
595 025500 005001          SRCH00: CLR     R1         ;INCASE OF ERROR (TYPTIM)
596 025502 005037 177776        CLR     PS
597 025506 012777 033526 003654        MOV     #ISR,@RMVEC  ;SETUP INTERRUPT VECTOR
598 025514 013704 031366        MOV     RMADR,R4    ;PICKUP ADDRESS OF RMCS1
599 025520 012764 000040 000010        MOV     #CLR,RMCS2(R4) ;MASSBUS INIT.
600 025526 005037 036604        CLR     DTADPB+10   ;TRACK=0 & SECTOR=0
601 025532 005037 036606        CLR     DTADPB+12   ;CYLINDER =0
602 025536 012737 000107 036576        MOV     #RECAL,DTADPB+2 ;COMMAND = RECALIBRATE
603 025544 005037 001222        CLR     $ESCAPE     ;NO ESCAPE ADDRESS
604 025550 004037 032054        JSR     RO,RM80     ;CALL THE DRIVER
605 025554 036574        DTADPB
606 025556 000440          BR      4$          ;DPB POINTER
607 025560 005737 036612          1$:    TST     DTADPB+16   ;QUEUE IS FULL
608 025564 001775          BEQ     1$          ;WAIT ON DONE
609 025566 100030          BPL     3$          ;TAKE NORMAL EXIT IF NO ERROR
610 025570 012737 025644 001222        MOV     #2$, $ESCAPE ;:ESCAPE TO 2$ ON ERROR
611 025576 013737 036606 001366        MOV     DTADPB+12,CYL.DS ;CYLINDER
        MOVB    DTADPB+11,TRK.DS ;TRACK
        MOVB    DTADPB+10,SEC.DS ;SECTOR
        MOV     #DTADPB+16,-(SP) ;STATUS/ERROR INDICATOR ADDRESS
        JSR     PC,ERINDX ;FORM DISPATCH INDEX
        ADD     (SP)+,PC   ;REPORT PROPER ERROR
        EMT     41       ;NON-EXIST DRIVE
    
```



```

025634 104042      EMT      42      :PORT TIMEOUT
025636 104043      EMT      43      :UNSAFE ERROR
025640 104044      EMT      44      :NON-I/O ERROR
025642 104045      EMT      45      :I/O ERROR
612 025644 0C5720  2$: TST      (R0)+   :ADJUST FOR ERROR EXIT
613 025646 000404  BR      4$      :GO TO THE EXIT
614 025650 005064  000006  3$: CLR      RMDA(R4) :TRACK AND SECTOR = 0
615 025654 005064  000034  CLR      RMDC(R4)  :CYLINDER = 0
616 025660 000200  4$: RTS      R0      :RETURN
617
618 ;THIS IS AN RTI WHICH IS USED BY THE TIMING TESTS & THE SERVO SETTLE DOWN TEST
619
620 025662 000002  DORTI: RTI      :RETURN FROM INTERRUPT
621
622 ;THIS ROUTINE WILL INITIALIZE THE TIMERS USED BY THE 'TIMING ROUTINES
623 :CALL
624 :
625 :
626 :
627 025664 104412  STRTMR: SAVREG  :SAVE R0-R5
628 025666 012700  001376  MOV      #TIM.UP,R0 :INITIALIZE TIME TABLES
629 025672 005020  1$: CLR      (R0)+   :CLEAR
630 025674 020027  001432  CMP      R0,#TIM.PT :DONE?
631 025700 103774  BLO      1$        :NO--BRANCH
632 025702 012710  043540  MOV      #BUFFER,(R0) :SETUP TIME TABLE POINTER
633 025706 012737  077777  001376  MOV      #^CBIT15,TIM.UP :SET MINIMUM TIME TO MAXIMUM
634 025714 012737  077777  001414  MOV      #^CBIT15,TIM.DN :POSITIVE NUMBER
635 025722 104413  RESREG  :RESTORE R0-R5
636 025724 000207  RTS      PC        :RETURN
637
638 ;THIS ROUTINE WILL ADD THE ELAPSED TIME TO THE AVERAGE COUNTER AND
639 ;MAINTAIN THE MINIMUM AND MAXIMUM TIMES.
640 ;NOTE: THIS ROUTINE DESTROYS R2
641 :CALL
642 :
643 :
644 :
645 :
646 :
647 :
648 025726 012702  001376  COUNT: MOV      #TIM.UP,R2 :PICKUP THE 'UP' POINTER
649 025732 005705  TST      R5        :USE IT?
650 025734 001402  BEQ      1$        :YES--BRANCH
651 025736 012702  001414  MOV      #TIM.DN,R2 :NO--PICKUP 'DOWN' POINTER
652 025742 027722  153572  1$: CMP      @PKC,(R2)+ :LESS THAN PREVIOUS LOW?
653 025746 002003  BGE      2$        :NO--BRANCH
654 025750 017762  153564  177776  MOV      @PKC,-2(R2) :YES--SAVE IT
655 025754 027763  153556  000004  2$: CMP      @PKC,4(R3) :LESS THAN THE LOW LIMIT?
656 025764 002001  BGE      3$        :NO--BRANCH
657 025766 005212  INC      (R2)      :YES--COUNT IT
658 025770 005722  3$: TST      (R2)+   :ADVANCE THE POINTER
659 025772 027722  153542  CMP      @PKC,(R2)+ :GREATER THAN PREVIOUS HIGH?
660 025776 003403  BLE      4$        :NO--BRANCH
661 026000 017762  153534  177776  MOV      @PKC,-2(R2) :YES--SAVE IT
662 026006 027763  153526  000006  4$: CMP      @PKC,6(R3) :GREATER THAN THE HIGH LIMIT?
663 026014 003401  BLE      5$        :NO--BRANCH
664 026016 005212  INC      (R2)      :YES--COUNT IT
    
```

```

665 026020 005722          5$:   TST      (R2)+      :ADVANCE THE POINTER
666 026022 067722 153512   ADD      @PKC,(R2)+    :ADD THIS COUNT TO THE TOTAL
667 026026 005522          ADC      (R2)+
668 026030 005212          INC      (R2)         :COUNT THIS READING
669 026032 022737 050044 001432 CMP      #BUFFER+<4*561.>,TIM.PT :SAVE THIS COUNT?
670 026040 101406          BLOS    6$           :NO--BRANCH
671 026042 017777 153472 153362 MOV      @PKC,@TIM.PT  :YES--WELL SAVE IT THEN
672 026050 062737 000002 001432 ADD      #2,TIM.PT    :ADVANCE THE POINTER
673 026056 00C207          6$:   RTS      PC       :RETURN
    
```

674  
 675 :THIS ROUTINE PRINTS THE SPEC OF ALL TIMING TESTS

```

676 :CALL
677 :       JSR      RO,SPTYP
678 :       TABLE ADDRESS
679 :
680 :TABLE: .WORD   ASCIZ MESSAGE POINTER
681 :       .WORD   MIN VALUE
682 :       .WORD   MAX VALUE
683 :
    
```

```

684 026060 012002          SPTYP: MOV      (R0)+,R2     :THE TABLE ADDRESS
685 026062 032777 000100 153064 BIT      #SW06,@SWR   :ALLOW PRINT
686 026070 001035          BNE      3$           :EXIT IF NOT
687 026072 104401 001231   TYPE    ,SCLF
688 026076 104401 001231   TYPE    ,SCLF
689 026102 012237 026110   MOV      (R2)+,1$
690 026106 104401          TYPE
691 026110 000000          1$:   .WORD   0
692 026112 012246          MOV      (R2)+,-(SP)  :LOAD MIN VALUE
693 026114 001410          BEQ      2$           :SKIP IF MIN VALUE IS 0
694 026116 104401 037674   TYPE    ,MSGMIN
695 026122 004737 022212   JSR      PC,$SB2D    :CONVERT TO DECIMAL
696 026126 004737 022442   JSR      PC,$SUPRS   :TYPE IT
697 026132 104401 037716   TYPE    ,MSGOUS     :0 US
698 026136 104401 037702   2$:   TYPE    ,MSGMAX
699 026142 011246          MOV      (R2)+,-(SP)  :MAXIMUM VALUE
700 026144 004737 022212   JSR      PC,$SB2D
701 026150 004737 022442   JSR      PC,$SUPRS
702 026154 104401 037716   TYPE    ,MSGOUS
703 026160 104401 001231   TYPE    ,SCLF      :CR-LF
704 026164 000200          3$:   RTS      RO
    
```

705  
 706  
 707  
 708 ::THIS ROUTINE IS USED TO TYPE THE MINIMUM,  
 709 :MAXIMUM, AND AVERAGE TIMES FOR THE TIMING TESTS  
 710 :IT WILL ALSO CHECK THE TIMES TO ENSURE  
 711 :THEY ARE WITHIN TOLERANCE AND IF NOT FLAG THE BAD TIMES.

```

712 :CALL
713 :       JSR      RO,TYPTIM :GO REPORT THE TIMES
714 :       TABLE :POINT TO THE PROPER TABLE
715 :       RETURN
    
```

```

716 :TABLE: MSGADR1 :ADDRESS OF ASCIZ MESSAGE NUMBER 1
717 :       MSGADR2 :ADDRESS OF ASCIZ MESSAGE NUMBER 2
718 :       MIN.ALLOWED :MINIMUM TIME ALLOWED
719 :       MAX.ALLOWED :MAXIMUM TIME ALLOWED
720 :
721 :
    
```



722	026166				TYPTIM:	MOV	R2,-(SP)	::PUSH R2 ON STACK	
	026166	010246				MOV	R3,-(SP)	::PUSH R3 ON STACK	
	026170	010346				MOV	R4,-(SP)	::PUSH R4 ON STACK	
	026172	010446				MOV	R5,-(SP)	::PUSH R5 ON STACK	
	026174	010546				MOV	(R0)+,R2	::PICKUP THE TABLE POINTER	
723	026176	012002				BIT	#SW06,@SWR	::INHIBIT TIME REPORTS?	
724	026200	032777	000100	152746		BNE	9\$	::YES--BRANCH	
725	026206	001154				MOV	(R2)+,2\$	::ADDRESS OF MESSAGE NUMBER 1	
726	026210	012237	026230			MOV	(R2)+,R5	::ADDRESS OF MESSAGE NUMBER 2	
727	026214	012205				MOV	(R2)+,R3	::PICKUP THE LOW LIMIT	
728	026216	012203				MOV	(R2),R2	::AND THE HIGH LIMIT	
729	026220	011202				MOV	#TIM.UP,R4	::PARAMETER POINTER	
730	026222	012704	001376			1\$:	TYPE	::TYPE THE MESSAGE	
731	026226	104401				2\$:	.WORD	0	::ASCIZ MESSAGE POINTER GOES HERE
732	026230	000000				TST	14(R4)	::DID ANY COUNTS OCCUR?	
733	026232	005764	000014			BEQ	8\$	::NO--BRANCH	
734	026236	001536				TYPE	,MSGMIN	::"MIN="	
735	026240	104401	037674			MOV	(R4)+,-(SP)	::PUT (R4)+ ON THE STACK	
736	026244	012446				JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026246	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026252	004737	022442			TYPE	,MSGOUS	::"0 US"	
737	026256	104401	037716			TST	(R4)+	::ANY SEEKS BELOW THE LOW LIMIT	
738	026262	005724				BEQ	3\$	::NO--BRANCH	
739	026264	001421				TYPE	,COMMA	::TYPE "	
740	026266	104401	037160			MOV	-2(R4),-(SP)	::PUT -2(R4) ON THE STACK	
741	026272	016446	177776			JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026276	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026302	004737	022442			TYPE	,MABOVE	::"BELOW THE MINIMUM OF"	
742	026306	104401	037723			MOV	R3,-(SP)	::PUT R3 ON THE STACK	
743	026312	010346				JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026314	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026320	004737	022442			TYPE	,MSGOUS	::TYPE "0 US"	
744	026324	104401	037716			3\$:	TYPE	,MSGMAX	::"MAX="
745	026330	104401	037702			MOV	(R4)+,-(SP)	::PUT (R4)+ ON THE STACK	
746	026334	012446				JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026336	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026342	004737	022442			TYPE	,MSGOUS	::TYPE "0 US"	
747	026346	104401	037716			TST	(R4)+	::ANY SEEKS ABOVE THE HIGH LIMIT	
748	026352	005724				BEQ	4\$	::NO--BRANCH	
749	026354	001421				TYPE	,COMMA	::TYPE "	
750	026356	104401	037160			MOV	-2(R4),-(SP)	::PUT -2(R4) ON THE STACK	
751	026362	016446	177776			JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026366	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026372	004737	022442			TYPE	,MABOVE	::"ABOVE THE MAXIMUM OF"	
752	026376	104401	037752			MOV	R2,-(SP)	::PUT R2 ON THE STACK	
753	026402	010246				JSR	PC,\$SB2D	::CHANGE TO DECIMAL ASCIZ	
	026404	004737	022212			JSR	PC,\$SUPRS	::TYPE WITHOUT LEADING ZEROS	
	026410	004737	022442			TYPE	,MSGOUS	::TYPE "0 US"	
754	026414	104401	037716			4\$:	TYPE	,MSGAVG	::"AVG="
755	026420	104401	037710			MOV	(R4)+,-(SP)	::FORM THE AVERAGE	
756	026424	012446				MOV	(R4)+,-(SP)		
757	026426	012446				MOV	(R4)+,-(SP)		
758	026430	012446				MOV	(R4)+,-(SP)		
759	026432	004737	022604			JSR	PC,\$DIV	::IS THE REMAINDER OVER HALF?	
760	026436	006126				ROL	(SP)+		
761	026440	100001				BPL	5\$	::NO--BRANCH	
762	026442	005216				INC	(SP)	::YES--ROUND UP	

```

763 026444          5$:      JSR      PC,$SB2D      ;CHANGE TO DECIMAL ASCIZ
      026444 004737 022212      JSR      PC,$SUPRS     ;TYPE WITHOUT LEADING ZEROS
      026450 004737 022442      TYPE     ,MSGOUS      ;TYPE '0 US'
764 026454 104401 037716      TYPE     ,COMMA       ;TYPE '' ''
765 026460 104401 037160      MOV      -2(R4),-(SP)    ;PUT -2(R4) ON THE STACK
766 026464 016446 177776      JSR      PC,$SB2D      ;CHANGE TO DECIMAL ASCIZ
      026470 004737 022212      JSR      PC,$SUPRS     ;TYPE WITHOUT LEADING ZEROS
      026474 004737 022442      CMP      #11,$STSNM    ;TEST 11 ?
767 026500 022737 000011 001116 BEQ      6$           ;BRANCH IF SO
768 026506 001403          TYPE     ,MSGSEK     ;TYPE 'SEEKS TIMED'
769 026510 104401 040021      BR       7$           ;TYPE IT
770 026514 000402          6$:      TYPE     ,MSGSCH     ;TYPE 'SEARCH TIME'
771 026516 104401 040001      7$:      MOV      R5,2$      ;NEXT MESSAGE POINTER
772          BEQ      9$           ;IF NONE EXIT
773 026522 010537 026230      CLR      R5           ;NO MORE THAN 2
774 026526 001404          BR       1$           ;TYPE 'NOT TIMED'
775 026530 005005          8$:      TYPE     ,MSGNOT
776 026532 000635          9$:      MOV      (SP)+,R5    ;;POP STACK INTO R5
777 026534 104401 040036      MOV      (SP)+,R4    ;;POP STACK INTO R4
778 026540          MOV      (SP)+,R3    ;;POP STACK INTO R3
      026540 012605          MOV      (SP)+,R2    ;;POP STACK INTO R2
      026542 012604          RTS      R0           ;EXIT
      026544 012603
      026546 012602
779 026550 000200
780
781      ;THIS SUBROUTINE WILL INCREMENT THE TRACK
782      ;NUMBER (R2) BY THE AMOUNT SPECIFIED BY 'IT'.
783      ;CALL
784      ;      JSR      R0,INCTRK
785      ;      RETURN1      ;TRACK NUMBER GREATER THAN LT15
786      ;      RETURN2     ;TRACK NUMBER INCREMENTED
787
788 INCTRK: CMP      R2,LT      ;LAST TRACK COMPLETED?
789          BEQ      2$           ;YES--EXIT
790          ADD      IT,R2      ;NO--UPDATE TRACK
791          CMP      R2,LT      ;TRACK TO BIG?
792          BLE      1$           ;NO--EXIT
793          MOV      LT,R2      ;YES--SET TRACK TO LAST TRACK
794          TST      (R0)+      ;ADJUST FOR RETURN 2
795          RTS      R0           ;RETURN
796
797
798      ;THIS SUBROUTINE WILL INCREMENT THE CYLINDER
799      ;NUMBER (R1) BY THE AMOUNT SPECIFIED BY 'IC'.
800      ;CALL
801      ;      JSR      R0,INCCYL
802      ;      RETURN1      ;CYLINDER NUMBER GREATER THAN LC15
803      ;      RETURN2     ;CYLINDER NUMBER INCREMENTED
804
805 INCCYL: CMP      R1,LC      ;LAST CYLINDER COMPLETED?
806          BEQ      2$           ;YES--EXIT
807          ADD      IC,R1      ;NO--UPDATE CYLINDER
808          CMP      R1,LC      ;CYLINDER TO BIG?
809          BLE      1$           ;NO--EXIT
810          MOV      LC,R1      ;YES--SET CYLINDER TO LAST CYLINDER
811          TST      (R0)+      ;ADJUST FOR RETURN 2
    
```

```

812 026630 000200      2$:   RTS      R0          ;RETURN
813
814                   ;THIS ROUTINE DECREASES THE SECTOR ADDRESS.
815                   ;CALL
816                   ;:   CLR      -(SP)      ;CLEAR THE STACK
817                   ;:   JSR      PC,DECSEC  ;SUBROUTINE ENTRY
818                   ;:   RETURN
819
820 026632 113766 036646 000002 DECSEC: MOV#B   RM,REG+RMDA,2(SP) ;PUT THE SECTOR ADDRESS ON THE STACK
821 026640 005366 000002      DEC      2(SP)      ;DECREMENT THE ADDRESS
822 026644 100003      BPL      1$          ;BR IF NOT CORRECTION NEEDED
823 026646 013766 002426 000002      MOV      PRMLT+16,2(SP) ;OVERFLOW OCCURED, FORCE TO MAXIMUM ADDRESS
824 026654 000207      1$:   RTS      PC          ;RETURN
825
1142
1143                   ;THIS ROUTINE IS USED TO INPUT THE 'CONTROL SWITCHES'.
1144                   ;IF SWR<07>=1 THE PRESENT SETTING WILL BE TYPED AND THE NEW
1145                   ;SETTING IS READ AND STORED.
1146                   ;NOTE: THIS ROUTINE DESTROYS R3 AND R4
1147                   ;CALL
1148                   ;:   JSR      PC,GETSWR
1149                   ;:   RETURN          ;(C.SWR)=DESIRED CONTROL SWITCHES
1150
1151 026656 032777 000200 152270 GETSWR: BIT      #SW07,@SWR      ;READ CONTROL SWITCHES?
1152 026664 001430      BEQ      20$          ;NO--BRANCH
1153 026666 104401 026674      TYPE     ,65$          ;:TYPE ASCIZ STRING
1154 026672 000410      BR       64$          ;:GET OVER THE ASCIZ
1155                   ;:65$: .ASCIZ <CRLF>/SET SWR<07>=0/
1156                   ;64$:
1157 026714 012703 036750      10$:   MOV      #MSG.CS,R3      ;"CONTROL SWITCHES="
1158 026720 013704 001314      MOV      C.SWR,R4        ;PRESENT CONTROL SWITCH SETTINGS
1159 026724 004037 026750      JSR      R0,GETNUM      ;GET THE NEW SWITCH SETTINGS
1160 026730 000771      BR       10$          ; COMMA
1161 026732 000240      NOP
1162 026734 013737 001314 001320      MOV      C.SWR,SAVCSW   ;SAVE PREVIOUS VALUE
1163 026742 010437 001314      MOV      R4,C.SWR      ; DOUBLE PERIOD--SAVE NEW SWITCH SETTING
1164 026746 000207      20$:   RTS      PC          ;RETURN FROM CALL
1165
1166                   ;THIS ROUTINE WILL TYPE AN ASCIZ MESSAGE AND THEN
1167                   ;INPUT AN ASCIZ STRING AND CHANGE THE STRING TO OCTAL
1168                   ;IF REQUIRED.
1169                   ;NOTE: THIS ROUTINE DESTROYS R1
1170                   ;CALL
1171                   ;:   MOV      #ADR,R3      ;ADDRESS OF ASCIZ MESSAGE
1172                   ;:   MOV      #NUM,R4      ;OCTAL NUMBER
1173                   ;:   JSR      R0,GETNUM
1174                   ;:   RETURN1          ;INPUT TERMINATED WITH A COMMA
1175                   ;:   RETURN2         ;WITH A PERIOD
1176                   ;:   RETURN3        ;WITH A DOUBLE PERIOD
1177                   ;:                   ;R4=INPUT NUMBER AND
1178                   ;:                   ;R2=R4*32 FOR ALL
1179                   ;:                   ;THREE RETURNS
1180
1181 026750 010337 026756      GETNUM: MOV      R3,2$      ;SAVE MESSAGE POINTER
1182 026754 104401      1$:   TYPE     ;TYPE THE MESSAGE
1183 026756 000000      2$:   .WORD    0          ;MESSAGE POINTER GOES HERE
1184 026760 010446      MOV      R4,-(SP)      ;:SAVE R4 FOR TYPEOUT
    
```

1182	026762	104402		TYPOC		::GO TYPE--OCTAL ASCII(ALL DIGITS)
1183	026764	104401	036773	TYPE	.SLASH	::/
1184	026770	104411		RDLIN		::READ AN ASCIZ STRING
1185	026772	012601		MOV	(SP)+,R1	::ADDRESS OF FIRST CHARACTER
	026774	004037	030644	JSR	RO,CK.CHR	::CHECK ONE CHARACTER
	027000	026754		1\$		::ILLEGAL CHARACTER
	027002	026754		1\$		::CARRIAGE RETURN
	027004	027016		4\$		::/
	027006	027042		8\$		::..
	027010	027050		9\$		::..
	027012	027014		3\$		::DIGIT 0-9
1186	027014	005301		3\$: DEC	R1	::DECREMENT THE INPUT POINTER
1187	027016			4\$:		
	027016	004037	031104	JSR	RO,CK.NUM	::CHECK THE NUMBER
	027022	026754		1\$		::ILLEGAL INPUT
	027024	027036		7\$		::TERMINATED WITH A "..." OR "CR"
	027026	027034		6\$		::TERMINATED WITH A "..."
	027030	027032		5\$		::TERMINATED WITH A "..."
1188	027032	005720		5\$: TST	(R0)+	::DOUBLE PERIOD
1189	027034	005720		6\$: TST	(R0)+	::SINGLE PERIOD
1190	027036	010204		7\$: MOV	R2,R4	::COMMA--SAVE INPUT NUMBER
1191	027040	000414		11\$		::GO TO EXIT
1192	027042	105711		8\$: TSTB	(R1)	::TERMINATOR AFTER A COMMA?
1193	027044	001343		BNE	1\$	::NO--LOOP
1194	027046	000411		BR	11\$	::YES--EXIT
1195	027050	105711		9\$: TSTB	(R1)	::TERMINATOR AFTER A PERIOD?
1196	027052	001406		BEQ	10\$	::YES--EXIT
1197	027054	122721	000056	CMPB	#'..,(R1)+	::NO--DOUBLE PERIOD?
1198	027060	001335		BNE	1\$	::NO--LOOP
1199	027062	105711		TSTB	(R1)	::YES--TERMINATOR?
1200	027064	001333		BNE	1\$	::NO--LOOP
1201	027066	005720		TST	(R0)+	::DOUBLE PERIOD
1202	027070	005720		10\$: TST	(R0)+	::PERIOD
1203	027072	010402		11\$: MOV	R4,R2	::COMMA--POSITION THE
1204	027074	000302		SWAB	R2	::NUMBER IN CASE IT
1205	027076	006202		ASR	R2	::IS THE PRIORITY LEVEL
1206	027100	006202		ASR	R2	
1207	027102	006202		ASR	R2	
1208	027104	000200		RTS	RO	::EXIT
1209						
1210						::THIS ROUTINE IS USED TO CHANGE OR MODIFY
1211						::THE TEST PARAMETERS. IT GIVES THE OPERATOR
1212						::THE CAPABILITY OF SPECIFYING WHICH DRIVES TO TEST, WHICH
1213						::TESTS TO RUN AND HOW MANY TIMES TO REPEAT EACH TEST
1214						
1215	027106	104412		GT.PRM: SAVREG		::SAVE R0 - R5
1216	027110	005037	001330	GT.PR1: CLR	DRVSEL	::NO DRIVE SELECTED
1217	027114	104401	027122	TYPE	.65\$	::TYPE ASCIZ STRING
	027120	000406		BR	.64\$	::GET OVER THE ASCIZ
				::.65\$: .ASCIZ	<CRLF>/DRIVE(S)=/	
				64\$:		
1218	027136	104411		RDLIN		::READ TTY
1219	027140	012601		MOV	(SP)+,R1	::ADDRESS OF ASCIZ STRING
1220	027142	004037	030644	JSR	RO,CK.CHR	::CHECK ONE CHARACTER
	027146	027110		GT.PR1		::ILLEGAL CHARACTER
	027150	027110		GT.PR1		::CARRIAGE RETURN
	027152	027110		GT.PR1		::/

	027154	027110			GT.PR1					
	027156	027110			GT.PR1					
	02716C	027162			1\$					:DIGIT 0-9
1221	027162	005301		1\$:	DEC	R1				
1222	027164			2\$:						
	027164	012702	000007		MOV	#7,R2				:UPPER LIMIT OF INPUT
	027170	004037	030720		JSR	RO,CK.DIG				:CHECK THE DIGIT(S)
	027174	027110			GT.PR1					:ILLEGAL INPUT
	027176	027110			GT.PR1					:INPUT TO LARGE
	027200	027206			3\$					:TERMINATED WITH A '...' OR 'CR'
	027202	027232			4\$					:TERMINATED WITH A '...'
	027204	027232			4\$					:TERMINATED WITH A '...'
1223	027206	156237	031356	001330	3\$:	BISB	ATABIT(R2),DRVSEL			:SET THE DRIVE SELECTED BIT
1224	027214	105741			TSTB	-(R1)				:WAS THE LINE TERMINATED?
1225	027216	001362			BNE	2\$				:NO-GET THE NEXT DRIVE
1226	027220	005037	001332		CLR	TSTNMS				:DESELECT ALL TESTS
1227	027224	005037	001334		CLR	TSTNMS+2				
1228	027230	000405			BR	GTTST1				:YES--SELECT TEST
1229	027232	156237	031356	001330	4\$:	BISB	ATABIT(R2),DRVSEL			:SET THE SELECTED DRIVE BITS
1230	027240	104413			GT.PR2:	RESREG				:RESTORE RO - R5
1231	027242	000207			RTS	PC				:EXIT
1232										
1233	027244				GTTST1:					
	027244	104401	027252		TYPE	,65\$				::TYPE ASCIZ STRING
	027250	000403			BR	64\$				::GET OVER THE ASCIZ
					::65\$:	.ASCIZ	/TEST=			
					64\$:					
	027260				RDLIN					:READ AN ASCIZ STRING
1234	027260	104411			MOV	(SP)+,R1				:POINTER TO R1
1235	027262	012601			CMPB	#'.,(R1)				:DOUBLE PERIOD?
1236	027264	122711	000056		BNE	1\$				:NO--BRANCH
1237	027270	001007			CMPB	#'.,1(R1)				
1238	027272	122761	000056	000001	BNE	1\$				
1239	027300	001003			TSTB	2(R1)				: 'CR'?
1240	027302	105761	000002		BEQ	GT.PR2				:YES--EXIT
1241	027306	001754			CLR	TSTNMS				:NO TEST SELECTED
1242	027310	005037	001332	1\$:	CLR	TSTNMS+2				
1243	027314	005037	001334		CLR	TSTNMS+2				
1244	027320	005037	001336		CLR	OPNFLG				:NO TESTS TO BE OPENED
1245	027324	005037	001340		CLR	OPNFLG+2				
1246										
1247	027330	121127	000123		GTTST2:	CMPB	(R1),#'S			:ALL SEEK TESTS?
1248	027334	001004			BNE	1\$				:NO--BRANCH
1249	027336	052737	000377	001332	BIS	#377,TSTNMS				:YES--SELECT TESTS 0-7
1250	027344	000525			BR	GTTST3				
1251										
1252	027346	121127	000124	1\$:	CMPB	(R1),#'T				:ALL TIMING TESTS?
1253	027352	001004			BNE	2\$				:NO--BRANCH
1254	027354	052737	013000	001332	BIS	#13000,TSTNMS				:YES--SELECT TESTS 11,12 & 14
1255	027362	000516			BR	GTTST3				
1256										
1257	027364	004037	030570	2\$:	JSR	RO,CK.OCT				:OCTAL DIGIT?
1258	027370	000514			BR	GTTST4				:NO--BRANCH
1259	027372	010205			MOV	R2,R5				:YES--SAVE IT
1260	027374	005201			INC	R1				:MOVE TO NEXT CHARACTER
1261	027376	004037	030570		JSR	RO,CK.OCT				:OCTAL DIGIT
1262	027402	000405			BR	3\$				:NO--BRANCH
1263	027404	005201			INC	R1				:MOVE TO NEXT CHARACTER

1264	027406	006305		ASL	R5		:SCALE HIGH DIGIT
1265	027410	006305		ASL	R5		
1266	027412	006305		ASL	R5		
1267	027414	060502		ADD	R5,R2		:COMBINE HIGH & LOW DIGITS
1268	027416						
1271	027416	020227	000015	3\$: CMP	R2,#15		:VALID TEST NUMBER?
1272	027422	003310		BGT	GTTST1		:NO--BRANCH
1273	027424	010237	027616	MOV	R2,6\$		:SAVE THE TEST NUMBER
1274	027430	010204		MOV	R2,R4		:CONVERT TEST NUMBER INTO AN INDEX
1275	027432	042704	000017	BIC	#17,R4		:CLEAR UNWANTED BITS
1276	027436	006204		ASR	R4		:SHIFT THE BITS
1277	027440	006204		ASR	R4		
1278	027442	006204		ASR	R4		
1279	027444	006302		ASL	R2		
1280	027446	056264	001560 001332	BIS	BITS(R2),TSTNMS(R4)		:SELECT TEST
1281	027454	121127	000055	CMFB	(R1),#'-		:TEST STRING?
1282	027460	001060		BNE	GTTST4		:NO--BRANCH
1283	027462	005201		INC	R1		:YES--MOVE TO NEXT CHARACTER
1284	027464	004037	070570	JSR	RO,CK.OCT		:OCTAL DIGIT?
1285	027470	000665		BR	GTTST1		:NO--BRANCH
1286	027472	010205		MOV	R2,R5		:YES--SAVE IT
1287	027474	005201		INC	R1		:MOVE TO NEXT CHARACTER
1288	027476	004037	030570	JSR	RO,CK.OCT		:OCTAL DIGIT?
1289	027502	000405		BR	4\$		:NO--BRANCH
1290	027504	005201		INC	R1		:YES--MOVE TO NEXT CHARACTER
1291	027506	006305		ASL	R5		:SCALE HIGH DIGIT
1292	027510	006305		ASL	R5		
1293	027512	006305		ASL	R5		
1294	027514	060502		ADD	R5,R2		:COMBINE HIGH & LOW DIGIT
1295	027516						
1298	027516	020227	000015	4\$: CMP	R2,#15		:VALID TEST NUMBER?
1299	027522	003250		BGT	GTTST1		:NO--BRANCH
1300	027524	023702	027616	CMF	6\$,R2		:IS THE FIRST NUMBER OF THE
1301							:STRING SMALLER THAN THE LAST?
1302	027530	002245		BGE	GTTST1		:NO--BRANCH
1303	027532	010246		MOV	R2,-(SP)		:SAVE ENDING TEST NUMBER
1304	027534	013702	027616	MOV	6\$,R2		:GET STARTING TEST NUMBER
1305	027540	012637	027616	MOV	(SP)+,6\$		:STORE ENDING TEST NUMBER
1306	027544	006337	027616	ASL	6\$		:SHIFT ENDING TEST NUMBER
1307	027550	006302		ASL	R2		:SHIFT TEST NUMBER
1308	027552	010204		MOV	R2,R4		:COPY TEST NUMBER INTO R4
1309	027554	042704	000037	5\$: BIC	#37,R4		:CLEAR LOWER BITS
1310	027560	006204		ASR	R4		:SHIFT THE TEST NUMBER
1311	027562	006204		ASR	R4		
1312	027564	006204		ASR	R4		
1313	027566	006204		ASR	R4		
1314	027570	056264	001560 001332	BIS	BITS(R2),TSTNMS(R4)		:SELECT THE TEST
1315	027576	062702	000002	ADD	#2,R2		:INCREMENT THE TEST NUMBER
1316	027602	020237	027616	CMF	R2,6\$		:SEE IF FINISHED
1317	027606	101761		BLOS	5\$		:BR IF NOT
1318	027610	162702	000002	SUB	#2,R2		:CORRECT TEST NUMBER
1319	027614	000402		BR	GTTST4		:CONTINUE
1320	027616	000000		6\$: .WORD	0		:STORE TEST NUMBER HERE
1321							
1322	027620	005201		GTTST3: INC	R1		:MOVE TO NEXT CHARACTER
1323	027622	121127	000056	GTTST4: CMFB	(R1),#'		:PERIOD'?
1324	027626	001461		BEQ	GTTST5		:YES--BRANCH

```

1325 027630 005737 001332      TST      TSTNMS      :ANY TEST SELECTED THIS CYCLE?
1326 027634 001005      BNE      1$        :BR IF YES
1327 027636 005737 001334      TST      TSTNMS+2  :ANY TEST SELECTED THIS CYCLE ?
1328 027642 001002      BNE      1$        :BR IF YES
1329 027644 000137 027244      JMP      GTTST1    :NO
1330
1331                               :CHECK TO OPEN TEST FOR PARAMETER CHANGE
1332
1333 027650 121127 000057      1$:      CMPB      (R1),#'/      :'OPEN'?
1334 027654 001024      BNE      4$        :NO--BRANCH
1335 027656 126127 177777 000123      CMPB      -1(R1),#'S      :ALL SEEK TESTS?
1336 027664 001004      BNE      2$        :NO--BRANCH
1337 027666 052737 000377 001336      BIS      #377,OPNFLG      :YES--OPEN TESTS 0-7
1338 027674 000421      BR       5$
1339 027676 126127 177777 000124      2$:      CMPB      -1(R1),#'T      :ALL TIMING TESTS?
1340 027704 001004      BNE      3$        :NO--BRANCH
1341 027706 052737 013000 001336      BIS      #13000,OPNFLG    :YES--OPEN TESTS 11,12 & 14
1342 027714 000411      BR       5$
1343 027716 056264 001560 001336      3$:      BIS      BITS(R2),OPNFLG(R4) ;YES--SET BITS FOR TEST TO OPEN
1344 027724 000405      BR       5$
1345
1346 027726 121127 000054      4$:      CMPB      (R1),#',      :'COMMA'?
1347 027732 001402      BEQ      5$        :BR IF YES
1348 027734 000137 027244      JMP      GTTST1    :NO
1349 027740 005201      5$:      INC       R1        :MOVE TO NEXT CHARACTER
1350 027742 105711      TSTB     (R1)      :'CR'?
1351 027744 001402      BEQ      6$        :BR IF 'CR'
1352 027746 000137 027330      JMP      GTTST2    :NO--GO GET NEXT CHARACTER
1353 027752 005737 001336      6$:      TST      OPNFLG    :ANY TESTS TO OPEN ?
1354 027754 001042      BNE      OPNTST    :BR IF YES
1355 027760 005737 001340      TST      OPNFLG+2  :ANY TESTS TO OPEN ?
1356 027764 001037      BNE      OPNTST    :BR IF YES
1357 027766 000137 027244      JMP      GTTST1    :NO--START AGAIN
1358
1359 027772 005201      GTTST5: INC      R1        :MOVE TO NEXT CHARACTER
1360 027774 121127 000056      CMPB     (R1),#'.      :'PERIOD'?
1361 030000 001414      BEQ      GTTST6    :YES--BRANCH
1362 030002 165711      TSTB     (R1)      :'CR'?
1363 030004 001402      BEQ      1$        :YES--BRANCH
1364 030006 000137 027244      JMP      GTTST1
1365 030012 005737 001336      1$:      TST      OPNFLG    :ANY TESTS TO OPEN ?
1366 030016 001022      BNE      OPNTST    :BR IF YES
1367 030020 005737 001340      TST      OPNFLG+2  :ANY TESTS TO OPEN ?
1368 030024 001017      BNE      OPNTST    :BR IF YES
1369 030026 000137 027240      JMP      GT.PR2    :NO--GO START TESTING
1370
1371 030032 005201      GTTST6: INC      R1        :MOVE TO NEXT CHARACTER
1372 030034 105711      TSTB     (R1)      :'CR'?
1373 030036 001402      BEQ      1$        :YES--BRANCH
1374 030040 000137 027244      JMP      GTTST1
1375 030044 005737 001336      1$:      TST      OPNFLG    :ANY TESTS TO OPEN ?
1376 030050 001005      BNE      OPNTST    :BR IF YES
1377 030052 005737 001340      TST      OPNFLG+2  :ANY TESTS TO OPEN ?
1378 030056 001002      BNE      OPNTST    :BR IF YES
1379 030060 000137 027240      JMP      GT.PR2    :NO--GO START TESTING
1380
1381                               :OPEN THE SELECTED TEST FOR CHANGES
    
```



```

1382
1383 030064 104412          OPNTST: SAVREG          ;SAVE R0 - R5
1384 030066 005027          CLR          (PC)+      ;START WITH TEST 0
1385 030070 000000          OPN.CT: .WORD 0         ;COUNT STORED HERE
1386 030072 000411          BR          OPN.2       ;SKIP THE INCREMENT
1387
1388 030074 005237 030070    OPN.1:  INC          OPN.CT      ;MOVE TO THE NEXT TEST
1389 030100 022737 000015 030070  CMP          #15,OPN.CT    ;TEST NUMBER TOO BIG?
1392 030106 002003          BGE          OPN.2       ;NO--OPEN THE NEXT TEST
1393 030110 104413          RESREG
1394 030112 000137 027244    JMP          GTTST1      ;RESTORE R0 - R5
1395                                     ;YES--GO ASK FOR MORE TESTS
1396 030116 013705 030070    OPN.2:  MOV          OPN.CT,R5  ;SETUP TO USE THE
1397 030122 006305          ASL          R5          ;TEST NUMBER AS AN INDEX
1398 030124 013703 030070    MOV          OPN.CT,R3    ;GET INDEX
1399 030130 042703 000017    BIC          #17,R3      ;CLEAR LOWER TEST BITS
1400 030134 006203          ASR          R3          ;SHIFT TEST NUMBER
1401 030136 006203          ASR          R3
1402 030140 006203          ASR          R3
1403 030142 036563 001560 001336 BIT          BITS(R5),OPNFLG(R3) ;OPEN THIS TEST?
1404 030150 001751          BEQ          OPN.1       ;NO--MOVE TO NEXT TEST
1405 030152 104401 030160    TYPE          ,65$      ;TYPE ASCIZ STRING
1406 030156 000404          BR          64$         ;GET OVER THE ASCIZ
1407 030170          ;:65$: .ASCIZ / TEST /
1408 030170 013746 030070    64$:  MOV          OPN.CT,-(SP) ;:SAVE OPN.CT FOR TYPEOUT
1409 030174 104403          TYPOS
1410 030176 002          .BYTE 2              ;:GO TYPE--OCTAL ASCII
1411 030177 000          .BYTE 0              ;:TYPE 2 DIGIT(S)
1412 030200 104401 001231    TYPE          ,$CRLF    ;:SUPPRESS LEADING ZEROS
1413 030204 016500 002352    MOV          PRMPT(R5),R0 ;:CR-LF
1414 030210 011046          MOV          (R0),-(SP)  ;:PICKUP PARAMETER POINTER
1415 030212 012702 002320    MOV          #PRM,R2     ;:SAVE THE VARIABLE INDICATOR
1416 030216 000405          BR          2$          ;:FIRST ADDRESS OF TABLE
1417 030220 006216          1$:  ASR          (SP)    ;:CHECK FOR A VARIABLE
1418 030222 103403          BCS          2$          ;:GO MOVE THIS ONE
1419 030224 001404          BEQ          OPNPRM     ;:DONE
1420 030226 005722          TST          (R2)+      ;:BUMP THE POINTER
1421 030230 000773          BR          1$
1422 030232 012022          2$:  MOV          (R0)+,(R2)+ ;:MOVE THIS VARIABLE INTO THE
1423 030234 000771          BR          1$          ;:COMMON AREA
1424
1425 030236 013716 002320    OPNPRM: MOV          PRM,(SP) ;:GET THE VARIABLE INDICATOR
1426 030242 005004          CLR          R4         ;:ZERO THE INDEX
1427 030244 006216          1$:  ASR          (SP)    ;:CHECK FOR A VARIABLE
1428 030246 103403          BCS          3$          ;:GO GET IT
1429 030250 001772          BEQ          OPNPRM     ;:OUT OF VARIABLES
1430 030252 005724          2$:  TST          (R4)+      ;:UPDATE THE INDEX
1431 030254 000773          BR          1$
1432 030256          3$:
1433 030256 104401 040053          TYPE          ,BLNKS2   ;:TYPE 2 BLANKS
1434 030262 016437 002434 030272 MOV          PRMSG(R4),4$ ;:TYPE THE NAME OF THIS VARIABLE
1435 030270 104401          TYPE
1436 030272 000000          4$:  .WORD 0
1437 030274 104401 036746          TYPE          ,MSG.EQ   ;:TYPE '='
1438 030300 016446 002322    MOV          RPT(R4),-(SP) ;:PUT RPT(R4) ON THE STACK
1439 030304 004737 022212    JSR          PC,$SB2    ;:CHANGE TO DECIMAL ASCIZ
    
```



1438	030310	004737	022442		JSR	PC,\$SUPRS		:TYPE WITHOUT LEADING ZEROS
			036773		TYPE	.SLASH		:TYPE ' / '
1439	030320	104411			RDLIN			
1440	030322	012601			MOV	(SP)+,R1		:READ AN ASCIZ STRING
1441	030324	004037	030644		JSR	RO,CK.CHR		:CHECK ONE CHARACTER
	030320	030256			3\$			:ILLEGAL CHARACTER
	030332	030252			2\$			:CARRIAGE RETURN
	030334	030402			9\$			: /
	030336	030344			5\$			: ..
	030340	030352			6\$			: ...
	030342	030400			8\$			:DIGIT 0-9
1442	030344	105711		5\$:	TSTB	(R1)		: 'CR'?
1443	030346	001343			BNE	3\$		:NO--STAY ON THIS VARIABLE
1444	030350	000740			BR	2\$		:YES--MOVE TO NEXT VARIABLE
1445	030352	105711		6\$:	TSTB	(R1)		:IS THERE A 'CR' AFTER THE PERIOD?
1446	030354	001002			BNE	7\$		:NO
1447	030356	000137	030436		JMP	OPN.N2		:YES--GO CLOSE THIS TEST
1448	030362	122721	000056	7\$:	CMPB	#'.,(R1)+		:DOUBLE PERIOD?
1449	030366	001333			BNE	3\$		:NO--GO ASK FOR THIS VARIABLE
1450	030370	105711			TSTB	(R1)		:YES--IS A 'CR' AFTER THE DOUBLE PERIOD?
1451	030372	001331			BNE	3\$		:NO--ASK FOR THIS VARIABLE AGAIN
1452	030374	000137	030454		JMP	OPN.X2		:YES--CLOSE ALL TEST
1453								
1454	030400	005301		8\$:	D--	R1		:BACK THE POINTER UP BY ONE
1455	030402			9\$:				
	030402	016402	002410		MOV	PRMLT(R4),R2		:UPPER LIMIT OF INPUT
	030406	004037	030720		JSR	RO,CK.DIG		:CHECK THE DIGIT(S)
	030412	030256			3\$			:ILLEGAL INPUT
	030414	030256			3\$			:INPUT TO LARGE
	030416	030424			10\$			:TERMINATED WITH A '...' OR 'CR'
	030420	030432			OPN.N1			:TERMINATED WITH A '...'
	030422	030450			OPN.X1			:TERMINATED WITH A '...'
1456	030424	010264	002322	10\$:	MOV	R2,RPT(R4)		:SAVE THIS VARIABLE
1457	030430	000710			BR	2\$		:MOVE TO NEXT VARIABLE
1458								
1530	030432	010264	002322	OPN.N1:	MOV	R2,RPT(R4)		:SAVE THIS VARIABLE
1531	030436	005726		OPN.N2:	TST	(SP)+		:CLEAN OFF THE STACK
1532	030440	004737	030510		JSR	PC,CLOSE		:CLOSE THIS TEST
1533	030444	000137	030674		JMP	OPN.1		:GO OPEN THE NEXT TEST
1534								
1535	030450	010264	002322	OPN.X1:	MOV	R2,RPT(R4)		:SAVE THIS VARIABLE
1536	030454	005726		OPN.X2:	TST	(SP)+		:CLEAN OFF THE STACK
1537	030456	004737	030510	1\$:	JSR	PC,CLOSE		:CLOSE THIS TEST
1538	030462	005725		2\$:	TST	(R5)+		:UPDATE THE INDEX
1539	030464	020527	000034		CMP	R5,#16*2		:INDEX TO BIG?
1540	030470	002403			BLT	3\$		:NO--BRANCH
1541	030472	104413			RESREG			:RESTORE R0 - R5
1542	030474	000137	027240		JMP	GT.PR2		:GO TO EXIT
1543	030500	036503	001560	3\$:	BIT	BITS(R5),R3		:IS THIS TEST OPEN FOR CHANGE?
1544	030504	001364			BNE	1\$		:YES--GO CLOSE IT
1545	030506	000765			BR	2\$		:NO--MOVE TO NEXT TEST
1546								
1547								:CLOSE CURRENT TEST THAT WAS OPEN FOR CHANGES
1548								
1549	030510	104412		CLOSE:	SAVREG			:SAVE R0 - R5
1550	030512	012700	002320		MOV	#PRM,R0		: 'FROM' ADDRESS
1551	030516	016501	002352		MOV	PRMPT(R5),R1		: 'TO' ADDRESS

1552 030522 012002  
 1553 030524 012103  
 1554 030526 012704 000001  
 1555 030532 C\*0402  
 1556 030534 001403  
 1557 030536 030403  
 1558 030540 001404  
 1559 030542 011011  
 1560 030544 030403  
 1561 030546 001401  
 1562 030550 005721  
 1563 030552 005720  
 1564 030554 006304  
 1565 030556 032704 002000  
 1566 030562 001763  
 1567 030564 104413  
 1568 030566 000207  
 1569  
 1570  
 1571  
 1572  
 1573  
 1574  
 1575  
 1576  
 1577  
 1578  
 1579 030570 121127 000060  
 1580 030574 103407  
 1581 030576 121127 000067  
 1582 030602 101004  
 1583 030604 111102  
 1584 030606 042702 177770  
 1585 030612 005720  
 1586 030614 000200  
 1587  
 1588  
 1589  
 1590  
 1591  
 1592  
 1593  
 1594  
 1595  
 1596  
 1597 030616 121127 000060  
 1598 030622 103407  
 1599 030624 121127 000071  
 1600 030630 101004  
 1601 030632 111102  
 1602 030634 042702 000060  
 1603 030640 005720  
 1604 030642 000200  
 1605  
 1606  
 1607  
 1608

```

MOV (R0)+,R2 ;'FROM' INDICATOR
MOV (R1)+,R3 ;'TO' INDICATOR
MOV #1,R4 ;TEST BIT START A 'RPT'
1$: BIT R4,R2 ;PARAMETER TO BE MOVED?
    BEQ 2$ ;NO--BRANCH
    BIT R4,R3 ;A PLACE TO PUT IT?
    BEQ 3$ ;NO--BRANCH
    MOV (R0),(R1) ;YES--MOVE 'FROM' TO 'TO'
2$: BIT R4,R3 ;'TO' PARAMETER?
    BEQ 3$ ;NO--BRANCH
    TST (R1)+ ;YES--UPDATE THE POINTER
3$: TST (R0)+ ;UPDATE FROM POINTER
    ASL R4 ;POSITION THE TEST BIT
    BIT #BIT10,R4 ;DONE?
    BEQ 1$ ;NO--BRANCH
    RESREG ;RESTORE R0 - R5
    RTS PC ;RETURN
    
```

:THIS ROUTINE IS USED TO CHECK IF AN  
 :ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.

```

:CALL
:   MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
:   JSR RO,CK.OCT ;CHECK THE CHARACTER
:   RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
:   RETURN2 ;CHARACTER IS IN R2 AS A
:           ;OCTAL DIGIT
    
```

```

CK.OCT: CMPB (R1),#'0 ;LESS THAN ZERO?
        BLO 1$ ;YES -- BRANCH
        CMPB (R1),#'7 ;GREATER THAN SEVEN?
        BHI 1$ ;YES -- BRANCH
        MOVB (R1),R2 ;GET THE CHARACTER
        BIC #^C7,R2 ;STRIP AWAY THE ASCII
1$: TST (R0)+ ;ADJUST FOR RETURN
    RTS R0 ;RETURN
    
```

:THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER  
 :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.

```

:CALL
:   MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
:   JSR RO,CK.DEC ;CHECK THE CHARACTER
:   RETURN1 ;NOT BETWEEN 0 AND 9
:   RETURN2 ;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
1$: TST (R0)+ ;ADJUST FOR RETURN
    RTS R0 ;RETURN
    
```

:THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO  
 :DETERMINE WHAT IT IS.  
 :CALL

```

1609          :      MOV      #ADR,R1      :ADDRESS OF ASCII CHARACTER
1610          :      JSR      R0,CK.CHR    :CHECK CHARACTER
1611          :      RETURN   ADR1          :UNKNOWN CHARACTER
1612          :      RETURN   ADR2          :CARRIAGE RETURN * (R1)=ADR+1
1613          :      RETURN   ADR3          :SLASH * (R1)=ADR+1
1614          :      RETURN   ADR4          :COMMA * (R1)=ADR+1
1615          :      RETURN   ADR5          :PERIOD * (R1)=ADR+1
1616          :      RETURN   ADR6          :DIGIT BETWEEN 0 AND 9.
1617          :      :R2 = DIGIT * (R1)=ADR+1
1618
1619 030644 105711      CK.CHR: TSTB      (R1)      :'"CARRIAGE RETURN"'?
1620 030646 001420      BEQ        4$          :YES -- BRANCH
1621 030650 121127 000057  CMPB      (R1),# '/'      :'"SLASH"'?
1622 030654 001414      BEQ        3$          :YES -- BRANCH
1623 030656 121127 000054  CMPB      (R1),# ','      :'"COMMA"'?
1624 030662 001410      BEQ        2$          :YES -- BRANCH
1625 030664 121127 000056  CMPB      (R1),# '.'      :'"PERIOD"'?
1626 030670 001404      BEQ        1$          :YES -- BRANCH
1627 030672 004037 030616  JSR      R0,CK.DEC      :'"DIGIT"'?
1628 030676 000406      BR        5$          :NO -- BRANCH
1629 030700 005720      TST      (R0)+        :DIGIT BETWEEN 0-9
1630 030702 005720      1$: TST      (R0)+        :PERIOD
1631 030704 005720      2$: TST      (R0)+        :COMMA
1632 030706 005720      3$: TST      (R0)+        :SLASH
1633 030710 005720      4$: TST      (R0)+        :CARRIAGE RETURN
1634 030712 005201      INC      R1          :MOVE POINTER TO NEXT CHARACTER
1635 030714 011000      5$: MOV      (R0),R0      :UNKNOWN CHARACTER
1636 030716 000200      RTS      R0          :RETURN
1637
1638          :THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
1639          :CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
1640          :CALL
1641          :      MOV      #ADR,R1      :ADDRESS OF ASCII STRING
1642          :      MOV      #NUM,R2      :MAX. MAGNITUDE OF INPUT NUMBER
1643          :      JSR      R0,CK.DIG      :CHECK DIGITS
1644          :      RETURN   ADR1          :ILLEGAL CHARACTER -- R2=?
1645          :      RETURN   ADR2          :INPUT NUMBER TO LARGE -- R2=?
1646          :      RETURN   ADR3          :'"COMMA"' -- R2 = NUMBER
1647          :      RETURN   ADR4          :'"PERIOD"' -- R2 = NUMBER
1648          :      RETURN   ADR5          :'"PERIOD-PERIOD"' -- R2 = NUMBER
1649
1650 030720 010446      CK.DIG: MOV      R4,-(SP)      :SAVE R4
1651 030722 010346      MOV      R3,-(SP)      :SAVE R3
1652 030724 010246      MOV      R2,-(SP)      :SAVE THE MAX. SIZE ON THE STACK
1653 030726 005002      CLR      R2          :START WITH 0
1654 030730 005003      CLR      R3
1655 030732 005004      CLR      R4
1656 030734 004037 030644  JSR      R0,CK.CHR      :CHECK ONE CHARACTER
1657          :      9$          :ILLEGAL CHARACTER
1658          :      9$          :CARRIAGE RETURN
1659          :      9$          :"/"
1660          :      9$          :":"
1661          :      9$          :";"
1662          :      9$          :"!"
1663          :      1$         :DIGIT 0-9
1664          :      1$: ASL      R3          :2
1665          :      MOV      R3,-(SP)      :SAVE *2
1666          :      ASL      R3          :4
    
```

```

1660 030762 006303          ASL      R3          ;8
1661 030764 062603          ADD      (SP)+,R3    ;(*8)+(*2)=*10.
1662 030766 060203          ADD      R2,R3      ;UPDATE THE INPUT NUMBER
1663 030770 004037 030644  JSR      R0,CK.CHR  ;CHECK ONE CHARACTER
                                9$          ;ILLEGAL CHARACTER
                                2$          ;CARRIAGE RETURN
                                9$          ;/
                                4$          ;
                                3$          ;
                                1$          ;DIGIT 0-9
1664 031010 005301          2$: DEC      R1      ;BACKUP THE CHARACTER POINTER
1665 031012 000401          BR       4$          ;CONTINUE
1666 031014 005724          3$: TST      (R4)+   ;'PERIOD'
1667 031016 005724          4$: TST      (R4)+   ;'COMMA' OR 'CR'
1668 031020 004037 030644  JSR      R0,CK.CHR  ;CHECK ONE CHARACTER
                                9$          ;ILLEGAL CHARACTER
                                7$          ;CARRIAGE RETURN
                                9$          ;/
                                9$          ;
                                5$          ;
                                6$          ;DIGIT 0-9
1669 031040 005724          5$: TST      (R4)+   ;'PERIOD-PERIOD'
1670 031042 105711          TSTB    (R1)        ;'CR'?
1671 031044 001405          BEQ     7$          ;YES--BRANCH
1672 031046 000410          BR      9$
1673 031050 125127 177776 000054 6$: CMPB    -2(R1),#' , ;WAS CHARACTER BEFORE THE DIGIT A COMMA?
1674 031056 001004          BNE    9$          ;NO--EXIT
1675 031060 020316          7$: CMP     R3,(SP)  ;INPUT TO LARGE?
1676 031062 101001          BHI    8$          ;YES -- BRANCH
1677 031064 060400          ADD     R4,R0      ;ADJUST RETURN ADDRESS
1678 031066 005720          8$: TST      (R0)+   ;
1679 031070 010302          9$: MOV     R3,R2    ;NUMBER TO R2
1680 031072 005726          TST    (SP)+       ;CLEAN MAX. SIZE OFF OF STACK
1681 031074 012603          MOV     (SP)+,R3   ;RESTORE R3
1682 031076 012604          MOV     (SP)+,R4   ;RESTORE R4
1683 031100 011000          MOV     (R0),R0    ;GET RETURN ADDRESS
1684 031102 000200          RTS     R0         ;RETURN
1685
1686          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
1687          ;AND FORMS AN OCTAL NUMBER IN R2
1688          ;CALL:
1689          ;     MOV     #ADR,R1          ;ADDRESS OF ASCIZ STRING
1690          ;     JSR     R0,CK.NUM       ;GO FORM THE NUMBER
1691          ;     RETURN ADR1            ;ILLEGAL CHARACTER IN THE INPUT STRING
1692          ;     RETURN ADR2            ;'COMMA' OR 'CR'--R2=NUMBER
1693          ;     RETURN ADR3            ;'PERIOD'--R2=NUMBER
1694          ;     RETURN ADR4            ;'PERIOD-PERIOD'--R2=NUMBER
1695
1696 031104 010346          CK.NUM: MOV     R3,-(SP) ;SAVE R3
1697 031106 005003          CLR     R3         ;START NUMBER AT ZERO
1698 031110 004037 030570  JSR     R0,CK.OCT  ;OCTAL DIGIT?
1699 031114 000440          BR     6$         ;NO--BRANCH
1700 031116 005201          1$: INC     R1      ;MOVE TO NEXT CHARACTER
1701 031120 006303          ASL     R3         ;FOR THE OCTAL NUMBER IN R3
1702 031122 103435          BCS    6$         ;DON'T LET IT GET TO BIG
1703 031124 006303          ASL     R3
1704 031126 103433          BCS    6$
    
```

1705	031130	006303		ASL	R3	
1706	031132	103431		BCS	6\$	
1707	031134	060203		ADD	R2,R3	
1708	031136	004037	030570	JSR	R0,CK.OCT	: IS THIS AN OCTAL DIGIT?
1709	031142	000401		BR	2\$	: NO--FIND OUT WHAT IT IS
1710	031144	000764		BR	1\$	: YES--MAKE IT PART OF THE NUMBER
1711	031146	010302		2\$: MOV	R3,R2	: SAVE THE OCTAL NUMBER
1712	031150	005003		CLR	R3	: START WITH ZERO INDEX
1713	031152	004037	030644	JSR	R0,CK.CHR	: CHECK ONE CHARACTER
	031156	031216		6\$		: ILLEGAL CHARACTER
	031160	031206		5\$		: CARRIAGE RETURN
	031162	031216		6\$		: '/'
	031164	031206		5\$		: ''
	031166	031172		3\$		: ''
	031170	031216		6\$		: DIGIT 0-9
1714	031172	005723		3\$: TST	(R3)+	: 'PERIOD'
1715	031174	121127	000056	CMPB	(R1),#'	: 'PERIOD-PERIOD'?
1716	031200	001002		BNE	5\$	: NO--BRANCH
1717	031202	005201		INC	R1	: YES--ADVANCE THE POINTER
1718	031204	005723		4\$: TST	(R3)+	: 'PERIOD-PERIOD'
1719	031206	005723		5\$: TST	(R3)+	: 'COMMA'
1720	031210	105711		TSTB	(R1)	: 'CR'?
1721	031212	001001		BNE	6\$	: NO--BRANCH
1722	031214	060300		ADD	R3,R0	: YES--SAVE THE OCTAL NUMBER
1723	031216	012603		6\$: MOV	(SP)+,R3	: RESTORE R3
1724	031220	011000		MOV	(R0),R0	: PICKUP EXIT ADDRESS
1725	031222	000200		RTS	R0	: RETURN

7  
17  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
53  
54  
55  
56  
57  
58  
59  
60  
63  
64  
65  
66  
67  
68  
69  
70  
73

.SBTTL SINGLE/DUAL PORT RH70/RM80 DRIVER (REV 0.17)

:COPYRIGHT (C) 1979,1981  
:DIGITAL EQUIPMENT CORP.  
:MAYNARD, MA 01754  
:AUTHOR(S): CHUCK HESS  
:REVISED BY: MIKE LEAVITT

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

.STORAGE FOR RMDs, RMER1 AND RMER2 ON AN ERROR '2'

:RMERRS = RMDs  
:RMERRS+2 = RMER1  
:RMERRS+4 = RMER2

031224 000000 000000 000000 RMERRS: .WORD 0,0,0

:TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)

:DRVACT=0 IF DRIVE IS IDLE  
:DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND  
:DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

031232	000	DRVACT: .BYTE	0	:DRIVE	0
031233	000	.BYTE	0	:DRIVE	1
031234	000	.BYTE	0	:DRIVE	2
031235	000	.BYTE	0	:DRIVE	3
031236	000	.BYTE	0	:DRIVE	4
031237	000	.BYTE	0	:DRIVE	5
031240	000	.BYTE	0	:DRIVE	6
031241	000	.BYTE	0	:DRIVE	7

:TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)

:DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT  
:DRVSTA>0 IF DRIVE IS ONLINE  
:DRVSTA<0 IF DRIVE IS UNSAFE

031242	000	DRVSTA: .BYTE	0	:DRIVE	0
031243	000	.BYTE	0	:DRIVE	1
031244	000	.BYTE	0	:DRIVE	2
031245	000	.BYTE	0	:DRIVE	3
031246	000	.BYTE	0	:DRIVE	4
031247	000	.BYTE	0	:DRIVE	5
031250	000	.BYTE	0	:DRIVE	6
031251	000	.BYTE	0	:DRIVE	7

:TABLE OF DRIVE TYPES (DRVSTYP=8 BYTES)

:DRVSTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)  
:DRVSTYP=1 IF DRIVE IS RM80  
:DRVSTYP=-1 IF NOT RM80

031252	000	DRVSTYP: .BYTE	0	:DRIVE	0
031253	000	.BYTE	0	:DRIVE	1
031254	000	.BYTE	0	:DRIVE	2
031255	000	.BYTE	0	:DRIVE	3

031256 000  
 031257 000  
 031260 000  
 031261 000

.BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7

74  
 75  
 76  
 77  
 78

:TABLE OF DUAL PORT INITIALIZATION INDICATORS  
 ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE  
 ;DPINT<0 IF INITIALIZATION IS IN PROGRESS

79 031262 000  
 82 031263 000  
 031264 000  
 031265 000  
 031266 000  
 031267 000  
 031270 000  
 031271 000

DPINT: .BYTE 0 ;DRIVE 0  
 .BYTE 0 ;DRIVE 1  
 .BYTE 0 ;DRIVE 2  
 .BYTE 0 ;DRIVE 3  
 .BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7

83  
 84  
 85  
 86  
 87

:TABLE OF PENDING DUAL PORT REQUESTS  
 ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE  
 ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE

88 031272 000  
 91 031273 000  
 031274 000  
 031275 000  
 031276 000  
 031277 000  
 031300 000  
 031301 000

DPRQS: .BYTE 0 ;DRIVE 0  
 .BYTE 0 ;DRIVE 1  
 .BYTE 0 ;DRIVE 2  
 .BYTE 0 ;DRIVE 3  
 .BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7

92  
 93  
 94  
 95  
 98

:DRIVE REQUEST QUE WORDS

031302 000000  
 031304 000000  
 031306 000000  
 031310 000000  
 031312 000000  
 031314 000000  
 031316 000000  
 031320 000000

QDRV: .WORD 0 ;DRIVE 0  
 .WORD 0 ;DRIVE 1  
 .WORD 0 ;DRIVE 2  
 .WORD 0 ;DRIVE 3  
 .WORD 0 ;DRIVE 4  
 .WORD 0 ;DRIVE 5  
 .WORD 0 ;DRIVE 6  
 .WORD 0 ;DRIVE 7

99  
 100  
 101  
 102  
 103

:TRANSFER WAIT FLAG (TRNSWT=1 WORD)  
 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF  
 ;'DPB' OF THE I/O OPERATION.

104 031322 000000

TRNSWT: .WORD 0

105  
 106  
 107  
 108  
 109

:SEARCH WAIT KEYS (SRCHWT=1 WORD)  
 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF  
 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O  
 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.  
 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.

111  
 112 031324 000000

SRCHWT: .WORD 0

113  
 114

:RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)

```

115           :ACTDRV=0 IF DRIVER IS INACTIVE
116           :ACTDRV>0 IF DRIVER IS ACTIVE
117
118 031326     000           ACTDRV: .BYTE  0
119
120           :SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
121           :ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
122           :ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
123
124 031327     000           ACTSTR: .BYTE  0
125
126           :SAVE REGISTERS FLAG (SAVEFG =1 WORD)
127           :SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
128           :OPERATION IS COMPLETED AS PER (DPB+14).
129           :SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
130           : (DPB+14), AFTER AN ERROR.
131
132 031330     000000       SAVEFG: .WORD  0
133
134           :SEEK FLAG (SEEKFG=1 WORD)
135           :SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
136           :FOR A DATA TRANSFER START A SEARCH COMMAND
137           :SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
138           :DISREGARD THE WINDOW
139
140 031332     177777       SEEKFG: .WORD  -1
141
142           :TIMEOUT TABLE (TIMER=8 WORDS)
143           :THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
144
145 031334     177777       TIMER:  .WORD  -1           :DRIVE 0
146 031336     177777       .WORD  -1           :DRIVE 1
147 031340     177777       .WORD  -1           :DRIVE 2
148 031342     177777       .WORD  -1           :DRIVE 3
149 031344     177777       .WORD  -1           :DRIVE 4
150 031346     177777       .WORD  -1           :DRIVE 5
151 031350     177777       .WORD  -1           :DRIVE 6
152 031352     177777       .WORD  -1           :DRIVE 7
153
154           :DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
155           :DTUW<0 IF NO DATA TRANSFER UNDERWAY
156           :DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
157
158 031354     177777       DTUW:  .WORD  -1
159
160           :ATTENTION BITS TABLE (ATABIT=8 BYTES)
161           :THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
162           :ATTENTION BIT
163
164 031356     001     002     004  ATABIT: .BYTE  1,2,4,10,20,40,100,200
165
166           :STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM80),
167           :RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
168
169 031366     176700       RMADR:  .WORD  176700
170 031370     000254     000240  RMVEC:  .WORD  254,5*32.
    
```



171  
172  
173 031374 000005  
174  
175  
176  
177 000000  
178 000002  
179 000004  
180 000006  
181 000010  
182 000012  
183 000014  
184 000016  
185 000020  
186 000022  
187 000024  
188 000026  
189 000030  
190 000032  
191 000034  
192 000036  
193 000040  
194 000042  
195 000044  
196 000046  
201

:SEARCH DIFFERENCE IS 5 FIVE SECTORS

MXLNDW: .WORD 5

:DEFINITIONS OF THE RH70/RM80 ADDRESS INDEXES

RMCS1	= 0	:CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
RMWC	= 2	:WORD COUNT REGISTER (NOT A DRIVE REG)
RMBA	= 4	:UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
RMDA	= 6	:DESIRED TRK/SEC ADDRESS REGISTER (DRIVE REG. 5)
RMCS2	= 10	:CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
RMDS	= 12	:DRIVE STATUS REGISTER (DRIVE REG 1)
RMER1	= 14	:ERROR REGISTER #1 (DRIVE REG. 2)
RMAS	= 16	:ATTENTION SUMMARY PSEUDOC REGISTER (DRIVE REG. 4)
RMLA	= 20	:LOOK AHEAD REGISTER (DRIVE REG. 7)
RMDB	= 22	:DATA BUFFER REGISTER (NOT A DRIVE REG.)
RMPR1	= 24	:MAINTAINABILITY REGISTER #1 (DRIVE REG. 3)
RMDT	= 26	:DRIVE TYPE REGISTER (DRIVE REG. 6)
RMSN	= 30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
RMOF	= 32	:OFFSET REGISTER (DRIVE REG. 11)
RMDC	= 34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
RMHR	= 36	: 'HOLDING REGISTER' (DRIVE REG. 13)
RMPR2	= 40	:MAINTENANCE REGISTER #2 (DRIVE REG. 14)
RMER2	= 42	:ERROR REGISTER #2 (DRIVE REG. 15)
RMEC1	= 44	:ECC POSITION REGISTER (DRIVE REG. 16)
RMEC2	= 46	:ECC PATTERN REGISTER (DRIVE REG. 17)

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 031376 104412
15 031400 013746 177776
16 031404 012737 000240 177776
17 031412 004737 036134
18 031416 012701 031224
19 031422 012702 031332
20 031426 005021
21 031430 020102
22 031432 101775
23 031434 012702 031354
24 031440 012721 177777
25 031444 020102
26 031446 101774
27 031450 005037 031242
28 031454 005037 031244
29 031460 005037 031246
30 031464 005037 031250
31 031470 013703 031370
32 031474 012723 033526
33 031500 013713 031372
34 031504 013704 031366
35 031510 012764 000C40 000010
36 031516 005001
37 031520 004037 031610
38 031524 000401
39 031526 000402
40 031530 105061 031242
41 031534 005201
42 031536 042701 177770
43 031542 001366
44 031544 012701 000007
45 031550 005037 177776
46 031554 105761 031262
47 031560 001405
48 031562 004737 036062
49 031566 105761 031262
50 031572 001375
51 031574 005301
52 031576 100366
53 031600 012637 177776
54 031604 104413
55 031606 000207
56
57

```

```

·RH70/RM80 DRIVER INITIALIZATION CODE
:THIS ROUTINE WILL DETERMINE WHICH RM80 DRIVES ARE
:AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
:TO THE PROPER STATE FOR EACH DRIVE.
:NOTE: THIS ROUTINE CALLS DRVINT

:CALL
:
JSR PC,RMINIT
RETURN

:NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED

RMINIT: SAVREG
MOV @#PS,-(SP)
MOV #<5*32.>,@#PS
JSR PC,CLRQUE
MOV #RMERRS,R1
MOV #SEEKFG,R2
1$: CLR (R1)+
CMP R1,R2
BLOS 1$
MOV #DTUW,R2
2$: MOV #-1,(R1)+
CMP R1,R2
BLOS 2$
CLR DRVSTA
CLR DRVSTA+2
CLR DRVSTA+4
CLR DRVSTA+6
MOV RMVEC,R3
MOV #ISR,(R3)+
MOV RMVEC+2,(R3)
MOV RMADR,R4
MOV #40,RMCS2(R4)
3$: JSR R0,DRVINT
BR 4$
BR 5$
4$: CLRB DRVSTA(R1)
5$: INC R1
BIC #^C7,R1
BNE 3$
MOV #7,R1
CLR @#PS
6$: TSTB DPINT(R1)
BEQ 8$
JSR PC,SET.IE
7$: TSTB DPINT(R1)
BNE 7$
8$: DEC R1
BPL 6$
MOV (SP)+,@#PS
RESREG
RTS PC

:SAVE R0 - R5
:SAVE THE PRESENT PROCESSOR STATUS
:CHANGE THE PRIORITY TO 5
:CLEAR ALL REQUEST QUEUES
:FIRST ADDRESS TO BE CLEARED
:LAST ADDRESS TO BE CLEARED
:CLEAR
:ARE WE DONE?
:BRANCH IF NO
:LAST ADDRESS
:INITIALIZE
:DONE?
:LOOP IF NO
:SET ALL DRIVES TO OFFLINE

:SETUP THE RH70/RM80 VECTOR
:FIRST ADDRESS OF RH70/RM80
:MASSBUS INIT
:START WITH DRIVE 0
:INIT THE DRIVE
:'DVA' NOT SET
:NORMAL RETURN
:SET DRIVE STATUS TO OFFLINE
:GO TO NEXT DRIVE
:MASK OUT UNUSED BITS
:BR IF MORE DRIVES TO GO
:START WITH DRIVE 7
:CLEAR THE PROCESSOR STATUS
:WAITING FOR DRIVE TO SWITCH PORTS ?
:BR NOT WAITING
:SET INTERRUPT
:DRIVE SWITCHED PORTS ?
:BR IF NOT
:GO TO THE NEXT DRIVE
:CHECK NEXT DRIVE
:RESTORE THE PROCESSOR STATUS
:RESTORE R0 - R5
:BYE-BYE

```

:DRIVE INITIALIZATION ROUTINE

```

58 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
59 ;AN RM80. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
60 ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
61 ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
62 ;DRVSTA IS SET TO THE PROPER CONDITION.
63
64 ;CALL
65
66     MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
67     MOV     RMADR,R4       ;UNIBUS ADDRESS OF RH70/RM80 (RMCS1)
68     JSR     RO,DRVINT      ;CALLED BY A JSR
69     RETURN1 ;ERROR OCCURRED ('NED')
70     RETURN2 ;NORMAL RETURN
71
72 DRVINT: MOV     R5,-(SP)    ;SAVE R5
73         CLRB   DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
74         CLRB   DRVTP(R1)  ;CLEAR THE DRIVE TYPE INDICATOR
75         MOV     R1,RMCS2(R4) ;SELECT A DRIVE
76         MOV     #11,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
77         BIT     #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
78         BEQ     1$        ;NO---BRANCH
79         JSR     PC,SET.IE  ;GO SET 'IE' WITHOUT A 'TRE'
80         BR     4$        ;LEAVE THIS ROUTINE
81         CLRB   DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
82         BIT     #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
83         BEQ     4$        ;BR IF DRIVE NOT AVAILABLE
84         JSR     RO,RD.RM   ;CALL THE READ ROUTINE
85         RMDT   5$        ;REGISTER OFFSET
86         5$        ;'NED' RETURN
87         MOV     (SP)+,R5   ;PUT DRIVE TYPE IN R5
88         MOV     #1,DRVTP(R1) ;SET RM80 INDICATOR
89         CMP     #20026,R5 ;IS IT A SINGLE PORT RM80?
90         BEQ     2$        ;BRANCH IF YES
91         CMP     #24026,R5 ;IS IT A DUAL PORT RM80?
92         BEQ     2$        ;BR IF YES
93         MOV     #-1,DRVTP(R1) ;SET INDICATOR TO 'OTHER'
94         BR     4$        ;EXIT
95         MOV     #121,-(SP) ;DO A 'READ-IN PRESET'
96         JSR     RO,WRT.RM  ;CALL THE WRITE ROUTINE
97         RMCST  5$        ;REGISTER OFFSET
98         5$        ;'NED' RETURN
99         MOV     #BIT12,-(SP) ;SET FMT16=1
100        JSR     RO,WRT.RM  ;CALL THE WRITE ROUTINE
101        RMOF   5$        ;REGISTER OFFSET
102        5$        ;'NED' RETURN
103        JSR     RO,RD.RM   ;CALL THE READ ROUTINE
104        RMDS   5$        ;REGISTER OFFSET
105        5$        ;'NED' RETURN
106        MOV     (SP)+,R5   ;AND SAVE IT IN R5
107        BPL    3$        ;BRANCH IF ATA=0
108        MOV     ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
109        JSR     RO,RD.RM   ;CALL THE READ ROUTINE
110        RMR1   5$        ;REGISTER OFFSET
111        5$        ;'NED' RETURN
112        ROL    (SP)+      ;IS IT UNSAFE?
113        BPL    3$        ;BR IF NOT
114        MOV     #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
115        BR     4$        ;EXIT
    
```

105	032030	005105		3\$:	COM	R5		;CHECK MOL, DPR, DRY, AND VV
106	032032	042705	167077		BIC	#*C<BIT12:BIT08:BIT07:BIT06>,R5		
107	032036	001003			BNE	4\$		;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
108	032040	112761	000001 031242		MOVB	#1,DRVSTA(R1)		;SET DRIVE STATUS TO ONLINE
109	032046	005720		4\$:	TST	(R0)+		;STEP OVER THE ERROR RETURN
110	032050	012605		5\$:	MOV	(SP)+,R5		;RESTORE R5
111	032052	000200			RTS	RO		;EXIT

```

1      :REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
2
3      :CALL
4
5      :
6      JSR      RO,RM80      :CALL THE RM80 DRIVER
7      PNTADR   :ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
8      RETURN1  :RETURN HERE IF QUEUE IS FULL
9      RETURN2  :RETURN HERE IF REQUEST IS IN QUEUE OR THERE
10     :IS AN ERROR CONDITION
11     032054  013746  177776      RM80:  MOV      @#PS,-(SP)      :SAVE THE CALLING STATUS
12     032060  013737  031372  177776  MOV      RMVEC+2,@#PS    :DON'T ALLOW ANY RM80 INTERRUPTS
13     032066  112737  000001  031326  MOVB     #1,ACTDRV      :SET 'ACTIVE DRIVER' FLAG
14     032074  104412      :SAVE R0 - R5
15     032076  011002      MOV      (R0),R2        :PICKUP THE DRIVE PARAMETER BLOCK POINTER
16     032100  005062  000016      CLR      16(R2)        :CLEAR THE STATUS/ERROR INDICATOR
17     032104  111201      MOVB     (R2),R1        :PICKUP THE DRIVE NUMBER
18     032106  013704  031366      MOV      RMADR,R4      :UNIBUS ADDRESS OF RMCS1
19     032112  105761  031242      TSTB    DRVSTA(R1)    :CHECK DRIVES STATUS
20     032116  003011      BGT      1$           :BRANCH IF ONLINE
21     032120  105761  031262      TSTB    DPINT(R1)     :TRYING TO INIT THE DRIVE
22     032124  001027      BNE      4$           :BR IF YES
23     032126  004037  031610      JSR      RO,DRVINT    :GO INIT. THE DRIVE
24     032132  000421      BR       3$           :ERROR RETURN
25     032134  105761  031242      TSTB    DRVSTA(R1)    :IS DRIVE STATUS ONLINE?
26     032140  003432      BLE      5$           :BR IF NOT
27     032142  105761  031272      1$:  TSTB    DPRQS(R1)    :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
28     032146  001016      BNE      4$           :BR IF YES
29     032150  010164  000010      MOV      R1,RMCS2(R4) :SELECT THE DRIVE
30     032154  004037  036200      JSR      RO,DRVQUE    :PUT THIS REQUEST IN QUEUE
31     032160  000445      BR       8$           :QUEUE IS FULL
32     032162  105761  031232      2$:  TSTB    DRVACT(R1)    :IS THIS DRIVE ACTIVE?
33     032166  001037      BNE      7$           :BR IF YES
34     032170  004737  032312      JSR      PC,OPT       :CALL THE OPTIMIZER
35     032174  000434      BR       7$
36     032176  004737  033410      3$:  JSR      PC,C18      :GO HANDLE THE 'NED'
37     032202  000431      BR       7$
38     032204  004037  036200      4$:  JSR      RO,DRVQUE    :PUT REQUEST IN QUEUE
39     032210  000431      BR       8$           :QUEUE IS FULL
40     032212  032714  000100      BIT     #BIT06,(R4)   :IS 'IE' SET ALREADY ?
41     032216  001023      BNE      7$           :BR IF IT IS
42     032220  004737  036062      JSR      PC,SET.IE    :SET INTERRUPT
43     032224  000420      BR       7$           :RETURN, REQUEST IN QUEUE
44     032226  105761  031242      5$:  TSTB    DRVSTA(R1)    :SEE IF DRIVE OFFLINE OR UNSAFE
45     032232  002412      BLT     6$           :BR IF UNSAFE
46     032234  012762  140000  000016      MOV      #BIT15!BIT14,16(R2) :SET OFFLINE ERROR INDICATOR
47     032242  105761  031252      TSTB    DRVTYP(R1)   :SEE IF OFFLINE OR NONEXISTENT
48     032246  001007      BNE      7$           :BR IF OFFLINE
49     032250  012762  100002  000016      MOV      #BIT15!BIT01,16(R2) :REPORT DRIVE NONEXISTENT
50     032256  000403      BR       7$           :GO TO EXIT
51     032260  012762  110000  000016      6$:  MOV      #BIT15!BIT12,16(R2) :DRIVE IS UNSAFE
52     032266  104413      7$:  RESREG   :RESTORE R0 - R5
53     032270  005720      TST     (R0)+        :SETUP FOR NORMAL RETURN
54     032272  000401      BR      9$          :FINISH UP, THEN EXIT
55     032274  104413      8$:  RESREG   :RESTORE R0 - R5
56     032276  005720      9$:  TST     (R0)+        :CORRECT THE RETURN ADDRESS
57     032300  105037  031326      CLRB    ACTDRV      :CLEAR 'ACTIVE DRIVER' FLAG
    
```

```

58 032304 012637 177776      MOV      (SP)+,@#PS      ;RETURN 'PS' TO USER LEVEL
59 032310 000200              RTS      RO              ;RETURN TO CALLER
60
61                          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
62
63                          ;CALL
64                          MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
65                          JSR      PC,OPT          ;SETUP A COMMAND
66
67 032312 104412      OPT:   SAVREG          ;SAVE R0 - R5
68 032314 013746      MOV      @#PS,-(SP)     ;SAVE PROC. STATUS
69 032320 004737      JSR      PC,GETREQ     ;GET 'DPB' POINTER OF REQUEST
70 032324 005702      TST      R2           ;IS THERE A REQUEST IN QUEUE?
71 032326 001510      BEQ      9$          ;NO--BRANCH TO EXIT
72 032330 032764 004000 000000  BIT      #BIT11,RMCS1(R4) ;IS DVA SET?
73 032336 001407      BEQ      2$          ;BRANCH IF NOT
74 032340 032764 000100 000012  BIT      #BIT6,RMDS(R4) ;IS VV SET ?
75 032346 001003      BNE      2$          ;BR IF IT IS
76 032350 004037 031610      1$:   JSR      RO,DRVINT     ;SEE IF DRIVE STILL ONLINE ?
77 032354 000473      BR       8$          ;'DVA' NOT SET
78 032356 105761 031242      2$:   TSTB     DRVSTA(R1) ;IS DRIVE ONLINE?
79 032362 003014      BGT      3$          ;YES--BRANCH
80 032364 004737      JSR      PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
81 032370 012762 140000 000016  MOV      #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
82 032376 105761 031242      TSTB     DRVSTA(R1)   ;IS DRIVE UNSAFE ?
83 032402 100067      BPL      10$         ;BR TO EXIT IF NOT
84 032404 012762 110000 000016  MOV      #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
85 032412 000463      BR       10$         ;BRANCH TO EXIT
86 032414 012746 000111      3$:   MOV      #111,-(SP)   ;LOAD COMMAND ONTO THE STACK
87 032420 004037 035632      JSR      RO,WRT.RM    ;CALL THE WRITE ROUTINE
      032424 000000      RMCS1          ;REGISTER OFFSET
      032426 032544      8$      ;'NED' RETURN
88 032430 032714 004000      BIT      #BIT11,(R4)  ;DRIVE AVAILABLE ?
89 032434 001432      BEQ      7$          ;BR IF NOT
90 032436 122762 000150 000002  CMPB     #150,2(R2)   ;IS THE REQUEST FOR I/O?
91 032444 002403      BLT      4$          ;YES--BRANCH
92 032446 004737 033032      JSR      PC,C14      ;CALL THE COMMAND INITIATOR
93 032452 000443      BR       10$         ;BRANCH TO EXIT
94 032454 005737 031354      4$:   TST      DTUW        ;DATA TRANSFER UNDERWAY?
95 032460 002015      BGE      6$          ;YES--GO START A SEARCH
96 032462 136137 031356 031324  BITB     ATABIT(R1),SRCHWT ;FINISHED A SEARCH ?
97 032470 001003      BNE      5$          ;IF NE, YES
98 032472 005737 031332      TST      SEEKFG      ;DO IMPLIED SEEKS?
99 032476 100006      BPL      6$          ;IF PL DO A SEARCH
100 032500 146137 031356 031324  5$:   BICB     ATABIT(R1),SRCHWT ;CLEAR 'SEARCH WAIT' KEY
101 032506 004737 032572      JSR      PC,C11      ;START A DATA TRANSFER
102 032512 000423      BR       10$         ;
103 032514 004737 032724      6$:   JSR      PC,C13      ;START A SEARCH
104 032520 000420      BR       10$         ;GO TO THE EXIT
105 032522 112761 177777 031272  7$:   MOVB     #-1,DPROQS(R1) ;SET PORT REQUEST INDICATOR
106 032530 010103      MOV      R1,R3       ;SET UP TO ADDRESS WORDS
107 032532 006303      ASL      R3          ;CONVERT TO WORD INDEX
108 032534 012763 035230 031334  MOV      #15000.,TIMER(R3) ;START 15 SECOND TIMER
109 032542 000402      BR       9$          ;EXIT
110 032544 004737 033410      8$:   JSR      PC,C18      ;PROCESS THE 'NED'
111 032550 032714 000100      9$:   BIT      #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
112 032554 001002      BNE      10$        ;BR IF SET
    
```

113 032556 004737 036062  
 114 032562 012637 177776  
 115 032566 104413  
 116 032570 000207

10\$: JSR PC,SET.IE  
 MOV (SP)+,a#PS  
 RESREG  
 RTS PC

:SFT 'IE' WITHOUT A 'TRE'  
 :RESTORE PROC. STATUS  
 :RESTORE R0 - R5

117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130

:COMMAND INITIATOR

:CALL  
 :  
 : MOV #DRVNUM,R1  
 : MOV #DPB,R2  
 : JSR PC,C1?

:DRIVE NUMBER  
 :ADDRESS OF DPB  
 :C1?= C11,C13, OR C14  
 :WHERE:  
 :C11=DATA TRANSFER  
 :C12=SEARCH REQUESTED BY DATA XFER  
 :C14=NOT DATA TRANSFER

:START A DATA TRANSFER

131 032572 004737 036234  
 132 032576 010237 031322  
 133 032602 010203  
 134 032604 013704 031366  
 135 032610 010164 000010  
 136 032614 062703 000004  
 137 032620 062704 000002  
 138 032624 012324  
 139 032626 012324  
 140 032630 012346  
 141 032632 004037 035632  
 032636 000006  
 032640 033410  
 142 032642 012346  
 143 032644 004037 035632  
 032650 000034  
 032652 033410  
 144 032654 004037 035552  
 032660 000032  
 032662 033410  
 145 032664 042716 001000  
 146 032670 004037 035632  
 032674 000032  
 032676 033410  
 147 032700 016246 000002  
 148 032704 004037 035632  
 032710 000000  
 032712 033410  
 149 032714 010137 031354  
 150 032720 000137 033366

C11: JSR PC,POPQUE  
 MOV R2,TRANSWT  
 MOV R2,R3  
 MOV RMADR,R4  
 MOV R1,RMCS2(R4)  
 ADD #4,R3  
 ADD #2,R4  
 MOV (R3)+,(R4)+  
 MOV (R3)+,(R4)+  
 MOV (R3)+,-(SP)  
 JSR R0,WRT.RM  
 RMDA  
 C18  
 MOV (R3)+,-(SP)  
 JSR R0,WRT.RM  
 RMDC  
 C18  
 JSR R0,RD.RM  
 RMOF  
 C18  
 BIC #BIT09,(SP)  
 JSR R0,WRT.RM  
 RMOF  
 C18  
 MOV 2(R2),-(SP)  
 JSR R0,WRT.RM  
 RMCS1  
 C18  
 MOV R1,DTUW  
 JMP C15

:REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE  
 :PUT REQ. IN TRANSFER WAIT QUEUE  
 :DPB ADDRESS TO R3  
 :RMCS1 ADDRESS  
 :SELECT DRIVE  
 :DESIRED WORD COUNT  
 :RMWC ADDRESS  
 :LOAD WORD COUNT  
 :LOAD BUFFER ADDRESS  
 :LOAD SECTOR AND TRACK  
 :CALL THE WRITE ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN  
 :LOAD CYLINDER ADDRESS  
 :CALL THE WRITE ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN  
 :CALL THE READ ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN  
 :CLEAR 'SKIP SECTOR ERROR INHIBIT'  
 :CALL THE WRITE ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN  
 :LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'  
 :CALL THE WRITE ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN  
 :SET 'DATA TRANSFER UNDERWAY'

:START A SEARCH

154 032724 013704 031366  
 155 032730 010164 000010  
 156 032734 016246 000012  
 157 032740 004037 035632  
 032744 000034  
 032746 033410

C13: MOV RMADR,R4  
 MOV R1,RMCS2(R4)  
 MOV 12(R2),-(SP)  
 JSR R0,WRT.RM  
 RMDC  
 C18

:RMCS1 ADDRESS  
 :SELECT DRIVE  
 :DESIRED CYLINDER ADDRESS  
 :CALL THE WRITE ROUTINE  
 :REGISTER OFFSET  
 : 'NED' RETURN

```

158 032750 110203 000010      MOVB    10(R2),R3      ;PICKUP SECTOR ADDRESS
159 032754 163703 031374      SUB     MXLNDW,R3     ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
160 032760 002002                BGE     1$
161 032762 062703 000037      ADD     #31.,R3
162 032766 010346                1$:    MOV     R3,-(SP)     ;COMBINE THE ADJUSTED SECTOR WITH
163 032770 116266 000011 000001  MOVB    11(R2),1(SP) ;THE DESIRED TRACK
164 032776 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMDA                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
165 033006 012746 000131      MOV     #131,-(SP)   ;START A SEARCH
166 033012 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMCS1               ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
167 033022 156137 031356 031324  BISB    ATAR'T(R1),SRCHWT ;SET 'SEARCH WAIT' KEY
168 033030 000556      BR     CIB
169
170                               ;INITIATE A NON-I/O OPERATION
171
172 033032 013704 031366      CI4:   MOV     RMADR,R4      ;RMCS1 ADDRESS
173 033036 010164 000010      MOV     R1,RMCS2(R4) ;SELECT DRIVE
174 033042 116203 000002      MOVB    2(R2),R3     ;PICKUP THE REQUESTED COMMAND
175 033046 122703 000131      CMPB    #131,R3     ;IS IT A SEARCH COMMAND?
176 033052 001007                BNE     1$           ;BRANCH IF NO
177 033054 016246 000010      MOV     10(R2),-(SP) ;LOAD DESIRED TRACK & SECTOR
178 033060 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMDA                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
179 033070 000403                BR     2$           ;GO LOAD CYLINDER
180 033072 122703 000105      1$:    CMPB    #105,R3   ;IS IT A SEEK COMMAND
181 033076 001007                BNE     3$           ;BRANCH IF NO
182 033100 016246 000012      2$:    MOV     12(R2),-(SP) ;LOAD DESIRED CYLINDER
183 033104 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMDC                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
184 033114 000517                BR     14$
185 033116 122703 000115      3$:    CMPB    #115,R3   ;IS IT AN 'OFFSET' COMMAND?
186 033122 001013                BNE     4$           ;BR IF NO
187 033124 004037 035552      JSR     R0,RD.RM     ;CALL THE READ ROUTINE
                                RMOF                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
188 033134 116216 000001      MOVB    1(R2),(SP)   ;BYTE WHEN LOADING THE REGISTER
189 033140 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMOF                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
190 033150 000501                BR     14$
191 033152 122703 000107      4$:    CMPB    #107,R3   ;IS IT A 'RECALIBRATE' COMMAND?
192 033156 001006                BNE     5$           ;IF NE, NO
193 033160 005046                CLR     -(SP)        ;CYLINDER ZERO
194 033162 004037 035632      JSR     R0,WRT.RM    ;CALL THE WRITE ROUTINE
                                RMDC                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
195 033172 000470                BR     14$          ;CONTINUE
196 033174 122703 000143      5$:    CMPB    #143,R3   ;IS IT A 'SET FORMAT' COMMAND?
197 033200 001014                BNE     6$           ;BRANCH IF NO
198 033202 004037 035552      JSR     R0,RD.RM     ;CALL THE READ ROUTINE
                                RMOF                ;REGISTER OFFSET
                                CIB                 ;'NED' RETURN
033206 000032
033210 033410
    
```



```

199 033212 116266 000001 000001      MOVB      1(R2),1(SP)      ;COMBINE 'FMT16','ECI','MCI', & 'SSEI'
200 033220 004037 035632              JSR      RO,WRT.RM      ;CALL THE WRITE ROUTINE
      033224 000032              RMOF                    ;REGISTER OFFSET
      033226 033410              CIB                    ;'NED' RETURN
201 033230 000436                    BR      12$
202 033232 122703 000141      6$:      CMPS      #141,R3      ;IS IT A 'GET REGISTER' COMMAND?
      033236 001023                    BNE     10$              ;BRANCH IF NO
204 033240 016203 000006      7$:      MOV      6(R2),R3      ;POINTS TO 1ST ADDRESS OF WHERE
      033244 116237 000010 033262      MOVB     10(R2),9$      ;TO PUT THE REGISTER(S)
      033252 116205 000011              MOVB     11(R2),R5      ;INIT. THE INDEX FOR THE FIRST REG.
208 033256 004037 035552      8$:      JSR      RO,RD.RM      ;INDEX OF LAST REG. TO MOVE
209 033262 000000      9$:      RMCS1                    ;READ RM80 REGISTER
      033264 033410              CIB                    ;INDEX OF REG. TO READ
211 033266 012623                    MOV      (SP)+,(R3)+    ;GET THE CONTENTS OF RH70/RM80 REG.
212 033270 023705 033262              CMP      9$,R5          ;LAST REG. BEEN READ?
213 033274 001414                    BEQ     12$              ;GET OUT IF YES
214 033276 062737 000002 033262      ADD      #2,9$          ;INCREASE THE INDEX BY 2
215 033304 000764                    BR      8$              ;LOOP--MORE TO READ
216 033306 122703 000145      10$:     CMPB     #145,R3      ;IS IT A 'SELECT DRIVE' COMMAND?
      033312 001405                    BEQ     12$              ;BRANCH IF YES
218 033314 010346                    MOV      R3,-(SP)      ;LOAD THE COMMAND
219 033316 004037 035632              JSR      RO,WRT.RM      ;CALL THE WRITE ROUTINE
      033322 000070              RMCS1                    ;REGISTER OFFSET
      033324 033410              CIB                    ;'NED' RETURN
220 033326 004737 036234      12$:     JSR      PC,POPQUE      ;REMOVE REQ. FROM QUEUE
221 033332 052762 000200 000016      BIS      #BIT07,16(R2)  ;SET THE 'DONE' BIT
222 033340 005737 031330              TST     SAVEFG          ;SAVE THE RH70/RM80 REGISTERS?
223 033344 100002                    BPL     13$              ;BRANCH IF NO
224 033346 004737 035744              JSR      PC,SVRH70      ;YES--GO SAVE THE REGISTERS
225 033352 000207      13$:     RTS      PC          ;RETURN TO USER
226
227      ;START A NON-DATA TRANSFER OPERATION
228
229 033354 010346      14$:     MOV      R3,-(SP)      ;LOAD THE COMMAND
230 033356 004037 035632              JSR      RO,WRT.RM      ;CALL THE WRITE ROUTINE
      033362 000000              RMCS1                    ;REGISTER OFFSET
      033364 033410              CIB                    ;'NED' RETURN
231
232      ;START THE COMMAND TIMER
233
234 033366 006301      C15:     ASL      R1          ;SET A 10 SECOND TIMER
235 033370 012761 023420 031334      MOV      #10000.,TIMER(R1)
236 033376 006201              ASR      R1
237 033400 112761 000001 031232      MOVB     #1,DRVACT(R1)  ;SET THE DRIVE ACTIVE
238 033406 000207              RTS      PC          ;RETURN TO THE USER
239
240      ;PROCESS A NON-EXISTENT DRIVE
241
242 033410 104412      C18:     SAVREG                    ;SAVE R0 - R5
243 033412 105761 031232      TSTB     DPVACT(R1)      ;DRIVE ACTIVE?
244 033416 001431              BEQ     3$              ;BRANCH IF NO
245 033420 013702 031322      MOV      TRNSWT,R2      ;GET THE 'TRANSFER WAIT' QUEUE
246 033424 020137 031354      CMP      R1,DTUW          ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
247 033430 001402              BEQ     1$              ;BRANCH IF YES
248 033432 004737 036222      JSR      PC,GETREQ      ;GET THE DPB POINTER
249 033436 005702      1$:      TST      R2          ;QUEUE ENTRY FOR DRIVE ?
    
```

250	033440	001403			BEQ	2\$		:BR IF NOT
251	033442	012762	100002	000016	MOV	#BIT15!BIT01, 6(R2)		:SET 'DRIVE NON-EXISTENT' INDICATOR
252	033450	012763	177777	031334	MOV	#-1, TIMER(R3)	2\$:	:STOP THE TIMER
253	033456	105061	031232		CLRB	DRVACT(R1)		:SET 'DRIVE ACTIVE' TO IDLE
254	033462	020137	031354		CMP	R1, DTUW		:IS THIS DRIVE SETUP FOR A TRANSFER
255	033466	001005			BNE	3\$		:BR IF NOT
256	033470	012737	177777	031354	MOV	#-1, DTUW		:RESET THE INDICATOR
257	033476	005037	031322		CLR	TRNSWT		:CLEAR THE TRANSFER QUEUE
258	033502	004737	036166		JSR	PC, EMPTYQ	3\$:	:CLEAR THE DRIVE'S QUEUE
259	033506	105061	031242		CLRB	DRVSTA(R1)		:SET DRIVE TO OFFLINE
260	033512	105061	031252		CLRB	DRVTYP(R1)		:CLEAR THE DRIVE TYPE INDICATOR
261	033516	004737	036062		JSR	PC, SET.IE	4\$:	:SET 'IE' WITHOUT 'TRE'
262	033522	104413			RESREG			:RESTORE R0 - R5
263	033524	000207			RTS	PC		:RETURN

```

1          ;INTERRUPT SERVICE ROUTINE
2
3 033526 112737 000001 C31326 ISR:  MOVB  #1,ACTDRV      ;SET 'ACTIVE DRIVER' FLAG
4 033534 104412          SAVREG          ;SAVE R0 - R5
5 033536 013704 031366          MOV    RMADR,R4      ;ADDRESS OF RMCS1
6 033542 013701 031354          MOV    DTUW,R1      ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
7 033546 002402          BLT    1$           ;BRANCH IF NO DATA TRANSFER UNDERWAY
8 033550 004737 033570          JSR    PC,TD        ;CALL TRANSFER DONE
9 033554 004737 034202          1$:  JSR    PC,SC        ;CALL SPECIAL CONDITIONS
10 033560 104413          RESREG          ;RESTORE R0 - R5
11 033562 105037 031326          CLRB  ACTDRV      ;CLEAR 'ACTIVE DRIVER' FLAG
12 033566 000002          RTI             ;RETURN
13
14          ;TRANSFER DONE ROUTINE
15
16 033570 105061 031232          TD:  CLRB  DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
17 033574 012737 177777 031354          MOV    #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
18 033602 006301          ASL    R1
19 033604 012761 177777 031334          MOV    #-1,TIMER(R1) ;CANCEL TIMEOUT
20 033612 006201          ASR    R1
21 033614 013702 031322          MOV    TRNSWT,R2   ;GET 'DPB' ADDRESS FROM THE
22 033620 005037 031322          CLR    TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
23 033624 052762 000200 000016          BIS    #BIT07,16(R2) ;SET DONE
24 033632 010164 000010          MOV    R1,RMCS2(R4) ;SELECT THE DRIVE
25 033636 004037 035552          JSR    R0,RD.RM    ;CALL THE READ ROUTINE
          RMCS1      ;REGISTER OFFSET
          CIB       ;'NED' RETURN
26 033646 006126          ROL    (SP)+      ;IS TRE=1 ?
27 033650 100413          BMI    3$         ;BR IF YES
28 033652 005737 031330          TST   SAVEFG      ;SAVE THE RH70/RM80 REGISTERS?
29 033656 100002          BPL    1$         ;BRANCH IF NO
30 033660 004737 035744          JSR    PC,SVRH70   ;YES--SAVE THE REGISTERS
38 033664 004737 032312          1$:  JSR    PC,OPT      ;CALL OPTIMIZER
40 033670 000207          RTS    PC         ;RETURN
41 033672 012714 000113          2$:  MOV    #113,(R4) ;RELEASE THE DRIVE
42 033676 000207          RTS    PC         ;RETURN
43
44          3$:
45 033700          JSR    R0,RD.RM    ;CALL THE READ ROUTINE
          RMER1      ;REGISTER OFFSET
          CIB       ;'NED' RETURN
46 033704 000014          BIT    #BIT8!BIT7,(SP)+ ;SEE IF HCRC OR HCE ERRORS
47 033706 033410          BNE    4$         ;IF NE, YES
48 033710 032726 000600          JSR    R0,RD.RM    ;CALL THE READ ROUTINE
          RMER2      ;REGISTER OFFSET
          CIB       ;'NED' RETURN
49 033714 001016          BIT    #SSE,(SP)+ ;SEE IF SKIP SECTOR ERROR
50 033716 004037 035552          BEQ    4$         ;IF EQ, NO
          JSR    R0,RD.RM ;CALL THE READ ROUTINE
          RMOF      ;REGISTER OFFSET
          CIB       ;'NED' RETURN
51 033722 000042          BIT    #SSEI,(SP)+ ;IS THE INHIBIT BIT ALREADY SET ?
          BEQ    SKIP ;IF EQ, NO
52 033724 033410          BEQ    SKIP
53 033726 032726 000040          BIS    #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
54 033732 001407          JSR    PC,EMPTYQ  ;EMPTY THE 'DRIVE'S WAIT' QUEUE
55 033734 004037 035552          JSR    PC,SVRH70  ;SAVE THE RH70/RM80 REGISTERS
56 033740 000032          MOV    #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
          CIB       ;'NED' RETURN
          RTS
    
```

```

60 034000 012714 000113          MOV    #113,(R4)      ;ISSUE A RELEASE TO THE DRIVE
61 034004 000207          RTS     PC           ;RETURN
62
63          ;SKIP SECTOR HANDLING ROUTINE
64
65 034006 010137 031354      SKIP:  MOV    R1,DTUW      ;LOAD ACTIVE DRIVE NUMBER
66 034012 010237 031322      MOV    R2,TRNSWT      ;RESTORE TRANSFER FLAG
67 034016 005062 000016      CLR    $STATUS(R2)    ;CLEAR THE DRIVER STATUS
68 034022 132762 000002 000002  BITB   #2,$COMND(R2)  ;SEE IF HEADER ORDER
72 034030 0C1405          BEQ    1$             ;IF EQ, NO
82 034032 016200 000004      MOV    $WCNT(R2),R0   ;STARTING WORD COUNT
84 034036 016203 000006      MOV    $BUF(R2),R3   ;STARTING BUFFER ADDRESS
101 034042 000417          BR     2$
102
103 034044          1$:
104 034044 004037 035552          JSR    R0,RD.RM      ;CALL THE READ ROUTINE
105 034050 000002          RMWC   ;REGISTER OFFSET
106 034052 033410          C18    ;'NED' RETURN
107 034054 166216 000004      SUB    $WCNT(R2),(SP) ;CALCULATE THE NUMBER OF WORD TRANSFERED
108 034060 042716 000377      BIC    #377,(SP)     ;LEAVE ONLY SECTOR MULTIPLES
109 034064 011603          MOV    (SP),R3       ;COPY THE DIFFERENCE
110 034066 066216 000004      ADD    $WCNT(R2),(SP) ;NEW WORD COUNT
111 034072 012600          MOV    (SP)+,R0      ;COPY THE WORD COUNT
112 034074 006303          ASL    R3            ;CONVERT WORD DIFFERENCE TO A BYTE DIFFERENCE
113 034076 066203 000006      ADD    $BUF(R2),R3   ;NEW BUFFER ADDRESS
114 034102 010346          2$:  MOV    R3,-(SP)      ;BUFFER ADDRESS
115 034104 004037 035632      JSR    R0,WRT.RM     ;CALL THE WRITE ROUTINE
116 034110 000004          RMWA   ;REGISTER OFFSET
117 034112 033410          C18    ;'NED' RETURN
118 034114 010046          MOV    R0,-(SP)     ;WORD COUNT
119 034116 004037 035632      JSR    R0,WRT.RM     ;CALL THE WRITE ROUTINE
120 034122 000002          RMWC   ;REGISTER OFFSET
121 034124 033410          C18    ;'NED' RETURN
122 034126 012714 040111      MOV    #40111,(R4)  ;CLEAR THE DRIVE
123 034132 004037 035552      JSR    R0,RD.RM     ;CALL THE READ ROUTINE
124 034136 000032          RMOF   ;REGISTER OFFSET
125 034140 033410          C18    ;'NED' RETURN
126 034142 052716 001000      BIS    #SSEI,(SP)   ;SET THE INHIBIT BIT
127 034146 004037 035632      JSR    R0,WRT.RM     ;CALL THE WRITE ROUTINE
128 034152 000032          RMOF   ;REGISTER OFFSET
129 034154 033410          C18    ;'NED' RETURN
130 034156 012762 000001 000020  MOV    #1,$SSENB(R2) ;INDICATE THAT SKIP SECTORING WAS ENABLED
131 034164 016246 000002      MOV    $COMND(R2),-(SP) ;GET THE ORIGINAL COMMAND
132 034170 004037 035632      JSR    R0,WRT.RM     ;CALL THE WRITE ROUTINE
133 034174 000000          RMCS1  ;REGISTER OFFSET
134 034176 033410          C18    ;'NED' RETURN
135 034200 000207          RTS     PC           ;RETURN
136
137          ;SPECIAL CONDITION ROUTINE
138
139 034202 116403 000016      SC:   MOVB   RMAS(R4),R3 ;READ 'RMAS'
140 034206 001014          BNE    2$           ;BRANCH IF ANY 'ATA' BITS SET
141 034210 0C4037 035552      JSR    R0,RD.RM     ;CALL THE READ ROUTINE
142 034214 000000          RMCS1  ;REGISTER OFFSET
143 034216 033410          C18    ;'NED' RETURN
144 034220 106126          ROLB   (SP)+       ;IS 'IE'=1?
145 034222 100405          BMI    1$           ;YES, NO DRIVES TO CHECK

```

```

158 034224 004037 036252      JSR    R0,ES.SAV      ;SAVE THE ADDRESS IN 'SESCAPE'
                                EMT    1                    ;REPORT AN ILLEGAL INTERRUPT
159 034232 004737 036062      JSR    PC,SET.IE      ;SET INTERRUPT ENABLE
160 034236 000207              1$:   RTS    PC        ;RETURN
161 034240 005046              2$:   CLR    -(SP)     ;PROCESS ALL DRIVES THAT HAVE
162 034242 110316              MOVB   R3,(SP)       ;AN 'ATA'=1
163 034244 012703 000001      MOV    #1,R3
164 034250 005001              CLR    R1

                                ;PROCESS ALL DRIVES WITH 'ATA' SET
165
166
167
168 034252 030316      SC3:   BIT    R3,(SP)   ;ATA=1?
169 034254 001005      BNE    SC5            ;YES--BRANCH
170 034256 005201      SC4:   INC    R1        ;MOVE TO THE NEXT DRIVE
171 034260 106303      ASLB   R3
172 034262 001373      BNE    SC3            ;BRANCH IF MORE TO CHECK?
173 034264 005726      TST   (SP)+          ;CLEAN OFF THE STACK
174 034266 000207      RTS    PC            ;RETURN TO USER
175
176
                                ;DETERMINE IF THE DRIVE WITH 'ATA' SET IS ACTIVE WITH A COMMAND
177
178 034270 105761 031262      SC5:   TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
179 034274 001402      BEQ    1$            ;BR IF NOT
180 034276 000137 034674      JMP    SC13          ;PROCESS THE DRIVE
181 034302 105761 031272      1$:   TSTB   DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
182 034306 001402      BEQ    2$            ;BR IF NOT
183 034310 000137 034674      JMP    SC13          ;START THE OUTSTANDING COMMAND
184 034314 105761 031232      2$:   TSTB   DRVACT(R1) ;DRIVE ACTIVE ?
185 034320 001022      BNE    SC6            ;BR IF ACTIVE
186
187
                                ;THE DRIVE WAS NOT ACTIVE, FIND OUT WHY IT INTERRUPTED
188
189 034322 004737 034632      JSR    PC,SC12       ;SAVE RMD5, RMER1 AND RMER2
190
                                ;ALSO DO A DRIVE INIT (DRVINT)
191 034326 105761 031262      TSTB   DPINT(R1)     ;TRYING TO INIT THE DRIVE ?
192 034332 001351      BNE    SC4            ;BR IF YES, CHECK ON MORE DRIVES
193 034334 032737 020000 031232  BIT    #BIT13,RMERRS+6 ;ADDRESS PLUG CHANGED ?
194 034342 001005      BNE    4$            ;BR IF YES
195 034344 011605      3$:   MOV    (SP),R5    ;PICKUP (RMA5) BEFORE THE ERROR CALL
196 034346 004037 036252      JSR    R0,ES.SAV     ;SAVE THE ADDRESS IN 'SESCAPE'
                                EMT    2                    ;REPORT THE UNEXPECTED ATTENTION
                                BR     SC4                    ;GO CHECK FOR MORE ATA'S
197 034354 000740
198 034356
                                4$:   JSR    R0,ES.SAV     ;SAVE THE ADDRESS IN 'SESCAPE'
                                EMT    5                    ;REPORT THE ADDRESS PLUG CHANGE
                                BR     SC4                    ;CHECK FOR MORE DRIVES
199 034364 000734
200
201
                                ;THE DRIVE COMPLETED A NON-I/O COMMAND
202
203 034366 006301      SC6:   ASL    R1        ;SETUP TO ADDRESS WORDS
204 034370 012761 177777 031334  MOV    #-1,TIMER(R1) ;STOP THE TIMER
205 034376 006201      ASR    R1            ;RESTORE THE DRIVE ADDRESS
206 034400 004737 036222      JSR    PC,GETREQ     ;GET THE DPB POINTER FROM THE QUEUE
207 034404 010164 000010      MOV    R1,RMCS2(R4) ;SELECT DRIVE
208 034410 004037 035552      JSR    R0,RD.RM      ;CALL THE READ ROUTINE
                                RMD5
                                SC8
                                ;REGISTER OFFSET
                                ;'NED' RETURN
    
```

```

209 034420 011605          MOV      (SP),R5          ;AND PUT IT IN R5
210 034422 006126          ROL      (SP)+           ;WAS THERE AN ERROR?
211 034424 100053          BPL      SC11           ;BR IF NO ERROR
212 034426 004037 035552    JSR      R0,RD.RM       ;CALL THE READ ROUTINE
                                RMER1          ;REGISTER OFFSET
                                SC8            ;'NED' RETURN
213 034436 012605          MOV      (SP)+,R5       ;AND SAVE IT IN R5
214 034440 004737 035744    JSR      PC,SVRH70      ;SAVE RH70/RM80 REGISTERS
215 034444 012746 000111    MOV      #111,-(SP)     ;ISSUE A DRIVE CLEAR
216 034450 004037 035632    JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
                                RMCS1         ;REGISTER OFFSET
                                SC8            ;'NED' RETURN
217 034460 006105          ROL      R5              ;WAS 'UNSAFE' CONDITION =1?
218 034462 100404          BMI      1$             ;BRANCH IF YES
219 034464 052762 100240 000016 BIS      #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
220 034472 000403          BR       2$             ;
221 034474 052762 100220 000016 1$: BIS      #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF UNSAFE ERROR
222 034502 105061 031232    2$: CLRB   DRVACT(R1)    ;SET DRIVE TO IDLE
223 034506 007737 036166    JSR      PC,EMPTYQ      ;DUMP THE QUEUE
224 034512 146137 031356 031324 BICB    ATABIT(R1),SRCHWT ;CLEAR THE SEARCH WAIT FLAG
225 034520 012746 000113    MOV      #113,-(SP)     ;RELEASE COMMAND
226 034524 004037 035632    JSR      R0,WRT.RM      ;CALL THE WRITE ROUTINE
                                RMCS1         ;REGISTER OFFSET
                                SC8            ;'NED' RETURN
227 034534 000137 034256    JMP      SC4             ;CHECK FOR MORE DRIVES
228
229          ;ISR 'NED' PROCESSOR
230
231 034540 004737 036166    SC8: JSR      PC,EMPTYQ    ;CLEAR THE DRIVE'S QUEUE
232 034544 004737 033410    JSR      PC,CIB         ;PROCESS THE 'NED'
233 034550 000137 034256    JMP      SC4             ;CHECK MORE DRIVES
234
235          ;NON-I/O COMMAND TERMINATION ROUTINE
236
237 034554 105061 031232    SC11: CLRB   DRVACT(R1)    ;SET DRIVE IDLE
238 034560 136137 031356 031324 BITB    ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
                                ;AN I/O COMMAND?
239
240 034566 001012          BNE      1$             ;BRANCH IF YES
241 034570 004737 036234    JSR      PC,POPQUE      ;REMOVE REQUEST FROM QUEUE
242 034574 052762 000200 000016 BIS      #BIT07,16(R2)    ;SET 'DONE' BIT
243 034602 005737 031330    TST     SAVEFG          ;SAVE THE REGISTERS?
244 034606 100002          BPL      1$             ;BRANCH IF NO
245 034610 004737 035744    JSR      PC,SVRH70      ;YES--SAVE ALL OF THE RH70/RM80 REG'S
246 034614 116164 031356 000016 1$: MOVB   ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
247 034622 004737 032312    JSR      PC,OPT         ;START A REQUEST
248 034624 000137 034256    JMP      SC4             ;CHECK FOR MORE DRIVES
249
250          ;ERROR PROCESSOR
251
252 034632 010164 000010    SC12: MOV      R1,RMCS2(R4) ;SELECT DRIVE
253 034636 016437 000012 031224 MOV      RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
254 034644 016437 000014 031226 MOV      RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
255 034652 016437 000042 031230 MOV      RMER2(R4),RMERRS+4
256 034660 004037 031610    JSR      R0,DRVINT      ;INIT. THE STATE OF THE DRIVE
257 034664 000401          BR       1$             ;TAKE ERROR EXIT
258 034666 000207          RTS      PC             ;RETURN
259 034670 005726          1$: TST     (SP)+         ;POP PC OFF OF THE STACK
    
```

```

260 034672 000722          BR      SCB          :PROCESS THE PARITY ERROR
261
262          :DUAL PORT REQUEST PROCESSOR
263
264 034674 006301          SC13:  ASL      R1          :SETUP TO ADDRESS WORDS
265 034676 012761 177777 031334  MOV     #-1,TIMER(R1) :STOP THE TIMER
266 034704 006201          ASR      R1          :
267 034706 010164 000010  MOV     R1,RMCS2(R4)  :SELECT THE DRIVE
268 034712 116164 031356 000016  MOVB   ATABIT(Ri),RMAS(R4) :CLEAR THE ATTENTION BIT
269 034720 032714 004000  BIT     #BIT11,(R4)   :DRIVE AVAILABLE ?
270 034724 001006          BNE     1$          :BR IF AVAILABLE
271 034726 006301          ASL      R1          :
272 034730 012761 035230 031334  MOV     #15000..TIMER(R1) :START 15 SECOND TIMER AGAIN
273 034736 006201          ASR      R1          :
274 034740 000433          BR      3$          :EXIT
275 034742 105761 031262          1$:  TSTB   DPINT(R1)   :INITIALIZING THE DRIVE ?
276 034746 001424          BEQ     2$          :BR IF NOT
277 034750 105061 031262          CLRB   DPINT(R1)   :CLEAR THE INIT INDICATOR
278 034754 004037 031610          JSR    R0,DRVINT   :GO INIT THE DRIVE
279 034760 000240          NOP                    :DUMMY PARITY ERROR RETURN
280 034762 105761 031242          TSTB   DRVSTA(R1)  :DRIVE ONLINE ?
281 034766 003014          BGT     2$          :BR IF YES -- START ORDER
282 034770 005702          TST    R2          :QUEUE ENTRY FOR THE DRIVE
283 034772 001416          BEQ     3$          :BR IF NOT
284 034774 004737 036222          JSR    PC,GETREQ   :GET DPB ADDRESS
285 035000 052762 140000 000016  BIS    #BIT15!BIT14,16(R2) :INFORM USER THAT DRIVE OFFLINE
286 035006 004737 035744          JSR    PC,SVRH70   :SAVE THE REGISTERS
287 035012 004737 036166          JSR    PC,EMPTYQ   :EMPTY THE REQUEST QUEUE
288 035016 000404          BR      3$          :
289 035020 105061 031272          2$:  CLRB   DPRQS(R1)  :CLEAR THE PORT REQUEST INDICATOR
290 035024 004737 032312          JSR    PC,OPT      :START THE PENDING REQUEST
291 035030 000137 034256          3$:  JMP     SC4        :PROCESS OTHER DRIVES
    
```



```

1
2
3
4
5
6 035034 005737 031326      :RM80 TIMER ROUTINE
7 035040 001030              :CALL
8 035042 112737 00G001 031327 :      MOV      #TIME,-(SP)      :ELAPSED TIME IN MILLISECONDS ON THE STACK
9 035050 104412              :      JSR      PC,RMTMR        :CALL RM80 TIME ROUTINE
10 035052 005001              RMTMR: TST      ACTDRV          :CHECK 'ACTDRV & ACTSTR'
11 035054 005003              BNE      4$                :IF NON ZERO EXIT
12 035056 005763 031334      MOV      #1,ACTSTR        :SET 'ACTSTR'
13 035062 002407              SAVREG                    :SAVE R0 - R5
14 035064 166663 000002 031334 :      CLR      R1            :START WITH DRIVE 0
15 035072 003003              CLR      R3
16 035074 004737 035126      1$:   TST      TIMER(R3)     :IS THE TIMER RUNNING?
17 035100 000405              BLT      2$                :BRANCH IF NO
18 035102 005201              SUB      2(SP),TIMER(R3)  :COUNT THE INTERVAL
19 035104 005723              BGT      2$                :BR IF NO SOFTWARE TIMEOUT
20 035106 022701 00LJ10      JSR      PC,STO           :CALL SOFTWARE TIMEOUT ROUTINE
21 035112 003361              BR       3$                :GO TO THE EXIT
22 035114 104413              2$:   INC      R1            :MOVE TO NEXT DRIVE
23 035116 105037 031327      TST      (R3)+            :OUT OF DRIVES?
24 035122 012616              CMP      #8.,R1           :BRANCH IF NO
25 035124 000207              BGT      1$                :RESTORE R0 - R5
                                :ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
                                :ADJUST THE STACK
                                :RETURN
                                :SOFTWARE TIMEOUT ROUTINE
                                :NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
                                :OR GREATER
                                :CALL:  STO      #DRVNUM,R1      :DRIVE NUMBER
                                :      MOV      PC,STO          :CALL
                                :      RETURN
37 035126 010146      STO:   MOV      R1,-(SP)     :SAVE R1
38 035130 010346      MOV      R3,-(SP)     :SAVE R3
39 035132 013704 031366      MOV      RMADR,R4     :GET ADDRESS OF 'RMCS1'
40 035136 010164 000010      MOV      R1,RMCS2(R4) :SELECT THE DRIVE
41 035142 004037 035552      JSR      R0,RD.RM     :CALL THE READ ROUTINE
                                :REGISTER OFFSET
                                :'NED' RETURN
42 035152 105726      TST      (SP)+        :IS 'DRY'=1?
43 035154 100473      BMI     ST02          :BR IF YES
44 035156 105761 031262      ST01: TST      DPINT(R1)    :TRYING TO INTIALIZE THE DRIVE ?
45 035162 001070      BNE     ST02          :BR IF YES
46 035164 105761 031272      TST      DPRQS(R1)    :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
47 035170 001065      BNE     ST02          :BR IF YES
48 035172 013702 031322      MOV      TRNSWT,R2    :PICKUP TRANSFER WAIT QUEUE
49 035176 020137 031354      CMP      R1,DTUW     :TRANSFER UNDERWAY ON THIS DRIVE?
50 035202 001402      BEQ     1$            :BRANCH IF YES
51 035204 004737 036222      JSR      PC,GETREQ    :GET DPB ADDRESS
52 035210 052762 101000 000016 1$:   BIS      #BIT15!BIT09,16(R2) :SET THE ERROR FLAGS
53 035216 004737 035744      JSR      PC,SVRH70   :SAVE RM70/RM80 REGISTERS
54 035222 012764 000040 000010      MOV      #BIT05,RMCS2(R4) :'INIT' THE MASS BUS
55 035230 105061 031232      CLRB   DRVACT(P1)    :DRIVE IS IDLE
    
```



56	035234	005001		CLR	R1		;START WITH DRIVE 0
57	035236	005003		CLR	R3		
58	035240	004037	031610	JSR	RO,DRVINT	2\$:	;INIT. THIS DRIVE
59	035244	000475		BR	STO5		;PARITY ERROR RETURN
60	035246	105761	031232	TSTB	DRVACT(R1)		;DRIVE IDLE BEFORE THE INIT.?
61	035252	001414		BEQ	4\$		;YES--BRANCH
62	035254	013702	031322	MOV	TRNSWT,R2		;GET TRANSFER WAIT QUEUE
63	035260	023701	031354	CMP	DTUW,R1		;WAS THERE I/O ON THIS DRIVE?
64	035264	001402		BEQ	3\$		;YES--BRANCH
65	035266	004737	036222	JSR	PC,GETREQ		;GET THE DPB POINTER FROM QUEUE
66	035272	052762	100400	BIS	#BIT15!BIT08,16(R2)	3\$:	;INFORM USER OF INIT.
67	035300	105061	031232	CLRB	DRVACT(R1)		;SET DRIVE ACTIVE TO IDLE
68	035304	012763	177777	MOV	#-1,TIMER(R3)	4\$:	;STOP THE TIMER
69	035312	005723		TST	(R3)+		;UPDATE THE INDEX
70	035314	005201		INC	R1		;INCREMENT THE DRIVE NUMBER
71	035316	022701	000010	CMP	#8.,R1		;LAST DRIVE BEEN CHECKED?
72	035322	003346		BGT	2\$		;NO--LOOP
73	035324	012737	177777	MOV	#-1,DTUW	031354	;NO DATA TRANSFERS UNDERWAY
74	035332	005037	031322	CLR	TRNSWT		;CLEAR TRANSFER WAIT QUEUE
75	035336	004737	036134	JSR	PC,CLRQUE		;CLEAR ALL REQUEST QUEUES
76	035342	000500		BR	STO9		;EXIT
77	035344	116405	000016	MOV	RMA5(R4),R5	STO2:	;READ ATTENTION REG
78	035350	136105	031356	BITB	ATABIT(R1),R5		;IS ATTENTION FOR THIS DRIVE UP ?
79	035354	001017		BNE	STO3		;YES--BRANCH
80	035356	105761	031262	TSTB	DPINT(R1)		;TRYING TO INITIALIZE THE DRIVE ?
81	035362	001031		BNE	STO6		;BR IF YES - DRIVE NOT ONLINE
82	035364	105761	031272	TSTB	DPRQS(R1)		;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
83	035370	001045		BNE	STO7		;BR IF YES - NO RESPONSE TO REQUEST
84	035372	020137	031354	CMP	R1,DTUW		;DATA TRANSFER UNDERWAY FOR THIS DRIVE
85	035376	001267		BNE	STO1		;BR IF NO
86	035400	004037	035552	JSR	RO,RD.RM		;CALL THE READ ROUTINE
	035404	000000		RMCS1			;REGISTER OFFSET
	035406	035440		STO5			; 'NED' RETURN
87	035410	105726		TSTB	(SP)+		
88	035412	100261		BPL	STO1		;BR IF 'RDY'=0
89	035414	105761	031262	TSTB	DPINT(R1)	STO3:	;INITIALIZING THE DRIVE ?
90	035420	001003		BNE	1\$		;BR IF INIT PENDING
91	035422	105761	031272	TSTB	DPRQS(R1)		;PORT REQUEST PENDING ?
92	035426	001446		BEQ	STO9		;BR IF NOT
93	035430	012763	177777	MOV	#-1,TIMER(R3)	1\$:	;STOP THE TIMER
94	035436	000442		BR	STO9		;EXIT
95	035440	004737	033410	JSR	PC,C18	STO5:	;GO HANDLE THE 'NED'
96	035444	000437		BR	STO9		
97	035446	105061	031262	CLRB	DPINT(R1)	STO6:	;CLEAR THE INITIALIZE INDICATOR
98	035452	105061	031242	CLRB	DRVSTA(R1)		;SET UNIT OFFLINE
99	035456	012763	177777	MOV	#-1,TIMER(R3)	031334	;STOP THE TIMER
100	035464	004737	036222	JSR	PC,GETREQ		;GET THE DPB ADDRESS
101	035470	005702		TST	R2		;REQUEST IN QUEUE ?
102	035472	001424		BEQ	STO9		;BR IF NOT
103	035474	052762	140000	BIS	#BIT15!BIT14,16(R2)	000016	;INFORM THE USER DRIVE NOT AVAILABLE
104	035502	000414		BR	STO8		;FINISH
105	035504	012763	177777	MOV	#-1,TIMER(R3)	031334	;STOP THE TIMER
106	035512	105061	031272	CLRB	DPRQS(R1)	STO7:	;CLEAR PORT REQUEST INDICATOR
107	035516	004737	036222	JSR	PC,GETREQ		;GET DPB ADDRESS
108	035522	005702		TST	R2		;QUEUE ENTRY FOR DRIVE ?
109	035524	001407		BEQ	STO9		;BR IF NONE
110	035526	012762	100004	MOV	#BIT15!BIT2,16(R2)	000016	;INFORM USER OF PORT REQUEST ERROR

111 035534	004737	036166	ST08:	JSR	PC,EMPTYQ	:CLEAR THE QUEUE FOR THE DRIVE
112 035540	004737	035744		JSR	PC,SVRH70	:SAVE THE REGISTERS
113 035544	012603		ST09:	MOV	(SP)+,R3	:RESTORE R3
114 035546	012601			MOV	(SP)+,R1	:RESTORE R1
115 035550	000207			RTS	PC	:RETURN

DATA

```

1      ;ROUTINE TO READ A RH70/RM80 REGISTER
2
3      ;CALL
4      JSR      RO,RD.RM      ;GO READ A REGISTER
5      INDEX   ERRADR        ;REG. INDEX FROM BASE
6      ;ERROR ADDRESS--PROCESS ERROR STARTING
7      ;AT THIS ADDRESS
8      RETURN                ;CONTENTS OF REG. IS ON THE STACK
9
10     RD.RM:  MOV      (SP),-(SP)      ;SAVE RO FOR RETURN
11     035552 011646      MOV      RMADR,RD.ADR      ;FORM THE DESIRED ADDRESS
12     035554 013737      ADD      (RO)+,RD.ADR      ;USING THE BASE AND THE INDEX
13     035562 062037      RD.RM1:  MOV      @ (PC)+,(PC)+      ;READ THE DESIRED REGISTER OF THE RM80
14     035570 000000      RD.ADR:  .WORD 0      ;ADDRESS IS FORMED HERE
15     035572 000000      RD.WRD:  .WORD 0      ;REG. CONTENTS PUT HERE
16     035574 013766      MOV      RD.WRD,2(SP)      ;RETURN IT TO THE USER
17     035602 013746      MOV      RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
18     035606 062716      ADD      #RMCS2,(SP)      ;FORM THE ADDRESS OF RMCS2
19     035612 032736      BIT      #BIT12,@(SP)+      ;CHECK THE 'NED' BIT
20     035616 001002      BNE     RD.RM3      ;IF NE, DRIVE NOT PRESENT
21     035620 005720      TST     (RO)+      ;ERROR FREE RETURN
22     035622 000402      BR      RD.RM4      ;EXIT
23     035624 011000      RD.RM3: MOV      (RO),RO      ;ERROR EXIT
24     035626 012616      MOV      (SP)+,(SP)
25     035630 000200      RD.RM4: RTS      RO
26
27     ;ROUTINE TO WRITE A REGISTER
28
29     ;CALL
30     MOV      DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
31     JSR      RO,WRT.RM      ;CALL THE ROUTINE TO LOAD (WRITE) THE REG.
32     INDEX   'NED' RETURN      ;INDEX OF THE REGISTER TO BE LOADED
33     ;ADDRESS TO RETURN TO IF 'NED' ERROR
34     RETURN                ;ERROR FREE RETURN
35
36     WRT.RM: MOV      2(SP),WRT.WD      ;SAVE THE WORD TO WRITE
37     035632 016637      MOV      (SP)+,(SP)      ;ADJUST THE STACK
38     035640 012616      MOV      (RO)+,WRT.AD      ;GET INDEX OF REGISTER TO BE WRITTEN
39     035642 012037      BNE     1$      ;BRANCH IF NOT RMCS1
40     035646 001015      CMPB   #150,WRT.WD      ;IS THE COMMAND FOR DATA TRANSFERS?
41     035650 122737      BLT    1$      ;YES--DON'T GET THE OLD A16 & A17, & PSEL
42     035656 002411      JSR      RO,RD.RM      ;CALL THE READ ROUTINE
43     035660 004037      RMCS1   ;REGISTER OFFSET
44     035664 000000      WRT.R3  ;'NED' RETURN
45     035666 035734      ;THE COMMAND BEFORE SENDING TO THE
46     ;THE RH70/RM80
47
48     SWAB    (SP)
49     BIC     #^C7,(SP)
50     MOVB   (SP)+,WRT.WD+1
51     1$:    ADD      RMADR,WRT.AD      ;FORM THE ADDRESS OF THE DISK REG.
52     WRT.R1: MOV      (PC)+,@(PC)+      ;LOAD THE DESIRED REG.
53     WRT.WD: .WORD 0      ;WORD TO WRITE GOES HERE
54     WRT.AD: .WORD 0      ;ADDRESS IS FORMED HERE
55     035710 012737      MOV      RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
56     035712 000000      ADD      #RMCS2,(SP)      ;FORM THE ADDRESS OF RMCS2
57     035714 000000      BIT      #BIT12,@(SP)+      ;CHECK THE 'NED' BIT
58     035716 013746      BEQ     WRT.R4      ;IF EQ, DRIVE IS PRESENT
59     035722 062716
60     035726 032736
61     035732 001402
    
```

```

56 035734 011000      WRT.R3: MOV      (R0),R0      ;TAKE THE 'NED' EXIT
57 035736 000401      BR        WRT.R5      ;EXIT
58 035740 005720      WRT.R4: TST      (R0)+      ;ADJUST FOR ERROR FREE EXIT
59 035742 000200      WRT.R5: RTS        R0
60
61                      ;ROUTINE TO SAVE THE RH70/RM80 REGISTERS
62
63                      ;CALL
64                      :
65                      :     MOV      #DPBADR,R2      ;DPB POINTER TO R2
66                      :     JSR      PC,SVRH70      ;SAVE THE DRIVES REG'S
67 035744 104412      SVRH70: SAVREG      ;SAVE R0 - R5
68 035746 005702      TST      R2          ;QUEUE ENTRY FOR THE DRIVE ?
69 035750 001442      BEQ      6$         ;BR IF NONE
70 035752 013704 031366  MOV      RMADR,R4
71 035756 111264 000010  MOVB     (R2),RMCS2(R4) ;SELECT DRIVE
72 035762 016203 000014  MOV      14(R2),R3    ;GET THE ERROR TABLE POINTER
73 035766 001433      BEQ      6$         ;EXIT IF NO ADDRESS
74 035770 005037 036024  CLR      3$         ;COUNTER & POINTER
75 035774 023727 036024 000022 1$:     CMP      3$,#RMDB     ;REACHED THE BUFFER REGISTER ?
76 036002 001006      BNE      2$         ;BR IF NOT
77 036004 032764 000200 000010  BIT      #BIT07,RMCS2(R4) ;'OR' SET ?
78 036012 001002      BNE      2$         ;BR IF SET
79 036014 005023      CLR      (R3)+      ;STORE RMDB AS ZEROES
80 036016 000405      BR        4$         ;CONTINUE
81 036020 004037 035552      2$:     JSR      R0,RD,RM    ;READ THE SELECTED REGISTER
82 036024 000000      3$:     .WORD     0      ;REGISTER INDEX
83 036026 036052      5$:     .WORD     0      ;ERROR RETURN ADDRESS
84 036030 012623      MOV      (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
85 036032 023727 036024 000046 4$:     CMP      3$,#RMEC2   ;REACHED THE END ?
86 036040 001406      BEQ      6$         ;BR IF YES
87 036042 062737 000002 036024  ADD      #2,3$      ;INCREMENT THE REGISTER INDEX
88 036050 000751      BR        1$         ;CONTINUE READING THE REGISTERS
89 036052 004737 033410      5$:     JSR      PC,CIB     ;PROCESS THE 'NED'
100 036056 104413      6$:     RESREG      ;RESTORE R0 - R5
102 036060 000207      RTS        PC      ;RETURN
103
104                      ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
105                      ;CALL
106                      :
107                      :     MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
108                      :     JSR      PC,SET.IE      ;SET 'IE'
109                      :     RETURN
110 036062 010446      SET.IE: MOV      R4,-(SP)   ;SAVE R4
111 036064 013704 031366  MOV      RMADR,R4    ;PICKUP ADDRESS OF RMCS1
112 036070 010164 000010  MOV      R1,RMCS2(R4) ;SELECT DRIVE
113 036074 011446      MOV      (R4),-(SP)  ;READ RMCS1
114 036076 052716 040000  BIS      #BIT14,(SP) ;SET THE 'TRE' BIT OF THE WORD READ
115 036102 000316      SWAB     (SP)      ;ADJUST FOR DATO
116 036104 112714 000100  MOVB     #BIT06,(R4) ;SET 'IE'
117 036110 032764 010000 000010  BIT      #BIT12,RMCS2(R4) ;IS 'NED'=1?
118 036116 001002      BNE      1$         ;YES--CLEAR 'TRE'
119 036120 005726      TST      (SP)+      ;CLEAN OFF THE STACK
120 036122 000402      BR        2$         ;
121 036124 112664 000001 1$:     MOVB     (SP)+,1(R4) ;CLEAR 'TRE'
122 036130 012604      2$:     MOV      (SP)+,R4  ;RESTORE R4
123 036132 000207      RTS        PC      ;RETURN TO CALLER
    
```

```

124
125          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
126          ;
127          ;CALL
128          ;      JSR      PC,CLRQUE
129          ;
130 036134 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
131 036136 012702 031302  MOV      #QDRV,R2      ;QUEUE BASE ADDRESS
137 036142 005022          CLR      (R2)+          ;CLEAR ENTRY
138 036144 005022          CLR      (R2)+          ;CLEAR ENTRY
139 036146 005022          CLR      (R2)+          ;CLEAR ENTRY
140 036150 005022          CLR      (R2)+          ;CLEAR ENTRY
141 036152 005022          CLR      (R2)+          ;CLEAR ENTRY
142 036154 005022          CLR      (R2)+          ;CLEAR ENTRY
143 036156 005022          CLR      (R2)+          ;CLEAR ENTRY
144 036160 005022          CLR      (R2)+          ;CLEAR ENTRY
139 036162 104413          RESREG          ;RESTORE R0 - R5
140 036164 000207          RTS      PC
141          ;
142          ;EMPTY THE QUEUE SPECIFIED BY R1
143          ;
144          ;CALL
145          ;      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
146          ;      JSR      PC,EMPTYQ
147          ;
148 036166 006301          EMPTYQ: ASL      R1
149 036170 005061 031302  CLR      QDRV(R1)      ;CLEAR DRIVE QUEUE
150 036174 006201          ASR      R1          ;RESTORE R1
151 036176 000207          RTS      PC
152          ;
153          ;ROUTINE TO PUT A REQUEST IN QUEUE
154          ;
155          ;CALL
156          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER
157          ;      MOV      #DPB,R2        ;ADDRESS OF PARAMETER BLOCK
158          ;      JSR      R0,DRVQUE      ;GO PUT REQUEST IN QUEUE
159          ;      RETURN1          ;RETURN HERE IF QUEUE IS FULL
160          ;      RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
161          ;
162 036200 006301          DRVQUE: ASL      R1
163 036202 005761 031302  TST      QDRV(R1)      ;TEST THE QUEUE ENTRY
164 036206 001003          BNE      1$          ;IF NE, QUEUE ENTRY ALREADY THERE
165 036210 010261 031302  MOV      R2,QDRV(R1)    ;ADD THE QUEUE ENTRY
166 036214 005720          TST      (R0)+          ;TAKE RETURN 2
167 036216 006201          1$: ASR      R1
168 036220 000200          2$: RTS      R0          ;RETURN TO USER
169          ;
170          ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
171          ;
172          ;CALL
173          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
174          ;      JSR      PC,GETREQ      ;GO GET THE REQUEST
175          ;      RETURN          ;R2='DPB' ADDRESS OF THE REQUEST
176          ;      ;R2=0 IF NO REQUEST IN QUEUE
177          ;
178 036222 006301          GETREQ: ASL      R1
179 036224 016102 031302  MOV      QDRV(R1),R2    ;GET THE REQUEST
    
```



.SBTTL GETADR - GET BUS ADDRESS AND VECTOR ADDRESS

:THIS ROUTINE IS USED TO ENSURE THE BUS ADDRESS OF THE RH/RM  
:IS SETUP TO READ THE PROPER VALUE. IT WILL ALSO READ THE  
:ADDRESS FROM THE TTY IF REQUIRED.

:NOTE: THIS ROUTINE DESTROYS R0-R4  
:CALL

		JSR RETURN	PC,GETADR		
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13	036276	005737	001324	GETADR:	TST BUSADR ;INPUT FROM TTY REQUESTED?
14	036302	001436			BEQ 5\$ ;NO--BRANCH
15	036304	005037	001324		CLR BUSADR ;YES--CLEAR THE REQUEST FLAG
16	036310	104401	001231		,SCLF ;CR-LF
17	036314	005227	177777		INC #1 ;FIRST TIME THRU ?
18	036320	001402			BEQ 1\$ ;BR IF YES
19	036322	104401	001231		TYPE ,SCLF ;CR-LF
20	036326	012700	001522	1\$:	MOV #RH.ADR,R0 ;FIRST ADDRESS
21	036332	012703	036466		MOV #MRMCS1,R3 ;'RMCS1='
22	036336	011004			MOV (R0),R4 ;PRESENT RMCS1 ADDRESS
23	036340	004037	026750		JSR R0,GETNUM ;GET NEW RMCS1
24	036344	000402			BR 2\$ ;COMMA
25	036346	000767			BR 1\$ ;PERIOD
26	036350	000412			BR 4\$ ;DOUBLE PERIOD
27	036352	010420		2\$:	MOV R4,(R0)+ ;SAVE NEW RMCS1
28	036354	012703	036476		MOV #MRMVEC,R3 ;'RMVEC='
29	036360	011004			MOV (R0),R4 ;PRESENT RH/RM VECTOR ADDRESS
30	036362	004037	026750		JSR R0,GETNUM ;GET NEW RMVEC
31	036366	000402			BR 3\$ ;COMMA
32	036370	000756			BR 1\$ ;PERIOD
33	036372	000401			BR 4\$ ;DOUBLE PERIOD
34	036374	010420		3\$:	MOV R4,(R0)+ ;SAVE NEW RMVEC
35	036376	010410		4\$:	MOV R4,(R0) ;SAVE INPUT
36	036400	013701	000004	5\$:	MOV ERRVEC,R1 ;SAVE THE ERROR VECTOR
37	036404	012737	036440		MOV #6\$,ERRVEC ;SETUP FOR TRAP
38	036412	005777	143104		TST @RH.ADR ;CHECK FOR RH/RM
39	036416	0137	000004		MOV R1,ERRVEC ;RESTORE ERROR VECTOR
40	036422	012700	001522		MOV #RH.ADR,R0 ;FIRST ADDRESS OF NEW PARAMETERS
41	036426	012701	031366		MOV #RMADR,R1 ;FIRST ADDRESS OF WHERE TO PUT THEM
42	036432	012021			MOV (R0)+,(R1)+ ;BUS ADDRESS
43	036434	012021			MOV (R0)+,(R1)+ ;VECTOR ADDRESS
44	036436	000207			RTS PC ;RETURN
45	036440	010137	000004	6\$:	MOV R1,ERRVEC ;RESTORE ERROR VECTOR
46	036444	022626			CMP (SP)+,(SP)+ ;CLEAN OFF THE STACK
47	036446	104010			EMT 10
48	036450	005737	000042		TST @#42 ;IS THERE A MONITOR?
49	036454	001724			BEQ 1\$ ;NO--GO ASK FOR ADDRESS
50	036456	005037	001330		CLR DRVSEL ;DESELECT ALL DRIVES FROM TEST
51	036462	000137	015142		JMP \$EOP ;RETURN TO \$EOP

52  
53 036466 200 122 115 MRMCS1: .ASCIIZ <CRLF>/RMCS1=/  
54 036476 200 122 115 MRMVEC: .ASCIIZ <CRLF>/RMVEC=/

S11  
EX1  
EX1  
EX1  
EX1  
EX1  
EX1  
EX1  
EX1  
FC  
FER  
FM1  
FS  
FT  
FMC  
F1  
F2  
F3  
F4  
F5  
GE1  
GE1  
GE1  
GE1  
GO  
GTS  
GT1  
GT1  
GT1  
GT1  
GT1  
GT.  
GT.  
GT.  
HCE  
HCI  
HCF  
HT  
IAE  
IBS  
IC  
IE  
ILF  
ILR  
INC  
INC  
INC  
INC  
IOT  
IR  
ISA  
IT  
ITE  
IVO  
LBC

SC  
SCO  
SCT  
SCO  
SC1  
SC1  
SC1  
SC1  
SC2  
SC3  
SC4  
SC5  
SC6  
SC8  
SEA  
SEC  
SEC  
SEE  
SEE  
SEK  
SEK  
SELI  
SER  
SET  
SET  
SET  
SHU  
SKCI  
SKI  
SKI  
SKS  
SKT  
SLA  
SPT  
SP1  
SP1  
SP1  
SP1  
SP1  
SRCI  
SRCI  
SRTI  
SRT  
SRV  
SSE  
SSE  
STA  
STA  
STA  
STA  
STA  
STA  
STA

.SBTTL DPB (DATA PARAMETER BLOCKS)

1					
2					
3	036506	000	DPB.A:	.BYTE	0
4	036507	000		.BYTE	0
5	036510	000		.BYTE	0
6	036511	000		.BYTE	0
7	036512	000000		.WORD	0
8	036514	043540		.WORD	BUFFER
9					
10	036516	000		.BYTE	0
11					
12	036517	000		.BYTE	0
13					
14	036520	000000		.WORD	0
15	036522	036640		.WORD	RM.REG
16					
17					
18					
19					
20					
21	036524	000000		.WORD	0
22					
23					
24					
25					
26	036526	000000		.WORD	0
27					
28	036530	000	DPB.B:	.BYTE	0
29	036531	000		.BYTE	0
30	036532	000		.BYTE	0
31	036533	000		.BYTE	0
32	036534	17777C		.WORD	-2
33	036536	043540		.WORD	BUFFER
34					
35	036540	000		.BYTE	0
36					
37	036541	000		.BYTE	0
38					
39	036542	000000		.WORD	0
40	036544	036640		.WORD	RM.REG
41					
42					
43					
44					
45					
46	036546	000000		.WORD	0
47					
48					
49					
50					
51	036550	000000		.WORD	0
52					
53	036552	000	DPB.C:	.BYTE	0
54	036553	000		.BYTE	0
55	036554	000		.BYTE	0
56	036555	000		.BYTE	0
57	036556	177776		.WORD	-2

: (0) DRIVE NUMBER  
: (1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI  
: (2) COMMAND  
: (3) PSEL AND A17 AND A16  
: (4) WORD COUNT (MUST BE NEG.)  
: (6) BUFFER ADDRESS OR REGISTER TABLE POINTER  
: (10) SECTOR ADDRESS OR FIRST REG. INDEX  
: (11) TRACK ADDRESS OR LAST REG. INDEX  
: (12) CYLINDER ADDRESS  
: (14) ERROR TABLE POINTER  
: POINTS TO THE FIRST OF TWENTY LOCATIONS OF WHERE THE DRIVER IS TO STORE THE RH/RM REGISTERS ON AN ERROR. IF LEFT ZERO REGISTERS ARE NOT SAVED.  
: (16) STATUS/ERROR INDICATOR  
: BIT15=1=>ERROR OCCURRED  
: BIT07=1=>DONE  
: BIT14-BIT09 AND BIT06-BIT03  
: INDICATE TYPE OF ERROR  
: SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

: (0) DRIVE NUMBER  
: (1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI  
: (2) COMMAND  
: (3) PSEL AND A17 AND A16  
: (4) WORD COUNT (MUST BE NEG.)  
: (6) BUFFER ADDRESS OR REGISTER TABLE POINTER  
: (10) SECTOR ADDRESS OR FIRST REG. INDEX  
: (11) TRACK ADDRESS OR LAST REG. INDEX  
: (12) CYLINDER ADDRESS  
: (14) ERROR TABLE POINTER  
: POINTS TO THE FIRST OF TWENTY LOCATIONS OF WHERE THE DRIVER IS TO STORE THE RH/RM REGISTERS ON AN ERROR. IF LEFT ZERO REGISTERS ARE NOT SAVED.  
: (16) STATUS/ERROR INDICATOR  
: BIT15=1=>ERROR OCCURRED  
: BIT07=1=>DONE  
: BIT14-BIT09 AND BIT06-BIT03  
: INDICATE TYPE OF ERROR  
: SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

: (0) DRIVE NUMBER  
: (1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI  
: (2) COMMAND  
: (3) PSEL AND A17 AND A16  
: (4) WORD COUNT (MUST BE NEG.)



58	036560	043540	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
59					;REGISTER TABLE POINTER
60	036562	000	.BYTE	0	;(10) SECTOR ADDRESS OR
61					;FIRST REG. INDEX
62	036563	000	.BYTE	0	;(11) TRACK ADDRESS OR
63					;LAST REG. INDEX
64	036564	000000	.WORD	0	;(12) CYLINDER ADDRESS
65	036566	036640	.WORD	RM.REG	;(14) ERROR TABLE POINTER
66					;POINTS TO THE FIRST OF TWENTY
67					;LOCATIONS OF WHERE THE DRIVER
68					;IS TO STORE THE RH/RM
69					;REGISTERS ON AN ERROR. IF LEFT
70					;ZERO REGISTERS ARE NOT SAVED.
71	036570	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
72					;BIT15=1=>ERROR OCCURRED
73					;BIT07=1=>DONE
74					;BIT14-BIT09 AND BIT06-BIT03
75					;INDICATE TYPE OF ERROR
76	036572	000000	.WORD	0	;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)
77					
78	036574	000	DTADPB: .BYTE	0	;(0) DRIVE NUMBER
79	036575	000	.BYTE	0	;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
80	036576	000	.BYTE	0	;(2) COMMAND
81	036577	000	.BYTE	0	;(3) PSEL AND A17 AND A16
82	036600	000000	.WORD	0	;(4) WORD COUNT (MUST BE NEG.)
83	036602	043540	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
84					;REGISTER TABLE POINTER
85	036604	000	.BYTE	0	;(10) SECTOR ADDRESS OR
86					;FIRST REG. INDEX
87	036605	000	.BYTE	0	;(11) TRACK ADDRESS OR
88					;LAST REG. INDEX
89	036606	000000	.WORD	0	;(12) CYLINDER ADDRESS
90	036610	036640	.WORD	RM.REG	;(14) ERROR TABLE POINTER
91					;POINTS TO THE FIRST OF TWENTY
92					;LOCATIONS OF WHERE THE DRIVER
93					;IS TO STORE THE RH/RM
94					;REGISTERS ON AN ERROR. IF LEFT
95					;ZERO REGISTERS ARE NOT SAVED.
96	036612	000000	.WORD	0	;(16) STATUS/ERROR INDICATOR
97					;BIT15=1=>ERROR OCCURRED
98					;BIT07=1=>DONE
99					;BIT14-BIT09 AND BIT06-BIT03
100					;INDICATE TYPE OF ERROR
101	036614	000000	.WORD	0	;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)
102					
103	036616	000	DPB.R: .BYTE	0	;(0) DRIVE NUMBER
104	036617	000	.BYTE	0	;(1) OFFSET VALUE OR FMT16, ECI, HCI, & SSEI
105	036620	107	.BYTE	RECAL	;(2) COMMAND
106	036621	000	.BYTE	0	;(3) PSEL AND A17 AND A16
107	036622	000000	.WORD	0	;(4) WORD COUNT (MUST BE NEG.)
108	036624	043540	.WORD	BUFFER	;(6) BUFFER ADDRESS OR
109					;REGISTER TABLE POINTER
110	036626	000	.BYTE	0	;(10) SECTOR ADDRESS OR
111					;FIRST REG. INDEX
112	036627	000	.BYTE	0	;(11) TRACK ADDRESS OR
113					;LAST REG. INDEX
114	036630	000000	.WORD	0	;(12) CYLINDER ADDRESS

SYME  
 SMTY  
 SMTY  
 SMTY  
 SMTY  
 SPXC  
 SML  
 SML  
 SOCM  
 SOCM  
 SOVE  
 SPAS  
 SPAS  
 SPSE  
 SQUE  
 SRAM  
 SRDC  
 SRDL  
 SRDS  
 . AE  
 ERRO  
 VIRT  
 DYNM  
 .A:C

```

115 036632 036640
116
117
118
119
120
121 036634 000000
122
123
124
125
126 036636 000000
127
128
129
130 036640 000000
131 036642 000000
132 036644 000000
133 036646 000000
134 036650 000000
135 036652 000000
136 036654 000000
137 036656 000000
138 036660 000000
139 036662 000000
140 036664 000000
141 036666 000000
142 036670 000000
143 036672 000000
144 036674 000000
145 036676 000000
146 036700 000000
147 036702 000000
148 036704 000000
149 036706 000000
150

```

```

        .WORD  RM.REG      ;(14) ERROR TABLE POINTER
                                ;POINTS TO THE FIRST OF TWENTY
                                ;LOCATIONS OF WHERE THE DRIVER
                                ;IS TO STORE THE RH/RM
                                ;REGISTERS ON AN ERROR. IF LEFT
                                ;ZERO REGISTERS ARE NOT SAVED.
121      .WORD  0          ;(16) STATUS/ERROR INDICATOR
                                ;BIT15=1=>ERROR OCCURRED
                                ;BIT07=1=>DONE
                                ;BIT14-BIT09 AND BIT06-BIT03
                                ;INDICATE TYPE OF ERROR
126      .WORD  0          ;SKIP SECTOR ENABLE INDICATOR (ENABLED =1)

;SAVE RH/RM REGISTERS HERE ON ERROR
RM.REG: .WORD  0          ;RMCS1 (776700) CONTROL & STATUS #1
        .WORD  0          ;RMWC (776702) WORD COUNT
        .WORD  0          ;RMB A (776704) BUS ADDRESS
        .WORD  0          ;RMDA (776706) DESIRED SECTOR/TRACK
        .WORD  0          ;RMCS2 (776710) CONTROL & STATUS #2
        .WORD  0          ;RMDS (776712) DISK STATUS
        .WORD  0          ;RMER1 (776714) ERROR REG. #1
        .WORD  0          ;RMAS (776716) ATTENTION SUMMARY
        .WORD  0          ;RMLA (776720) LOOK AHEAD
        .WORD  0          ;RMDB (776722) DATA BUFFER
        .WORD  0          ;RMMR1 (776724) MAINTAINABILITY
        .WORD  0          ;RMDT (776726) DRIVE TYPE
        .WORD  0          ;RMSN (776730) SERIAL NUMBER
        .WORD  0          ;RMOF (776732) OFFSET
        .WORD  0          ;RMDC (776734) DESIRED CYLINDER
        .WORD  0          ;RMRH (776736) CURRENT CYLINDER
        .WORD  0          ;RMMR2 (776740) ERROR REG #2
        .WORD  0          ;RMER2 (776742) ERROR REG #3
        .WORD  0          ;RMEC1 (776744) ECC POSITION
        .WORD  0          ;RMEC2 (776746) ECC PATTERN

        .EVEN

```

CZFI  
CRG  
SSCE  
SOFI  
S40C  
SAPT  
SAST  
SATI  
SATY  
SAUT  
SBAS  
SBDA  
SBDD  
SBEL  
SBUF  
SCHA  
SCKS  
SCM1  
SCM2  
SCM3  
SCM4  
SCM1  
SCNT  
SCNT  
SCNT  
SCOM  
SCPL  
SCRL  
SCYL  
SDB2  
SDBL  
SDEI  
SDEV  
SDEV  
SDIV  
SDOA  
SDTB  
SEND  
SEND  
SENU  
SENV  
SENV  
SEOP  
SEOP  
SERP  
SERP  
SERP  
SERP

```

1
2
3 036710 122 000 MSG.P: .ASCIZ /R/
4 036712 106 103 000 MSG.FC: .ASCIZ /FC/
5 036715 114 103 000 MSG.LC: .ASCIZ /LC/
6 036720 111 103 000 MSG.IC: .ASCIZ /IC/
7 036723 106 124 000 MSG.FT: .ASCIZ /FT/
8 036726 114 124 000 MSG.LT: .ASCIZ /LT/
9 036731 111 124 000 MSG.IT: .ASCIZ /IT/
10 036734 106 123 000 MSG.FS: .ASCIZ /FS/
11 036737 114 123 000 MSG.LS: .ASCIZ /LS/
12 036742 120 101 124 MSG.PAT: .ASCIZ /PAT/
13 036746 075 000 MSG.EQ: .ASCIZ /=/
14 036750 200 103 117 MSG.CS: .ASCIZ <CRLF>/CONTROL SWITCHES=/
15
16 036773 040 057 040 SLASH: .ASCIZ @ / @
17 036777 200 125 116 UNSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
18 037015 040 117 106 UNTOFF: .ASCIZ / OFFLINE/
19 037026 040 117 116 UNTON: .ASCIZ / ONLINE/
20 037036 040 116 117 NOTPRS: .ASCIZ / NOT PRESENT/
21 037053 040 125 116 NOTSAF: .ASCIZ / UNSAFE/
22 037063 040 116 117 NOTRM: .ASCIZ @ NOT AN RM80@
23 037100 040 114 117 LODEV: .ASCIZ / LOAD DEVICE/
24 037115 122 115 070 $RM80: .ASCIZ /RM80/
25 037122 200 104 122 DRIVES: .ASCIZ <CRLF>/DRIVE(S) TO BE TESTED, /
26 037153 116 117 116 NONE: .ASCIZ /NONE/
27 037160 054 040 000 COMMA: .ASCIZ /, /
28 037163 072 040 000 COLON: .ASCIZ /:/
29 037166 200 116 117 NOCLOCK: .ASCIZ <CRLF>/NO KW11-P CLOCK, TIMING TESTS WILL NOT BE PERFORMED/
30 037253 115 102 101 SERIAL: .ASCIZ @MBA S/N: @
31 037265 200 124 105 MSGTST: .ASCIZ <CRLF>/TEST/
32 037273 200 104 122 MSDRIV: .ASCIZ <CRLF>/DRIVE/
33 037302 040 104 122 DROP: .ASCIZ / DROPPED/
34 037313 105 130 103 EXCEED: .ASCIZ /EXCEEDED MAXIMUM ERROR LIMIT/<CRLF>
35 037351 200 116 117 NODRVS: .ASCIZ <CRLF>/NO DRIVES TO TEST/<CRLF>
36 037375 200 116 117 NOTEST: .ASCIZ <CRLF>/NO TESTS SPECIFIED/<CRLF>
37
38 037422 200 012 122 ROTATE: .ASCIZ <CRLF><LF>/ROTATIONAL SPEED TIMES/
39 037453 200 012 117 ONECYL: .ASCIZ <CRLF><LF>/ONE CYLINDER SEEK TIMES/<CRLF>/ * FORWARD/
40 037520 200 012 101 AVERGE: .ASCIZ <CRLF><LF>/AVERAGE SEEK TIMES/<CRLF>/ * FORWARD/
41 037560 200 012 115 MXSEEK: .ASCIZ <CRLF><LF>/MAXIMUM SEEK TIMES/
42 037604 200 040 052 FWD: .ASCIZ <CRLF>/ * FORWARD/
43 037620 200 040 052 REV: .ASCIZ <CRLF>/ * REVERSE/
44 037634 200 012 101 MEASUR: .ASCIZ <CRLF><LF>/AVERAGE SEEK TIME MEASUREMENT/
45
46 037674 200 115 111 MSGMIN: .ASCIZ <CRLF>/MIN=/
47 037702 200 115 101 MSGMAX: .ASCIZ <CRLF>/MAX=/
48 037710 200 101 126 MSGAVG: .ASCIZ <CRLF>/AVG=/
49 037716 060 040 125 MSGOUS: .ASCIZ /0 US/
50 037723 040 102 105 MBELOW: .ASCIZ / BELOW THE MINIMUM OF /
51 037752 040 101 102 MABOVE: .ASCIZ / ABOVE THE MAXIMUM OF /
52 040001 040 123 105 MSGSCH: .ASCIZ / SEARCHES TIMED/
53 040021 040 123 105 MSGSEK: .ASCIZ / SEEKS TIMED/
54 040036 040 116 117 MSGNOT: .ASCIZ / NOT TIMED/
55 040051 040 BLNKS4: .ASCII / /
56 040052 040 BLNKS3: .ASCII / /
57 040053 040 BLNKS2: .ASCII / /

```

SE .

SETA  
SETE  
SFAY  
SFIL  
SFIL  
SFMT  
SGDA  
SGDD  
SGET  
SGTS  
SHD  
SHIB  
SHIN  
SICH  
SINT  
SITE  
SLF  
SLFL  
SLOW  
SLPA

SLPE

SPAD  
SPAD  
SPAD  
SPAD  
SPAD

SPAH  
SPAH  
SPAH  
SPHA  
SPHL  
SPHE  
SPSG  
SPSG  
SPSW  
SPTY  
SPTY  
SPTY  
SPYC  
SNUL  
SNWT

SOCN

ASCIZ MESSAGES

58	040054	040	000	BLNK51:	.ASCIZ	//
59	040055	101	114	114	MSG11X:	.ASCIZ /ALLOWABLE ROTATIONAL SPEED LIMITS FOR RM80/
60	040131	101	114	114	MSG12X:	.ASCIZ /ALLOWABLE ONE CYLINDER SEEK LIMIT/
61	040173	101	114	114	MSG13X:	.ASCIZ /ALLOWABLE AVERAGE SEEK TIME LIMIT/
62	040235	101	114	114	MSG14X:	.ASCIZ /ALLOWABLE MAXIMUM (FORWARD) SEEK TIME LIMIT/

SOPK  
SOVE  
SPAS  
SPAS  
SPSE  
SQUE  
SR2A  
SRAN  
SRDC  
SRDD  
SRDL  
SRDC  
SRDS  
\$REG  
\$REG  
\$REC  
\$REG  
\$REG  
\$REG  
\$REG  
\$REG  
\$RES  
\$RHE  
\$RMB  
\$RTN  
\$SAV  
\$SB2  
\$SCC  
\$SEC  
\$SET  
  
\$SSE  
\$STL  
  
\$SUP  
  
\$S/L  
\$SVP  
\$SWR  
  
\$SWR  
\$SWR  
\$STAT  
\$TES  
  
\$TIM  
  
\$TKE  
\$TKC  
\$TKI  
\$TKC

ERROR HEADER (EM) MESSAGES

.SBTTL ERROR HEADER (EM) MESSAGES

1						
2						
3	040311	122	110	040	EM1:	.ASCIZ /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
4	040363	125	116	105	EM2:	.ASCIZ /UNEXPECTED ATTENTION OCCURRED/
5	040421	101	104	104	EM5:	.ASCIZ /ADDRESS PLUG CHANGE BIT SET/
6	040455	122	110	040	EM10:	.ASCIZ /RH CONTROLLER FAILED TO RESPOND TO ADDRESSING/
7	040533	104	122	111	EM11:	.ASCIZ /DRIVE SELECTED IS NOT ONLINE/
8	040570	111	115	120	EM12:	.ASCIZ /IMPROPER HEADER DATA/
9	040615	104	101	124	EM13:	.ASCIZ /DATA COMPARE FAILURE/
10	040642	104	111	123	EM17:	.ASCIZ /DISK ERROR IN TIMING TEST/
11	040674	103	114	117	EM20:	.ASCIZ /CLOCK (KW11-P) OVERFLOW IN TIMING TEST/
12	040743	104	111	123	EM23:	.ASCIZ /DISK ERROR DURING SEEK/
13	040772	123	105	105	EM24:	.ASCIZ /SEEK NOT COMPLETE WITHIN 120 MS/
14	041032	122	110	057	EM41:	.ASCIZ /RH/RM ERROR/
15	041046	106	101	124	EM46:	.ASCIZ /FATAL WRITE CHECK ERROR/

STKQ  
STKQ  
STKQ  
STKS  
STKS  
STMP  
STMP  
STMP  
STN

STMP  
STPB  
STPF  
STPS  
STRA  
STRA  
STRK  
STRP

STRP  
STST  
STST

STTY  
STYP  
STYP  
STYP  
STYP  
STYP  
STYP  
STYP  
STYP  
SUNI  
SUNI  
SUSW  
SVEC  
SVEC  
SWCN  
SXOF  
SXON



```
1  
2  
3 041503 105 122 122 DH1: .ASCIZ /ERR PC RMAS/  
4 041520 105 122 122 DH2: .ASCIZ /ERR PC DRIVE RMAS RMDS RMER1 RMMR2 RMER2/  
5 041605 122 115 103 DH10: .ASCIZ /RMCS1 ERR PC/  
6 041624 104 122 111 DH11: .ASCIZ /DRIVE ERR PC/  
7 041643 124 105 123 DH12: .ASCIZ /TEST ERR PC TST PC DRIVE CYLNDR TRACK SECTOR/  
8 041732 107 104 103 DH12A: .ASCIZ /GDCYL GDTRK GDSCTR BDCYL BDTRK BDSCTR/  
9 042011 107 104 104 DH13A: .ASCIZ /GDDAT BDDAT WRDCNT GDADR BDADR/  
10 042057 124 105 123 DH17: .ASCIZ /TEST ERR PC DRIVE RMCS1 RMDS RMER1 RMMR2 RMER2/  
11 042154 124 105 123 DH21: .ASCIZ /TEST ERR PC TST PC DRIVE CYLNDR TRACK/  
12 042232 107 104 104 DH21A: .ASCIZ /GDDAT BDDAT WRDCNT SECTOR/  
13 042271 124 105 123 DH23: .ASCIZ /TEST ERR PC DRIVE CYLNDR RMCS1 RMCS2 RMDS/  
14 042356 122 115 105 DH23A: .ASCIZ /RMER1 RMMR2 RMER2 RMDC RMHR/  
15 042423 124 105 123 DH41: .ASCIZ /TEST ERR PC TST PC DRIVE/  
16 042461 124 105 123 DH42: .ASCIZ /TEST ERR PC TST PC DRIVE RMCS1 RMCS2 RMDS/  
17 042546 122 115 105 DH43A: .ASCIZ /RMER1 RMMR2 RMER2/  
18 042574 122 115 103 DH44A: .ASCIZ /RMCS1 RMCS2 RMDS RMHR RMDC RMDA/  
19 042650 122 115 105 DH44B: .ASCIZ /RMER1 RMMR2 RMER2/  
20 042676 122 115 105 DH45A: .ASCIZ /RMER1 RMMR2 RMER2 RMWC RMBA RMDB/  
21 .EVEN
```

(R...  
AT1  
AT2  
AT3  
AT4  
AT5  
AT6  
AT7  
ATA  
ATAB  
ATES  
AUNI  
AUSWI  
AVEC  
AVEC  
AVEC  
AVERI  
BADTI  
BAI  
BASFI  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT0  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT1  
BIT2  
BIT3  
BIT4  
BIT5  
BIT6  
BIT7  
BIT8  
BIT9  
BITS  
BLNK  
BLNK  
BLNK  
BLNK  
BPTV  
BSE  
BUFF

.SBTTL DATA TABLE (DT)

Line	Key	DT1	DT2	DT3	DT4	DT5	DT6	DT7	DT8	DT9	DT10	DT11	DT12	DT13	DT14	DT15	DT16	DT17	DT18	DT19	DT20	DT21	DT22	DT23	DT24	DT25	DT26	DT27	DT28	DT29	DT30	
1																																
2																																
3	042754	001132	001204			DT1:	.WORD	\$ERRPC,\$REG3																								
4	042760	001132	001200	001204		DT2:	.WORD	\$ERRPC,\$REG1,\$REG7, RMERRS, RMERRS+2, RMERRS+6, RMERRS+4																								
5	042776	001212	001132	001200		DT5:	.WORD	\$TMP0,\$ERRPC,\$REG1,\$REG5, RMERRS, RMERRS+2, RMERRS+6, RMERRS+4																								
6	043016	001522	001132			DT10:	.WORD	RM.ADR,\$ERRPC																								
7	043022	001202	001132			DT11:	.WORD	\$REG2,\$ERRPC																								
8	043026	001212	001132	001176		DT12:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS																								
9	043044	001366	001372	001370		DT12A:	.WORD	CYL.DS,TRK.DS,SEC.DS,CYL.RD,TRK.RD,SEC.RD																								
10	043060	001212	001132	001176		DT13:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS																								
11	043076	001140	001142	001206		DT13A:	.WORD	\$GDDAT,\$BDDAT,\$REG4,\$GDADR,\$BDADR																								
12	043110	001212	001132	001352		DT17:	.WORD	\$TMP0,\$ERRPC,CHKDRV, RM.REG, RM.REG+12, RM.REG+14, RM.REG+40, RM.REG+42																								
13	043130	001212	001132	001176		DT21:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS																								
14	043144	001200	001142	001206		DT21A:	.WORD	\$REG1,\$BDDAT,\$REG4,\$REG1																								
15	043154	001212	001132	001352		DT23:	.WORD	\$TMP0,\$ERRPC,CHKDRV,CYL.DS, RM.REG, RM.REG+10, RM.REG+12																								
16	043172	036654	036700	036702		DT23A:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+34, RM.REG+36																								
17	043204	001212	001132	001176		DT41:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV																								
18	043214	001212	001132	001176		DT42:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV, RM.REG, RM.REG+10, RM.REG+12																								
19	043232	001212	001132	001176		DT43:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV, RM.REG, RM.REG+10, RM.REG+12																								
20	043250	036654	036700	036702		DT43A:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42																								
21	043256	001212	001132	001176		DT44:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS																								
22	043274	036640	036650	036652		DT44A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06																								
23	043310	036654	036700	036702		DT44B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42																								
24	043316	001212	001132	001176		DT45:	.WORD	\$TMP0,\$ERRPC,\$REG0,CHKDRV,CYL.DS,TRK.DS,SEC.DS																								
25	043334	036640	036650	036652		DT45A:	.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+36, RM.REG+34, RM.REG+06																								
26	043350	036654	036700	036702		DT45B:	.WORD	RM.REG+14, RM.REG+40, RM.REG+42, RM.REG+2, RM.REG+4, RM.REG+22																								

BUSAD  
BYPAS

C.SWR

CALL.  
CALL.

CALL.  
CALL.  
CHANG  
CHKDR

C11  
C13  
C14  
C15  
C18

CK.CH  
CK.DE  
CK.DI  
CK.NU  
CK.OC  
CKSWR  
CLKST  
CLOSE  
CLR  
CLRQU  
CNTCL

CNTRL  
COLON  
COMMA  
CONT  
COUNT  
CPSAV  
CR  
CRLF

CYINC  
CYL.D  
CYL.R  
DCK  
DDISP  
DECSE  
DECSK  
DELTA  
DF1  
DF10  
DF11  
DF12  
DF13  
DF14



DATA FORMAT (DF) TABLE

.SBTTL DATA FORMAT (DF) TABLE

Line	Address	Value	DF Label	Format	Notes
1					
2					
3	043364	000001	DF1:	.WORD 1	;NUMBER OF DATA HEADERS ;NUMBER OF WORDS IN DATA TABLE ;ALL 3 NUMBERS ARE OCTAL
4	043366	002		.BYTE 2	
5	043367	000		.BYTE 0	
6					
7	043370	000001	DF2:	.WORD 1	
8	043372	007		.BYTE 7	
9	043373	000		.BYTE 0	
10					
11	043374	000001	DF3:	.WORD 1	
12	043376	004		.BYTE 4	
13	043377	000		.BYTE 0	
14					
15	043400	000001	DF4:	.WORD 1	
16	043402	005		.BYTE 5	
17	043403	000		.BYTE 0	
18					
19	043404	000001	DF10:	.WORD 1	
20	043406	002		.BYTE 2	
21	043407	000		.BYTE 0	
22					
23	043410	000001	DF11:	.WORD 1	
24	043412	002		.BYTE 2	
25	043413	000		.BYTE 0	
26					
27	043414	000002	DF12:	.WORD 2	;2 DH'S TO BE TYPED ;7 DATA WORDS FOLLOW THE 1ST DH ;WORDS 1-4 ARE OCTAL 5-7 ARE DECIMAL ;ADDRESS OF 2ND DH ;6 DATA WORDS FOLLOW THE 2ND DH ;ALL WORDS ARE OCTAL
28	043416	007		.BYTE 7	
29	043417	160		.BYTE 160	
30	043420	041732		.WORD DH12A	
31	043422	006		.BYTE 6	
32	043423	000		.BYTE 0	
33					
34	043424	000002	DF13:	.WORD 2	
35	043426	007		.BYTE 7	
36	043427	160		.BYTE 160	
37	043430	042011		.WORD DH13A	
38	043432	005		.BYTE 5	
39	043433	004		.BYTE 4	
40					;WORD 3 IS DECIMAL
41	043434	000000	DF14:	.WORD 0	
42	043436	005		.BYTE 5	
43	043437	004		.BYTE 4	
44					;WORD 3 IS DECIMAL
45	043440	000001	DF17:	.WORD 1	
46	043442	010		.BYTE ^D8	
47	043443	000		.BYTE 0	
48					
49	043444	000002	DF21:	.WORD 2	
50	043446	006		.BYTE 6	
51	043447	060		.BYTE 60	
52	043450	042232		.WORD DH21A	
53	043452	004		.BYTE 4	
54	043453	014		.BYTE 14	
55					
56	043454	000000	DF22:	.WORD 0	
57	043456	004		.BYTE 4	

DF17  
DF2  
DF21  
DF22  
DF23  
DF3  
DF4  
DF41  
DF42  
DF43  
DF44  
DF45  
DFLT  
DH1  
DH10  
DH11  
DH12  
DH12A  
DH13A  
DH17  
DH2  
DH21  
DH21A  
DH23  
DH23A  
DH41  
DH42  
DH43A  
DH44A  
DH44B  
DH45A  
DISPL  
DISPR  
DLT  
DMD  
DORTI  
DPB.A  
  
DPB.B  
  
DPB.C  
  
DPB.R  
DPE  
DPINT  
DPR  
DPRQS  
DRIVE  
DROP  
DRQ  
DRVAC  
DRVCL

58	043457	014		.BYTE	14	
59						
60	043460	000002	DF23:	.WORD	2	
61	043462	007		.BYTE	7	
62	043463	010		.BYTE	10	
63	043464	042356		.WORD	DH23A	;WORD 4 IS DECIMAL
64	043466	005		.BYTE	5	
65	043467	000		.BYTE	0	
66						
67						
68	043470	000001	DF41:	.WORD	1	
69	043472	004		.BYTE	4	
70	043473	000		.BYTE	0	
71						
72	043474	000001	DF42:	.WORD	1	
73	043476	007		.BYTE	7	
74	043477	000		.BYTE	0	
75						
76	043500	000002	DF43:	.WORD	2	
77	043502	007		.BYTE	7	
78	043503	000		.BYTE	0	
79	043504	042546		.WORD	DH43A	
80	043506	003		.BYTE	3	
81	043507	000		.BYTE	0	
82						
83	043510	000003	DF44:	.WORD	3	
84	043512	007		.BYTE	7	
85	043513	160		.BYTE	160	
86	043514	042574		.WORD	DH44A	
87	043516	006		.BYTE	6	
88	043517	000		.BYTE	0	
89	043520	042650		.WORD	DH44B	
90	043522	003		.BYTE	3	
91	043523	000		.BYTE	0	
92						
93	043524	000003	DF45:	.WORD	3	
94	043526	007		.BYTE	7	
95	043527	160		.BYTE	160	
96	043530	042574		.WORD	DH44A	
97	043532	006		.BYTE	6	
98	043533	000		.BYTE	0	
99	043534	042676		.WORD	DH45A	
100	043536	006		.BYTE	6	
101	043537	000		.BYTE	0	
102						
103			.SETTL	START OF READ/WRITE BUFFER		
104	043540		BUFFER:			
105						
106		000200	.END	200		

SYMBOL TABLE

ABASE = 000000	AT4 = 000020	C13 = 032724	DH17 = 042057	DT13 = 043060
ACDW1 = 000000	AT5 = 000040	C14 = 033032	DH2 = 041520	DT13A = 043076
ACDW2 = 000000	AT6 = 000100	C15 = 033366	DH21 = 042154	DT17 = 043110
ACPUOP = 000000	AT7 = 000200	C18 = 033410	DH21A = 042232	DT2 = 042760
ACTDRV = 031326	AUNIT = 000000	CKSWR = 104407	DH23 = 042271	DT21 = 043130
ACTSTR = 031327	AUSWR = 000000	CK.CHR = 030644	DH23A = 042356	DT21A = 043144
ADDW0 = 000000	AVECT1 = 000000	CK.DEC = 030616	DH41 = 042423	DT23 = 043154
ADDW1 = 000000	AVECT2 = 000000	CK.DIG = 030720	DH42 = 042461	DT23A = 043172
ADDW10 = 000000	AVERGE = 037520	CK.NUM = 031104	DH43A = 042546	DT41 = 043204
ADDW11 = 000000	A16 = 000400	CK.OCT = 030570	DH44A = 042574	DT42 = 043214
ADDW12 = 000000	A17 = 001000	CLKSTA = 001342	DH44B = 042650	DT43 = 043232
ADDW13 = 000000	BADTMO = 003176	CLOSE = 030510	DH45A = 042676	DT43A = 043250
ADDW14 = 000000	BAI = 000010	CLR = 000040	DISPLA = 001156	DT44 = 043256
ADDW15 = 000000	BASFLG = 001466	CLRQUE = 036134	DISPRE = 000174	DT44A = 043274
ADDW2 = 000000	BITS = 001560	CNTCLR = 023146	DLT = 100000	DT44B = 043310
ADDW3 = 000000	BIT0 = 000001	CNTRLC = 001322	DMD = 000001	DT45 = 043316
ADDW4 = 000000	BIT00 = 000001	COLON = 037163	DORTI = 025662	DT45A = 043334
ACDW5 = 000000	BIT01 = 000002	COMMA = 037160	DPB.A = 036506	DT45B = 043350
ADDW6 = 000000	BIT02 = 000007	CONT = 000100	DPB.B = 036530	DT5 = 042776
ADDW7 = 000000	BIT03 = 000010	COUNT = 025726	DPB.C = 036552	DVA = 004000
ADDW8 = 000000	BIT04 = 000020	CPSAVE = 016332	DPB.R = 036616	DVC = 000200
ADDW9 = 000000	BIT05 = 000040	CR = 000015	DPE = 000010	EBL = 020000
ADEVCT = 000000	BIT06 = 000100	CRLF = 000200	DPINT = 031262	ECH = 000100
ADEVPM = 000000	BIT07 = 000200	CYINC = 001510	DPR = 000400	ECI = 004000
AENV = 000000	BIT08 = 000400	CYL.DS = 001366	DPROS = 031272	ECRC = 001000
AENVPM = 000000	BIT09 = 001000	CYL.RD = 001360	DRIVES = 037122	EECC = 000020
AFATAL = 000000	BIT1 = 000002	C.SWR = 001314	DROP = 037302	EMPTYQ = 036166
AMADR1 = 000000	BIT10 = 002000	DCK = 100000	DRQ = 004000	EMTVEC = 000030
AMADR2 = 000000	BIT11 = 004000	DDISP = 177570	DRVACT = 031232	EM1 = 040311
AMADR3 = 000000	BIT12 = 010000	DECSEC = 026632	DRVCLR = 000111	EM10 = 040455
AMADR4 = 000000	BIT13 = 020000	DECSK = 006330	DRVINT = 031610	EM11 = 040533
AMAMS1 = 000000	BIT14 = 040000	DELTA = 001450	DRVMSK = 001354	EM12 = 040570
AMAMS2 = 000000	BIT15 = 100000	DFLT = 002456	DRVQUE = 036200	EM13 = 040615
AMAMS3 = 000000	BIT2 = 000004	DF1 = 043364	DRVSEL = 001330	EM17 = 040642
AMAMS4 = 000000	BIT3 = 000010	DF10 = 043404	DRVSTA = 031242	EM2 = 040363
AMSGAD = 000000	BIT4 = 000020	DF11 = 043410	DRVSTYP = 031252	EM20 = 040674
AMSGLG = 000000	BIT5 = 000040	DF12 = 043414	DRY = 000200	EM23 = 040743
AMSGTY = 000000	BIT6 = 000100	DF13 = 043424	DSWR = 177570	EM24 = 040772
AMTYP1 = 000000	BIT7 = 000200	DF14 = 043434	DTADPB = 036574	EM41 = 041032
AMTYP2 = 000000	BIT8 = 000400	DF17 = 043440	DTE = 010000	EM46 = 041046
AMTYP3 = 000000	BIT9 = 001000	DF2 = 043370	DTG = 000020	EM5 = 040421
AMTYP4 = 000000	BLNKS1 = 040054	DF21 = 043444	DTO = 020000	ERINDX = 025062
AOE = 001000	BLNKS2 = 040053	DF22 = 043454	DTUW = 031354	ERMAX = 001316
APASS = 000000	BLNKS3 = 040052	DF23 = 043460	DT00 = 000001	ERR = 040000
APRIOR = 000000	BLNKS4 = 040051	DF3 = 043374	DT01 = 000002	ERRCN = 001512
APTCSU = 000040	BPTVEC = 000014	DF4 = 043400	DT02 = 000004	ERROR = 104000
APTENV = 000001	BSE = 100000	DF41 = 043470	DT03 = 000010	ERRVEC = 000004
APTSIZ = 000200	BUFFER = 043540	DF42 = 043474	DT04 = 000020	FRR.CT = 001464
APTPO = 000100	BUSADR = 001324	DF43 = 043500	DT05 = 000040	ESRC = 004000
ASWREG = 000000	BYPASS = 001350	DF44 = 043510	DT06 = 000100	ES.SAV = 036252
ATA = 100000	CALL.A = 024076	DF45 = 043524	DT07 = 000200	EXCEED = 037313
ATABIT = 031356	CALL.B = 024244	DH1 = 041503	DT08 = 000400	EXIT.A = 011410
ATESTN = 000000	CALL.C = 024466	DH10 = 041605	DT1 = 042754	EXIT0 = 005702
ATO = 000001	CALL.R = 024710	DH11 = 041624	DT10 = 043016	EXIT1 = 006110
AT1 = 000002	CHANGE = 000001	DH12 = 041643	DT11 = 043022	EXIT10 = 010562
AT2 = 000004	CHKDRV = 001352	DH12A = 041732	DT12 = 043026	EXIT11 = 011424
AT3 = 000010	C11 = 032572	DH13A = 042011	DT12A = 043044	EXIT12 = 012126

SYMBOL TABLE

EXIT13	012666	LC	002326	MSG.FT	036723	OPT	032312	RDY	= 000200
EXIT14	013436	LDCMD	024032	MSG.IC	036720	OR	= 000200	RD.ADR	035570
EXIT15	015140	LF	= 000012	MSG.IT	036731	PACK	= 000123	RD.RM	035552
EXIT2	006352	LKS	001546	MSG.LC	036715	PAR	= 000010	RD.FM1	035566
EXIT3	006572	LKV	001542	MSG.LS	036737	PAT	= 002344	RD.RM3	035624
EXIT4	007204	LODEV	037100	MSG.LT	036726	PDA	= 000400	RD.RM4	035630
EXIT5	007430	LODFLT	023516	MSG.PA	036742	PFECH	016754	RD.WRD	035572
EXIT6	007720	LODPRM	023734	MSG.R	036710	PFECH1	016764	READ	= 000171
EXIT7	010316	LOP.CK	025022	MSGOUS	037716	PFECH2	017046	READHD	= 000173
FC	002324	LPB	001556	MSG11X	040056	PFECH3	017100	READIN	= 000121
FER	= 000020	LPS	001554	MSG12X	040131	PFECH4	017110	RECAL	= 000107
FMT16	= 010000	LPTAVL	001326	MSG13X	040173	PFTSTN	017114	RELEAS	= 000113
FS	002340	LP.AVL	023176	MSG14X	040235	PGE	= 002000	RESREG	= 104413
FT	002332	LS	002342	MUR	= 001000	PGM	= 001000	RESVEC	= 000010
FWD	037604	LSC	= 004000	MWP	= 000010	PHA	= 000200	REV	037620
F1	= 000002	LSIT	= 000002	MXF	= 001000	PIP	= 020000	REX	= 010000
F2	= 000004	LST	= 002000	MXSEEK	037560	PIRQ	= 177772	RHVEC	001524
F3	= 000010	LSTPK	001374	MXSTAL	001462	PIRQVE	= 000240	RH.ADR	001522
F4	= 000020	LT	002334	MXLNDW	031374	PKB	001536	RMADR	031366
F5	= 000040	MABOVE	037752	M.DPID	022652	PKC	001510	RMAS	= 000016
GETADR	036276	MBELOW	037123	M.DP40	022710	PKCS	001534	RMAA	= 000004
GETNUM	026750	MCLK	= 004000	M.DP41	022744	PKV	001530	RMCS1	= 000000
GETREG	= 000141	MCPE	= 020000	M.DP42	022754	PLFS	= 002000	RMCS2	= 000010
GETREQ	036222	MDF	= 000100	M.DP44	023006	POPQUE	036234	RMDA	= 000006
GETSWR	026656	MEASUR	037634	M.DP50	023020	PRM	002320	RMDB	= 000022
GN	= 000001	MI	= 000004	NBA	= 100000	PRMLMT	002410	RMDC	= 000034
GTSWR	= 104406	MOC	= 000400	NC1	002346	PRMSG	002434	RMD5	= 000012
GTTST1	027244	MOH	= 020000	NC2	002350	PRMPT	002352	RMDT	= 000026
GTTST2	027330	MOL	= 010000	NED	= 010000	PRMO	002726	RMEC1	= 000044
GTTST3	027620	MPE	= 000400	NEM	= 004000	PRM1	002742	RMEC2	= 000046
GTTST4	027622	MRD	= 002000	NOASSY	= 000001	PRM10	003106	RMERRS	031224
GTTST5	027772	MRMCS1	036466	NOCLOK	037166	PRM11	003124	RMER1	= 000014
GTTST6	030032	MRMVEC	036476	NODRVS	037351	PRM12	003136	RMER2	= 000042
GT.PRM	027106	MS	= 000040	NONE	037153	PRM13	003146	RMHR	= 000036
GT.PR1	027110	MSC	= 000002	NOTEST	037375	PRM14	003156	RMINIT	031376
GT.PR2	027240	MSDRIV	037273	NOTPRS	037036	PRM15	003166	RMLA	= 000020
HCE	= 000200	MSEN	= 010000	NOTRM	037063	PRM2	002764	RMPR1	= 000024
HCI	= 002000	MSER	= 000200	NOTSAF	037053	PRM3	003002	RMPR2	= 000040
HCRC	= 000400	MSGAVG	037710	OBCK	= 100000	PRM4	003020	RMOF	= 000032
HT	= 000011	MSGB01	041445	OBEN	= 040000	PRM5	003036	RMR	= 000004
IAE	= 002000	MSGB02	041375	OCC	= 100000	PRM6	003054	RMSN	= 000030
IBSAVE	016334	MSGB04	041355	OFD	= 000200	PRM7	003072	RMTMR	035034
IC	002330	MSGB05	041305	OFFSET	= 000115	PRO	= 000000	RMVEC	031370
IE	= 000100	MSGB06	041241	OM	= 000001	PR1	= 000040	RMVC	= 000002
ILF	= 000001	MSGB08	041177	ONECYL	037453	PR2	= 000100	RM.REG	036640
ILR	= 000002	MSGB09	041140	OPE	= 020000	PR3	= 000140	RM80	032054
INCCYL	026602	MSGB14	041076	OPI	= 020000	PR4	= 000200	ROTATE	037422
INCEC	023030	MSGMAX	037702	OPNFLC	001336	PR5	= 000240	RPT	002322
INCSK	006266	MSGMIN	037674	OPNPRM	030236	PR6	= 000300	RSTART	005102
INCRK	026552	MSGNOT	040036	OPNTST	030064	PR7	= 000340	RSTRT1	005132
IOTVEC	= 000020	MSGSCH	040001	OPN.CT	030070	PS	= 177776	RTC	= 000117
IR	= 000100	MSGSEK	040021	OPN.N1	030432	PSEL	= 002000	RTURN	015406
ISR	033526	MSGTST	037265	OPN.N2	030436	PSW	= 177776	R6	= 000006
IT	002336	MSG.CS	036750	OPN.X1	030450	PURVEC	= 000024	R7	= 000007
ITEM41	002240	MSG.EG	036746	OPN.X2	030454	QDRV	031302	SAVCSW	001320
IVC	= 010000	MSG.FC	036712	OPN.1	030074	RDCHR	= 104410	SAVEFG	031330
LBC	= 002000	MSG.FS	036734	OPN.2	030116	RDLIN	= 104411	SAVREG	= 104412

## SYMBOL TABLE

SC	034202	STATBL	001746	TEST13	012340	UNS	= 040000	\$DEVCT	001244
SCOPE =	000004	STKLMT=	177774	TEST14	013100	UNSTAT	036777	\$DEVM	001312
SCTRWC=	177400	STO	035126	TEST15	013644	UNTOFF	037015	\$DIV	022604
SCO	= 000100	STO1	035156	TEST2	006246	UNTON	037026	\$DOAGN	015376
SC1	= 000200	STO2	035344	TEST3	006510	UPE	= 020000	\$DTBL	020104
SC11	034554	STO3	035414	TEST4	006744	US1	= 000001	\$ENDAD	015366
SC12	034632	STO5	035440	TEST5	007342	US2	= 000002	\$ENDCT	015322
SC13	034674	STO6	035446	TEST6	007566	US4	= 000004	\$EMULL	015402
SC2	= 000400	STO7	035504	TEST7	010056	VERIFY	025302	\$ENV	001254
SC3	034252	STO8	035534	TICKMS	001344	VV	= 000100	\$ENVM	001255
SC4	034256	STO9	035544	TICKUS	001346	WC	= 000040	\$EOP	015142
SC5	034270	STRTMR	025664	TIMER	031334	WCE	= 040000	\$EOPCT	015314
SC6	034366	STRT1A	003274	TIM.DN	001414	WCEFLG	001434	\$ERFLG	001117
SC8	034540	STRT2A	003316	TIM.PT	001432	WCF	= 000040	\$ERMAX	001131
SEARCH=	000131	ST.CLK	023240	TIM.UP	001376	WD	= 000010	\$ERROR	015674
SFC.DS	001370	ST.LCL	023450	TKVEC =	000060	WLE	= 004000	\$ERRPC	001132
JEC.RD	001364	ST.PCL	023406	TPB	001552	WRCKD =	000151	\$ERRTB	002000
SEEK =	000105	SVADR	001440	TPS	001550	WRCKHD=	000153	\$ERTTL	001126
SEEKFG	031332	SVRH70	035744	TPS50	011422	WRITE =	000161	\$ESCAP	001222
SEKCNT	001446	SVSTAT	001356	TPS60	011404	WRL =	004000	\$ETABL	001254
SEKTMR	001444	SWR	001154	TPVEC =	000064	WRTHD =	000163	\$ETEND	001314
SELDRV=	000145	SWREG	000176	TP50	011414	WRT.AD	035714	\$FATAL	001236
SERIAL	037253	SWO	= 000001	TP60	011376	WRT.RM	035632	\$FFLG	015672
SETFOR=	000143	SW00	= 000001	TRAPVE=	000034	WRT.R1	035710	\$FILLC	001172
SETVEC	004356	SW01	= 000002	TRCKWC	001452	WRT.R3	035734	\$FILLS	001171
SET.IE	036062	SW02	= 000004	TRE =	040000	WRT.R4	035740	\$FMT =	000001
SHUT	021410	SW03	= 000010	TRK.DS	001372	WRT.R5	035742	\$GDADR	001134
SKCNT	001502	SW04	= 000020	TRK.RD	001362	WRT.WD	035712	\$GDDAT	001140
SKI =	04000C	SW05	= 000040	TRNSWT	031322	XXDP	001470	\$GET42	015356
SKIP	034006	SW06	= 000100	TRTVEC=	000014	SAPTHD	001100	\$GTSWR	020524
SKSIZ	001506	SW07	= 000200	TSTMS	001332	SATYC	015452	SHD =	000000
SKTIM	001472	SW08	= 000400	TST0	005442	SATY1	015426	SHIBTS	001100
SLAS4	036773	SW09	= 001000	TST1	005704	SATY3	015434	SHINUM	022600
SPTYT	026060	SW1	= 000002	TST10	010320	SATY4	015444	SICNT	001120
SP11A	001710	SW10	= 002000	TST11	010564	SAUTOB	001150	SINTAG	001151
SP11B	001716	SW11	= 004000	TST12	011426	SBASE	001310	SITEMB	001130
SP12	001724	SW12	= 010000	TST13	012130	SBADR	001136	SLF	001232
SP13	001732	SW13	= 020000	TST14	012670	SBDDAT	001142	SLFLG	015671
SP14	001740	SW14	= 040000	TST15	013440	SBELL	001224	SLOWUM	022602
SRCHWT	031324	SW15	= 100000	TST2	006112	SBUF =	000006	SLPADR	001122
SRCHOO	025500	SW2	= 000004	TST3	006354	SCHARC	017446	SLPERR	001124
SRTDRV	004570	SW3	= 000010	TST4	006574	SCKSWR	020434	SMADR1	001266
SRTINT	004300	SW4	= 000020	TST5	007206	SCMTAG	000114	SMADR2	001272
SRVCLK	023504	SW5	= 000040	TST6	007432	SCM1 =	000006	SMADR3	001276
SSE =	000040	SW6	= 000100	TST7	007722	SCM2 =	000014	SMADR4	001302
SSE1 =	001000	SW7	= 000200	TYPDS =	104405	SCM3 =	000006	SMAIL	001234
STACK =	001100	SW8	= 000400	TYPE =	104401	SCM4 =	000003	SMAMS1	001264
STALL	025220	SW9	= 001000	TYPERR	016336	SCNTLC	021346	SMAMS2	001270
STALL0	001436	TAB.XY=	001114	TYPOC =	104402	SCNTLG	021360	SMAMS3	001274
STALL1	001454	TAP =	040000	TYPON =	104404	SCNTLU	021353	SMAMS4	001300
STALL2	001456	TBITVE=	000014	TYPOS =	104403	SCOPND=	000002	SMEADR	001102
STALL3	001460	TD	033570	TYPTIM	026166	SCPUOP	001262	SMFLG	015670
START	003324	TEST0	005612	T11A	001640	SCRFL	001231	SMNEW	021376
START1	003270	TEST1	006074	T11B	001650	SCYL =	000012	SMSGAD	001250
START2	003312	TEST10	010504	T12	001660	SDBLK	020114	SMSGLG	001252
START3	003260	TEST11	011102	T13	001670	SDB2D	022246	SMSGTY	001234
START4	003302	TEST12	011636	T14	001700	SDECVL	022426	SMSWR	021365





























LST	4-142#								
LSTRK	7-0#	32-197*							
LT	9-11#	13-85	13-362	32-788	32-791	32-793			
M.DP40	31-25#	31-46							
M.DP41	31-30	31-37#							
M.DP42	31-36	31-41#							
M.DP44	31-48	31-53#							
M.DP50	31-23	31-57#							
M.DPID	14-583	30-31	31-14#						
MABOVE	32-752	41-51#							
MBELOW	32-742	41-50#							
MCLK	4-204#								
MCPE	4-89#								
MDF	4-199#								
MEASUR	14-583	41-44#							
MI	4-195#								
MOC	4-201#								
MOH	4-236#								
MOL	4-144#								
MPE	4-111#								
MRD	4-203#								
MRMCS1	39-21	39-53#							
MRMVEC	39-28	39-54#							
MS	4-198#								
MSC	4-194#								
MSDRIV	12-47	32-16	41-32#						
MSEN	4-205#								
MSER	4-200#								
MSG.CS	32-:54	41-14#							
MSG.EQ	32->36	41-13#							
MSG.FC	9-41	41-4#							
MSG.FS	9-47	41-10#							
MSG.FT	9-44	41-7#							
MSG.IC	9-43	41-6#							
MSG.IT	9-46	41-9#							
MSG.LC	9-42	41-5#							
MSG.LS	9-48	41-11#							
MSG.LT	9-45	41-8#							
MSG.PA	41-12#								
MSG.R	9-40	41-3#							
MSGOUS	14-583	32-697	32-702	32-737	32-744	32-747	32-754	32-764	41-49#
MSG11X	7-0	7-0	41-59#						
MSG12X	7-0	41-60#							
MSG13X	7-0	41-61#							
MSG14X	7-0	41-62#							
MSGAVG	14-583	32-755	41-48#						
MSGBO1	7-0	43-10#							
MSGBO2	7-0	43-9#							
MSGBO4	7-0	43-8#							
MSGBO5	7-0	43-7#							
MSGBO6	7-0	43-6#							
MSGBO8	7-0	43-5#							
MSGBO9	7-0	43-4#							
MSGB14	7-0	43-3#							
MSGMAX	32-698	32-745	41-47#						
MSGMIN	32-694	32-735	41-46#						

















CZRNGAO RM80 FCTNL PT4 MACRO V04.00 15-JAN-82 07:05:59 PAGE S-20  
CROSS REFERENCE TABLE (CREF V04.00 )

WRT.R4	38-55	38-58#												
WRT.R5	38-57	38-59#												
WRT.RM	34-93	34-95	35-87	35-141	35-143	35-146	35-148	35-157	35-164	35-166	35-178	35-183	35-189	35-194
	35-200	35-219	35-230	36-117	36-119	36-123	36-126	36-216	36-226	38-36#				
WRT.WD	38-36*	38-40	38-47*	38-50#										
WRTHD	4-304#													
XXDP	7-0#	10-49*	10-52*	10-53	10-55*	10-60	10-71	10-123	10-125	11-27	11-29			

ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789



GETPRI	4-80#													
GETSWR	4-80#	10-37	10-37#											
LOOP	9-541#	13-119	13-126	13-151	13-154	13-185	13-206	13-250	13-286	13-339	13-341			
MORE.S	9-512#	13-44	13-79	13-109	13-141	13-170	13-239	13-274	13-316	13-380	14-23	14-140	14-221	14-305
	14-388													
MORETA	5-15#	6-0	8-1#											
MSG	13-37#	13-44	13-72#	13-79	13-98#	13-109	13-134#	13-141	13-162#	13-170	13-228#	13-239	13-265#	13-274
	13-307#	13-316	13-369#	13-380	14-13#	14-23	14-131#	14-140	14-207#	14-221	14-289#	14-305	14-377#	14-388
MULT	4-80#													
NEWTST	4-80#	13-44	13-79	13-109	13-141	13-170	13-239	13-274	13-316	13-380	14-23	14-140	14-221	14-305
	14-388													
POP	4-80#	16-1	16-1	21-1	24-1	29-1	32-173	32-199	32-778					
PUSH	4-80#	16-1	16-1	16-1	21-1	24-1	29-1	32-130	32-182	32-722				
READ	33-8#	34-83	34-96	34-100	35-144	35-187	35-198	36-25	36-44	36-47	36-50	36-103	36-121	36-155
	36-208	36-212	37-41	37-86	38-42									
REPORT	4-80#	9-482#	14-204	14-286	14-374									
SAV.RH	9-466#	14-84	14-93	14-103	14-177	14-195	14-254	14-266	14-277	14-340	14-352	14-365	14-467	14-515
	14-593													
SET.TN	9-487#	13-44	13-79	13-109	13-141	13-170	13-239	13-274	13-316	13-380	14-23	14-140	14-221	14-305
	14-388													
SETPRI	4-80#	22-1												
SETTRA	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1	25-1#		
SETUP	4-80#	10-32												
SKIP	4-80#													
SLASH	4-80#	13-1	13-25	13-29	13-35	14-3	14-11							
STARS	4-80#	4-289	5-9	5-12	5-12	5-12	6-0	6-0	6-0	8-219	8-232	13-44	13-44	13-79
	13-79	13-109	13-109	13-141	13-141	13-170	13-170	13-239	13-239	13-274	13-274	13-316	13-316	13-380
	13-380	14-23	14-23	14-140	14-140	14-221	14-221	14-305	14-305	14-388	14-388	15-1	16-1	17-1
	19-1	20-1	21-1	22-1	22-1	22-1	22-1	22-1	23-1	24-1	25-1	26-1	27-1	28-1
	29-1	33-36												
SWRSU	4-80#	10-32	10-32#											
TRMTRP	25-1#													
TSTTYP	4-17#	13-45	13-80	13-110	13-142	13-171	13-240	13-275	13-317	13-381	14-24	14-141	14-222	14-306
	14-389													
TYPB2D	9-474#	32-736	32-741	32-743	32-746	32-751	32-753	32-763	32-766	32->37				
TYPBIN	4-80#													
TYPDEC	4-80#	18-86												
TYPEND	14-602#	15-1												
TYPNAM	4-80#	10-37												
TYPNUM	4-80#													
TYPOCS	4-80#	10-103	11-62	12-48	13-45	13-80	13-110	13-142	13-171	13-240	13-275	13-317	13-381	14-24
	14-141	14-222	14-306	14-389	15-1	32-17	32->06							
TYPOCT	4-80#	15-1	18-84	22-1	32-:81									
TYPTXT	4-80#	10-6	10-58	10-69	10-75	15-1	15-1	15-1	15-1	18-52	18-66	32-:53	32-<17	32-<33
	32->05													
WRITE	33-18#	34-93	34-95	35-87	35-141	35-143	35-146	35-148	35-157	35-164	35-166	35-178	35-183	35-189
	35-194	35-200	35-219	35-230	36-117	36-119	36-123	36-126	36-216	36-226				